

3rd Edition

SYSTEMS ANALYSIS AND DESIGN

BIBLIOTEX
Digital Library

Systems Analysis and Design

3rd Edition

BIBLIOTEX
Digital Library
www.bibliotex.com

SYSTEMS ANALYSIS AND DESIGN

3RD EDITION



www.bibliotex.com
email: info@bibliotex.com

e-book Edition 2023
ISBN: 978-1-98468-218-5 (e-book)

This book contains information obtained from highly regarded resources. Reprinted material sources are indicated. Copyright for individual articles remains with the authors as indicated and published under Creative Commons License. A Wide variety of references are listed. Reasonable efforts have been made to publish reliable data and views articulated in the chapters are those of the individual contributors, and not necessarily those of the editors or publishers. Editors or publishers are not responsible for the accuracy of the information in the published chapters or consequences of their use. The publisher assumes no responsibility for any damage or grievance to the persons or property arising out of the use of any materials, instructions, methods or thoughts in the book. The editors and the publisher have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission has not been obtained. If any copyright holder has not been acknowledged, please write to us so we may rectify.

Notice: Registered trademark of products or corporate names are used only for explanation and identification without intent of infringement.

© 2023 3G E-learning LLC

In Collaboration with 3G E-Learning LLC. Originally Published in printed book format by 3G E-Learning LLC with ISBN 978-1-98467-968-0

EDITORIAL BOARD



Aleksandar Mratinković was born on May 5, 1988. in Arandjelovac, Serbia. He has graduated on Economic high school (2007), The College of Tourism in Belgrade (2013), and also has a master degree of Psychology (Faculty of Philosophy, University of Novi Sad). He has been engaged in different fields of psychology (Developmental Psychology, Clinical Psychology, Educational Psychology and Industrial Psychology) and has published several scientific works.



Dan Piestun (PhD) is currently a startup entrepreneur in Israel working on the interface of Agriculture and Biomedical Sciences and was formerly president-CEO of the National Institute of Agricultural Research (INIA) in Uruguay. Dan is a widely published scientist who has received many honours during his career including being a two-time recipient of the Amit Golda Meir Prize from the Hebrew University of Jerusalem, his areas of expertise includes stem cell molecular biology, plant and animal genetics and bioinformatics. Dan's passion for applied science and technological solutions did not stop him from pursuing a deep connection to the farmer, his family and nature. Among some of his interest and practices counts enjoying working as a beekeeper and onboard fishing.



Hazem Shawky Fouda has a PhD. in Agriculture Sciences, obtained his PhD. From the Faculty of Agriculture, Alexandria University in 2008, He is working in Cotton Arbitration & Testing General Organization (CATGO).



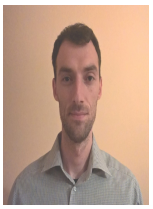
Felecia Killings is the Founder and CEO of LiyahAmore Publishing, a publishing company committed to providing technical and educational services and products to Christian Authors. She operates as the Senior Editor and Writer, the Senior Writing Coach, the Content Marketing Specialist, Editor-in-Chief to the company's quarterly magazine, the Executive and Host of an international virtual network, and the Executive Director of the company's online school for Authors. She is a former high-school English instructor and professional development professor. She possesses a Master of Arts degree in Education and a Bachelor's degree in English and African American studies.



Dr. Sandra El Hajj, Ph.D. in Health Sciences from Nova Southeastern University, Florida, USA is a health professional specialized in Preventive and Global Health. With her 12 years of education obtained from one of the most prominent universities in Beirut, in addition to two leading universities in the State of Florida (USA), Dr. Sandra made sure to incorporate interdisciplinary and multicultural approaches in her work. Her long years of studies helped her create her own miniature world of knowledge linking together the healthcare field with Medical Research, Statistics, Food Technology, Environmental & Occupational Health, Preventive Health and most noteworthy her precious last degree of Global Health. Till today, she is the first and only doctor specialized in Global Health in the Middle East area.



Fozia Parveen has a Dphil in Sustainable Water Engineering from the University of Oxford. Prior to this she has received MS in Environmental Sciences from National University of Science and Technology (NUST), Islamabad Pakistan and BS in Environmental Sciences from Fatima Jinnah Women University (FJWU), Rawalpindi.



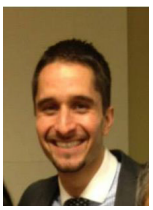
Igor Krunic 2003-2007 in the School of Economics. After graduating in 2007, he went on to study at The College of Tourism, at the University of Belgrade where he got his bachelor degree in 2010. He was active as a third-year student representative in the student parliament.Then he went on the Faculty of science, at the University of Novi Sad where he successfully defended his master's thesis in 2013. The crown of his study was the work titled Opportunities for development of cultural tourism in Cacak". Later on, he became part of a multinational company where he got promoted to a deputy director of logistic. Nowadays he is a consultant and writer of academic subjects in the field of tourism.



Dr. Jovan Pehcevski obtained his PhD in Computer Science from RMIT University in Melbourne, Australia in 2007. His research interests include big data, business intelligence and predictive analytics, data and information science, information retrieval, XML, web services and service-oriented architectures, and relational and NoSQL database systems. He has published over 30 journal and conference papers and he also serves as a journal and conference reviewer. He is currently working as a Dean and Associate Professor at European University in Skopje, Macedonia.



Dr. Tanjina Nur finished her PhD in Civil and Environmental Engineering in 2014 from University of Technology Sydney (UTS). Now she is working as Post-Doctoral Researcher in the Centre for Technology in Water and Wastewater (CTWW) and published about eight International journal papers with 80 citations. Her research interest is wastewater treatment technology using adsorption process.



Stephen obtained his PhD from the University of North Carolina at Charlotte in 2013 where his graduate research focused on cancer immunology and the tumor microenvironment. He received postdoctoral training in regenerative and translational medicine, specifically gastrointestinal tissue engineering, at the Wake Forest Institute of Regenerative Medicine. Currently, Stephen is an instructor for anatomy and physiology and biology at Forsyth Technical Community College.



Michelle holds a Masters of Business Administration from the University of Phoenix, with a concentration in Human Resources Management. She is a professional author and has had numerous articles published in the Henry County Times and has written and revised several employee handbooks for various YMCA organizations throughout the United States.

TABLE OF CONTENTS

Preface xi

CHAPTER 1 INTRODUCTION TO SYSTEM ANALYSIS AND DESIGN 1

Introduction 2
Concept of System Analysis 3
Characterization 4
Characteristics of a System 5
Elements of a System 8
Benefits of Systems Analysis 9
System Development Life Cycle (SDLC) 11
Feasibility Study 14
Analysis 17
Design 19
Development (Coding) 20
Testing 21
Implementation 23
Maintenance and Support 24
Rapid Application Development (RAD) 25
Prototyping 28
Following are the steps in prototyping 29
Conclusion 30

CHAPTER 2 SYSTEM ENGINEERING 47

Introduction 48
System Engineering: An Overview 49
The System Engineering Hierarchy 49
System Modeling 51
Information Engineering 52
Business Process Engineering 55
Information Strategy Planning 56
Enterprise Modeling 58
Business-Level Data Modeling 60

Business Area Analysis 62
Process Modeling 64
Information Flow Modeling 64
System Product Engineering: An Overview 66
Product Engineering 66
Modeling the System Architecture 76
System Modeling and Simulation 79
System Specification 80

CHAPTER 3 ANALYSIS MODELING 89

Introduction 90
Essential of Analysis Modeling 90
Elements of Analysis Model 90
Principles of Analysis Model 92
Rule of Thumb for Analysis Model 93
Analysis Modeling Approaches 93
Elements of Analysis Modeling 94
Importance of Analysis Modeling 95
Data Modeling 95
Cardinality and Modality 98
Entity-Relationship Diagrams 99
Functional Modeling and Information Flow 100
Data Flow Diagrams 100
Extensions for Real-Time Systems 103
Ward and Mellor Extensions 103
Hatley and Pirbhai Extensions 105
Behavioral Modeling 106
Creating an Entity Relationship Diagram 109
Creating a Data Flow Model 110
Creating a Control Flow Model 111
The Control Specification 114
The process Specification 115
Data Dictionary 115
SADT 116

CHAPTER 4 BUSINESS PROCESS MODELING AND DATA ANALYTICS 119

Introduction 120
Business Process Modeling Techniques 120
Business Process Modeling Notation (BPMN) 121
UML Diagrams 123
Flowchart Technique 124

Data Flow Diagrams – Yourdon's Technique 125
Role Activity Diagrams – RAD 126
Role Interaction Diagrams – RID 126
Gantt Charts 127
Integrated Definition for Function Modeling (IDEF) 128
Colored Petri Nets (CPN) 129
Object Oriented Methods 129
Workflow Technique 130
Simulation 130
Business Process Modelling and Workflow Management 131
Preliminaries 132
Workflow Design 134
Modeling and Analysis Using Discrete-Event Simulation 136
Business Process Integration 138
Importance of Business Reference Model 140
Business Process Reengineering 142
Activity Flow–Oriented Approaches 144
Role of Data Analytics in Manufacturing 148
Big Data Analytics in Manufacturing 151
Big Data: How it is Affecting Manufacturing 153
Big Data Examples in the Manufacturing Industry 154

CHAPTER 5 CAPABILITY MODEL FOR BUSINESS ANALYTICS 169

Introduction 170
Concept of Capability Model for Business Analytics 170
Analytic Capability Assessment 174
Capability Assessment Process 179
Predictive Business Analytics and Decision Management 183
Predictive Modeling 184
Decision Analysis and Optimization 185
Transaction Profiling 186
Challenging Established Business Mindsets to Exploit Analytics 186
Abandoned Technological Skills 187
Refined Business Skills 188
Maximizing Data Lakes: Exploiting Analytics 190
Building an Analytical Organization 191
Conclusion 192

CHAPTER 6 OBJECT–ORIENTED SYSTEM DEVELOPMENT 197

Introduction **198**
Procedure-Oriented Paradigms **198**
Classic software development life cycle 200
Development Tools 201
Object-Oriented Notation and Graphs 205
Object–Oriented Analysis **206**
Problem Understanding 206
Requirements Specification 207
Identification of Objects 207
Data flow diagram 208
Textual Analysis 208
Establishing Interconnections 209
Object-Oriented Design **209**
Class Dependencies 211
Organization of Class Hierarchies 213
Design of Classes 214
Design of Member Functions 215
Design of the Driver Program 216
Implementation 216
Prototyping Paradigm 217

CHAPTER 7 USER INTERFACE DESIGN 221

Introduction **222**
Principles for User Interface Design **222**
Layout 223
Content Awareness 225
User Interface Design Process **226**
Use Scenario Development 227
Interface Structure Design 229
Interface Standards Design 230
Interface Design Prototyping **233**
Storyboard 233
Navigation Design **238**
Basic Principles 239
Types of Navigation Controls 240
Input Design **244**
Basic Principles 244
Types Of Inputs 247
Output Design **249**
Basic Principles 249

Types of Outputs 251
Conclusion 257

CHAPTER 8 BUSINESS INTELLIGENCE USER MODEL 261

Introduction 262
Business Intelligence Opportunity Analysis 263
Business Intelligence Opportunity Analysis: Practical Considerations 264
Content Management System 267
Advantages of Content Management System 268
End-User Segmentation 269
Basic Reporting and Querying 271
Online Analytical Processing (OLAP) 273
Data Mining 284
Dashboard 284
Characteristics of a Dashboard 285
The Benefits of Business Dashboards 286
Advanced/Emerging BI Technologies 287

CHAPTER 9 SYSTEM SECURITY 295

Introduction 296
Basic Fundamental of System Security 297
Security System Goal 298
Types of Program Threats 298
Types of System Threats 300
Security Measures Taken 301
Hash Functions in System Security 302
Assurance of System Security 303
System Assurance 305
Software Assurance 306
Hardware Assurance 306
System Description 307
Discipline Management 308
Security Domains 309
Security System Development Life Cycle 310
Security System Handle Cyber Security Threats and Vulnerabilities 312
Port Security in Computer Network 317
Security in Distributed System 319
Information System Security 321
Balancing Information Security and Access 322
Tools of Information Security 322
Information System and Security 323
Active and Passive attacks in Information Security 324

Difference between Information Security and Network Security 328
Conclusion **329**

INDEX **333**

PREFACE

The key to business success is the ability to collect, organize and interpret information. Systems analysis and design is a proven methodology that helps both large and small organizations realize the full benefits of leveraging information. As a systems analyst, the person in the organization most involved with systems analysis and design, you will enjoy a rich career path that will enhance both your computing and interpersonal skills. Organizations are complex systems made up of interconnected and interlocking subsystems. Changes in one part of the system have both expected and unanticipated consequences in other parts of the system. Systems are created to solve problems. You can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design (SAD) mainly deals with the activities of software development. System approval is a way of thinking about the analysis and design of computerized applications. It provides a framework for visualizing the organizational and environmental factors affecting a system. When a computer is introduced into an organization, various functions and malfunctions affect both the user and the organization. System design is the process of planning a new business system or a system that will replace or complement an existing system. But before this planning can be done, we must thoroughly understand the legacy system and determine how computers can best be used to make its operations more effective. In this dynamic world, the subject System Analysis and Design (SAD) mainly deals with the activities of software development. Systems analysts solve business problems by analyzing information systems requirements and designing such systems using analysis and design techniques.

The text in this edition is revised and restructured and new topics are added dealing with the concepts, skills, methodologies, techniques, tools, and perspectives essential for systems analysts.

Chapter 1 starts with the introduction to system analysis and design. Furthermore, it delves systems, analysis modelling, and the software development processes that empower timely and reasonable development of software systems and integrate the strategies and tools together. It also introduces the basic principles and processes of interface design and discusses how to design the interface structure and standard.

Chapter 2 is System Engineering. Systems engineering enables building, analyzing and managing a system—be it electrical, mechanical, chemical, biological or one involving business processes and logistics.

Chapter 3 focuses on Analysis Modeling. Models represent the information that software transform, the function that enables this transformation, the features that the user's desired, and the behavior of the system as transformation is taking place.

Chapter 4 sheds light on Business Process Modeling and Data Analytics. Business process modeling is mainly used to map a workflow so anyone can understand, analyze and make positive changes to that workflow or process.

Chapter 5 focuses on Capability Model for Business Analytics. The capability model is a decomposition of what a business does and can do in a logical and granular grouping. A business capability model is one of the critical business architecture deliverables and a bridge between business and IT, and a foundation for enterprise transformation.

Chapter 6 sheds light on OO System Development. In Object-Oriented Development, we apply object orientation across every system development activity such as requirement analysis, design, programming, testing, and maintenance.

Chapter 7 focuses on User Interface Design. UI design prioritizes the user's visual experience. A good user interface is functional, reliable, and enjoyable to use. User interface design should minimize the effort that the user has to invest interacting with a product and help users accomplish their goals with ease.

Chapter 8 covers the Business Intelligence User Model. The chapter focuses on business intelligence user model that offers opportunities for businesses to transform raw data into meaningful and useful information, in order to build an effective strategic plan, as well as create tactical and operational insights for decision-making within a given timeframe.

Chapter 9 is System Security. System security describes the controls and safeguards that an organization takes to ensure its networks and resources are safe from downtime, interference or malicious intrusion.

CHAPTER 1

INTRODUCTION TO SYSTEM ANALYSIS AND DESIGN



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Explain the concept of system analysis
2. Discuss on system development life cycle (SDLC)
3. Learn about the rapid application development (RAD)

INTRODUCTION

System analysis and design is a process that many companies use to evaluate particular business situations and develop ways to improve them through more optimal methods. Companies may use this process to reshape their organization or meet business objectives related to growth and profitability. System analysis and design also typically emphasize how systems act, their relationships to other subsystems and the ability of both to meet a specific goal. This often involves analyzing a system's performance and the quality of its output.

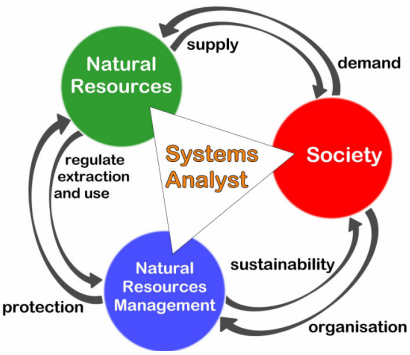
System analysis refers to the process of gathering data, interpreting information, identifying issues and using the results to recommend or develop possible system improvements. During this stage, companies may also evaluate future business needs and how improvements may answer them. System design involves the process in which an organization, in an appropriate situation, develops a newer system or strategy to complement or replace an existing one. This design and development cycle includes planning, analysis, design, implementation and maintenance.

The most common benefit of system analysis and design is improving upon a previous system and enjoying increased operational efficiency. Here's a list of other benefits you and your employing organization may enjoy from this practice:

- Enabling comprehension of complicated structures
- Allowing for better management of any business changes
- Aligning the organization with its environment and strategic priorities
- Minimizing IT issues and reducing the workload of IT employees
- Reducing costs in certain areas, saving the organization money and resources for use in other departments
- Identifying potential risks and threats to the processes before they arise
- Improving the overall quality of the system
- Improving the usability of the system by employees
- Increasing productivity and customer satisfaction

CONCEPT OF SYSTEM ANALYSIS

The word system is widely used. It has become fashionable to attach the word system to add a contemporary flair when referring to things or processes. People speak of exercise system, investment system, delivery system, information system, education system, computer system etc. System may be referred to any set of components, which function in interrelated manner for a common cause or objective.



Systems development can generally be thought of as having two major components: Systems analysis and Systems design. System design is the process of planning a new business system or one to replace or complement an existing system. But before this planning can be done, we must thoroughly understand the old system and determine how computers can best be used to make its operation more effective. System analysis, then, is the process of gathering and interpreting facts, diagnosing problems, and using the information to recommend improvements to the system. This is the job of the systems analyst.

For example, the stockroom operation of clothing store. To better control its inventory and gain access to more up – to – date information about stock levels and reordering, the store asks a system analyst, to "computerize" its stockroom operations. Before one can design a system to capture data, update files, and produce reports, one need to know more about the store operations: what forms are being used to store information manually, such as requisitions, purchase orders, and invoices and what reports are being produced and how they are being used.

Characterization

The term system is derived from the Greek word *systema*, which means an organized relationship among functioning units or components. A system exists because it is designed to achieve one or more objectives. We come into daily contact with the transportation system, the telephone system, the accounting system, the production system, and, for over two decades, the computer system. Similarly, we talk of the business system and of the organization as a system consisting of interrelated departments (subsystems) such as production, sales, personnel, and an information system. None of these subsystems is of much use as a single, independent unit. When they are properly coordinated, however, the firm can function effectively and profitably.



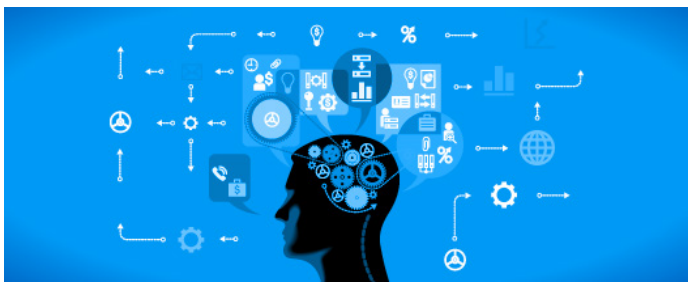
There are more than a hundred definitions of the word system, but most seem to have a common thread that suggests that a system is an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective. The word component may refer to physical parts (engines, wings of aircraft, car), managerial steps (planning, organizing and controlling), or a system in a multi-level structure. The component may be simple or complex, basic or advanced. They may be single computer with a keyboard, memory, and printer or a series of intelligent terminals linked to a mainframe. In either case, each component is part of the total system and has to do its share of work for the system to achieve the intended goal. This orientation requires an orderly grouping of the components for the design of a successful system.

The study of systems concepts, then, has three basic implications:

1. A system must be designed to achieve a predetermined objective.
2. Interrelationships and interdependence must exist among the components.
3. The objectives of the organization as a whole have a higher priority than the objectives of its subsystems.

Characteristics of a System

Our definition of a system suggests some characteristics that are present in all systems: organization (order), interaction, interdependence, integration and a central objective.



Organization

Organization implies structure and order. It is the arrangement of components that helps to achieve objectives. In the design of a business system, *for example*, the hierarchical relationships starting with the president on top and leading downward to the blue – collar workers represents the organization structure. Such an arrangement portrays a system–subsystem relationship, defines the authority structure, specifies the formal flow of communication and formalizes the chain of command. Likewise, a computer system is designed around an input device, a central processing unit, an output device and one or more storage units. When linked together they work as a whole system for producing information.

Interaction

Interaction refers to the manner in which each component functions with other components of the system. In an organization, *for example*, purchasing must interact with production, advertising with sales and payroll with personnel.



Interdependence

Interdependence means that parts of the organization or computer system depend on one another. They are coordinated and linked together according to a plan. One subsystem depends on the input of another subsystem for proper functioning: that is, the output of one subsystem is the required input for another subsystem. This interdependence is crucial in systems work.

An integrated information system is designed to serve the needs of authorized users (department heads, managers, etc.) for quick access and retrieval via remote terminals. The interdependence between the personnel subsystem and the organization's users is obvious.

In summary, no subsystem can function in isolation because it is dependent on the data (inputs) it receives from other subsystems to perform its required tasks. Interdependence is further illustrated by the activities and support of systems analysts, programmers, and the operations staff in a computer center. A decision to computerize an application is initiated by the user, analyzed and designed by the analyst, programmed and tested by the programmer, and run by the computer operator. None of these persons can perform property without the required input from others in the computer center subsystem. (See Figure 1.1)

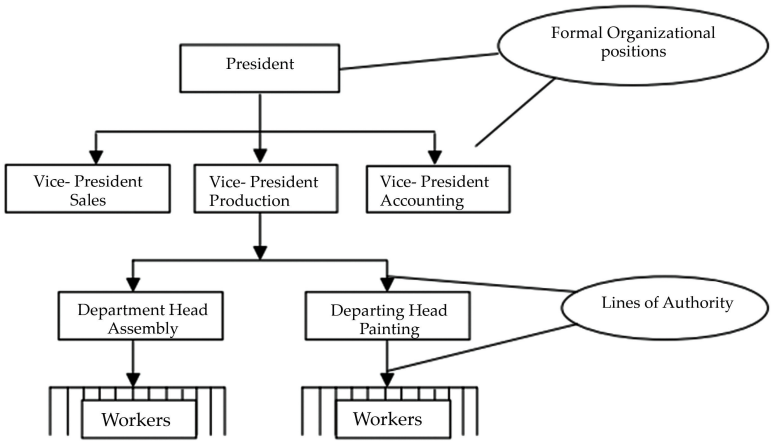


Figure 1.1: Organization structure.

Integration

Integration refers to the holism of systems. Synthesis follows analysis to achieve the central objective of the organization. Integration is concerned with how a system is tied together. It is more than sharing a physical part or location. It means that parts of the system work together within the system even though each part performs a unique function. Successful integration will typically produce a synergistic effect and greater total impact than if each component works separately.

"Viewed abstractly, systems analysis implies rigorous thinking, hopefully quantitative, regarding the gains and the resource-expenditures involved in a particular course of action -- to insure that scarce resources are employed productively rather than wastefully."

—James R. Schlesinger

Central Objective

The last characteristic of a system is its central objective. Objectives may be real or stated. Although a stated objective may be the real objective, it is not uncommon for an organization to state one objective and operates to achieve another. The important point is that users must know the central objective of a computer application early in the analysis for a successful design and conversion. Political as well as organizational considerations often cloud the real objective.

Elements of a System

In most cases, systems analysts operate in a dynamic environment where change is a way of life. The environment may be a business firm, a business application, or a computer system.

To reconstruct a system, the following key elements must be considered:

- Outputs and inputs
- Processor(s)
- Control
- Feedback
- Environment
- Boundaries and interface

Outputs and Inputs

A major objective of a system is to produce an output that has value to its user. Whatever the nature of the output (goods, services, or information), it must be in line with the expectations of the intended user. Inputs are the elements (material, human resources, and information) that enter the system for processing. Output is the outcome of processing. A system feeds on input to produce output in much the same way that a business brings in human, financial, and material resources to produce goods and services. It is important to point out here that determining the output is a first step in specifying the nature, amount, and regularity of the input needed to operate a system. *For example*, in systems analysis, the first concern is to determine the user's requirements of a proposed computer system that is, specification of the output that the computer is expected to provide for meeting user requirements.

Processor(s)

The processor is the element of a system that involves the actual transformation of input into output. It is the operational component of a system. Processors may modify the input totally or partially, depending on the specifications of the output. This means that as the output specifications change so does the processing. In some cases, input is also modified to enable the processor to handle the transformation.

Control

The control element guides the system. It is the decision making subsystem that controls the pattern of activities governing input, processing, and output. In an organizational context, management as a decision making body controls the inflow, handling and outflow of activities that affect the welfare of the business. In a computer system, the operating system and accompanying software influence the behavior of the system. Output specifications determine what and how much input is needed to keep the system in balance.



In systems analysis, knowing the attitudes of the individual who controls the area for which a computer is being considered can make a difference between the success and failure of the installation. Management support is required for securing control and supporting the objective of the proposed change.

Benefits of Systems Analysis

There are many reasons why you might want to analyze a system. These include learning to use systems that somebody else created, for planning new systems, and reducing errors when problem-solving.

Sometimes systems analysis is a necessity. *For example*, if you buy a company and want to hire your own staff, you might find that you are now an owner of a series of systems you know nothing about. It might be impossible to use the systems until they are analyzed, especially if all of the original staff is gone. This can happen on a much smaller scale when you start a new job, or move to a new department. If there is not anyone there

to explain how a system works, you might have to figure it out for yourself.

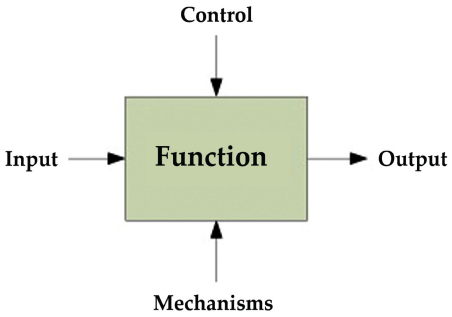


Figure 1.2: Systems analysis involves looking at inputs, outputs, mechanisms and controls

Installing a new system, whether it is a home entertainment center, a factory production line, or way of working in an office, requires proper planning. Without this planning and systems analysis, the change might not work. You need to understand how the current system works, before you install a new one. Otherwise there may be problems, and the project could be a failure.

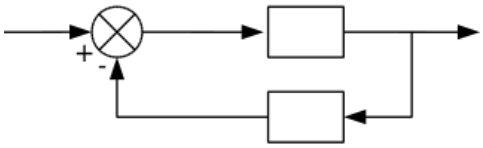


Figure 1.3: Diagrams can help with systems analysis.

Systems analysis also leads to fewer mistakes when problem-solving. A person might be able to fix a problem within five minutes, but then days later discover that this fix caused more problems than it solved. The better you understand the system, the less likely this is to happen.

Examples of Systems Analysis

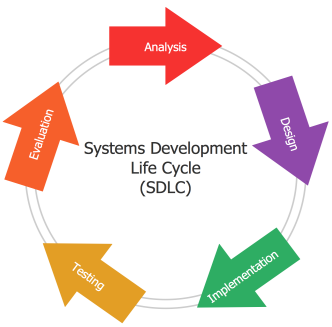
Let's say you are an air conditioning repair person and have been called in to replace a faulty system. Perhaps the air conditioning system is unusually complex or of a type you have never seen

before. You need to analyze the system to understand it before you can figure out what went wrong.

Or perhaps you are a computer programmer and you have been asked to change a program so that it can send notifications to cell phones, instead of just emails. You find yourself faced with pages and pages of code, little of which you have any idea of what it does. The very first step is to make sense of it all: to break it down piece by piece and figure out how it works.

SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

The SDLC (System Development Life Cycle), just as the name implies, is defined as the process (as a whole) of developing system or software to meet certain requirements. It covers many activities; starts from understanding why the system should be built, studying the project feasibility, analyzing problems, choosing the system design and architecture, implementing and testing it, up to delivering the system as product to the user. The SDLC is a process of gradual refinement, meaning that it is done through several development phases.



Each phase continues and refines what's done in the phase. Commonly known development phases in SDLC are:

Planning: It is the process of understanding why the system should be built and defining its requirements. It also includes feasibility study from several different perspectives, technical, economic, and organization feasibility aspects.

Analysis: This phase includes activities such as problems identifying and analysis, and even predicting potential problems

that may arise in the future regarding the system. The deliverables / products of this phase will drive how the system will be built and guide the developers' works.

Design: System analysis leads to design decision, which exactly determines how the system operates in terms of process, data, hardware, network infrastructures, user interface, and other important factors in the system environment.

Implementation: This is probably the most resource-, cost-, and time-consuming phase of all. This is when the system is actually built, tested, and finally installed. It also includes activities such as user training and system maintenance. Some experts like to separate them into different phases deployment and maintenance. However the four phases are the most commonly known and accepted steps. The SDLC tries to achieve high quality system that meets or exceeds the requirements. Many methodologies have been developed and introduced in order to implement SDLC; some of them also try to improve other known methodology.

Although each method follows certain different techniques and steps, they are all must go into the same development phases described above. There are many system development methods known today, but most of them basically are extended from three main methodologies which are Structured Design, RAD (Rapid Application Development), and Object-oriented Analysis and Design.

The Systems Development Life Cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies have been developed to guide the processes involved including the waterfall model (the original SDLC method), rapid application development (RAD), joint application development (JAD), the fountain model and the spiral model. Mostly, several models are combined into some sort of hybrid methodology. Documentation is crucial regardless of the type of model chosen or devised for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely particular plan was followed.

The Figure 1.4 is the classic Waterfall model methodology, which is the first SDLC method and it describes the various phases involved in development. (See Figure 1.4)

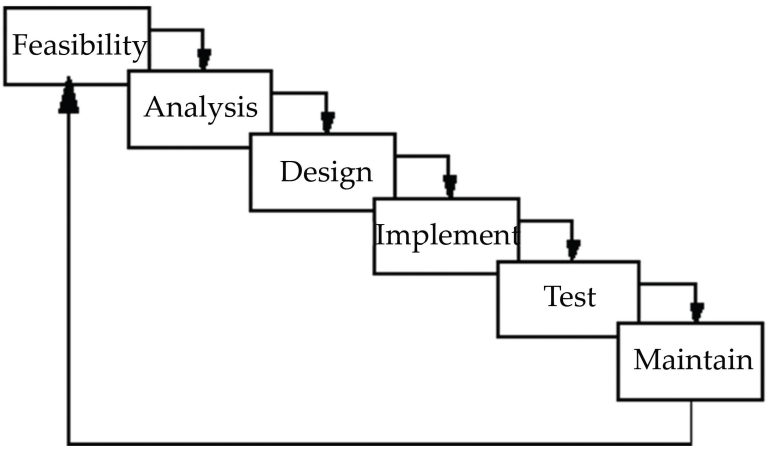


Figure 1.4: Waterfall model.

Problem Definition

Problem Initiation includes define necessary input, output, storage etc. Define what the problem really is state a goal to be achieved.

A problem initiation will describe:

- required input (what data has to be acquired to produce the output?)
- required output (i.e. what information is the system supposed to produce?)

Problem analysis breaks the problem down into its parts and describes them. Note that this step does not care what solution will be used to solve the problem. The analysis lays down the basic requirements that the eventual solution must achieve (a logical design).

"With the subsequent strong support from cybernetics , the concepts of systems thinking and systems theory became integral parts of the established scientific language, and led to numerous new methodologies and applications -- systems engineering, systems analysis, systems dynamics, and so on."

—Fritjof Capra

During problem initiation, one of the first things to do is to define the problem correctly. If we get this wrong (or skip it completely) everything we do afterwards could be a complete waste of time and money.

The most important task in creating a software product is extracting the requirements. Customers typically know what they want, but not what software should do, while incomplete, ambiguous or contradictory requirements are recognized by skilled and experienced software engineers. Frequently demonstrating live code may help reduce the risk that the requirements are incorrect. Problem initiation is the task of precisely describing the software to be written, possibly in a rigorous way. In practice, most successful specifications are written to understand and fine-tune applications that were already well-developed, although safety-critical software systems are often carefully specified prior to application development. Specifications are most important for external interfaces that must remain stable.

Here are some possible definitions of problems:

- The existing system has a poor response time, i.e. it is slow.
- It is unable to handle workload.
- The problem of cost, i.e. the existing system is not economical.
- The problem of accuracy and reliability.
- The requisite information is not produced by the existing system.
- The problem of security.

Similarly, a system analyst should provide a rough estimate of the cost involved for the system development. This is again a very important question that too often is not asked until it is quite late in the system development process.

Feasibility Study

The feasibility study is used to determine if the project should get the go-ahead. If the project is to proceed, the feasibility study will produce a project plan and budget estimates for the future stages of development.

The literal meaning of feasibility is vitality. This study is undertaken to know the likelihood of the system being useful to the organization. Feasibility study, basically, is a high-level capsule version of the entire process, intended to answer a number of questions like what is the problem. Is the problem even worth solving? However, as the name indicates in preliminary investigation, feasibility study should be relatively brief, as the objective at this stage is not only to get an idea of the scope. The findings of this study should be formally presented to the user management. This presentation marks a crucial decision point in the life of the project. If the management approves the project, the feasibility study report represents an excellent model of the system analyst's understanding of the problem and provides a clear sense of direction for the subsequent development of the system.



The feasibility of a proposed system can be assessed in terms of four major categories, as summarized:

1. *Organizational Feasibility*:

The extent to which a proposed information system supports the objective of the organization's strategic plan for information systems determines the organizational feasibility of the system project. The information system must be taken as a sub-set of the whole organization.

2. *Economic Feasibility*:

In this study, costs and returns are evaluated to know whether

returns justify the investment in the system project. The economic questions raised by the analyst during the preliminary investigation are for the purpose of estimating the following:

a. The cost of conducting a full system investigation.
b. The cost of hardware and software for the class of application being considered.
c. The benefits in the form of reduced costs, improved customer service, improved resource utilization or fewer costly errors.



3. *Technical Feasibility*:

Whether reliable hardware and software, capable of meeting the needs of the proposed system can be acquired or developed by the organization in the required time is a major concern of the technical feasibility. In the other words, technical feasibility includes questions like:

- Does the necessary technology exist to do what is suggested and can it be acquired?
- Does the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate responses to inquiries, regardless of the number of locations and users?

- Can the system be expanded?
- Is there any technical surety of accuracy, reliability, ease of access and data security?

4. *Operational Feasibility*:

The willingness and the ability of the management, employees, customers, suppliers, etc., to operate, use and support a proposed system come under operational feasibility. In the other words, the test of operational feasibility asks if the system will work when it is developed and installed. Are there major barriers to implementation? The following questions are asked in operational feasibility.

- Is there sufficient support from the management? From employees? From customers? From suppliers?
- Are current business methods acceptable to the users?
- Have the users been involved in the planning and development of the system project?

Operational feasibility would pass the test if the system is developed as per rules, regulations, laws, organizational culture, union agreements, etc., and above all with the active involvement of the users. Besides these four main categories, the system should also be accessed in terms of legal feasibility and schedule feasibility. Whereas legal feasibility refers to the viability if the system from the legal point of view, i.e. it checks whether the system abides by all laws and regulations of the land, the schedule feasibility evaluates the probability of completing the system in the time allowed for its development, since for the system to be useful, it must be finished well before the actual requirement of its usage.

For the determining feasibility, a project proposal must pass all these tests. Otherwise, it is not a feasible project.

Analysis

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design

(how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.



Primary objective of Analysis phase is to understand the user's needs and develop requirements for Software development.

It involves activities like:

- Gathering Information
- Define Software requirement.
- Prioritize requirements.
- Generate and Evaluate alternatives.
- Review recommendations with management.

Before we start trying to solve a problem it's important to study the existing system before embarking on major changes.

Consider the Analysis phase like a visit to the doctor. We would be pretty worried if we told the doctor we had a headache and the doctor immediately started merrily injecting us with various things before even looking at us or asking we any questions. Such behavior is likely to cause more problems than it solves, so doctors always analyze their patients observing, questioning, and testing - before beginning any treatment.

So also do problem solvers study the system they intend to change, and the organization it's in, before they decide what needs to be done? By thoroughly understanding a system, its operation, its context, its strengths and weaknesses, one can better decide how to start improving it. There's not much good getting heavily into a project if the whole thing is a silly idea to start with. The preliminary investigation is an early test of whether the project should even be started.

Design

The finished design of a solution should contain:

- Data structure (e.g. field names, data types and lengths, file naming, folder structure schemes etc.).
- How the data is to be acquired (what procedures and equipment will be needed?)
- Data input procedures and equipment (e.g. keyboard? barcode reader? ICR/OMR?)
- Interfaces (e.g. what will a data entry screen look like? Will people need to leave the main screen to access functions? How will menus be organized into commands and submenus? What shortcut keys will be used? Will we use a text box, list box, combo box, tick box for a particular item of data entry? What color scheme will be used? What navigation scheme will be used? What icons represent what meaning? Will the layout of the data entry form help users enter data in the required order and the required format?

Control Procedures

What validation rules will be used on what fields to check for data reasonableness, existence or format?) What will different error messages say? How can output be checked for accuracy (e.g. an average can be compared with the data items from which it was calculated). How can procedural errors or problems be detected? (E.g. an order may be cross-checked against the stock database to ensure the ordered item is in stock, or whether it needs to be backordered and the potential customer notified of the delay)

- What workloads and capacities the system must be capable of - e.g. storage capacities, number of transactions per hour, disaster-recovery abilities
- Documentation and training requirements for different types of users
- Validation and storage methods to be used
- How to produce the output (i.e. processing actions)
- Procedures to be followed to use the solution
- Backup requirements and procedures - what needs to be backed up, how often, how backups are stored, what backup scheme will be used?
- How the solution is to be tested to ensure it works properly - what needs to be tested? Functionality, presentation, usability, accessibility, communication of message. How will we test?

A data flow diagram is used to describe the flow of data through a complete data-processing system. Different graphic symbols represent the clerical operations involved and the different input, storage, and output equipment required. Although the flow chart may indicate the specific programs used, no details are given of how the programs process the data. Gantt Charts is detailed timeline of events in a project laid out. In short it is a schedule of Software Development Life Cycle. Structure chart consist of a top-down description of a process and its sub-processes. Data Dictionary - describes (for example) a database's fields, types, lengths, validation rules, formulae.

Development (Coding)

The design must be translated into a machine-readable form. The code generation step performs this task. If the design is performed in a detailed manner, code generation can be accomplished without much complication. Programming tools like compilers, interpreters, debuggers etc. are used to generate the code. Different high level programming languages like C, C++, Pascal, and Java are used for coding. With respect to the type of application, the right programming language is chosen.

Software coding standards are language-specific programming rules that greatly reduce the probability of introducing errors

into applications, regardless of which software development model is being used to create that application.

Languages used:

- Java
- C/C++
- Web-related scripting: HTML, JavaScript
- Database related: MS-SQL, Oracle.

Testing

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/ beta testing).

Software Testing

Software testing is the process of checking software, to verify that it satisfies its requirements and to detect errors. Software testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs.



Testing can never completely establish the correctness of computer software. Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against a specification. A primary purpose for testing is to detect software failures so that defects may be uncovered and corrected.

"It is necessary to learn how to do a systems analysis of your life, to learn about the effects of places, people, jobs. There are millions of things that go into the study of meditation."

—Frederick Lenz

Testing Methods

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

Black Box Testing: Black box testing treats the software as a black-box without any understanding of internal behavior. It aims to test the functionality according to the requirements. Thus, the tester inputs data and only sees the output from the test object. This level of testing usually requires thorough test cases to be provided to the tester who then can simply verify that for a given input, the output value (or behavior), is the same as the expected value specified in the test case. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix etc.

White Box Testing

White box testing, however, is when the tester has access to the internal data structures and algorithms. (And the code that implement these)

Types of white box testing

The following types of white box testing exist:

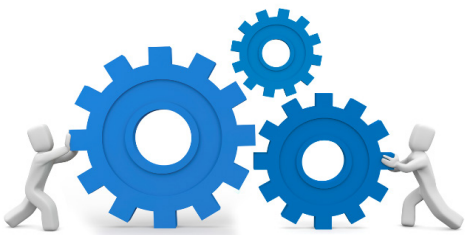
- Code coverage - creating tests to satisfy some criteria of code coverage. *For example*, the test designer can

create tests to cause all statements in the program to be executed at least once.

- Mutation testing methods.
- Fault injection methods.
- Static testing - White box testing includes all static testing.

Implementation

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like C, C++, Pascal, and Java are used for coding. With respect to the type of application, the right programming language is chosen.



In SDLC, implementation refers to post-development process of guiding a client to use the software or hardware that was purchased. This includes Requirements Analysis, Scope Analysis, Customizations, Systems Integrations, User Policies, User Training and Delivery.

Software Implementations involve several professionals like Business Analysts, Technical Analysts, Solution Architect, and Project Managers. Analysts in the implementation phase acts as the intermediate between user and developers. The implementation phase includes:

- Hardware and software installation
- User Training
- Documentation

Primary objectives of Implementation phase are to ensure that:

- Software is installed - The systems are placed and used by actual users.
- The users are all trained - Training is provided to the users of the system usually through workshops or online
- The business is benefiting

Maintenance and Support

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.



Maintenance includes activities like keeping the system up to date with the changes in the organization and ensuring it meets the goals of the organization by:

- Building a help desk to support the system users – having a team available to aid technical difficulties and answer questions
- Implementing changes to the system when necessary.

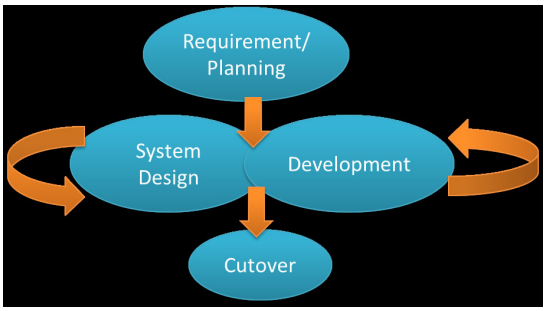
System maintenance involves the monitoring, evaluating and modifying of a system to make desirable or necessary improvements. In other words, maintenance includes enhancements, modifications or any change from the original specifications. Therefore, the information analyst should take change as his/her responsibility so as to keep the functioning at an acceptable level.

Software needs to be maintained not because some of its modules or programs wear out and need to be replaced, but because there are often some residual errors remaining in the system which have to be removed as soon they are discovered. This is an on-going process, until the system stabilizes.

Maintaining and enhancing software to cope with newly discovered problems or new requirements can take far more time than the initial development of the software. Not only may it be necessary to add code that does not fit the original design but just determining how software works at some point after it is completed may require significant effort by a software engineer. About ⅔ of all software engineering work is maintenance, but this statistic can be misleading. A small part of that is fixing bugs. Most maintenance is extending systems to do new things, which in many ways can be considered new work.

RAPID APPLICATION DEVELOPMENT (RAD)

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product. In RAD model the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery.



Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process. RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype. The most important aspect for

this model to be successful is to make sure that the prototypes developed are reusable.

The RAD model is rapid application development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

Diagram of RAD-model

RAD model distributes the analysis, design, build, and test phases into a series of short, iterative development cycles.

Following are the phases of RAD Model:

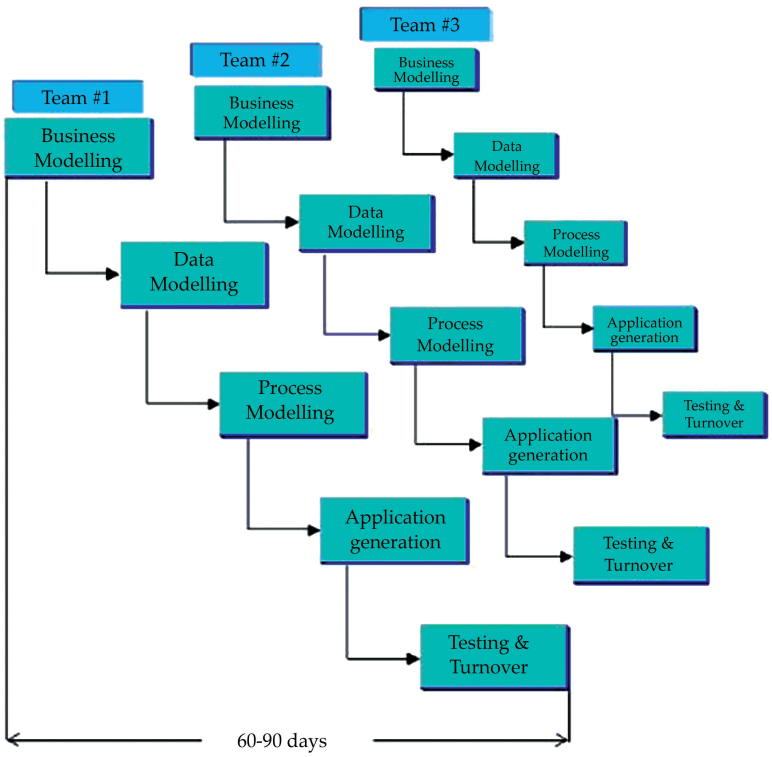


Figure 1.5: RAD model.

The phases in the rapid application development (RAD) model are:

Business Modeling: The information flow is identified between various business functions.

Data Modeling: Information gathered from business modeling is used to define data objects that are needed for the business.

Process Modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

Application Generation: Automated tools are used to convert process models into code and the actual system.

Testing and Turnover: Test new components and all the interfaces.

Advantages of the RAD model

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

When to use RAD model?

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers

for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.

- The RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

"The logical and extralogical exercises you do in meditation are very similar to advanced systems analysis and programming."

—Frederick Lenz

PROTOTYPING

In systems analysis a prototype is a model of the system (or subsystem) under analysis. A system can be anything from the food ordering system at a restaurant to the air traffic control system of a major airport. Prototypes of these systems can take many forms. They can be paper-based or computer-based. They can model the entire system with real data or just a few screens with sample data. Prototyping is the process of developing prototypes. It is a methodology in its own right and a technique and supplemental methodology to other methodologies. You will focus on the ways in which prototyping is used as a technique and a supplemental methodology to the systems development life cycle (SDLC)

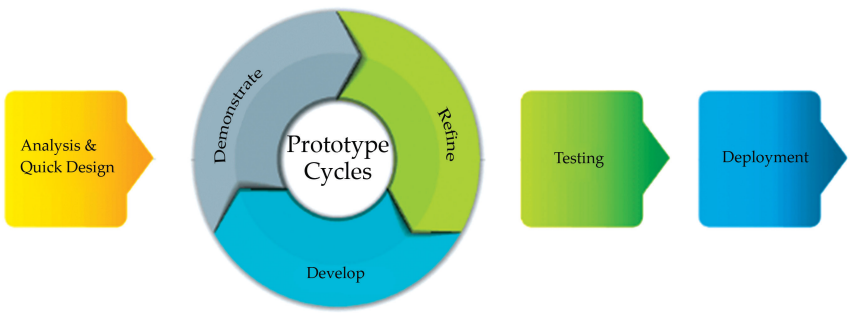


Figure 1.6: systems analysis a prototype is a model

A survey of MIS managers in fortune 1000 firms suggests that there are four prototyping methodologies in use today which supplement the traditional systems development life cycle:

- *Illustrative*: Produces only mockups of reports and screens.
- *Simulated*: Simulates some system functions but does not use real data or a database, model not implemented.
- *Functional*: Performs some actual system functions and uses real data and/or a database, model not implemented.
- *Evolutionary*: Produces model(s) that become part of the final operational system.

Others suggest such categorizations as evolutionary versus throw-away. Evolutionary is similar to mentioned. It produces a model that evolves throughout the development of the system and eventually becomes the final system. Throw-away would encompass the other three methodologies previously mentioned. A throw-away prototype is just what it sounds like. Once its purpose is fulfilled it is thrown away.

Another way that prototypes are classified is by the fidelity of the prototype, or the degree to which the prototype represents the appearance and interaction of the system. A low-fidelity prototype is one that is quickly constructed to depict concepts, design alternatives, and screen layouts. Medium-fidelity prototypes partially simulate the system interaction and functionality. High-fidelity prototypes are fully interactive, simulating much of the functionality in the final product. The chart below suggests techniques that could be used at different fidelity levels.

Following are the steps in prototyping:

There are several steps in the prototypes.

Requirement Gathering and Refinement

The objective and the plans are laid by the customer and the analyst/ developer. They identify the requirements for making the system more feasible.

Develop Working Model

The user and the developer identify the data which will be needed for the system. They also predict the output that the system will yield. This helps the management to estimate the

overall costs for the productions and how profitable the system will be to the organization. In this phase a rough idea of the cost is given by the developer and the tools that will be required for the development of system.

Customer Evaluation

The user actually works on the prototypes in order to evaluate its operations and features. According to the working of the system the user gives positive or negative feedback to the developer.

"There is a change in the view about how condominiums are designed. People used to make prototype apartments and reproduce these boxes hundreds and even thousands of times, and then ask people to conform."

—Santiago Calatrava

Review Prototype

In this phase a feedback from the user's side is obtained from users and the required suggestions if any are made by the users. The developer must understand the needs of the uses properly and likewise the necessary modifications.

The prototyping approach is best suited for the smaller organizations where capital is an issue.

CONCLUSION

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

Systems design is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

CASE STUDY

THE APPLICATION OF THE SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC) IN 21ST CENTURY HEALTH CARE: SOMETHING OLD, SOMETHING NEW?

The systems development life cycle, in its variant forms, remains one of the oldest and yet still widely used methods of software development and acquisition methods in the information technology (IT) arena. While it has evolved over the years in response to ever-changing scenarios and paradigm shifts pertaining to the building or acquiring of software, its central tenants are as applicable today as they ever were. Life-cycle stages have gone through iterations of different names and number of steps, but at the core the SDLC is resilient in its tried-and-true deployment in business, industry, and government. In fact, the SDLC has been called one of the two dominant systems development methodologies today, along with prototyping (Piccoli, 2012). Thus, learning about the SDLC remains important to the students of today as well as tomorrow.

This paper describes the use of the SDLC in a real-world heath care setting involving a principle component of a regional hospital care facility. The paper can be used as a pedagogical tool in a systems analysis and design course, or in an upper-division or graduate course as a case study of the implementation of the SDLC in practice. First, a review of the SDLC is provided, followed by a description of the case study environment. Next, the application of the methodology is described in detail. Following, inferences and observations from the project are presented, along with lessons learned. Finally, the paper concludes with implications for the three areas of research, practice, and pedagogy, as well as suggestions for future research.

Background

The SDLC has been a part of the IT community since the inception of the modern digital computer. A course in Systems Analysis and Design is requisite in most Management Information Systems programs. While such classes offer an overview of many different means of developing or acquiring software (e.g., prototyping, extreme programming, rapid application development (RAD), joint application development (JAD), etc.), at their heart such programs still devote a considerable amount of time to the SDLC, as they should. As this paper will show, following the steps and stages of the methodology is still a valid method of insuring the successful deployment of software. While the SDLC, and systems analysis and design in general, has evolved over the years, at its heart it remains a robust methodology for developing software and systems.

Early treatises of the SDLC promoted the rigorous delineation of necessary steps to follow for any kind of software project. The Waterfall Model (Boehm, 1976) is one of the most well-known forms. In this classic representation, the methodology involves seven sequential steps: 1) System Requirements and Validation; 2) Software Requirements and Validation; 3) Preliminary Design and Validation; 4) Detailed Design and Validation; 5) Code, Debug, Deployment, and Test; 6) Test, Preoperations, Validation Test; and 7) Operations, Maintenance, Revalidation. In the original description of the Boehm-Waterfall software engineering methodology, there is an interactive backstep between each stage. Thus the Boehm-Waterfall is a combination of a sequential methodology with an interactive backstep.

Other early works were patterned after the Waterfall Model, with varying numbers of steps and not-markedly-different names for each stage. For example, Gore and Stubbe (1983) advocated a four-step approach consisting of the study phase, the design phase, the development phase, and the operation phase. Martin and McClure (1988) described it as a multistep process consisting of five basic sequential phases: analysis, design, code, test, and maintain. Another widely used text (Whitten, Bentley, and Ho, 1986) during the 1980s advocated an eight-step method. Beginning with 1) Survey the Situation, it was followed by 2) Study Current System; 3) Determine User Requirements; 4) Evaluate Alternative Solutions; 5) Design New System; 6) Select New Computer Equipment and Software; 7) Construct New System; and 8) Deliver New System.

Almost two decades later, a book by the same set of authors in general also advocated an eight step series of phases, although the names of the stages changed somewhat (albeit not significantly). The methodology proceeded through the steps of Scope definition, Problem analysis, Requirements analysis, Logical design, Decision analysis, Physical design and integration, Construction and testing, and ending with Installation and delivery. It is interesting to note that nearly 20 years later, the naming conventions used in the newer text are almost synonymous with those in the older work. The Whitten and Bentley (2008) text, in its present form, still breaks up the process into eight stages. While there is no consensus in the naming (or number) of stages (e.g., many systems analysis and design textbooks advocate their own nomenclature, McMurtrey (1997) reviewed the various forms of the life cycle in his dissertation work and came up with a generic SDLC involving the phases of Analysis, Design, Coding, Testing, Implementation, and Maintenance.

Even one of the most current and popular systems analysis and design textbooks (Kendall and Kendall, 2011) does not depart from tradition, emphasizing that the SDLC is still primarily comprised of seven phases. Although not immune to criticism, Hoffer, George, and Valacich (2011) believe that the view of systems analysis and design taking place in a cycle continues to be pervasive and true. Thus, while the

SDLC has evolved over the years under the guise of different combinations of naming conventions and numbers of steps or stages, it remains true to form as a well-tested methodology for software development and acquisition. We now turn our attention to how it was utilized in a present-day health care setting.

Case Study Setting

The present investigation regards the selection of a software package by a medium-size regional hospital for use in the Home Health segment of their organization. The hospital (to be referred to in this monograph by a fictitious name, General Hospital) is located in the central portion of a southern state in the USA, within 30 minutes of the state capital. Its constituents reside in the largest SMSA (standard metropolitan statistical area) in the state and consist of both rural, suburban, and city residents. The 149-bed facility is a state-of-the-art institution, as 91% of their 23 quality measures are better than the national average ("Where to Find Care", 2010). Services offered include Emergency Department, Hospice, Intensive Care Unit (ICU), Obstetrics, Open Heart Surgery, and Pediatrics. Additional components of General Hospital consist of an Imaging Center, a Rehabilitation Hospital, Four Primary Care Clinics, a Health and Fitness Center (one of the largest in the nation with more than 70,000 square feet and 7,000 members), a Wound Healing Center, regional Therapy Centers, and Home Care (the focal point of this study).

There are more than 120 physicians on the active medical staff, over 1,400 employees and in excess of 100 volunteers ("General Hospital", 2010). In short, it is representative of many similar patient care facilities around the nation and the world. As such, it provides a rich environment for the investigation of using the SDLC in a 21st century health care institution.

Home Health and Study Overview

Home Health, or Home Care, is the portion of health care that is carried out at the patient's home or residence. It is a participatory arrangement that eliminates the need for constant trips to the hospital for routine procedures. For example, patients take their own blood pressure (or heart rate, glucose level, etc.) using a device hooked up near their bed at home. The results are transmitted to the hospital (or in this case, the Home Health facility near General Hospital) electronically and are immediately processed, inspected, and monitored by attending staff.

In addition, there is a Lifeline feature available to elderly or other homebound individuals. The unit includes a button worn on a necklace or bracelet that the patient can push should they need assistance ("Home Health", 2010). Periodically, clinicians (e.g., nurses, physical therapists, etc.) will visit the patient in their home to monitor their progress and perform routine inspections and maintenance on the technology.

The author was approached by his neighbor, a retired accounting faculty member who is a volunteer at General Hospital. He had been asked by hospital administration to investigate the acquisition, and eventual purchase, of software to facilitate and help coordinate the Home Health care portion of their business. After an initial meeting to offer help and familiarize ourselves with the task at hand, we met with staff (i.e., both management and the end-users) at the Home Health facility to begin our research.

The SDLC in Action

The author, having taught the SAD course many times, recognized from the outset that this particular project would indeed follow the stages of the traditional SDLC. While we would not be responsible for some of the steps (e.g., testing, and training of staff), we would follow many of the others in a lockstep fashion, thus, the task was an adaptation of the SDLC (i.e., a software acquisition project) as opposed to a software development project involving all the stages. For students, it is important to see that they benefit from understanding that the core ideas of the SDLC can be adapted to fit a "buy" (rather than "make") situation. Their knowledge of the SDLC can be applied to a non-development context. The systematic approach is adaptable, which makes the knowledge more valuable. In this project, we used a modified version of the SDLC that corresponds to the form advocated by McMurtrey (1997). Consequently, we proceed in this monograph in the same fashion that the project was presented to us: step by step in line with the SDLC.

Analysis

Problem Definition

The first step in the Systems Development Life Cycle is the Problem Definition component of the Analysis phase. One would be hard-pressed to offer a solution to a problem that was not fully defined. The Home Health portion of General Hospital had been reorganized as a separate, subsidiary unit located near the main hospital in its own standalone facility. Furthermore, the software they were using was at least seven years old and could simply not keep up with all the changes in billing practices and Medicare requirements and payments. The current system was not scalable to the growing needs and transformation within the environment. Thus, in addition to specific desirable criteria of the chosen software (described in the following section), our explicit purpose in helping General was twofold: 1) to modernize their operations with current technology; and 2) to provide the best patient care available to their clients in the Home Health arena.

A precursor to the Analysis stage, often mentioned in textbooks and of great importance in a practical setting, is the Feasibility Study. This preface to the beginning of the Analysis phase is oftentimes broken down into three areas of feasibility:

- Technical (Do we have the necessary resources and infrastructure to support the software if it is acquired?), Economic (Do we have the financial resources to pay for it, including support and maintenance?)
- Operational (Do we have properly trained individuals who can operate and use the software?).

Fortunately, these questions had all been answered in the affirmative before we joined the project. The Director of Information Technology at General Hospital budgeted $250,000 for procurement (thus meeting the criteria for economic feasibility); General's IT infrastructure was more than adequate and up to date with regard to supporting the new software (technical feasibility); and support staff and potential end users were well trained and enthusiastic about adopting the new technology (operational feasibility). Given that the Feasibility Study portion of the SDLC was complete, we endeavored forthwith into the project details.

Requirements Analysis

In the Requirements Analysis portion of the Analysis stage, great care is taken to ensure that the proposed system meets the objectives put forth by management. To that end, we met with the various stakeholders (i.e., the Director of the Home Care facility and potential end-users) to map out the requirements needed from the new system. Copious notes were taken at these meetings, and a conscientious effort to synthesize our recollections was done. Afterwards, the requirements were collated into a spreadsheet for ease of inspection. Several key requirements are described here:

MEDITECH Compatible: This was the first, and one of the most important requirements, at least from a technological viewpoint. MEDITECH (Medical Information Technology, Inc.) has been a leading software vendor in the health care informatics industry for 40 years. It is the flagship product used at General Hospital and is described as the number one health care vendor in the United States with approximately 25% market share. All Meditech platforms are certified EMR/EHR systems. "With an Electronic Health Record, a patient's record follows her electronically. From the physician's office, to the hospital, to her home-based care, and to any other place she receives health services, and she and her doctors can access all of this information and communicate with a smartphone or computer". Because of its strategic importance to General, and its overall large footprint in the entire infrastructure and day-to-day operations, it was imperative that the new software would be Meditech-compatible.

Point of Care Documentation: Electronic medical record (EMR) point-of-care (POC) documentation in patients› rooms is a recent shift in technology use in hospitals. POC documentation reduces inefficiencies, decreases the probability of errors, promotes information transfer, and encourages the caregiver to be at the bedside or, in the case of home care, on the receiving end of the transmission.

OASIS Analyzer: OASIS is a system developed by the Centers for Medicare & Medicaid Services (CMS), formerly an agency of the U.S. Department of Health and Human Services, as part of the required home care assessment for reimbursing health care providers. OASIS combines 20 data elements to measure case-mix across 3 domains–clinical severity, functional status and utilization factors ("Medical Dictionary", 2010). This module allows staff to work more intelligently, allowing them to easily analyze outcomes data in an effort to move toward improved clinical and financial results ("Butte Home Health", 2009). Given its strategic link to Medicare and Medicaid reimbursement, OASIS Analyzer was a "must have" feature of the new software.

Physician Portal: The chosen software package must have an entryway for the attending, resident, or primary caregiver physician to interact with the system in a seamless fashion. Such a gateway will facilitate efficient patient care by enabling the physician to have immediate access to critical patient data and history.

Other "Must Haves" of the New Software: Special billing and accounts receivable modules tailored to Home Health; real-time reports and built-in digital dashboards to provide business intelligence (e.g., OASIS Analyzer); schedule optimization; and last, but certainly not least, the system must be user friendly.

Desirable, But Not Absolutely Necessary Features: Security (advanced, beyond the normal user identification and password type); trial period available (i.e., could General try it out for a limited time before fully committing to the contract?).

Other Items of interest During the Analysis Phase: Several other issues were important in this phase:

- Is the proposed solution a Home Health-only product, or is it part of a larger, perhaps enterprise-wide system?
- Are there other modules available (e.g., financial, clinical, hospice; applications to synchronize the system with a PDA (Personal Digital Assistant) or smart phone)?
- Is there a web demo available to view online; or, even better, is there an opportunity to participate in a live, hands-on demonstration of the software under real or simulated conditions?

We also made note of other observations that might be helpful in selecting final candidates to be considered for site visits. To gain insight into the experience, dependability, and professionalism of the vendors, we also kept track of information such as: experience (i.e., number of years in business); number of clients or customers; revenues; and helpfulness (return e-mails and/or phone calls within a timely manner or at all). Finally, some anecdotal evidence was gathered to help us evaluate each vendor as a potential finalist. For instance, Vendor A had an Implementation/ Installation Team to assist with that stage of the software deployment; they also maintained a Knowledge Base (database) of Use Cases/List Cases describing the most frequently occurring problems or pitfalls. Vendor C sponsored an annual User Conference where users could share experiences with using the product, as well as provide feedback to be incorporated into future releases. To that end, Vendor C also had a user representative on their Product Advisory Board. Vendor E offered a "cloud computing" choice, in that the product was hosted in their data center. (A potential buyer did not have to choose the web-enabled solution.) Vendor E's offering was part of an enterprise solution, and could be synchronized with a PDA or smart phone.

Design

As previously noted, for this particular case study of software selection, the researchers did not have to proceed through each step of the SDLC since the software products already existed. Thus, the Design stage of the SDLC has already been carried out by the vendors. In a similar vein, the coding, testing, and debugging of program modules had too been performed by each vendor candidate. Thus, after painstakingly analyzing all the wares, features, pros and cons, and costs and benefits associated with each product, we were now ready to make a choice: we would whittle our list of five potential vendors down to the two that we felt met our needs and showed the most interest and promise.

The Choice

The principle investigators arranged another meeting with the primary stakeholders of General Hospital's Home Health division. After all, although we had done the research, they were the ones that would be using the system for the foreseeable future. As such, it only made sense that they be heavily involved. This is in line with what is put forth in systems analysis and design textbooks: user involvement is a key component to system success. Having carefully reviewed our research notes, in addition to the various brochures, websites, proposals, communications, and related documents from each of our shortlist of five vendors, together as a group we made our decision. We would invite Vendor B for a site visit and demonstration.

Vendor B was very professional, courteous, prompt, and conscientious during their visit. One thing that greatly supported their case was that their primary business model focused on Home Health software. It was, and still is, their core competency. In contrast, one other vendor (not on our original short list of five) came and made a very polished presentation, in the words of the Director. However, this company was a multi-billion dollar concern, of which Home Health software was only a small part. Thus the choice was made to go with Vendor B.

Ironically, this seller's product was not Meditech compatible, which was one of the most important criteria for selection. However, through the use of a middleware company that had considerable experience in designing interfaces to be used in a Meditech environment, a suitable arrangement was made and a customized solution was developed and put into use. The middleware vendor had done business with General before and, therefore, was familiar with their needs.

Implementation

As is taught in SAD classes, the implementation stage of the SDLC usually follows one of four main forms. These are, according to Valacich, George, and Hoffer (2009): 1) Direct Installation (sometimes also referred to as Direct Cutover, Abrupt, or Cold Turkey method) where the old system is simply removed and replaced with the new software, perhaps over the weekend; 2) Parallel Installation, when the old and new systems are run side-by-side until at some point (the "go live" date) use of the former software is eliminated; 3) Single Location Installation (or the Pilot approach) involves using one site (or several sites if the software rollout is to be nationwide or international involving hundreds of locations) as beta or test installations to identify any bugs or usage problems before committing to the new software on a large scale; and 4) Phased Installation, which is the process of integrating segments of program modules into stages of implementation, ensuring that each block works before the whole software product is implemented in its entirety. The Home Care unit of General Hospital utilized the Parallel Installation method for approximately 60 days before the "go live" date. Clinicians would "double enter" patient records and admissions data into both the old and new systems to ensure that the new database was populated, while at the same time maintaining patient care with the former product until its disposal. The Director of the Home Care facility noted that this process took longer than anticipated but was well worth it in the long run. Once the "go live" date was reached the new system performed quite well, with a minimal amount of disruption.

Training of staff commenced two weeks before the "go live" date. Of the approximately 25 users, half were trained the first week and the rest the next.

Clinicians had to perform a live visit with one of their patients using the new system. Thus they would already have experience with it in a hands-on environment before switching to the new product and committing to it on a full-time basis.

It is again worth noting that the implementation method, Parallel Installation, follows from the SDLC and is what is taught in modern-day SAD courses. Thus, it was satisfying to the researchers that textbook concepts were being utilized in "real world" situations. It also reinforced that teaching the SDLC was in line with current curriculum guidelines and should continue.

Maintenance/Support

Software upgrades (called "code loads" by the vendor) are performed every six weeks. The Director reported that these advancements were not disruptive to everyday operations. Such upgrades are especially important in the health care industry, as changes to Medicare and billing practices are common occurrences. The Director also noted that all end users, including nurses, physical therapists, physicians, and other staff, were very happy with the new system and, collectively, had no major complaints about it. General Hospital expects to use the software for the foreseeable future, with no plans to have to embark on another project of this magnitude for quite some time.

Discussion

Many inferences and observations were gleaned by both the researchers and hospital staff during the course of the investigation. First, we all learned that we must "do our homework"; that is, much research and analysis had to be performed to get up to speed on the project. For instance, while the principle investigators both had doctoral degrees in business administration, and one of them (the author) had taught the systems analysis and design course for over ten years at two different institutions, neither of us had any practical experience in the Home Health arena. Thus, we had to familiarize ourselves with the current environment as well as grasp an understanding of the criteria set forth by the stakeholders (both end-users and management). This was an important lesson learned, because we teach our students (in the SAD class) that they must not only familiarize themselves with the application at hand, but they must also interact with the users. Much research has been conducted in the area of user involvement and its relationship to system success. Therefore it was satisfying, from a pedagogical standpoint, to know that concepts taught in a classroom setting were being utilized in a real-world environment.

It was also very enlightening, from the standpoint of business school professors, to see how the core functional areas of study (e.g., marketing, management, accounting,

etc., not to mention MIS) were also highly integral to the project at hand. During our research on the various vendor companies, we were subjected to a myriad of different marketing campaigns and promotional brochures, which typically touted their wares as the "best" on the market. Key, integral components (such as billing, scheduling, business intelligence, patient care, electronic medical records (EMR), etc.) that are critical success factors in almost any business were promoted and we were made keenly aware of their strategic importance. Again, this was very rewarding from the point of view from business school professors: we were very pleased that our graduates and students are learning all of these concepts (and more) as core competencies in the curriculum. Finally, probably the most positive outcome from the project was that patient care will be improved as a result of this endeavor. Following that, it was enlightening that an adaptation of the SDLC was applied to a healthcare setting and it achieved positive results. This showed that the SDLC, in part or in whole, is alive and well and is an important part of the MIS world in both practice and academia. In addition, key outcomes regarding each were identified and are elaborated upon in the following section.

Implications for Practice, Research And Pedagogy

Implications for Practice

This project, and case study, was an application of pedagogy on a real-world systems analysis project. As such, it has implications for practice. First, it showed that concepts learned in a classroom environment (such as the SDLC in the systems analysis and design course) can be effectively applied in a business (or in our case, a health care) environment. It was very satisfying for us, as business school professors, to see instructional topics successfully employed to solve a real-world problem. For practitioners, such as any organization looking to acquire a software package, we hope that we have shown that if one applies due diligence to their research effort that positive outcomes can be achieved. Our findings might also help practitioners appreciate that tried and true methods, such as the SDLC, are applicable to projects of a similar nature, and not just academic exercises to fulfill curriculum requirements. We find this among the most gratifying implications.

Implications for Research

This project could be used as the beginning of a longitudinal study into the life cycle of the Home Health software product selected. It is customary to note that

maintenance can consume half of the IS budget when it comes to software, especially large-scale systems. It would be interesting to track this project, in real time, to see if that is indeed the case. Furthermore, an often-neglected phase of the SDLC is the stage at the very end: disposal of the system. By following the present study to the end, it would be enlightening (from all three viewpoints of research, practice, and pedagogy) to see what happens at the end of the software's useful life. Additional future research might investigate the utilization of the SDLC in different contexts, or even other settings with the healthcare arena.

Implications for Pedagogy

Insights for the SAD Course

After learning so much about real-world software acquisition throughout this voluntary consulting project, the author has utilized it in classroom settings. First, the obvious connection with the SAD course was made. To that end, in addition to another semester-long project they work on in a group setting, the students pick an application domain (such as a veterinary clinic, a dentist's office, a movie rental store, etc.) and perform a research effort not unlike the one described in this monograph. Afterwards, a presentation is made to the class whereby three to five candidate vendors are shown, along with the associated criteria used, and then one is chosen. Reasons are given for the selection and additional questions are asked, if necessary. This exercise gives the students a real-world look at application software through the lens of the SDLC.

While some SAD professors are able to engage local businesses to provide more of a "real-world" application by allowing students to literally develop a system, such an endeavor was not possible at the time of this study. The benefits of such an approach are, or course, that it provides students "real world" experience and applying concepts learned in school to practical uses. The drawback is that it requires a substantial commitment from the business and oftentimes the proprietors pull back from the project if they get too busy with other things. Thus, the decision was made to allow students to pick an application domain, under the assumption that they had been contracted by the owners to acquire a system for them.

Such an exercise enables students to engage in what Houghton and Ruth (2010) call "deep learning". They note that such an approach is much more appropriate when the learning material presented involves going beyond simple facts and into what lies below the surface. Indeed, this particular exercise for the SAD students was not rote memorization of facts at a surface level; it forced them to perform critical thinking and analysis at a much greater depth of understanding. Although the students were

not able to complete a "real world" project to the extent that other educators have reported (e.g., Grant, Malloy, Murphy, Foreman, and Robinson (2010), the experience did allow students to tackle a contemporary project and simulate the solving of it with real-world solutions. This gave them a much greater appreciation for the task of procuring software than just reading about it in textbooks. The educational benefits of using real-world projects are well established both in the United States (Grant et al., 2010) and internationally (Magboo and Magboo, 2003).

From an IS curriculum standpoint, this form of exercise by SAD students helps bridge the well-known gap between theory and practice (Andriole, 2006). As was shown in this monograph, the SDLC is a theory that has widespread application in practice. The project performed by students in the SAD class reinforces what Parker, LeRouge, and Trimmer (2005) described in their paper on alternative instructional strategies in an IS curriculum.

Insights for IS Courses, SAD and non-SAD

Other insights gained, by the SAD students as well as the core MIS course, have to do with what the author teaches during the requisite chapter on software. In class, I present this topic as "the software dilemma". This description is tantamount to the recognition that when acquiring software, businesses must make one of three choices (in general). The options are "make" versus "buy" versus "outsource" when it comes to acquiring software. (There is also a hybrid approach that involves customizing purchased software.)

Briefly explained, the "make" option presupposes that the organization has an IT staff that can do their own, custom, programming. The "buy" alternative relates to what was described in this paper, in that General Hospital did not have the resources to devote to developing software for their Home Health segment, and as such enlisted the researchers to assist in that endeavor. The "outsource" choice alludes to several different options available, under this umbrella, on the modern-day IT landscape. The decision to outsource could range from an application service provider (ASP) delivering the solution over the internet (or the "cloud") to complete transfer of the IT operation to a hosting provider or even a server co-location vendor.

Thus, a project like this one could be used in the core MIS course to further illustrate problems and potential pitfalls faced by businesses, small and large, when it comes to software acquisition. Instructors could use the features of this case study to focus on whatever portion of it they thought best: project management, budgeting, personnel requirements, marketing, etc. It could even be used in a marketing class to investigate the ways in which vendors, offering similar solutions to standard problems, differentiate themselves through various marketing channels and strategies.

Furthermore, the case study is ripe for discussion pertaining to a plethora of business school topics, from economics and accounting to customer relationship management. The case is especially rich fodder for the MIS curriculum: not only systems analysis and design, but programming and database classes can find useful, practical, real-world issues surrounding this case that can be used as "teaching tools" to the students.

Finally, a case study like this one could even be used in an operations management, or project management, setting. The discovery of issues, such as those raised in this paper, could be fruitful research for both undergraduate and graduate students alike. A team project, along with a group presentation as the finale, would also give students much-needed experience in public speaking and would help prepare them for the boardrooms of tomorrow.

Conclusion

Two business school professors, one an MIS scholar and the other retired from the accounting faculty, were called upon by a local hospital to assist with the procurement of software for the Home Health area. These academics were up to the challenge, and pleasantly assisted the hospital in their quest. While both researchers hold terminal degrees, each learned quite a bit from the application of principles taught in the classroom (e.g., the SDLC) to the complexities surrounding real-world utilization of them. Great insights were gained, in a variety of areas, and have since been shown as relevant to future practitioners (i.e., students) in the business world. It is hoped that others, in both academe and commerce, will benefit from the results and salient observations from this study.

Questions

1. How to execute the SDLC in action?
2. Define the SDLC implications for practice, research and pedagogy.

KEY TERMS

Development cycle	System analysis	Delivery system
Education system	Production system	Central processing unit
Computer operator	Project feasibility	Rapid application development

DISCUSSION QUESTIONS

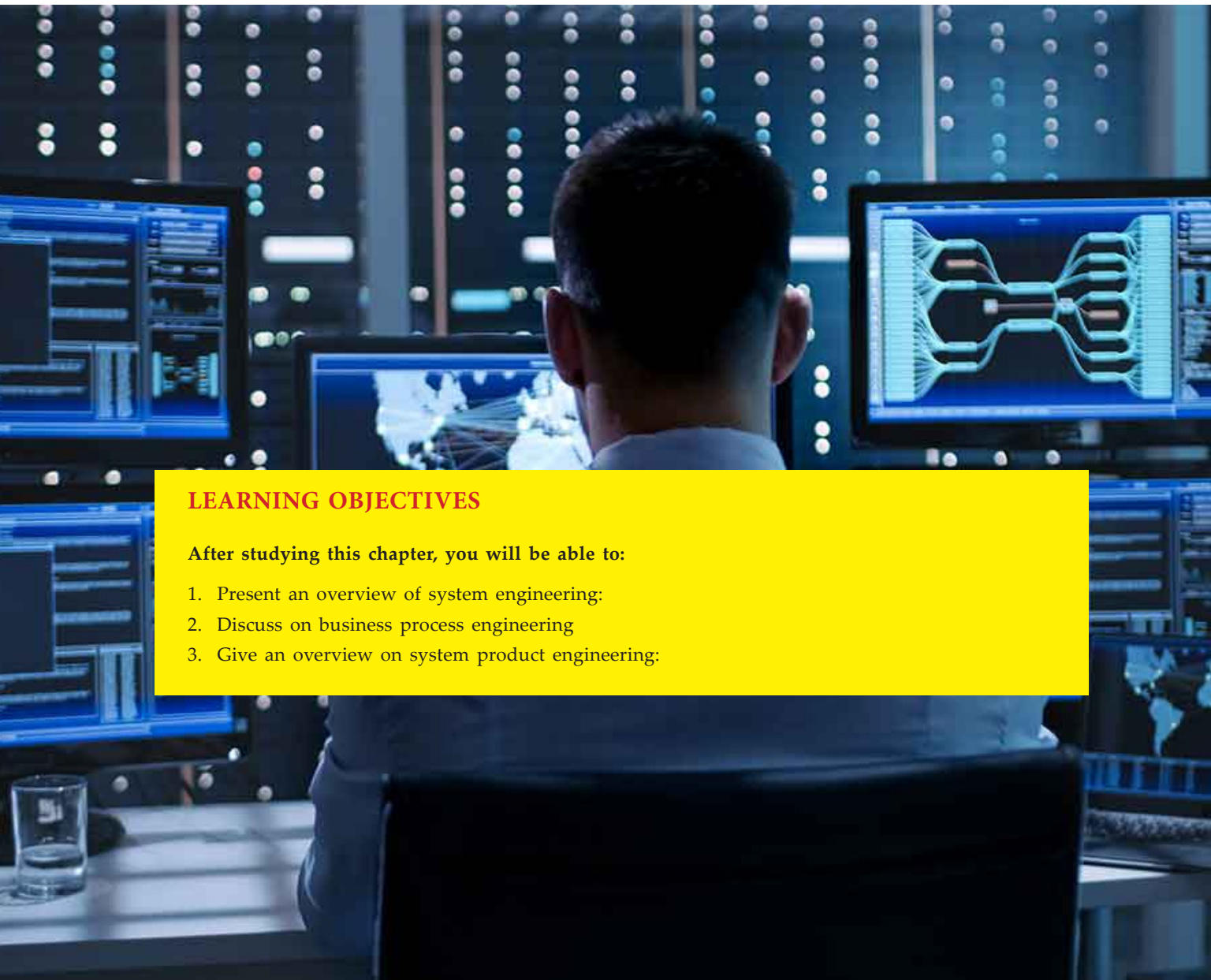
1. What are the characteristics of a system? Explain.
2. Discuss on the elements of a system.
3. What do you understand by feasibility study?
4. Explain the advantages and disadvantage of the RAD model.
5. How can we evaluate feasibility of proposed system? Explain.

REFERENCES

1. Ahern DM, Clouse A, Turner R (2003) CMMI distilled: a practical introduction to integrated process improvement. The SEI Series in Software Engineering, Addison-Wesley.
2. Dobing B, Parsons J (2000) Understanding the Role of Use Cases in UML: A Review and Research Agenda. Journal of Database Management 11: 28-36.
3. Hay D (2012) UML and Data Modeling: A Reconciliation. Technics Publications.
4. Hungerford BC, Hevner AR, Collins RW (2004) Reviewing Software Diagrams: A Cognitive Study. IEEE Transactions on Software Engineering 30: 82-96.
5. Langer AM (2007) Analysis and Design of Information Systems. Software engineering.
6. Parsons J, Saunders C (2004) Cognitive Heuristics in Software Engineering applying and Extending Anchoring and Adjustment to Artifact Reuse. IEEE Transactions on Software Engineering 30: 873-888.
7. Sydow A, Tzafestas SG (2000) Systems Analysis Modelling Simulation - Special issue on systems modelling, analysis and design. Systems Analysis Modelling Simulation, USA.
8. Tilley S, Parveen T (2012) Software Testing in the Cloud: Perspectives on an Emerging Discipline. Idea Group,U.S.A.
9. Wang S, Wang H (2012) Information Systems Analysis and Design. Universal Publishers, USA.
10. Wasson CS (2005) System Analysis, Design, and Development: Concepts, Principles, and Practices. Wiley Series in Systems Engineering and Management.

CHAPTER 2

SYSTEM ENGINEERING



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Present an overview of system engineering:
2. Discuss on business process engineering
3. Give an overview on system product engineering:

INTRODUCTION

Systems engineering enables building, analyzing and managing a system—be it electrical, mechanical, chemical, biological or one involving business processes and logistics. Traditionally, engineering disciplines are defined through physical attributes: electrical engineers work with circuits and transistors, aeronautical engineers with airplanes and rockets. Systems engineering transcends the physical nature of what is designed or managed—if "it" consists of multiple interacting components performing a function that cannot be achieved by any one component alone, then "it" is a system, and systems engineers can work to understand and improve it.

Systems Engineering focuses on:

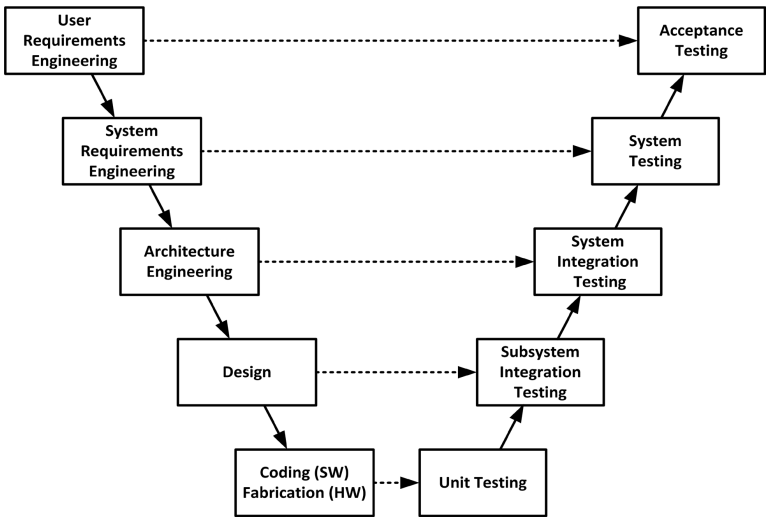
- Establishing, balancing and integrating stakeholders' goals, purpose and success criteria, and defining actual or anticipated customer needs, operational concept and required functionality, starting early in the development cycle;
- Establishing an appropriate lifecycle model, process approach and governance structures, considering the levels of complexity, uncertainty, change, and variety;
- Generating and evaluating alternative solution concepts and architectures;
- Base lining and modelling requirements and selected solution architecture for each phase of the endeavor;
- Performing design synthesis and system verification and validation;
- While considering both the problem and solution domains, taking into account necessary enabling systems and services, identifying the role that the parts and the relationships between the parts play with respect to the overall behavior and performance of the system, and determining how to balance all of these factors to achieve a satisfactory outcome.

Systems Engineering provides facilitation, guidance and leadership to integrate the relevant disciplines and specialty groups into a cohesive effort, forming an appropriately structured development process that proceeds from concept to production, operation, evolution and eventual disposal.

Systems Engineering considers both the business and the technical needs of customers with the goal of providing a quality solution that meets the needs of users and other stakeholders, is fit for the intended purpose in real-world operation, and avoids or minimizes adverse unintended consequences.

SYSTEM ENGINEERING: AN OVERVIEW

A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results.



The System Engineering Hierarchy

Regardless of its domain of focus, system engineering encompasses a collection of top-down and bottom up methods to navigate the hierarchy illustrated in Figure 2.1. The system engineering process usually begins with a "world view" That is the entire business of product domain is examined to ensure that the proper business or technology context established. The world view is refined to focus more fully on specific domain of interest. Within a specific domain, the need for targeted system elements (for example. data, software, hardware, and people) is analyzed. Finally the analysis design and construction of a targeted system element

is initiated. At the top of the hierarchy a very broad context is established and at the bottom, detailed technical activities performed by the relevant engineering discipline (for example. hardware or software engineering) are conducted.

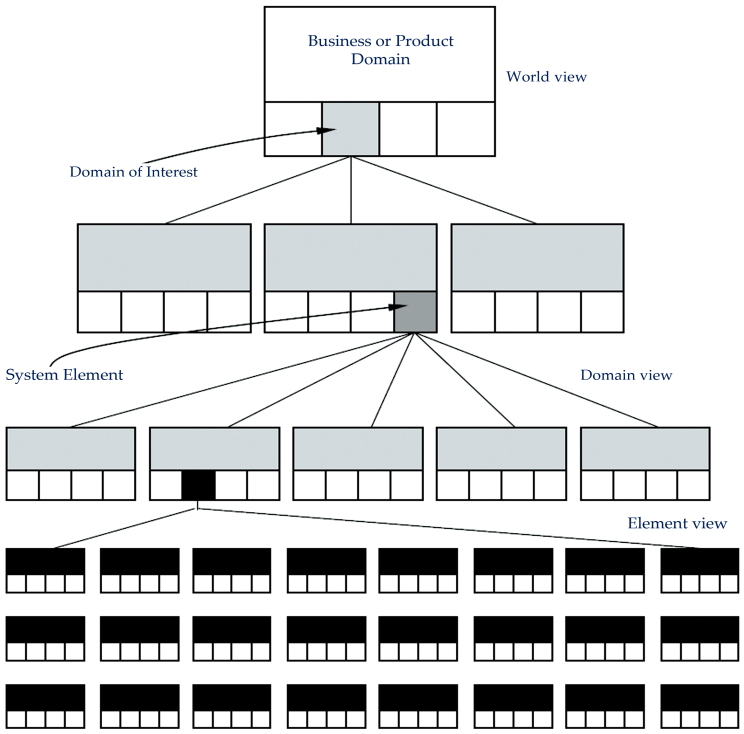


Figure 2.1: The system engineering hierarchy.

Stated in a slightly more formal manner the world view (WV) is composed of a set of systems in its own right.

$WV = \{D1, D2, D3 \ldots D_n\}$

Each domain is composed of specific elements (E_j) each of which serves some role in accomplishing the objectives and goals of the domain:

$D_i = \{E1, E2, E3 \ldots E_m\}$

Finally, each element is implemented by specifying the technical components (C_k) that achieve the necessary function for an element.

$E_j = \{C1, C2, C3 \ldots C_k\}$

In the software context, a component could be a computer program a reusable program component, a module, a class or object or even a programming language statement.

It is important to note that the system engineer narrows the focus of work as he or she moves downward in the hierarchy. However the world view portrays a clear definition of overall functionality that will enable the engineer to understand the domain and ultimately the system or product in the proper context.

System Modeling

System engineering is a modeling process. Whether the focus is on the world view or the detailed view the engineer creates models.

- Define the processes that serve the needs of the view under consideration
- Represent the behavior of the processes and the assumptions on which the behavior is based
- Explicitly define both exogenous and endogenous input to the model
- Represent all linkages (including output) that will enable the engineer to better understand the view.

To construct a system model, the engineer should consider a number or restraining factors:

1. Assumption that reduce the number of possible permutations and variations thus enabling a model to reflect the problem in a reasonable manner. As an example consider a three-dimensional rendering product used by the entertainment industry to create realistic animation. One domain of the product enables the representation of 3D human forms. Input to this domain encompasses the ability to input movement from a live human "actor" from video or by the creation of graphical models. The system engineering makes certain assumptions about the range of allowable human movement so that the range of inputs and processing can be limited.
2. Simplifications that enable the model to be created in timely manner. To illustrate, consider an office products

company that sells and services a broad range of copiers fax machines and related equipment. The system engineer is modeling the needs of the service organization and is working to understand the flow of information that spawns a service order. Although a service order can be derived from much origin, the engineer categorizes only two sources: internal demand or external request. This enables a simplified partitioning of input that is required to generate the work order.

3. Limitation that helps to bound the system. For example an aircraft avionics system is being modeled for a next generation aircraft. Since the aircraft will be a two-engine design, all monitoring domains for propulsion will be modeled to accommodate a maximum of two engines and associated redundant systems.
4. Constraints that will guide the manner in which the model is created and the approach taken when the model is implemented. For example, the technology infrastructure for the three-dimensional rendering system is a single Power PC-based processor.
5. Preferences that indicate the preferred architecture for all data, functions, and technology. The preferred solution sometimes comes into conflict with other restraining factors. Yet, customer satisfaction is often predicated on the degree to which the preferred approach is realized.

The resultant system model may call for a completed automated solution, a semi-automated solution or a manual approach. In fact it is often possible to characterize models of each type that serve as alternative solutions to the problem at hand. In essence the system engineer simply modifies the relative influence of different system elements to derive models of each type.

Information Engineering

The goal of information engineering (IE) is to define architecture that will enable a business to use information effectively. In addition information engineering works to create an overall plan for implementing that architecture.



Three different architecture must be analyzed and designed within the context of business objectives and goals:

- Data architecture
- Application architecture
- Technology infrastructure

The data architecture provides a framework for the information needs of a business or business function.

The individual building blocks of the architecture are the data objects that are used by the business. The data objects flow between business functions are organized within a database and are transformed to provide information that serves the needs of the business.

The application architecture encompasses those elements of a system that transform objects within the data architecture for some business purpose. We normally consider the application architecture to be the system of programs that performs this transformation. However in a broader context, the application architecture might incorporate the role of people and business procedures that have not been automated.

The technology infrastructure provides the foundation for the data and application architectures. The infrastructure encompasses the hardware and software that are used to support application and data. The includes computers and computer networks, telecommunication links, storage technologies and the architecture that has been designed to implement these technologies.

The model the system architectures a hierarchy of information engineering activities is defined. As shown in the Figure 2.2 the world view is achieved through information strategy planning

(ISP) ISP views the entire business as an entity and isolates the domains of the business that are important to the overall enterprise. ISP defines the data objects that are visible at the enterprise level, their relationships and how they flow between the business domains.

The domain view is addressed with an IE activity called business area analysis (BAA). A hare describes BAA in the following manner:

BAA is concerned with identifying in detail data and function requirements of selected business area identified during ISP and ascertaining their interactions. It is only concerned with specifying what is required in a business area.

As the information engineer begins BAA the focus narrows to a specific business domain. BAA views the business area as an entity and isolates the business functions and procedures that enable the business area to meet its objectives and goals. BAA like ISP defines data objects their relationship and how data flow. But at this level these characteristic are all bounded by the business area being analyzed. The outcome of NAA is to isolate areas of opportunity in which information systems may support the business area.

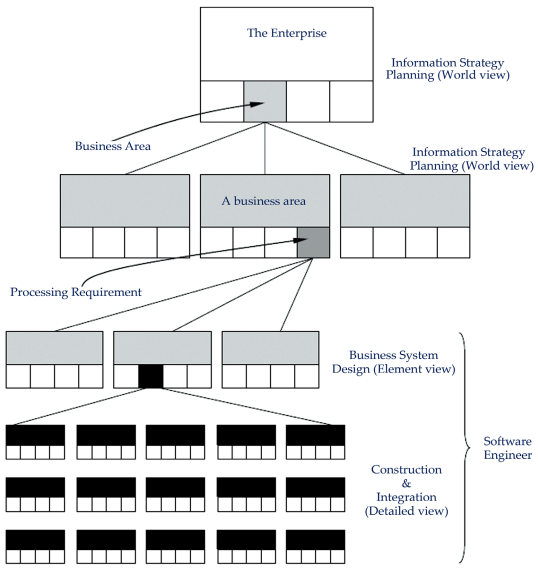


Figure 2.2: The Information Engineering Hierarchy.

Once an information system has been isolated for further development, IE makes a transition into software engineering. By invoking a business system design step the basic requirements of a specific information system are modeled and these requirements are translated into data architecture, applications architecture and technology infrastructure.

The final IE step construction and integration (C&I) focuses on implementation detail. The architecture and infrastructure are implemented by constructing an appropriate database and internal data structures by building application using program components and by selecting appropriate elements of a technology infrastructure to support the design created during BSD. Each of these system components must then be integrated to form a complete information system into the business area context, performing all user training and logistics support to achieve a smooth transition.

BUSINESS PROCESS ENGINEERING

Business process engineering is the science of helping a business run more efficiently. A business process engineer analyzes the goals and mission of a business and examines the processes used to fulfill them. The engineer then creates a cohesive business process model to help the company reach its goals as efficiently as possible.

A business process is a series of tasks a business uses to achieve a defined goal. Business process engineering follows the belief that in order to make an improvement, tasks in the series cannot be isolated. They must be viewed as part of a whole and evaluated on how and if they contribute to the overall mission.

To begin business process engineering for a company, an engineer first assesses the mission, goals, and customer needs to be met. He or she makes sure that all of these are clearly defined and that the company is operating under realistic assumptions. The process will not be effective if the process is based on faulty principles.



Once the mission is clearly defined, the engineer examines how and how well the business is achieving that mission. He or she looks at the business processes currently in place and rates the effectiveness of the individual tasks and the process as a whole. Business process engineers will redesign the whole process to make sure every step works toward the goal as efficiently as possible. He or she may recommend that some steps are removed altogether if they are not helping the company achieve its goals.

Instead of trying to automate their processes, companies should focus on whether the process is necessary at all and how all the individual processes fit together.

The idea spread quickly. As information systems, networks, and technology developed, more business were re-engineered to incorporate these technologies. Business process engineering has improved the customer service, product quality, and speed of many businesses.

This science does have some problems. Some people dislike it because it can lead to mass layoffs, as the process is streamlined. Engineering cannot solve all problems because it always assumes that the process is to blame for failure while ignoring all other influences on business success.

Information Strategy Planning

The first information engineering steps is information strategy planning (ISP). The overall objectives of ISP are (1) to define strategic business objectives and goals, (2) to isolate the critical success factors that will enable the business to achieve these goals and objectives, (3) to analyze the impact of technology and automation on the goals and objectives and (4) to analyze

existing information to determine its role in achieving goals and objectives. ISP also creates a business-level data model that defines key data objects and their relationship to one another and to various business areas.

The terms 'objectives' and 'goals' take on a specific meaning in ISP. An objective is a general statement of direction.

Objectives tend to be strategic. Goals are tactical

Critical success factors can be tied to an objective or to individual goals. A CSF must be present if the objective or goal is to be achieved. Therefore management planning must accommodate it. For example,

CSFs for the manufacturing objective noted might be:

- Total quality management strategy for the manufacturing organization
- Worker training and motivation
- Higher-reliability machines
- Higher-quality parts
- A 'sales plan' to convince suppliers to reduce prices
- Availability of engineering staff

Technology impact analysis examines objectives and goals and provides an indication of those technologies that will have a direct or indirect impact on achieving them successfully. The information engineer addresses the following questions: How critical is the technology to the achievement of a business objective? Is the technology available today? How will the technology change the way business is conducted? What are the direct and indirect costs? How should the business adapt or extend objectives and goals to accommodate the technology?

Because every business area makes some use of information technologies, ISP must also identify what currently exists and how it is currently used to achieve objectives and goals. Business process reengineering is an activity that examines existing systems with the intent of reengineering them to better meet business needs.

Enterprise Modeling

Enterprise modeling creates a three-dimensional view of a business. The first dimension addresses the organizational structure and the functions that are performed within the business area defined by the organizational structure. The second dimension decomposes business function to isolate the processes that make function happen. Finally, the third dimension relates objectives, goals and CSFs to the organization and its functions. In addition, enterprise modeling creates a business-levels data model that defines data objects and their relationships to other elements of the enterprise model.

The business organization is defined in a classical business unit hierarchy. Each box in the org chart represents a business area of the company. Like all hierarchies, it is generally possible to refine the boxes within the org chart until small working groups or even individuals are noted. However, for ISP purpose, business areas are all that is required.

Business functions are identified and the processes that are required to implement the business functions are defined. Each of the business functions is then related to the business area that has responsibility for it. In general a business function is some ongoing activity that must be accomplished to support the overall business. It can usually be as a noun phrase. A business process is a transform that accepts specific inputs and produces specific outputs. It can generally be as a verb phrases.

Business Function

- Product development and engineering
- Marketing
- Demographic research
- Market analysis
- Forecasting
- Product specification
- Product engineering
- Technology research
- New product development
- System analysis

- Component engineering
- Hardware engineering
- Software engineering
- Human engineering
- Product V & V
- Quality assurance

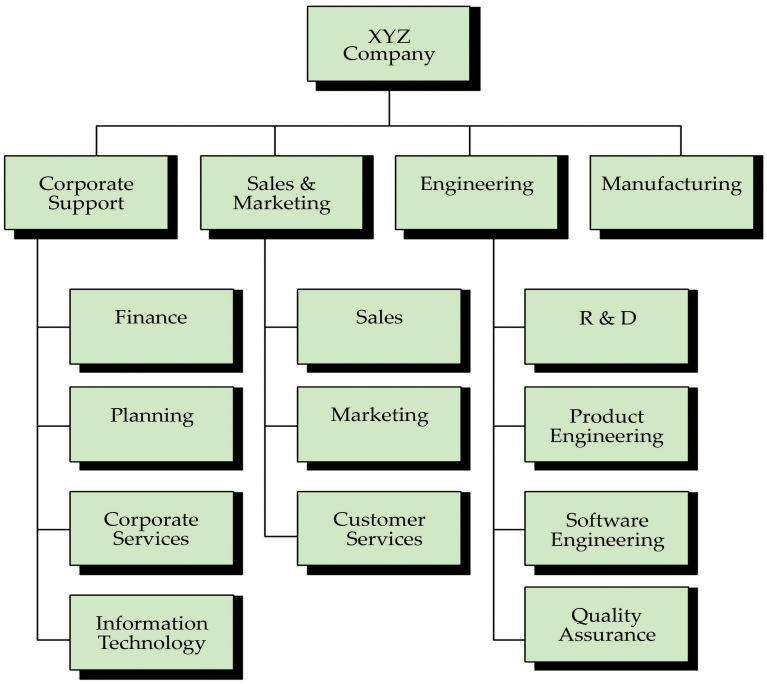


Figure 2.3: Deriving an organizational chart and coupling business areas to function.

To illustrate how a business functions is refined into a set of supporting processes, consider the market analysis function shown figure 2.3. A process refinement follows:

Market Analysis:

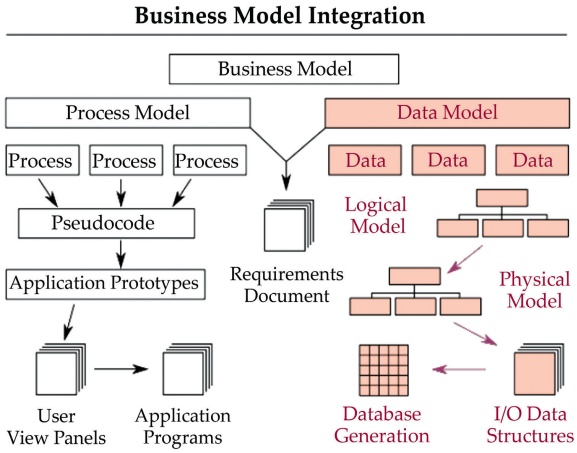
- Collect data on all sales inquiries
- Collect data on all sales
- Analyze data on inquiries and sales
- Develop buyer profile

- Compare profile to demographic research
- Study industry buying trends
- Establish focus groups to determine best sales message
- Design rough sales materials
- Test sales materials and refine
- Finalize sales approach

Each of the bulleted process steps could be further refined to provide a detailed road map for accomplishing the business function. During ISP, the information engineer does not become concerned with areas of automation opportunity. The intent is simply to understand and model the business.

Business-Level Data Modeling

Business-level data modeling is an enterprise modeling activity that focuses on the data objects (also called entities) that are required to achieve the business functions. At the business level, typical data objects include producers and consumers of information (for example a customer), things (for example a report), occurrences of events (for example a sales conference), organization units for example., sales and marketing), places (for example manufacturing cell), or information structures (for example an employee file). A data object contains a set of attributes that define some aspect, quality, characteristic or descriptor of the data that is being.



For example during enterprise modeling an information engineer might define the data object customer.

To more fully describe customer, the following attributes are defined:

Object: Customer

Attributes:

Name

Company name

Job classification and purchase authority

Business addresses and contact information

Product interest(s)

Past purchase(s)

Date of last contact

Status of contact

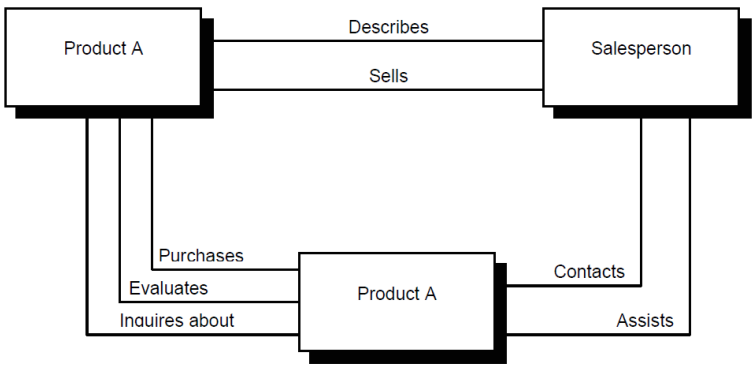


Figure 2.4: Depicting relationship among business level data objects.

Once a set of data objects is defined, their relationships are identified. A relationship indicates how objects are connected to one another. As an example, consider the objects customer, product A and salesperson. An information engineer creates a diagram (Figure 2.4) to depict these relationships. Referring to the figure relationships imply a connection between data objects. In general, relationships can be read in either direction; hence

a customer purchase products A and product A is purchased by a customer. In reality additional information is provided as part of the data model. The culmination of the ISP activity is the creation of a series of cross-reference matrices that establish the overall relationships between the organization (and its business areas), business objectives and goals, business functions, and data objects. Examples of such matrices are shown in figure 2.5.

Business Area Analysis

Information engineering, Martin describes business area analysis in the following manner: Business areas analysis (BAA) establishes a detailed framework for building an information-based enterprise. It takes one business area at a time and analyzes it in detail. It uses diagrams and matrices to model and record the data and activities in the enterprise and to give a clear understanding of the elaborate and subtle ways in which the information aspects of the enterprise interrelate.

During BAA, our focus shifts from the world view to the domain view. To model 'the elaborate and subtle ways in which the information aspects of the enterprise interrelate' the information engineer must depict how data objects (during ISP and refined during BAA) are used and transformed within each business area and how the business functions and processes within each business area transform these data objects. In essence, both exogenous and endogenous data are analyzed and modeled for each business area.

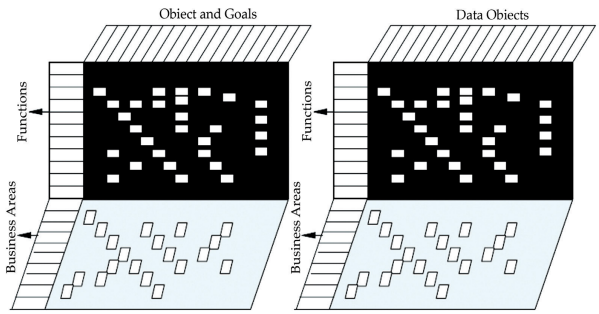


Figure 2.5: Typical cross-reference matrices used during ISP.

To accomplish this work, BAA makes use a number of different models;

- Data models (now refined to the business area level)
- Process flow models
- Process decomposition diagrams
- A variety of cross-reference matrices

The data objects defined during ISP are refined for use within each business area. For example the data object customer in the preceding is used by the sales department. After evaluation of the needs of the sales department (an analysis of the sales domain), the original definition of customer is further refined to meet the needs of sales:

Object: Customer

Attributes:

Name

Company name → Object Company

Job classification and purchase authority

Business addresses and contact information

Product interest(s)

Past purchase(s)

Date of last contact – record of contacts

Status of contact – status of last contact

→Next contact date

→Recommended nature of contact

The attribute company name has been modified to point to another object called Company. This object contains not only the company name but additional information about the size of the company, its purchasing requirements, and the name of other contacts and so on. This information will be useful in the sales domain. Other attributes have been modified and added as noted.

Process Modeling

The work performed within a business area encompasses a set of business functions that are further refined into business processes. To illustrate consider a simplified version of the sales function. The processes that occur to accomplish sales are:

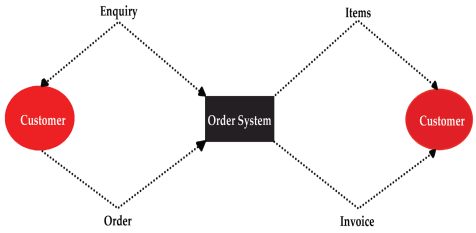
Sales Function

- Establish customer contact
- Provide product literature and related information
- Address questions and concerns
- Provide evaluation product
- Accept sales order
- Check availability of configuration ordered
- Prepare delivery order
- Confirm configuration, pricing , ship data with customer
- Transmit delivery order to fulfillment department
- Follow up with customer.

A process flow diagram (Figure 2.6) can be developed for this sequence of processing. It should be noted that each business function relevant to the business area can be refined in a similar manner.

Information Flow Modeling

The process flow model is integrated with the data model to provide an indication of how information flows through a business area. Input and output data objects are shown for each process, providing an indication of how the process transforms information to accomplish a business function.



Once a complete set of process flow models has been created the information engineer (along with others) examines how the existing process can be reengineered and where existing information systems or applications might be modified or replaced by more efficient information technologies. The revised process model is used as a basis for the specification of new or revised software to support the business function.

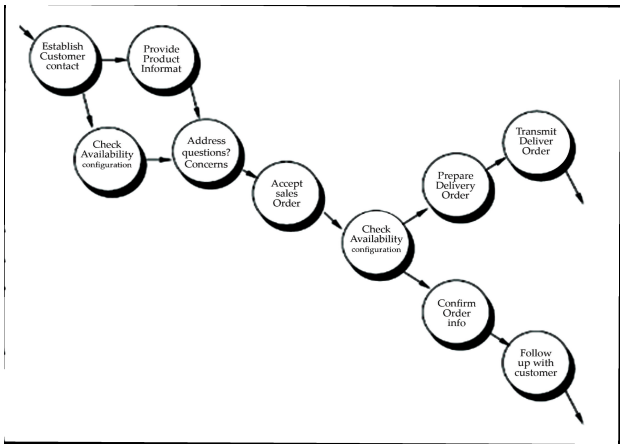


Figure 2.6: Process flow model for the sales functions.

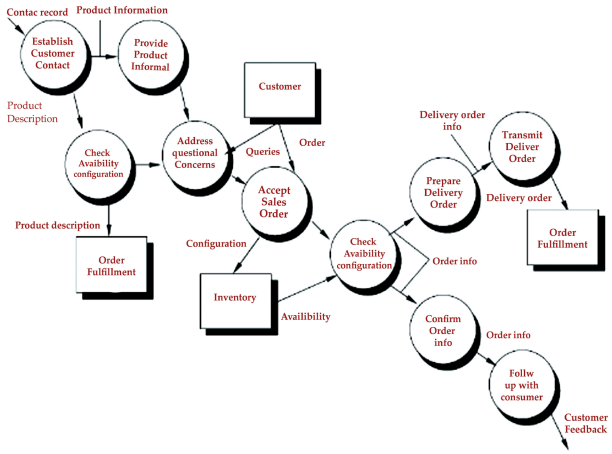


Figure 2.7: Adding information flow the process flow model for the sales functions.

The domain view established during BAA serves as the basis for business system design and construction and integration –IE seeps that are actually part of the software engineering process.

SYSTEM PRODUCT ENGINEERING: AN OVERVIEW

A product engineer is a person that can design, develop, and manage new product ideas for corporations or individual inventors. Being an engineer is not always required, but the person must be familiar with all phases of the Product Development Cycle, and keep up with the latest technologies. Also, the designer has to combine technical knowledge, human factors, and creativity in order to make a product successful in the marketplace.



Product Engineering

Product engineering (also called system engineering) is a problem solving activity. Desired product data, function, and behavior are uncovered analyzed and allocated to individual engineering components. The system engineer begins with customer-defined objectives and goals for the product and proceeds to model this requirement in a manner that allocates them to as set of engineering components software, hardware, data and people. The components are tied together with a support infrastructure the technology required to integrate the components and the information that is used to support the components.

The genesis of most new products and systems begins with a rather nebulous concept of desired function.

Therefore the system engineer mist bound the product requirements by identifying the scope of function and performance

desired. For example, it is not enough to say that the control software for the robot in a manufacturing automation system will "respond rapidly if a parts tray is empty. " The system engineer mist defines

(1) what indicates an empty tray to the robot,

(2) the precise time bounds within which software response is expected and

(3) what form the response mistake. That is the system engineer mist describes the events that drive the behavior of the robot the nature of the behavior and the quantitative bounds placed on the behavior.

Once function, performance, constraints and interfaces are bounded the system engineer moves onto a task that is called allocation. During allocation function is assigned to one or more engineering components. Often alternative allocations are proposed and evaluated. To illustrate the process of allocation we consider a macro element of the factory automation system the conveyor line sorting system (CLSS). The system engineer is presented with the following statement of objectives for CLSS.

The boxes will pass by a sorting station where they will be identified.

Based on an identification number printed on the side of the box (an equivalent bar code is provided), the boxes will be shunted into the appropriate bins. Boxes pass in random order and are evenly spaced. The line is moving slowly.

CLSS is depicted schematically continuing make a list of questions that you would ask if you were the system engineer.

Among the many questions that should be asked and answered are the following:

1. How many different identification number must be processed and what is their form?
2. What is the speed of the conveyor line in feet per second and what is the distance between boxes in feet?
3. How far is the sorting station from the bins?
4. How far apart are the bins?

5. What should happen if a box doesn't have an identification number or an incorrect number is present?
6. What happens when a bin fills to capacity?
7. Is information about box destination and bin contents to be passed elsewhere in the factory automation system? Is real-time data acquisition required?
8. What error/failure rate is acceptable?
9. What pieces to the conveyor line system currently exist and are operational?
10. What schedule and budgetary constraints are imposed?

The system engineer does not ask the customer how the task is to be done; rather the engineer asks what is required.

Assuming reasonable answers, the system engineer develops a number of alternative allocations. Note that function and performance are assigned to different generic system elements in each allocation.

Allocation 1: A sorting operator is training and place at the sorting station location. He/She reads the box and places it into an appropriate bin.

Allocation 1 represents a purely manual (but nevertheless, effective) solutions to the CLSS problem. The primary engineering component is people (the sorting operator). The person performs all sorting functions. Some documentation (in the form of a table relating identification number to bin location and procedural description for operator training) may be required. Therefore this allocation uses only the people and documentation elements.

Allocation 2: A bar code reader and controller are placed at the sorting station. Bar code output is passed to a programmable controller that controls a technical shunting mechanism the shunt slides the box to the appropriate bin.

For allocation 2, hardware (bar code reader, programmable control, shunt hardware, etc.,) software (for the bar code reader and programmable controller) and database (a look-up table that relates box ID with bin location) components are used to provide a fully automated solution. It is likely that each of these components may have corresponding manuals and other documentation adding another component.

Allocation 3: A bar code reader and controller are placed at the sorting station. Bar code output is passed to a robot arm that grasps a box and moves it to the appropriate bin location.

Allocation 3 makes use of one macro element— the robot. Like allocation 2, this allocation uses hardware, software, a database and documentation as engineering components. The robot is a macro element of

CLSS and it contains a set of engineering components.

By examining the three alternative allocations for CLSS it should be obvious that the same function can be allocated to different components. In order to choose the most effective allocation a set of trade-off criteria should be applied to each alternative.

The following trade-off criteria govern the selection of a product configuration based on a specific allocation of function and performance to generic system elements:

Project considerations: Can the configuration be built within pre-established cost and schedule bound? What is the risk associated with cost and schedule estimates?

Business considerations: Does the configuration represent the most profitable solution? Can it be marketed successfully? Will ultimate payoff justify development risk?

Technical analysis: Does the technology exist develop all elements of the system? Are function and performance assured? Can the configuration be adequately maintained? Does technical resource exist? What is the risk associated with the technology?

Manufacturing evaluation: Are manufacturing facilities and equipment available? Is there a shortage of necessary component? Can quality assurance be adequately performed?

Human Issues: Are trained personnel available for development and manufacture? Do political problems exist? Does the customer understand what the system is to accomplish?

Environmental Interfaces: Does the proposed configuration properly interface with the system's external environment? Are machine-to-machine and human-to-machine communication handled in an intelligent manner?

Legal Considerations: Does this configuration introduce undue liability risk? Can proprietary aspects be adequately protected? Is there potential infringement?

We examine some of these issues in more detail late in this lecture.

It is important to note that the system engineer should also consider off- the shelf solutions to the customer's problem. Does an equivalent system already exist? Can major parts of a solution be purchased from a third party?

The application of trade-off criteria results in the selection of a specific system configuration and the specification of function and performance allocated to hardware software, people, databases, documentation and procedures. Essentially the scope of function and performance is allocated to each engineering component of the product. The role of hardware engineering, software engineering, human engineering and database engineering is to refine scope and procedure an operational product component that can be properly integrated with other components.

System Analysis

System analysis is conducted with the following objectives in mind:

1. identify the customer's need;
2. evaluate the system concept for feasibility;
3. perform economic and technical analysis;
4. allocate function to hardware, software, people, database and other system elements;
5. establish cost and schedule constraints; and
6. create a system definition that forms the foundation for all subsequent engineering work. Hardware and software expertise (as well as human and database engineering) are required to successfully attain the objectives listed.

Identification of Need

The first step of the system analysis process involves the identification of need. The analyst (system engineer) meets with the customer and the end user (if different from the customer).

The customer may be a representative of an outside company, the marketing department of the analyst's company (when a product is being defined), or another technical department (when an internal system is to be developed). Like information engineering the intent is to understand the product's objective(s) and to define the goals required to meet the objective(s).

Once overall goals are identified, the analyst moves on to an evaluation of supplementary information: Does the technology exist to build the system? What special development and manufacturing resources will be required? What bounds have been placed on costs and schedule? If the new system is actually a product to be developed for sale to many customers the following questions are also asked: what is the potential market for the product? How does this product compare with competitive products? What position does this product take in the overall product line of the company? Information gathered during the needs identification step is specified in a system concept document. The original concept document is sometimes prepared by the customer before meetings with the analyst. Invariably customer-analyst communication results in modification to the document.

Feasibility Study

All projects are feasible given unlimited resources and infinite time! Unfortunately, the development of a computer-based system or product is more likely plagued by a scarcity of resources and difficult (if not downright unrealistic) delivery dates. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. Months or years of effort thousand for millions of dollars and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase.

Feasibility and risk analysis are related in many ways. If project risk is great the feasibility of producing quality software is reduced. During product engineering however, we concentrate our attention on four primary areas of interest:

Economic feasibility: An evaluation of development cost weighed against the ultimate income or benefit derived from the developed system or product.

Technical Feasibility: A study of function, performance and constraints that may affect the ability to achieve an acceptable system.

Legal feasibility: A determination of any infringement violation or liability that could result from development of the system.

Alternative: An evaluation of alternative approaches to the development of the system or product.

A feasibility study is not warranted for systems in which economic justification is obvious technical risk is low, few legal problems are expected and no reasonable alternative exists. However if any of the preceding conditions fail a study of that area should be conducted.

Economic justification is generally the "bottom-line" consideration for most systems (notable exceptions sometimes include national defense systems, systems mandated by law, and high-technology applications such as the space program). Economic justification includes a broad range of concerns that include cost benefits analysis long-term corporate income strategies impact on other profit centers or products cost of resources needed for development and potential market growth.

Technical feasibility is frequently the most difficult area to assess at this stage of the product engineering process. Because objectives functions and performance are somewhat hazy anything seems possible if the "right" assumptions are made. It is essential that the process of analysis and definition be conducted in parallel with an assessment of technical feasibility. In this way concrete specification may be judged, as they are determined.

The considerations that are normally associated with technical feasibility include:

Development risk: Can the system element be designed so that necessary function and performance are achieved within the constraints uncovered during analysis?

Resource availability: Is skilled staff available to develop the system element in question? Are other necessary resources (hardware and software) available to build the system?

Technology: Has the relevant technology progressed to a state that will support the system?

Developers of computer-based systems are optimists by nature. However, during an evaluation of technical feasibility a cynical if not pessimistic attitude should prevail. Misjudgment at this stage can be disastrous.

Legal feasibility encompasses a broad range of concerns that include contracts, liability, infringement and myriad other traps frequently unknown to technical staff. The degree to which alternatives are considered is often limited by cost and time constraints; however a legitimate but "unsponsored" variation should not be buried.

The feasibility study may be documented as a separate report to upper management and included as and to the system specification. Although the format of a feasibility study may vary the outline provided in covers most important topics.

The feasibility study is reviewed first by project management (to assess content reliability) and by upper management (to assess project status). The study should result in a "go/no-go" decision. It should be noted that other go/no-go decisions will be made during the planning, specification, and development steps of both hardware and software engineering.

Economic Analysis

Among the most important information contained in feasibility study is cost-benefit analysis an assessment of the economic justification for a computer-based system project. Cost-benefit analysis delineates costs for project development and weighs them against tangible. (i.e., measurable directly in dollars) and intangible benefits of a system.

Cost-benefit analysis is complicated by criteria that vary with the characteristics of the system to be developed, the relative size of the project and the expected return on investment desired as part of a company's strategic plan. In addition many benefits derived from computer-based systems are intangible

(for example better design quality through iterative optimization, increased customer satisfaction through programmable control, and better business decisions through reformatted and reanalyzed sales data).

Direct quantitative comparisons may be difficult to achieve.

Analysis of benefits will differ depending on system characteristics. To illustrate, consider the benefits for management information systems most data-processing systems are developed with 'better information quantity, quality, timeliness or organization' as a primary objectives. Therefore the concentrate on information access and its impact on the user environment. The benefits that might be associated with an engineering-scientific analysis program or a computer-based product could differ substantially.

Costs associated with the development of a computer-based system the analyst can estimate each cost and then use development and ongoing costs to determine a return on investment, a break-even point and a payback period.

The following excerpt may best characterize cost-benefit analysis:

Like political rhetoric after the election, the cost-benefit analysis may be forgotten after the project implementation begins. However it is extremely important because it has been the vehicle by which management approval has been obtained.

Only by sending the time to evaluate feasibility do we reduce the chances for extreme embarrassment (or worse) at later stages of a system project. Effort spent on a feasibility analysis that results in cancellation of a proposed project is not wasted effort.

Technical Analysis

During technical analysis, the analyst evaluates the technical merits of the system concept, at the same time collecting additional information about performance, reliability, maintainability and reducibility. This system analysis step also includes a limited amount of research and design.

Technical analysis begins with an assessment of the technical viability of the proposed system. What technologies are required to accomplish system function and performance? What new material, methods, algorithms are processes are required and what is their development risk? How will these technology issues affect cost?

The tools available for technical analysis are derived from mathematical modeling and optimization techniques, probability and statistics queuing theory and control theory – to name a few. It is important to note, however that analytical evaluation is not always possible. Modeling (either mathematical or physical) is an effective mechanism for technical analysis of computer-based systems.

Blanchard and Fabrycky define a set of criteria for the use of models during technical analysis of systems:

- The model should represent the dynamics of the system configuration being evaluated in a way that is simple enough to understand and manipulate and yet close enough to the operating reality to yield results.
- The models should highlight those factors that are most relevant to the problem at hand and suppress (with discretion) those that are not as important.
- Model design should be simple enough to allow for timely implementation in problem solving. Unless the model can be utilized in a timely and efficient manner by the analyst or the manager, it is of little value. If the model is large and highly complex, it may be appropriate to develop a series of smaller models in which the output of one can be tied to the input of another. Also it may be desirable to evaluate a specific element of the system independently from other elements.
- Models design should incorporate provisions for ease of modification and/or expansion to permit the evaluation of additional factors as required. Successful model development often includes a series of trials before the overall objective is met. Initial attempts may suggest information gaps which are not immediately apparent and consequently may suggest beneficial changes.

The results obtained from technical analysis form the basis for another 'go/no-go' decision on the system.

If technical risk is severe, if models indicate that desired function or performance cannot be achieved, if the pieces just won't fit together smoothly – it's back to the drawing board!

Modeling the System Architecture

Every computer-based system can be modeling as information transforms using input-processing output architecture. Hatley and Pirbhai have extended this view to include two additional system features-user interface processing and maintenance and self-test processing. Although these additional features are not present for every computer-based system, they are very common, and their specification makes any system model more robust. Using a representation of input, processing, output, user interface processing, and self-test processing, a system engineer can create a model of system components that sets a foundation for later requirements analysis and design steps in each of the engineering disciplines.

To develop the system model, an architecture template is used. The system engineer allocates system elements to each of five processing regions within the template (1) user interface,(2) input, (3) system function and control, (4) output and (5) maintenance and self-test.

Like nearly all modeling techniques used in system and software engineering, the architecture template enables the analyst to create a hierarchy of detail. An architecture context diagram (ACD) resides at the top level of the hierarchy.

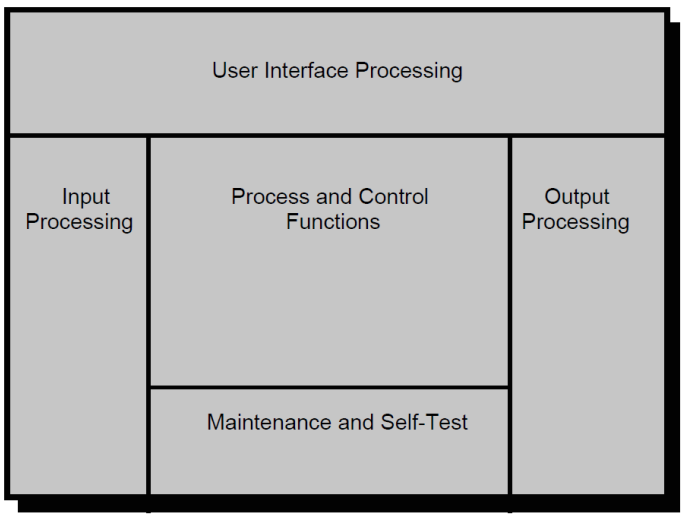


Figure 2.8: Architecture Template.

The context diagram 'establishes the information boundary between the system being implemented and the environment in which the system is to operate'. That is the ACD defines all external producers of information used by the system, all external consumers of information created by the system and all entities that communicate through the interface or perform maintenance and self-test.

To illustrate the use of the ACD, consider an extended version of the conveyor line sorting system (CLSS). The extended version makes use of personal computer at the sorting station site. The PC executes all CLSS software; interfaces with the bar code reader to read part number on each box; interfaces with the conveyor line monitoring equipment to acquire conveyor line speed; stores all part number sorted; interacts with a sorting station operator to produce a variety of reports and diagnostics; sends control signals to the shutting hardware to sort the boxes; and communicates with a central factory automation mainframe. The ACD for CLSS is shown in figure 2.9.

Each box shown in figure 2.9 represents an external entity-that is a producer or consumer of information form the system. For example, the barcode reader produces information that is input to the CLSS system. The symbol for the entire system (or at lower levels, major subsystems) is a rectangle with rounded corners. Hence, CLSS is represented in the ACD represent information (data and control) as it moves from the external environment into the CLSS system. The external entity bar code reader produces input information that is labeled bar code. In essence, the ACD places any system into the context of its external environment.

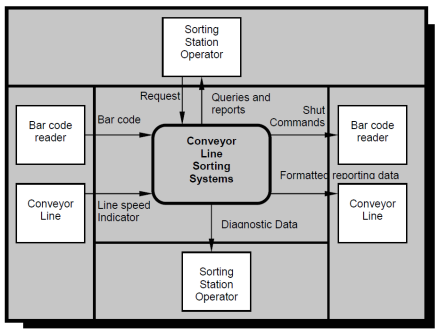


Figure 2.9: Architecture context diagram for CLSS.

The system engineer refines the architecture context diagram by considering the shaded rectangle in figure 2.9 in more detail. The major subsystems that enable the conveyor line sorting system to function within the context defined by the ACD are identified. In figure 2.10 the major subsystems are defined in an architecture flow diagram (AFD) that is derived from the ACD. Information flow across the regions of the ACD is used to guide the system engineer in developing the AFD – a more detailed 'schematic' for CLSS. The architecture flow diagram shows major subsystems and important lines of information (data and control) flow. In addition, the architecture template partitions the subsystem processing into each of the five processing regions At this stage, each of the subsystems can contain one or more system elements (for example hardware, software, people) as allocated by the system engineer.

The initial architecture flow diagram (AFD) becomes the top node of a hierarchy of AFDs. Each rounded rectangle in the original AFD can be expanded into another architecture template dedicated solely to it.

This process is illustrated systematically in figure 2.11. Each of the AFDs for the system can be used as a starting point for subsequent engineering steps for the subsystem. Subsystems and the information that flows between them can be specified (bounded) for subsequent engineering work. A narrative description of each subsystem and a definition of all data that flow between subsystems become important elements of the system specification.

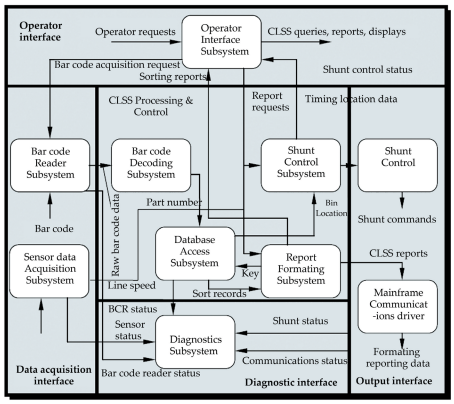


Figure 2.11: Architecture flow diagrams for CLSS.

System Modeling and Simulation

Almost three decades ago, R.M. Graham made a distressing comment about the way we built computer-based systems: "We build systems like the Wright brothers built airplanes build the whole thing, push it off a cliff, let it crash and start over again". In fact, for at least one class of system the reactive system - we continue to do this today.

Many computer-based systems interact with the real worked in a reactive fashion. That is real world events are monitored by the hardware and software that comprise the computer-based system, and based on these events, the system imposes control on the machines, processes and even people who cause the event to occur. Real-time and embedded systems often fall into the reactive systems category.

Unfortunately, the developers of reactive systems sometimes struggle to make them perform properly.

Until recently, it has been difficult to predict the performance, efficiently and behavior of such systems prior to building them. In a very real sense, the construction of many real-time systems was an adventure in 'flying'. Surprise (most of them unpleasant) was not discovered until the system was built and 'pushed off a cliff'. If the system crashed due to incorrect function, inappropriate behavior, or poor performance, we picked up the pieces and started over again.

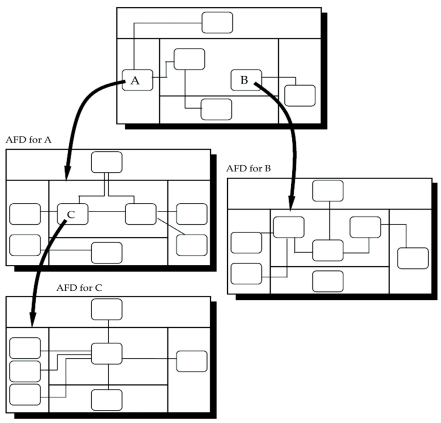


Figure 2.12: Building an AFD hierarchy.

Many systems in the reactive category control machines and/or processes (for example commercial airliners or petroleum refineries) that must operate with an extremely high degree of reliability. If the system fails, significant economic or human loss could occur. For this reason, the approach by Graham is both painful and dangerous.

System Specification

The system specification is a document that serves as the foundation for hardware engineering, software engineering, data base engineering and human engineering. It describes the function and performance of a computer-based system and the constraints that will govern its development. The specification bounds each allocated system element. For example, it provides the software engineer with an indication of the role of software within the context of the system as a whole and the various subsystems in the architecture flow diagrams. The system specification also describes the information (data and control) that is input to and output from the system.

It should be noted, however that this is but one of many outlines that can be used to define a system description document. The actual format and content may be dictated by software or system engineering standards or local custom and preferences.

CASE STUDY

SYSTEMS ENGINEERING: IMPROVING PATIENT SAFETY AND SYSTEM PRODUCTIVITY BY REDUCING LEAD-TIMES FOR BLOOD TESTS

Clarkson and Dodds summarized the development of the quality improvement in the NHS In order to improve the impact of the changes at macro-system level, ie hospital and health economy, they propose that system engineering, as applied to other complex systems eg telecommunications and global supply chains, should be used to diagnose and focus on the constraints within the macro-system and to coordinate and synchronize the improvements by the individuals and departments concerned.

This paper recounts how 'hard' systems engineering (involving manpower and machines) was used to bridge the gap between the current and required performance of the process for blood tests. In improving the previously unrecognized delays for blood tests, we uncovered the 'soft' system or 'simple rules' that govern the individuals and departments and these need to be understood and changed too.

Problem Description and Method

A high-level map of the reported patients' journeys through the hospital and observations of seven ward rounds revealed that many decisions made on ward rounds were based on blood tests taken up to 24 hours previously. Not only was this previously unrecognized constraint seen as a major safety issue (one patient died waiting for a gentamicin level and another required transfusion following a consultant's decision to discharge), the clinicians and managers recognized that these delays compromised the whole healthcare system as patients waiting in hospital beds for clinical decisions would delay new emergency and elective patients getting to their specialist wards.

A consultant physician led a multidisciplinary team of a porter, a junior doctor, a ward-based nurse, a phlebotomist, the laboratory manager, a laboratory technician and a laboratory scientist. The team agreed to meet weekly and follow Lean's A3 problem solving process.

The team mapped the process from requesting the correct blood test through to acting on the result. The team members had not previously recognized the impact of their individual and departments' work-schedules on the end-to-end process. The reasons for the delays were then explored and coordinated interventions tested. An initial demand and workload analysis were performed for the laboratory part of the process and continued after workflow changes.

Results and Analysis

The high-level results of the process mapping and analysis for an inpatient (IP) are shown in Fig 1 and Table 1.

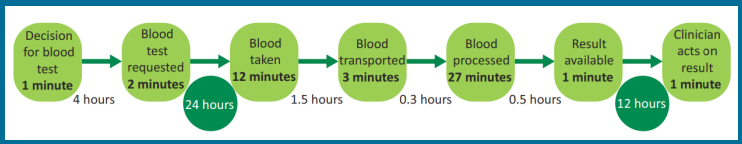


Figure 1. The end-to-end blood test process. The average cycle times taken to perform each task and the delays between the tasks were measured by the team.

Table 1. Reason for delays, consequences of work-around and the interventions required

Delay	Reason	Work around	Intervention required
4 hours delay to making the request.	Not enough time to complete blood forms on the ward round, so forms completed after the 4-hour ward round.	Junior doctor leaves ward round to complete form (and take blood) missing vital decisions regarding patient care. The alternative is to request tests the day before – many of which may not be correct or necessary on the following day.	Forms available on the trolley and completed by any doctor on the ward round (including the consultant when necessary). Forms dated with the required date of the test.
24 hours delay to blood taken.	The phlebotomists' rounds happened while the ward round happened. So, any blood test requests made after the ward round were taken 24 hours later.	Doctor taking blood takes 12 minutes: 10 minutes to find the kit and gloves with many interruptions. So, doctor asks phlebotomists to take 'urgent' samples.	Phlebotomists start at 7.30am rather than 8.00am to get the samples to the lab. Every bay on every ward has a fully stocked trolley for taking blood and putting up drips.
1.5 hours delay to transport.	The phlebotomists (and other clinical staff) left specimens in the ward out-tray for the porters rounds which occurred every 1.5 hours.	Doctors and nurses phone the porters to take urgent samples to the lab. Porters prioritise these calls.	Between 7.30 to 9.00am two porters 'chase' the six phlebotomists from ward to lab and back to next ward so that smaller batches are delivered continuously to the lab.
20 minutes delay to processing.	Big batches of work arriving in the lab to be registered.	Lab technicians prioritise 'urgent' samples from A&E over ward, OP and GP samples.	Six technicians continuously register samples from all sources as they arrive.
Half hour delay to verification and release.	Lab scientist looks at the results every half hour before releasing a batch electronically onto the clinical info system.	Lab staff stop work to search for 'urgent' requests and results.	One lab scientist continuously reads, verifies and releases results from 8.00am to 12.00pm.

12 hours delay to action.	Results released in the afternoon are acted on the next morning.	Night staff and out-of-hours GP called urgently to deal with unexpected and dangerously abnormal results for patients they don't know.	All samples and results cleared and released by 4.00pm so requesting doctors can act on the result.

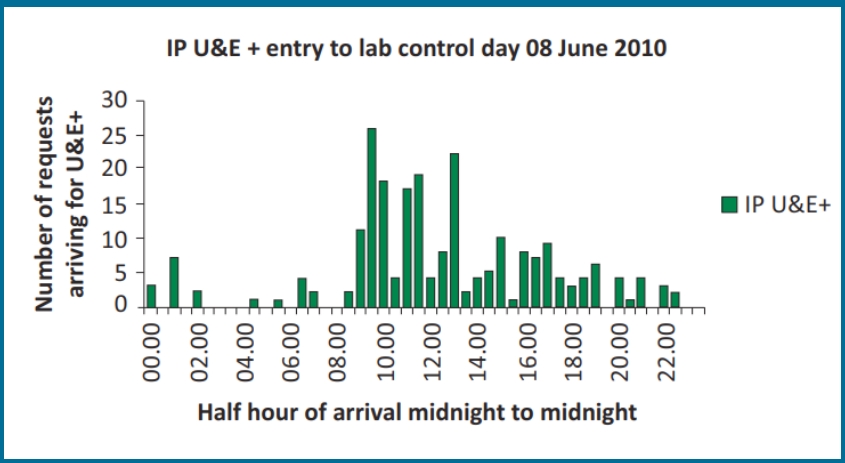
A&E = accident and emergency; GP = general practitioner; OP = outpatient

A cycle time is the time from a resource starting a task to being ready to start the same task for the next patient. The sum of average cycle times for each task in the end-to-end process showed that, in theory, the minimum lead-time was 47 minutes. However, the invisible delays between the tasks in the process caused by the working schedules and policies meant that the lead-time could be over 40 hours. This explained why many clinical decisions were incorrect. Clinicians assumed that they were making decisions on *today's* test, whereas their decisions were based on, at best, a blood sample taken the previous day.

The team realized that the main delays were not in the laboratory as they initially thought, and that their own workarounds didn't help and wasted both resource and time.

They had to design an end-to-end system to ensure that results for all IP blood samples requested on the day of the ward rounds were available before the end of the ward rounds at 12.00pm. This also should not compromise time to reporting for patients in accident and emergency (A&E), outpatients (OP) and general practice (GP), important additional system elements.

The laboratory staff retrieved the data for all blood requests that included urea and electrolytes. Fig 2 shows the demand into the lab before and after the changes to the upstream ward, phlebotomy and portering processes as shown in Table 1.



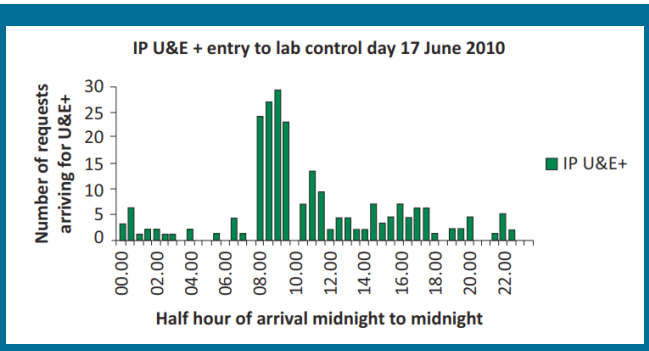


Figure 2. The daily pattern of arrivals into the lab before and after the interventions upstream. IP = inpatient; U&E = urea and electrolytes.

The peak demand for the ward samples now arrives in the laboratory earlier between 8.00am and 9.30am.

The mean and variation in demand, based on total laboratory data (A&E, IP, OP and GP samples) after the upstream changes were made to the IP process is shown in Fig 3.

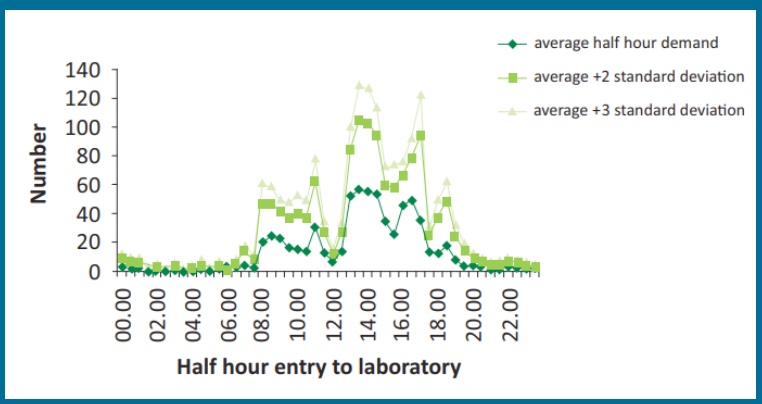


Figure 3. The variation in total demand into the laboratory. Half hourly demand profile entry to laboratory for 7 days; June to August 2010.

To ensure that the lab could provide the results within an hour of receiving the samples on 95% of days, we planned the lab flow capacity to meet the average plus two standard deviations of the half hourly demand. The period of highest demand is 2.00pm ie 100 samples / 30 minutes, or one sample every 18 seconds and this is when samples from GP and OP clinics start to arrive. In the morning, the rate of demand is half the afternoon rate ie one sample every 36 seconds.

The effective cycle time in seconds is calculated for each step in the laboratory process and revealed two flow constraints (Table 2). First, the sorting machine (track)

can't meet the afternoon demand of one sample every 18 seconds. Its main purpose is to sort the big batches from the centrifuges so that the samples can be put in order of their priority into the two analysers. It also receives and caps the processed samples returning from the analysers prior to storage. The immediate solution was to free up the sorting machine by scheduling the capping-off to after 8.00pm and delivering samples directly to the analysers.

The second constraint was centrifugation. The best effective cycle time that the three centrifuges in parallel could achieve is 6.25 seconds, and therefore this is not a resource constraint. The problem here was the queue-batch-queue policy that creates a time-trap for the samples. To cope with the increasing demand, the lab had replaced several smaller centrifuges with the three larger ones and the policy was to wait until each centrifuge was full of its maximum load of 32 samples before starting it. This meant that in the morning, the first sample of the batch of 32 could wait nearly 20 minutes for the centrifuge to start, and the afternoon it could wait nearly 10 minutes. To mitigate the impact of these delays, one of the three centrifuges was reserved for the urgent samples from A&E and this machine was run irregularly and intermittently according to unclear rules.

So, the rest of the samples now competed for the remaining two centrifuges and urgent samples from the A&E and IP wards taking priority over OP and GP samples. This time-trap at the centrifuges could be reduced by starting each of the three centrifuges regularly at 3.33 minute intervals with however-many samples were waiting. It also became apparent to the lab team that they could free up some of their resource in the morning to join the phlebotomy and portering team on the wards, thus improving the flow into the lab without compromising the flow through the lab (Fig 4).

Table 2. The laboratory cycle times and flow capacity (manpower and machines) available

Laboratory process step	**Average cycle time: time from the start of one specimen to the start of the next specimen at the same process step.**	**Batch quantity**	**Capacity available**	**Effective cycle time in seconds**
Sorting	5 seconds	1	One person	5
Log-in	90 seconds	1	Six people and six computers	15
Centrifuge	10 minutes	32	One person managing three machines (each with a max of 32 samples)	6.25

Sorting machine (track)	20 minutes	32	One person and one machine	37.5
Analyser	30 seconds	1	One person and two machines	15
Verification	20 seconds	Policy of only checking results every 30 minutes	One person and two computers	20

Samples are grouped by source (IP, A&E, GP and OP) and ordered along the x-axis by their authorization date and time (result available to clinicians). The mean is shown as a green line and the red lines show the range of variation either side of the mean (mean ±3 sigma).

The white dots are for blood results that have only been released the day after the sample was received. The raw data reveals that these samples arrived in the lab during the night shift (8.00pm to 8.00am) and were not deemed urgent by the clinical or lab staff or required transfer for unusual or more complex testing at the central laboratory 14 miles away (e.g. gentamicin). These results are clearly from a different system (outside mean +3 sigma) and have been excluded from the statistical analysis for those samples arriving into the lab during the normal working hours 8.00am to 8.00pm.

The summary tables show that the mean and sigma (variation) in laboratory cycle times for IP samples reduced indicating that the IP system was more responsive and predictable, with negligible impact on the samples from A&E and GP, but the mean and variation for lead-times for those from OP also improved.

Though improved, there is still variation in flow though the blood test system that needs to be addressed. The red dots indicate 'special causes' i.e. statistically significant changes in the pattern of behavior of the blood testing process demonstrating that it is not yet stable and resilient to the numerous other known and unknown causes of variation. For example, lab flow remains compromised at lunchtime, when the bulk of GP ad OP samples arrive at 2.00pm and when the laboratory scientist authorizing the results can only just cope with the peak flow rate in the afternoon.

Questions

1. Define the problem description and method for system engineering.
2. Examine the laboratory cycle times and flow capacity.

KEY TERMS

Business processes
Software engineering
Information engineering
Business area analysis
Architecture flow diagram
Electrical engineers
A programming language
Information strategy planning
Conveyor line sorting system

DISCUSSION QUESTIONS

1. Give an overview on system modeling.
2. What is the information strategy planning?
3. Determine the enterprise modeling process.
4. Define the business-level data modeling.
5. What do you understand by the business area analysis?

REFERENCES

1. Donna Rhodes; Daniel Hastings (March 2004). "The Case for Evolving Systems Engineering as a Field within Engineering Systems". MIT Engineering Systems Symposium.
2. Hollnagel, E. & Woods, D. D. (2005) Joint cognitive systems: The foundations of cognitive systems engineering. Taylor & Francis
3. Parnell, G.S., Driscoll, P.J., Henderson, D.L. (eds.), Decision Making in Systems Engineering and Management, 2nd. ed., Hoboken, NJ: Wiley, 2011. This is a textbook for undergraduate students of engineering.
4. Umbrello, Steven (5 April 2021). "Coupling levels of abstraction in understanding meaningful human control of autonomous weapons: a two-tiered approach". Ethics and Information Technology. 23 (3): 455–464.

CHAPTER 3

ANALYSIS MODELING



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Define the essential of analysis modeling
2. Discuss on functional modeling and information flow
3. Learn about the behavioral modeling
4. Creating a data flow model
5. Give an overview on data dictionary

INTRODUCTION

Analysis Model is a technical representation of the system. It acts as a link between system description and design model. In Analysis Modelling, information, behavior, and functions of the system are defined and translated into the architecture, component, and interface level design in the design modeling.

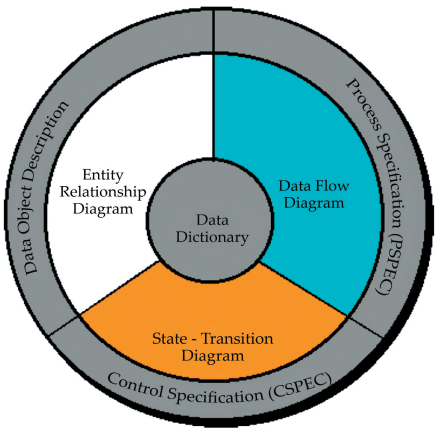
Models represent the information that software transform, the function that enables this transformation, the features that the user's desired, and the behavior of the system as transformation is taking place. Analysis modeling represents the user's requirements by depicting the software in three different domains: information domain, functional domain, and behavioral domain.

ESSENTIAL OF ANALYSIS MODELING

The analysis model is designed by a software engineer or modeler or system analysts, or project manager. To describe the requirement, it uses diagrammatic form and text.

Elements of Analysis Model

Analysis model operates as a link between the 'system description' and the 'design model'. In the analysis model, information, functions and the behavior of the system is defined and these are translated into the architecture, interface and component level design in the 'design modeling'.



- **Data Dictionary:** It is a repository that consists of a description of all data objects used or produced by the software. It stores the collection of data present in the software. It is a very crucial element of the analysis model. It acts as a centralized repository and also helps in modeling data objects defined during software requirements.
- **Entity Relationship Diagram (ERD):** It depicts the relationship between data objects and is used in conducting data modeling activities. The attributes of each object in the Entity-Relationship Diagram can be described using Data object description. It provides the basis for activity related to data design.
- **Data Flow Diagram (DFD):** It depicts the functions that transform data flow and it also shows how data is transformed when moving from input to output. It provides the additional information which is used during the analysis of the information domain and serves as a basis for the modeling of function. It also enables the engineer to develop models of functional and information domains at the same time.
- **State Transition Diagram:** It shows various modes of behavior (states) of the system and also shows the transitions from one state to another state in the system. It also provides the details of how the system behaves due to the consequences of external events. It represents the behavior of a system by presenting its states and the events that cause the system to change state. It also describes what actions are taken due to the occurrence of a particular event.
- **Process Specification:** It stores the description of each function present in the data flow diagram. It describes the input to a function, the algorithm that is applied for the transformation of input, and the output that is produced. It also shows regulations and barriers imposed on the performance characteristics that are applicable to the process and layout constraints that could influence the way in which the process will be implemented.
- **Control Specification:** It stores additional information about the control aspects of the software. It is used to indicate how the software behaves when an event occurs

and which processes are invoked due to the occurrence of the event. It also provides the details of the processes which are executed to manage events.

- **Data Object Description:** It stores and provides complete knowledge about a data object present and used in the software. It also gives us the details of attributes of the data object present in the Entity Relationship Diagram. Hence, it incorporates all the data objects and their attributes.

Principles of Analysis Model

Principles are necessary to build an effective analysis model.

- **Represent Information Domain:** The information domain represents the data flow into the system from an end-user or external system and devices, the data flow out of the system through the user interface or network interface and the data stores that collect and organize the permanently required data objects.
- **Represent Software Functions:** Software functions are responsible for the data flow in the system. It also provides support for those features that are user-visible and they are beneficial to the end-users. They also affect internal and external system elements.
- **Represent Software Behavior:** Software behavior represents the way the software interacts with the external environment. Input provided by the users, control data provided by the external system it was monitoring data collected over a network cause the software to behave indefinite way.
- **Partition of Above Three Representations:** The model that depicts information, behavior, and function should be partitioned in a layered manner to get a detailed view of the system. And this's very much helpful in the complex problems which are difficult to solve. A large complex problem is divided into subproblems until each subproblem is relatively easy to understand. This concept is called partitioning and is the key task of analysis modeling.

- **Move from Essence Towards Implementation:** Analysis modeling describes the problem from the user's perspective. The essence of the problem is described without any consideration of how a solution will be implemented, and implementation detail indicates how the essence will be implemented.

Rule of Thumb for Analysis Model

Arlow and Neustadt create rules of thumbs for analysis modeling.

These rules are as follows:

- The model should focus on requirements visible in a business domain.
- Each element of the analysis model should add to all requirements and provide insight into the system's information domain function and behavior.
- Some functions and not functional models are required to more time they need to design after the design process of the software.
- It gives assurance that the analysis model provides value to all stock holders.
- Minimizing coupling throughout the system. It is important to represent a relationship between classes and functions. Functions are interconnected to each other very tightly then efforts are put to reduce this interconnectedness.
- It keeps the model as simple as it can be.

Analysis Modeling Approaches

Given below are the approaches mentioned:

- **Context Analysis:** It is an external viewpoint that shows the context or environment of the system.
- **Structured Analysis:** It is a structural viewpoint that shows the architecture of the system or data. It involves the data model like the entity-relationship diagram. It considers data and the processes that transform the data into separate entities called as data objects. Data objects define the attributes and relationships. This process tells

us how we can manipulate and transform the data. It also represents the flow of data objects.

- **Behavioral Analysis:** It is behavioral viewpoint that shows the behavior of the system. Software engineers choose one approach: the best of software requirements and the most effective bridge to software design. The analysis model derives modeling elements. The specification of each element may differ from project to project.
- **Object-Oriented Analysis:** It focus on the definition of classes. It provides object-oriented design methodology such as UML and Unified process analysis. It performs collaboration of classes to fulfill the user's requirements.

"Mathematics as an expression of the human mind reflects the active will, the contemplative reason, and the desire for aesthetic perfection. Its basic elements are logic and intuition, analysis and construction, generality and individuality."

—Richard Courant

Elements of Analysis Modeling

Given below are the elements mentioned:

- **Data Dictionary:** Data dictionaries are lists of all the names used in the system models; description of the entities, their attributes and relationships are included.
- **Data Flow Diagram:** Data flow diagram model the data processing of the system.
- **State Transition Diagram:** It represents the dynamic model, which shows changes in the state that an object goes through during its lifetime in response to an event.
- **Entity Relationship Diagram:** It consists of information required for each entity or data objects and shows a relationship between objects. It shows the structure of the data in table format.
- **Data Object Description:** It represents the composite information of one entities such that the software understands it.

Importance of Analysis Modeling

Given below are the importance mentioned:

- The analysis model is multidimensional.
- The design modelling phase completely depends upon the analysis model.
- If any deficiency remains in the analysis model, then errors will be found in the ultimate product to be built.

Data Modeling

Data modeling methods make use of the entity relationship diagram (ERD). In the context of structured analysis, the ERD defines all data that are input, stored, transformed and produced within an application. The entity-relationship diagram focuses solely on data (and therefore satisfies the first operational analysis principle), representing a "data network" that exists for a given system. Unlike the data flow diagram data modeling considers data independently of the processing that transforms the data.

Data Objects, Attributes and Relationships

Data model consists of three interrelated pieces of information: the data objects the attributes that describe the data object and the relationship that connect data objects to one another.

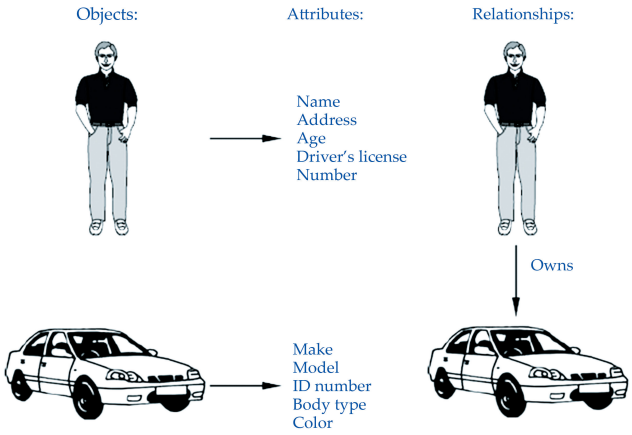


Figure 3.1: Data objects, attributes, and relationships

Data Objects

A data object is a representation of almost any composite information that must be understood by software. By composite information we mean something that has a number of different properties or attributes. Therefore "width "would not be a valid data object, but dimensions (incorporating height, width and depth) could be defined as an object.

A data object can be an external entity (*for example*, anything that produces or consumes information), a thing (*for example*, a report or a display), an organizational unit (*for example*, accounting department), a place (*for example*, a warehouse), or a structure (*for example*, a file). *For example,* a person or a car can be viewed as a data object in the sense that either can be defined in terms of a set of attribute. The data object description incorporates the data object and all of its attributes.

Data objects are related to one another. *For example*, person can own car where the relationship own connotes a specific connection between person and car. The relationships are always defined by the context of the problem that is being analyzed.

A data object encapsulates data only there is no reference within a data object to operations that act on the data. Therefore, the data object can be represented. The headings in the table reflect attribute of the object. A car is defined in terms of make, model, ID# body type, color and owner. The body of the table represents specific instances of the data object. *For example*, a Chevy Corvette is an instance of the data object car.

Attributes

Attributes define the properties of a data object and take on one of three different characteristics.

They can be used to:

- Name an instance of the data object,
- Describe the instance, or
- Make reference to another instance in another table.

In addition, one or more of the attributes must be defined as an identifier that is the identifier attribute becomes a "key" when we want to find an instance of the data object. In some

case, values for the identifier(s) are unique, although this is not a requirement. Referring to the data object car, a reasonable identifier might be the ID#

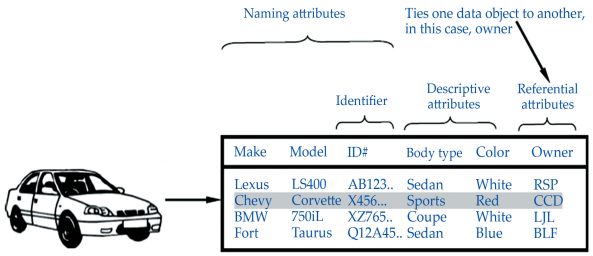


Figure 3.2: Tabulator representation of data objects.

The set of attributes that is appropriate for a given data object is determined through an understanding of the problem context. The attributes for car might serve well for an application that would be used by a Department of Motor Vehicles, but these attributes would be useless for an automobile company that needs manufacturing control software. In the latter case the attributes for car might also include ID#, body type, and color, but many additional attributes (*for example*, interior code, driver train type, trim package designator, transmission type) would have to be added to make car a meaningful object in the manufacturing control context.

"In the last analysis, the individual person is responsible for living his own life and for 'finding himself.' If he persists in shifting his responsibility to somebody else, he fails to find out the meaning of his own existence."

—Thomas Merton

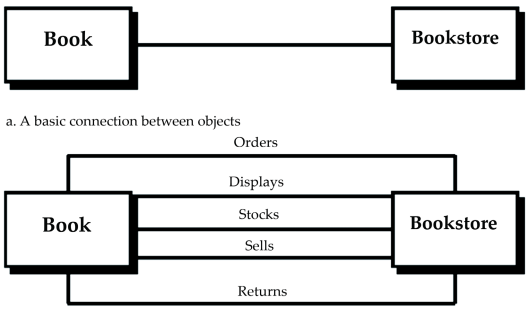


Figure 3.3: Relationships.

Relationship

Data object are connected to one another in a variety of different ways. Consider two data objects, book and bookstore. Those objects can be represented using the simple notation illustrated. A connection is established between book and bookstore because the two objects are related. But what are the relationships? To determine the answer, we must understand the role of books and bookstores within the context of the software to be built. We can define a set object relationship pairs that define the relevant relationships.

- A bookstore orders books
- A bookstore displays books
- A bookstore stocks books
- A bookstore sells books
- A bookstore returns books

The relationships orders, displays, stocks, sells, and returns define the relevant connections between book and bookstore. It is important to note that object relationship pairs are bi-directional; that is they can be read in either direction. A bookstore orders books or books are ordered by a bookstore.

Cardinality and Modality

The basic elements of data modeling data objects, attributes and relationships provide the basis for understanding the information domain of a problem. However additional information related to these basic elements must also be understood.

Cardinality

Cardinality is the specification of the number occurrence of one that can be related to the number of occurrences of anther (object). Cardinality is usually expressed as simply 'one 'or 'many' For example a husband can have only one wife while a parent can have many children. Taking into consideration all combination of 'one' and 'many' two (objects) can be related as

- One-to-one (1:1)—An occurrence of (object) 'A' can relate or one and only one occurrence of (object) 'B' can relate

to only one occurrence of 'A'. *For example*, a husband can have only one wife and a wife only one husband.

- One-to-many (1:N)—One occurrence of (object) 'A' can relate to more than one occurrences of (object) of 'B'; but an occurrence of 'B' can relate to only one occurrence of 'A'. *For example*, a mother can have many children, but a child can have only one mother.
- Many-to-many – An occurrence of (object) 'A' can relate to one or more occurrences of 'B'; while an occurrence of 'B' can relate to one or more occurrences of 'A'. *For example* an uncle can have many nephews, while a nephew can have many uncles.

Modality

The modality of a relationship is zero if there is no explicit need for the relationship to occur to the relationship is optional. To illustrate consider software that is used by a local telephone company to process requests for field service. A customer indicates that there is a problem. If the problem is diagnosed as relatively simple, a single repair action occurs. However if the problem is complex, multiple repair actions may be required.

Entity-Relationship Diagrams

The object-relationship pair is the cornerstone of the data model. These pairs can be represented graphically using the entity relationship diagram (ERD). The ERD was originally proposed by Peter Chen for the design of relational database systems and has been extended by others. A set of primary components is identified for the ER: data objects, attributes, relationships and various type indicators. The primary purpose of the ERD is to represent data objects and their relationships

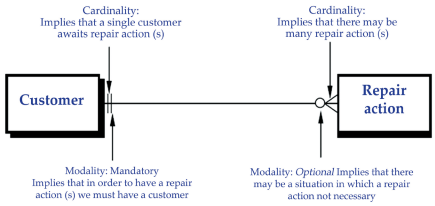


Figure 3.4: Cardinality and modality.

Data objects are represented by a labeled rectangle. Relationships are indicated with a labeled line connecting objects. In some variations of the ERD the connecting line contains a diamond that is labeled with the relationship connection between data objects and relationships are established using a variety of special symbols that indicate cardinality and modality.

FUNCTIONAL MODELING AND INFORMATION FLOW

Information is transformed as it flows through a computer based system. The system accepts input in a variety of forms; applies hardware, software and human elements to transform input into output and produces output in a variety of forms. Input may be a control signal transmitted by a transducer a series of numbers typed by a human operator a packet of information transmitted on a network link, or a voluminous data file retrieved form a CD-ROM (Compact Disc, read-only-memory), The transforms may comprise approach of an expert system. Output may light a single LED or produce a 200-pate report. In effect we can create a flow model for any computer-based system, regardless of size and complexity. Structured analysis began as an information flow modeling technique. A computer-based system is represented as an information transform. One or more inputs shown as labeled arrows originate from external entities represented as a box. The input drives the transform to produce output information (also represented as labeled arrows) that a is passed to the external entity. The key is to represent the information fed into and produced by the transform.

Data Flow Diagrams

Data flow diagram (DFD) represents the flows of data between different processes in a business. It is a graphical technique that depicts information flow and the transforms that are applied as data move form input to output. It provides a simple, intuitive method for describing business processes without focusing on the details of computer systems. DFDs are attractive technique because they provide what users do rather than what computers do.

As information moves through software it sis modified by a series of transformations. A data flow diagram is a graphical technique that depicts information flow and the transforms that are applied as data move form input to output. The basic form of a data flow diagram is illustrated in figure 3.5. The DFD is also known as a data flow graph or a bubble chart.

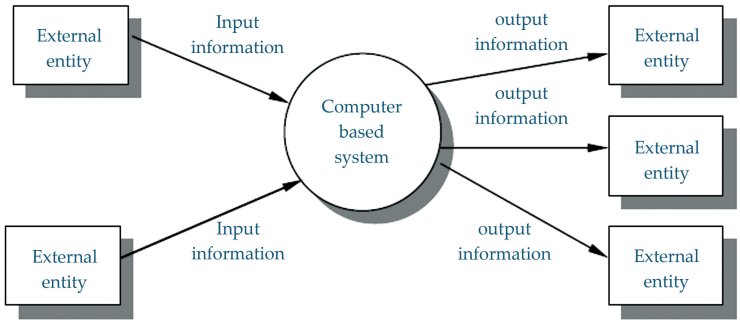


Figure 3.5: Information flow.

DFDs only involve four symbols.

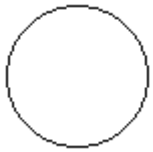
These are:

- Process
- Data object
- Data store
- External entity

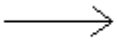
"The core of the person is what he or she loves, and that is bound up with what they worship - that insight recalibrates the radar for cultural analysis. The rituals and practices that form our loves spill out well beyond the sanctuary. Many secular liturgies are trying to get us to love some other kingdom and some other gods."

—Dallas Willard

Process: Transform of incoming data flow(s) to outgoing flow(s).



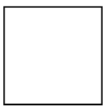
Data Flow: Movement of data in the system.



Data Store: Data repositories for data that are not moving. It may be as simple as a buffer or a queue or a s sophisticated as a relational database.



External Entity: Sources of destinations outside the specified system boundary.



The data flow diagram may be used to represent a system or software at any level of abstraction. In fact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Therefore the DFD provides a mechanism for functional modeling as well as information flow modeling. In so doing it satisfies the second operational analysis principle (i.e., creating a functional model). A level 0 DFD also called a fundamental system model or a context model represents the entire software element as a single bubble input and output data indicated by incoming outgoing arrows respectively.

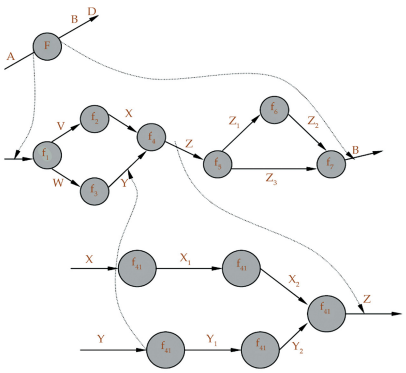


Figure 3.6: Information flow refinement.

Additional processes and information flow paths are represented as the level 0 DFD is partitioned to reveal more detail. *For example* a level 1 DFD might contain five or six bubbles with interconnecting arrows. Each of the processes represented at level 1 are sub functions of the overall system depicted in each context model.

Extensions for Real-Time Systems

Many software applications are time dependent and process as much or more control oriented information as data. Aircraft avionics manufacturing process control consumer products and industrial instrumentation are but a few of hundreds of real-time software applications.

Ward and Mellor Extensions

The system must monitor time-continuous information generated by some real world process. *For example* a real time test monitoring system for gas turbine engines might be required to monitor turbine speed, combustor temperature and a variety of pressure probes on a continuous basis.

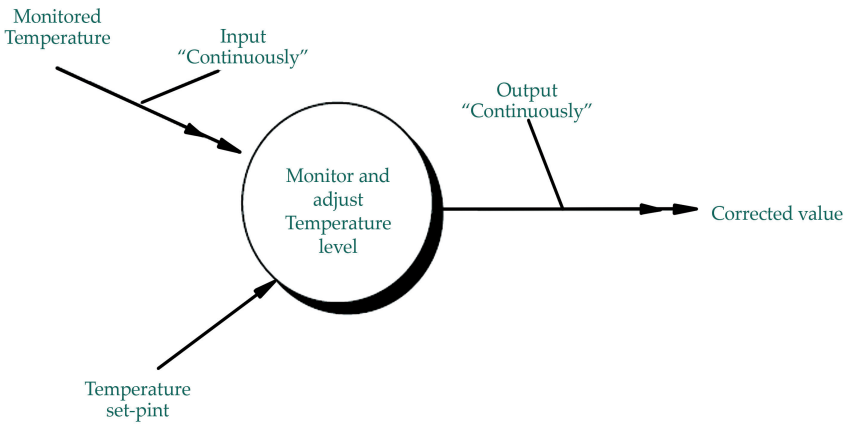


Figure 3.7: Time continuous data flow.

Conventional data flow notation does not make a distinction between discrete data and time continuous data. An extension to basic structured analysis notation, a mechanism for representing time-continuous flow, and a single headed arrow in used to

indicate discrete data flow. Monitored temperature is measured continuously while a single value for temperature set point is also provided.

The distinction ion between discrete and time-continuous data flow has important implication for both the system engineer and the software designer. During the creation of the system model, a system engineer will be better able to isolate those processes that may be performance critical (it is likely that the input and output of time-continuous data will be performance sensitive). As the physical or implementation model is created, the designer must establish a mechanism for collection of time-continuous data. Obviously, the digital system collects data in a quasi-continuous fashion using techniques such as high-speed polling. The notation indicates where analog to digital hardware will be required and which transforms are likely to demand high-performance software.

In conventional data flow diagrams, control or event flows are not represented explicitly. In fact, the analyst is cautioned to specifically exclude the representation of control flow from the data flow diagram. This exclusion is overly restrictive when real-time applications are considered and for this reason, a specialized notation for representing event flows and control processing has been developed. Continuing the convention established for data flow diagrams, data flow is represented using a solid arrow. Control flow however is represented using a dashed or shaded arrow. A process that handles only control flows, called a control process, is similarly represented using a dashed bubble.

Control flow can be input directly to a conventional process or into a control process. Figure 3.8 illustrates control flow and processing as it would be represented using Ward and Mellor notation. The figure illustrates a top-level view of data and control flow for a manufacturing cell. As components to be assembled by a robot are placed on fixtures, a status bit is set within a parts status buffer (a control store) that indicates the presence or absence of each component. Even information contained within the parts status buffer is passed as a bit string to a process, monitor fixture and operator interface. The process will read operator commands only when the control information, bit string, indicates that all fixtures contain components. An event flag, start/stop flag, is sent to robot initiation control, a

control process that enables further command processing. Other data flows occur as a consequence of the process activate event that is sent to process robot commands.

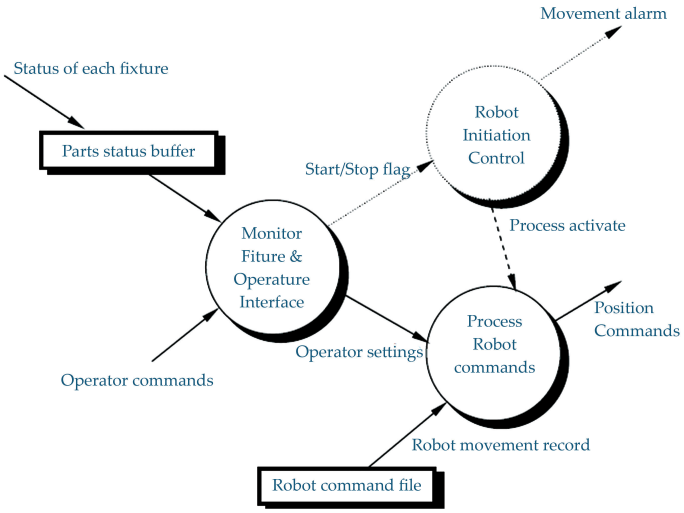


Figure 3.8: Data and control flows using ward and Mellor notation.

In some situations multiple instance of the same control or data transformation process may occur in a real-time system. This can occur in a multitasking environment when tasks are spawned as a result of internal processing or external events.

"Analysis does not set out to make pathological reactions impossible, but to give the patient's ego freedom to decide one way or another."

—Sigmund Freud

Hatley and Pirbhai Extensions

The Hatley and Pirbhai extensions to basic structured analysis notation focus less on the creation of additional graphical symbols and more on the representation and specification of the control oriented aspects of the software. Unlike ward and Mellor Hately and Pirbhai suggest that dashed and solid notation be represented separately. Therefore a control flow diagram (CFD) is defined. The CFD contains the same process as the DFD, but shows control flow rather than data flow. Instead of representing control process directly within the flow model, a notational reference

to a control specification is used. In essence the solid bar can be viewed as a "window "into an "executive" that controls the process represented in the DFD based on the event that is passed through the window.

Data flow diagrams are used to represent data and the processes that manipulate it. Control flow diagrams show how events flow among processes and illustrate those external events that cause various process to be activated.

The interrelationship between the process various processes and control models. The process model is connected to the control model through data conditions. The control model is connected to the process model through process activation information contained in the CSPEC.

A data condition occurs whenever data input to a process results in a control output. The process check and convert pressure implements the algorithm described in the PSPEC pseudo code shown. When the absolute tank pressure is greater than an allowable maximum. Note that when hatley and pirbhai notation is used, the data flow is shown as part of a DFD. While the control flow is noted separately as part of a control flow diagram. To determine what happens when this event occurs, we must check the CSPEC.

BEHAVIORAL MODELING

Behavioral modeling is an operational principle for all requirements analysis methods. Yet only extended versions of structured analysis provide a notation for this type of modeling. The state transition diagram represents the behavior of a system by depicting its states and the events that cause the system to change state.

In addition, the STD indicated what actions are taken as a consequence of a particular event. A state is any observable mode of behavior. *For example* states for a monitoring and control system for pressure vessel might be monitoring state, alarm state, pressure release state, and so on. Each on these states represents a mode of behavior of the system. A state transition diagram indicated how the system moves from state to state.

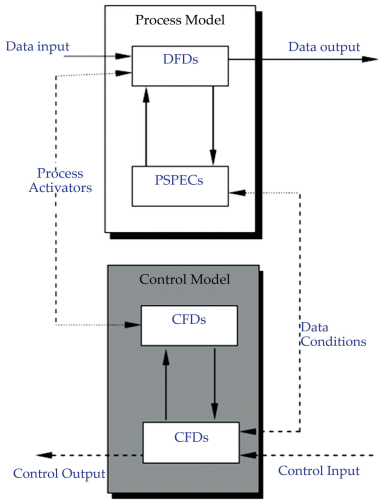


Figure 3.9: The relationship between data and control model.

To illustrate the use of the hatley and pirbhai control behavioral extension, consider software embedded within an office photocopying machine. The photocopier performs a number of functions that are implied by the level 1 DFD shown in figure 3.10. It should be noted that additional refinement of the data flow and definition of each data item would be required.

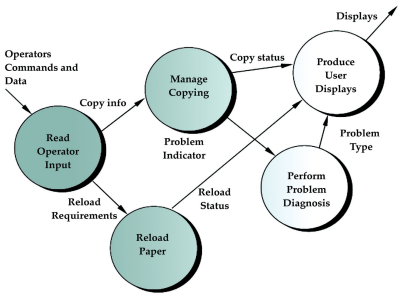


Figure 3.10: Level 1 CFD for photocopier Software.

The control flow for the photocopier software is shown entering and exiting individual process and the CSPEC "window". This implies that each of these events will cause some process represented in the CFD to be activated. If we were to examine the CSPEC internals, the start/stop event would be shown to activate/deactivate the manage copying process. Similarly, the jammed event (part of paper feed status) would activate perform problem diagnosis. It should be noted that all vertical bars within the CFD refer to the same CSPEC.

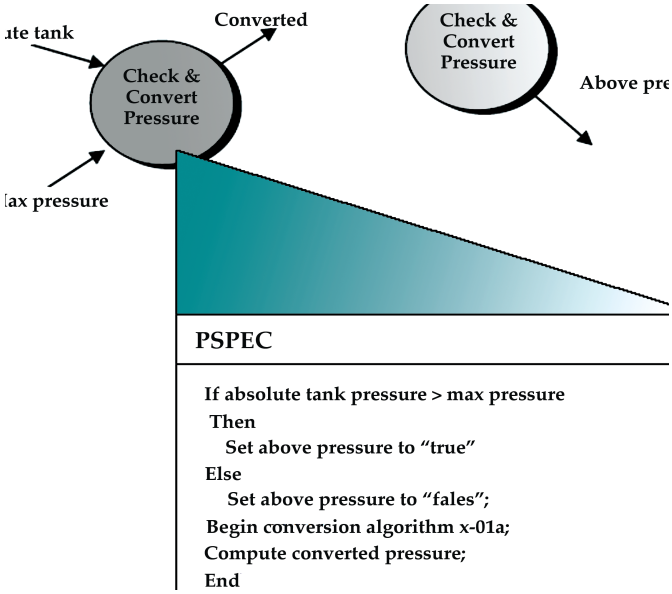


Figure 3.11: Data conditions.

An event flow can be input directly into a process as shown with reprofault. However, this flow does not activate the process, but rather provides control information for the process algorithm. Data flow arrows have been lightly shaded for illustrative purposes, but in reality they are not shown as part of a control flow diagram.

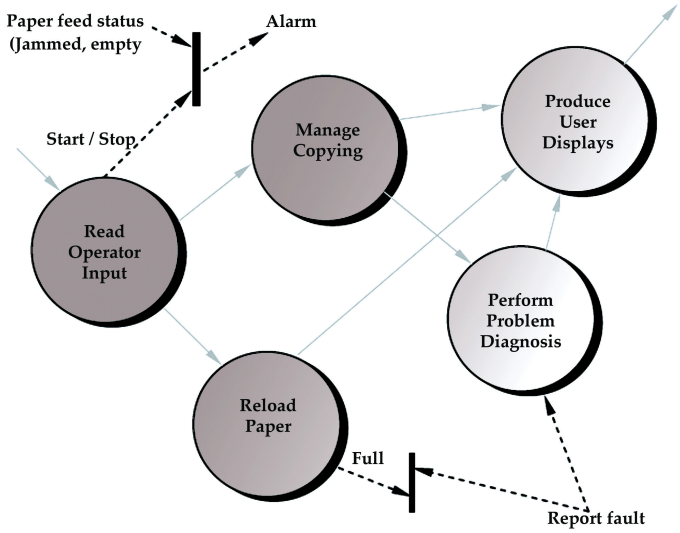


Figure 3.12: Level 1 CFD for photocopier software.

Creating an Entity Relationship Diagram

The entity relationship diagram enables a software engineer to fully specify the data objects that are input and output from a system, the attributes and define the properties of these objects, and the relationships between objects. Like most elements of the analysis model, the ERD is constructed in an iterative manner.

The following approach is taken;

- During requirements gathering, customers are asked to list the "things" that the application or business process addresses. These "things" evolve into a list of input and output data objects as well as external entities that produce or consume information.
- Taking the objects one at a time, the analyst and customer define whether or not a connection exists between the data object and other objects.
- Wherever a connection exists, the analyst and customer create one or more object relationship pairs.
- For each object relationship pair, cardinality and modality are explored.
- Step 2 through 4 are continued iteratively until all object relationship pairs have been defined. It is common to discover omissions as this process continues. New objects and relationships will invariably be added as the number of iterations grows.
- The attributes of each entity are defined.
- An entity relationship diagram is formalized and reviewed.
- Steps 1 through 7 are repeated until data modeling is complete

During installation the Safe Home control panel is used to 'program' and configure the system. Each sensor is assigned a number and type, a master password in programmed for arming and disarming the system, and telephone number(s) are input for dialing when a sensor event occurs.

When a sensor event is recognized, the software invokes an audible alarm attached to the system. After a delay time that is specified by the homeowner during system configuration

activities, the software dials a telephone number of a monitoring service, provides information about the location, reporting and the nature of the event that has been detected. The telephone number will be redialed every 20 seconds until telephone connection is obtained.

"The ideal of behaviorism is to eliminate coercion: to apply controls by changing the environment in such a way as to reinforce the kind of behavior that benefits everyone."

—B. F. Skinner

Discuss in between the analyst and the customer indicate the following list of things that are relevant to the problem.

- Homeowner
- Control panel
- Sensors
- Security system
- Monitoring service

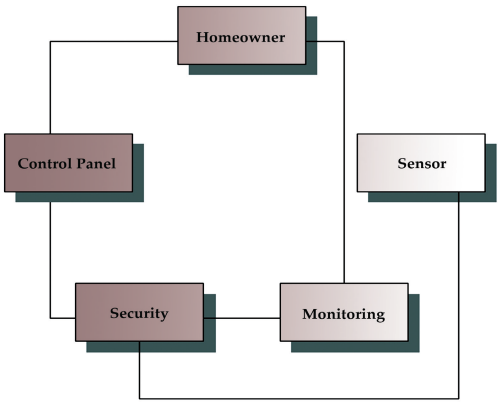


Figure 3.13: Establishing connections.

CREATING A DATA FLOW MODEL

The data flow diagram (DFD) enables the software engineer to develop models of the information domain and functional domain at the same time. As the DFD is refined into greater levels of detail, the analyst performs an implicit functional decomposition of the system thereby accomplishing the fourth operational analysis principle. At the same time, the DFD refinement results

in a corresponding refinement of data as it moves through the processes that embody the application.

A few simple guidelines can aid immeasurably during derivation of a data flow diagram;

- The level data flow diagram should depict the software/ system as a dingle bubble;
- Primary input and output should be carefully noted;
- Refinement should begin by isolating candidate processes data objects and stores to be represented at the next level;
- All arrows and bubbles should be labeled with meaningful names;
- Information flow continuity must be maintained from level to level; and
- One bubble data time should be refined.

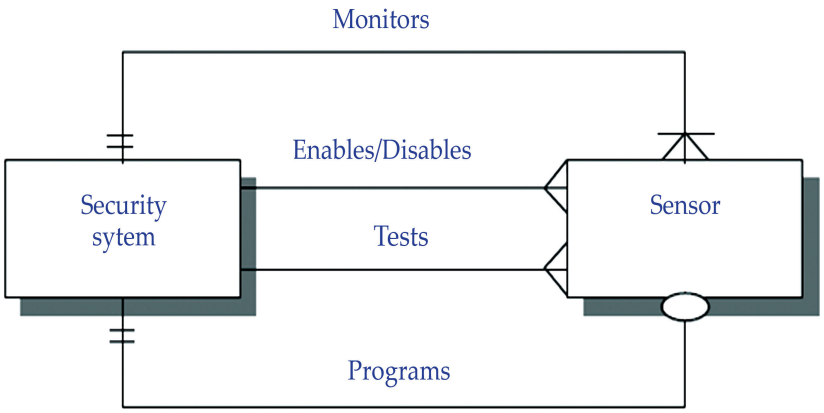


Figure 3.14: Developing relationships and cardinality / modality.

The primary external entities (boxes) produce information for use by the system and consume information generated by the system. The labeled arrows represent data objects or data object type hierarchies.

Creating a Control Flow Model

For many types of data processing applications the data model and the data flow diagram, are all that is necessary to obtain meaningful insight into software requirement. As we have already noted however there exists a large class of applications that

are driven by events rather than data that produce control information rather than reports or displays and that process information with heavy concern for time performance,. Such application require the use of control flow modeling in addition to data flow modeling.

The graphical notation required to create a control flow diagram (CFD) was presented. To review the approach for creating a CFED a data flow model is "stripped" of all data flow arrows10 Events and control items (dashed arrows) are then added to the diagram and a "window" (a vertical bar) into the control specification is shown.

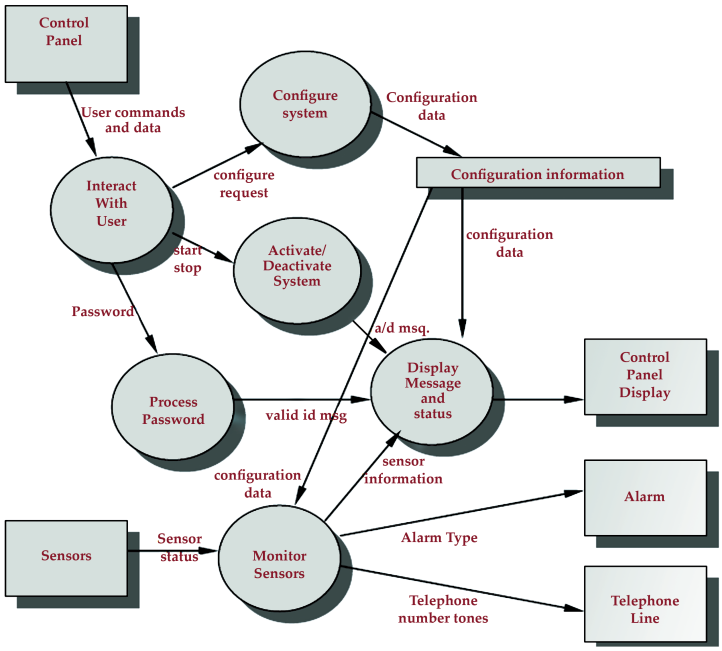


Figure 3.15: Level 1 DFD for safe home.

We have already noted that an event or control item is implemented as a Boolean value (*for example*, true or false, on or off, 1 or 0) or a discrete list of conditions (empty, jammed, full).

To select potential candidate events the following guidelines are suggested.

- List all sensors that are "read " by the software
- List all interrupt conditions
- List all "switches" that are actuated by an operator

- List all data conditions.
- Recalling the noun-verb parse that was applied to the processing narrative review all "control items" as possible CSPEC inputs/outputs.
- Describe the behavior of a system by identifying its states; identify how each state is reached and define the transitions between states.
- Focus on possible omissions a very common error in specifying control

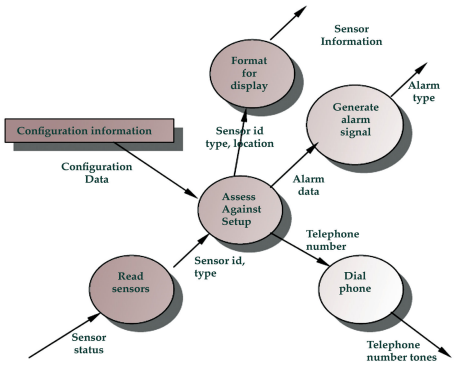


Figure 3.16: Level 2 DFD that refines the monitor sensors process.

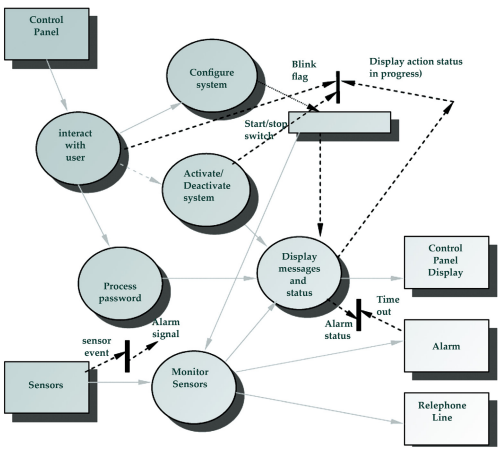


Figure 3.17: Level 1 CFD for safe home.

A level 1 CFD for Safe Home software is illustrated in Figure 3.17 Among the events and control items noted are sensor event (i.e., a sensor has been tripped), blink flag (a signal to blink the LCD display) and start/stop switch (a signal to turn the system

on or off) An event flowing into the CSPEC window from the outside world implies that the CSPEC will activate one or more of the processes shown in the CFD. When a control item emanates from a process and flows into the CSPEC window, control and activation of some other process or an outside entity is implied.

"A graphic representation of data abstracted from the banks of every computer in the human system. Unthinkable complexity. Lines of light ranged in the nonspace of the mind, clusters and constellations of data. Like city lights, receding."

—William Gibson

The Control Specification

The control specification (CSPEC) represents the behavior of the system (at the level from which it has been referenced) in two different ways. The CSPEC contains a state transition diagram (STD) that is a sequential specification of behavior It can also contain a process activation table (PAT) a combinatorial specification of behavior. The underlying attributes of the CSPEC were introduced. It is now time to consider an example of this important modeling notation for structured analysis.

Figure 3.18 depicts a state-transition diagram for the level 1 flow model for Safe Home. The labeled transition arrows indicate how the system responds to events as it traverses the four states defined at this level. By studying the STD a software engineer can determine the behavior of the system and more important can ascertain whether there are "holes" in the specified behavior.

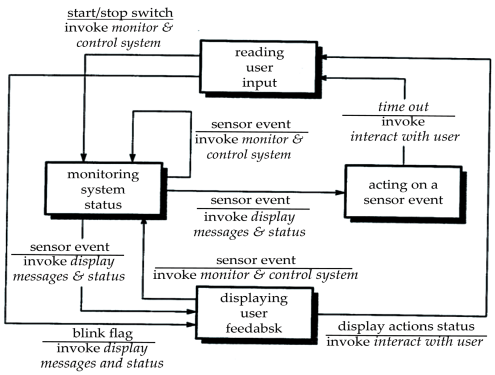


Figure 3.18: State transition diagram for safe home.

The process Specification

The process specification (PSPEC) is used to describe all flow model processes that appear at the final level of refinement. The content of the process specification can include narrative text a program design language (PDL) description of the process algorithm mathematical equations tables diagrams or charts By providing a PSPEC to accompany each bubble in flow model, the software engineer creates a "minispec" that can serve as a first step in the creation of the software requirements specification and as a guide for design of the program component that will implement the process.

DATA DICTIONARY

The analysis model encompasses representations of data objects function and control. In each representation data objects and / or control items play a role. Therefore it is necessary to provide an organized approach for representing the characteristics of each data objects and control item. This is accomplished with the data dictionary.

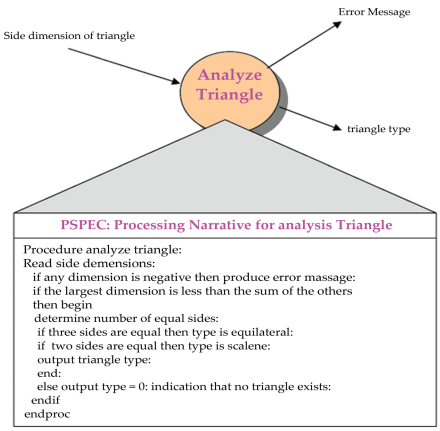


Figure 3.19: Process specification for a DFD process.

The data dictionary is an organized listing of all data elements that are pertinent to the system with precise rigorous definitions so that both user and system analyst will have a common understanding of inputs outputs components of stores and (even) intermediate calculations.

Today the data dictionary is almost always implemented as part of a CASE "structured analysis and design too;" Although the format of dictionaries varies from tool to tool most contain the following information:

- Name—the primary name of the data or control item, the data store or an external entity.
- Alias—other names used for the first entry
- Where-used /how-used—a listing of the processes that use the data or control item and how it is used (*for example*, input to the process output from the process ,as a store as an external entity)
- Content description—a notation for representing content
- Supplementary information—other information about data types preset values(if known) restrictions or limitations etc.,

SADT

Structured analysis and design technique (SADT) is a technique that has been widely used as annotation for system definition, process representations, software requirements analysis and system/software design SADT consists of procedures that allow the analyst to decompose software(or system) functions a graphical notation the SADT datagram that communicates the relationships of information (data and control) and function within software and project control guidelines for applying the methodology.

The SADT methodology encompasses automated tools to support analysis procedures and a well-defined organizational harness through which the tools are applied. Review and milestone are specified allowing validating of developer-customer communication.

Once a data object or control item name and its aliases are entered into the data dictionary consistency in naming can be enforced. That is if an analysis team member decides to name a newly derived data item xyz but xyz is already in the dictionary the CASE tool supporting the dictionary posts a warning to indicate duplicate names. This improves the consistency of the analysis model and helps to reduce errors.

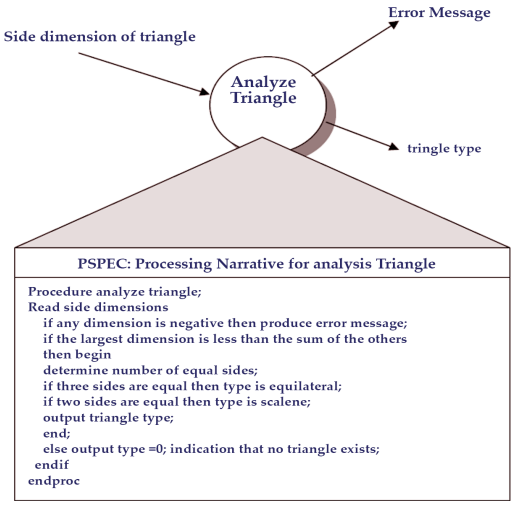


Figure 3.20: Process specification using PDL for a DFD process.

Where-used/how-used" information is recorded automatically from the flow models. When a dictionary entry is created, the CASE tool scans DFDs and CFDs to determine which processes use the data or control information and most important benefits for dictionary. During analysis there is an almost continuous stream of changes. For large projects it is often quite difficult to determine the impact of a changes. For large projects it is often quite difficult to determine the impact of a change.

KEY TERMS

Entity relationship diagram
Technical representation
Unified process analysis
Software engineer
Data objects
Operational analysis
Software transform
Behavioral domain
Software behavior

DISCUSSION QUESTIONS

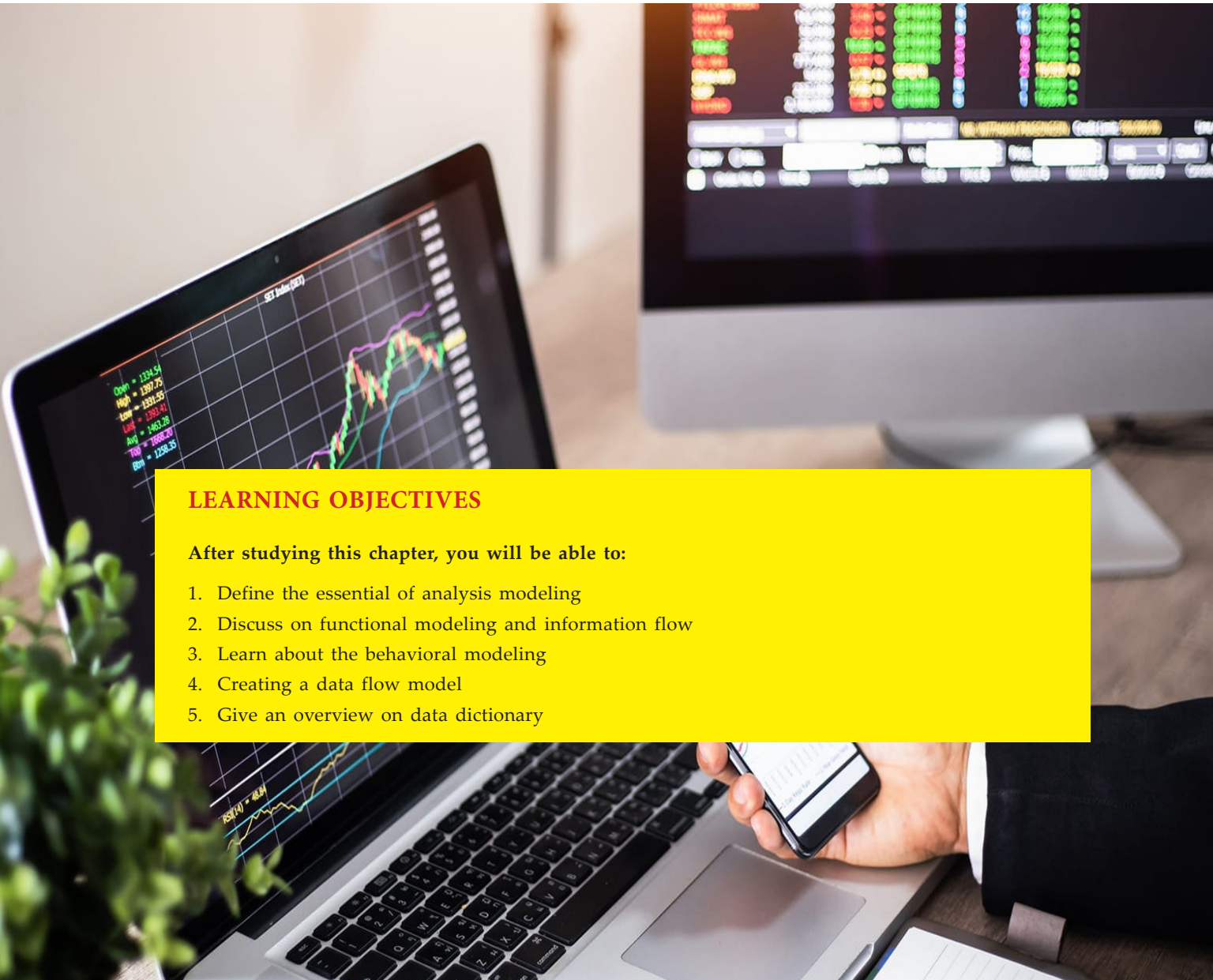
1. What is analysis modelling? What are its different elements? Explain.
2. Define attributes. Write its importance.
3. Write short notes on cardinality and modality.
4. Discuss about entity-relationship diagrams.
5. Explain data flow diagram.

REFERENCES

1. Galeon, J., 2013. System Analysis and Design: Systems Development Tools and Techniques. [online] Jessevimgaleon.blogspot.co.uk. Available at: <http://jessevimgaleon.blogspot.co.uk/2013/07/systems-development-tools-and-techniques.html> [Accessed 22 September 2015].
2. Jerome, Carlos. Building Better Business Systems with Scenarios. Software Development. June 2000 p. 56-60.
3. SearchSoftwareQuality. 2007. What is spiral model (spiral lifecycle model) ? - Definition from WhatIs.com. [online] Available at: <http://searchsoftwarequality.techtarget.com/definition/spiral-model> [Accessed 23 September 2015].
4. SearchSoftwareQuality. 2007. What is waterfall model? - Definition from WhatIs.com. [online] Available at: <http://searchsoftwarequality.techtarget.com/definition/waterfall-model> [Accessed 23 September 2015].

CHAPTER 4

BUSINESS PROCESS MODELING AND DATA ANALYTICS



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Define the essential of analysis modeling
2. Discuss on functional modeling and information flow
3. Learn about the behavioral modeling
4. Creating a data flow model
5. Give an overview on data dictionary

INTRODUCTION

Business process modeling (or) process modeling, is the analytical representation or put simply an illustration of an organization's business processes. Modeling processes is a critical component for effective business process management.

Process modeling software gives an analytical representation of 'as-is' processes in an organization and contrasts it with 'to-be' processes for making them more efficient.

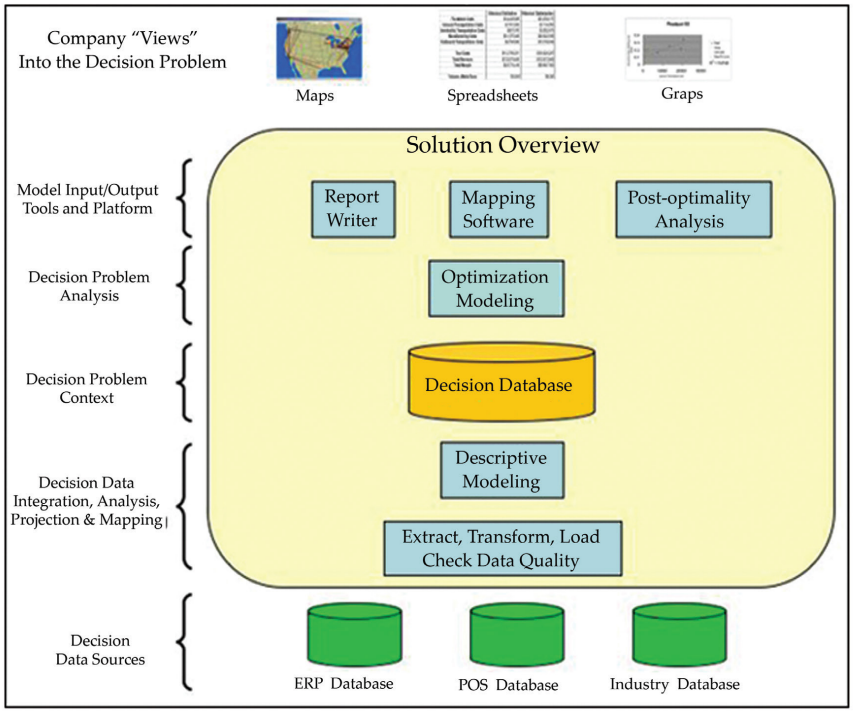
A business process model is a graphical representation of a business process or workflow and its related sub-processes. Process modeling generates comprehensive, quantitative activity diagrams and flowcharts containing critical insights into the functioning of a given process, including the following:

- Events and activities that occur within a workflow
- Who owns or initiates those events and activities
- Decision points and the different paths workflows can take based on their outcomes
- Devices involved in the process
- Timelines of the overall process and each step in the process
- Success and failure rates of the process

BUSINESS PROCESS MODELING TECHNIQUES

Business process modeling is mainly used to map a workflow so anyone can understand, analyze and make positive changes to that workflow or process. Usage of diagram helps you to visualize this process and make better decisions. Most of the legacy systems are outdated or used sparingly. However it must be noted that some of them are used far more effectively in other industries and for different purposes.

Your first step in modeling is actually pen and paper. However, to actually run a business process, you will need to digitize that process in a way that a workflow engine can understand.



Business process modeling software allow you to represent your process in a digital way that can then be transferred to a live automated process.

There are many benefits to business process modeling:

- Gives everyone a clear understanding of how the process works
- Provides consistency and controls the process
- Identifies and eliminates redundancies and inefficiencies
- Sets a clear starting and ending to the process

Business process modeling can also help you group similar processes together and anticipate how they should operate. The primary objective of business process modeling tools is to analyze how things are right now and simulate how they should be carried out to achieve better results.

Business Process Modeling Notation (BPMN)

Simply put BPMN is a graphical representation of your business process using standard objects. If you want to get more technical, it can also be defined as a set of graphical objects and rules

defining available connections between the objects.

BPMN consists of the following basic building blocks;

- Flow objects: events (circles), activities (rectangles with rounded corners), and gateways (diamonds)
- Connecting objects: mainly comprising arrows, these indicate sequence flow (filled arrows), message flow (dashed arrows), and associations
- Swim lanes: pools (graphic container) and lanes (sub-partition of the pool)
- Artifacts: data objects, groups, and annotations

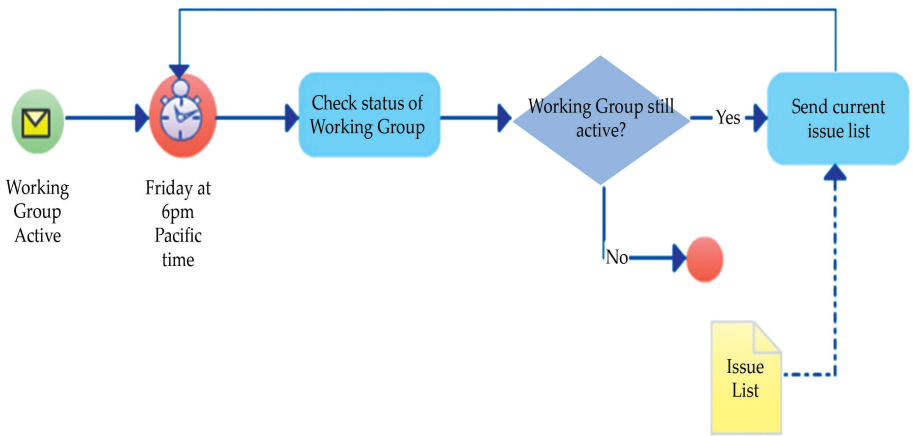


Figure 1: A business process modeled using BPMN.

Creately has a separate library with the latest BPMN 2.0 object included in the library. Plus you get intelligent grouping elements, professionally designed BPMN templates and much more. Check out how Creately makes it easier to model using BPMN.

The biggest advantage of BPMN is that it's a standard with well-defined syntax. So many business analysts are familiar with it which makes collaboration much easier. Also most modeling tools support BPMN which makes it much easier to share and edit if even using different software. All the together makes BPMN the most popular business process modeling technique at the moment.

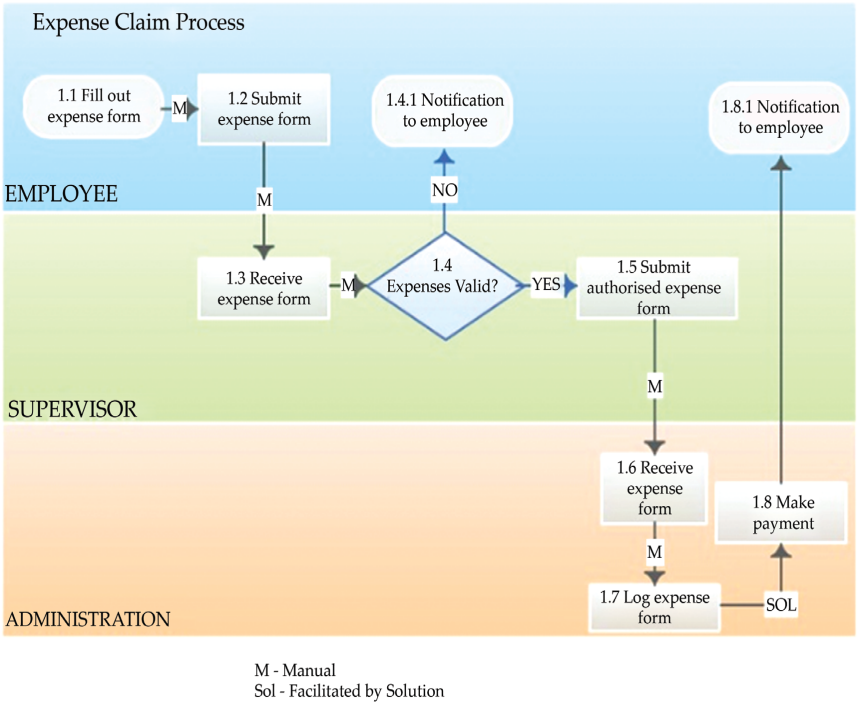


Figure 2: A process modeled using BPMN that has swim lanes

UML Diagrams

UML is a modelling language mainly used for specification, visualization, development and documenting of software systems. But business professionals have adapted it as a powerful business process modeling technique.

With 14 different UML diagram types it offers a flexible and powerful way to visualize almost any business process. They are typically used for modelling the detailed logic of a business process. In many ways UML diagrams are the object-oriented equivalent of flow charts.

As mentioned above one of its main advantages is its flexibility. But with 14 different diagram types some might find it difficult to understand the diagrams. Add to that the same process can be modeled using different UML diagrams. So probably not the most popular choice among analysts.

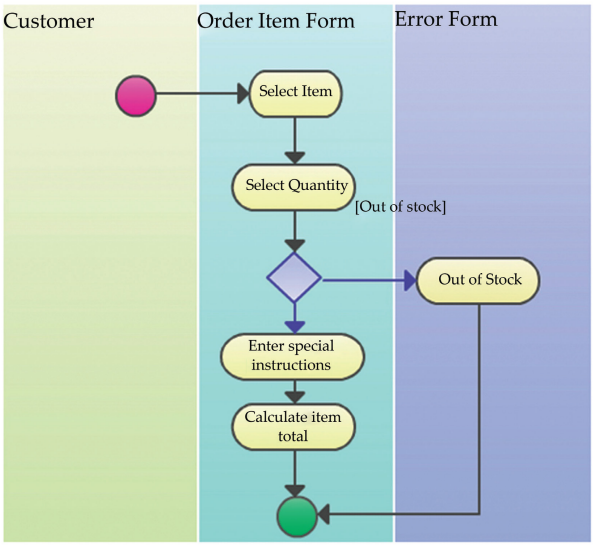


Figure 3: UML a popular business process modeling technique

The UML software supports drawing of all UML diagram types. We have templates, separate libraries for various UML diagrams and knowledge article to guide you through the process.

Business process modeling goes back a long time and BPMN and UML evolved from some techniques used a while back. However, some of these earlier techniques are still been used in business process modeling. Listed below are some of those techniques that are still used today.

Flowchart Technique

Flowcharts are probably the most popular diagram type in the world. Because it has few standard symbols it can be easily understood by many. Simplicity makes it powerful and an effective tool in fact BPMN can be considered as an advanced version of the basic flowchart technique. Also most drawing software support creation of flowcharts it is used by a much wider audience as well.

Flowchart uses a sequential flow of actions and does not support a breakdown of the activities. The Flow Chart model is possibly the first process notation. It has frequently been used over many years although there is no exact date for its origin.

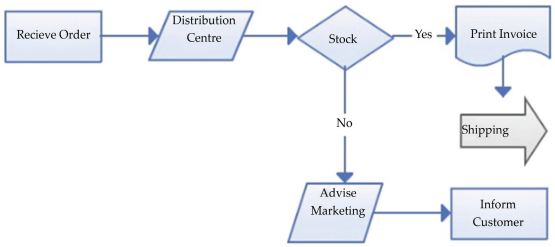


Figure 4: A simple flowchart with processes, decision blocks etc.

"All we are doing is looking at the timeline from the moment a customer gives us an order to the point we collect the cash. And we are reducing that timeline in the value stream by removing non value-added wastes."

—Taiichi Ohno

Data Flow Diagrams – Yourdon's Technique

Data flow diagrams (DFD) show the flow of data or information from one place to another. DFDs describe the processes showing how these processes link together through data stores and how the processes relate to the users and the outside world.

They are used to record the processes analyses as a part of the design documentation. A DFD can be seen as a method of organizing data from its raw state. DFDs are the backbone of structured analysis that was developed in the early sixties by Yourdon.

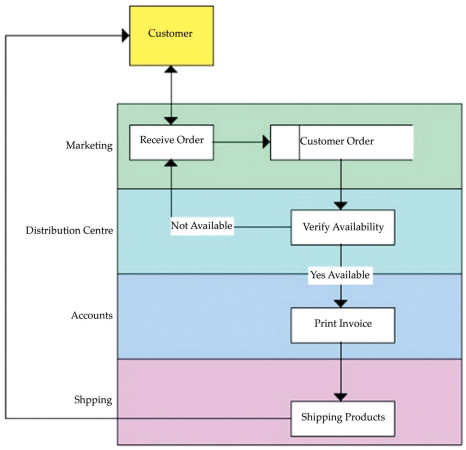


Figure 5: A DFD diagram used in modeling

Role Activity Diagrams – RAD

Roles are abstract notations of behavior describing a desired behavior within the organization. They are often organizational functions. They also include software systems, customers and suppliers. RADs provide a different perspective of the process and are particularly useful in supporting communication. They are intuitive to read, easy to understand and presents a detailed view of the process and permitting activities in parallel.

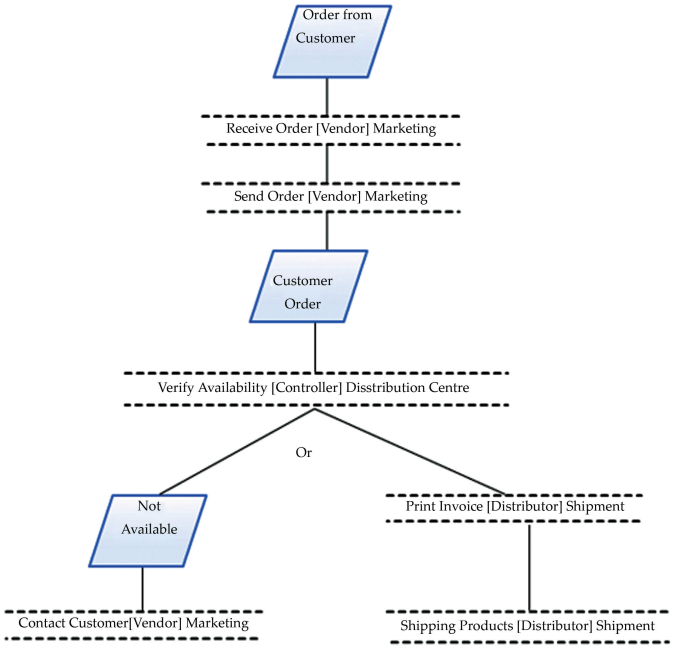


Figure 6: Role activity diagram used in business process modeling.

Role Interaction Diagrams – RID

Activities are connected to roles in a type of matrix. Activities are shown vertically on the left axis and the roles are shown horizontally at the top. Text and symbols are used together in order to represent the process.

Although slightly more complex than flow diagrams, RIDs are fairly intuitive to understand, easy to read but they tend to be messy, with many arrows pointing left and right and are therefore quite hard to build.

Inputs to, and outputs from the activities are not modeled. Therefore, important information is lost. RIDs are not as flexible as flowcharts, for example. They have quite rigid notation. But compared with other modelling techniques, RIDs are nevertheless flexible.

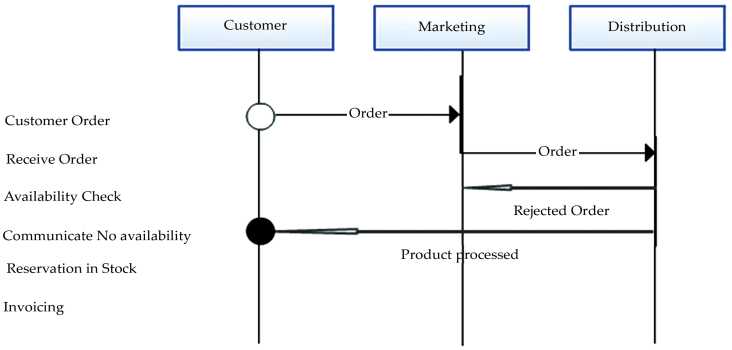


Figure 7: Role interaction diagrams offer flexibility but somewhat harder to draw.

Gantt Charts

Gantt charts relate a list of activities to a time scale. Although it can be used to represent a process graphically it's strength lies in the ability to monitor the current situation, project timeline and resource allocation. Although things like dependencies can be highlighted in it the modeling part is severely lacking. Probably the reason why it isn't popular anymore. If you're using to monitor timelines then you'll be pleased to know Creately supports drawing Gantt charts.

A Gantt chart is a matrix that lists on the vertical axis all the tasks or activities to be performed in a process. Each row contains a single activity identification, which usually consists of a number and a name. The horizontal axis is headed by columns indicating estimated activity duration, skill level needed to perform the activity, and the name of the person assigned to the activity, followed by one column for each period in the project's duration. Each period may be expressed in hours, days, weeks, months, and other time units. In some cases it may be necessary to label the period columns as period 1, period 2, and so on.

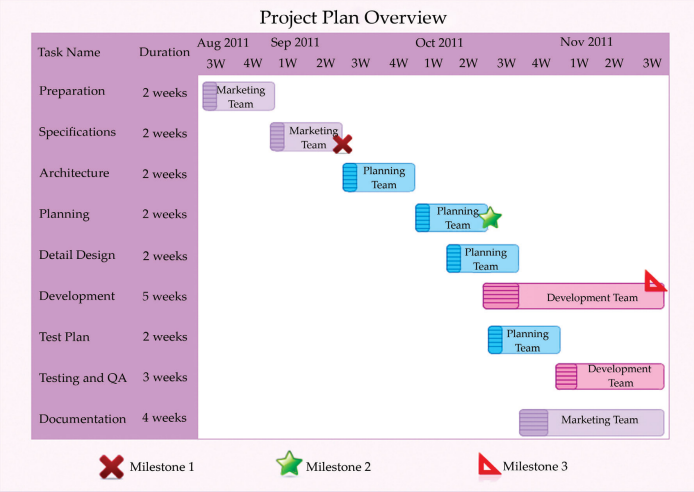


Figure 8: Gantt charts are used for timelines in processes

Integrated Definition for Function Modeling (IDEF)

IDEF is a family of methods that supports a paradigm capable of addressing the modelling needs of an enterprise and its business areas. The IDEF family is used according to different applications. The most important parts are: $IDEF_0$, $IDEF_1$, $IDEF_{1X}$, $IDEF_2$, $IDEF_3$, $IDEF_4$ and $IDEF_5$. However, for business process modelling, the most useful versions are $IDEF_0$ and $IDEF_3$.

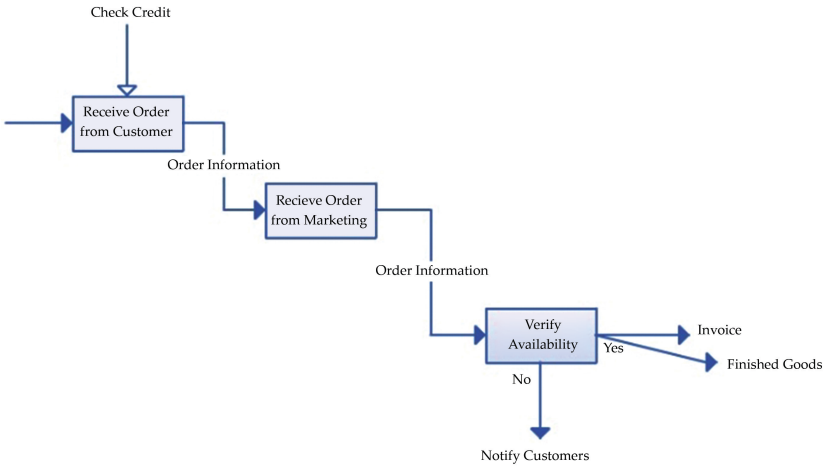


Figure 9: The IDEF model.

Colored Petri Nets (CPN)

Colored Petri nets are a graphical oriented language for design, specification, simulation and verification of systems. It is particularly well suited for systems that consist of a number of processes, which communicate and synchronize.

Colored nets are extended Petri nets in which symbols are differentiated by 'COLORS'. A CPN model consists of a set of modules which each contain a network of places, transitions and arcs. The graphical representation makes it easy to see the basic structure of a complex CPN model, i.e. to understand how the individual processes interact with each other. CP-nets have a formal, mathematical representation with a well-defined syntax and semantics. This representation is the foundation for the definition of the different behavioral properties and the analysis methods.

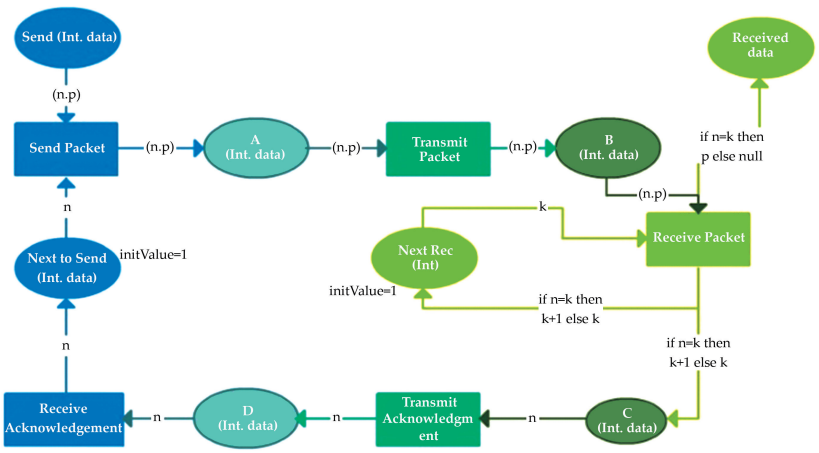


Figure 10: A diagram modeled using Colored Petri-Net

Object Oriented Methods

This method is based on three concepts: objects that represent a real-world entity. An object has a state, i.e. one of the possible conditions in which the object may exist represented by the values of the properties (attributes). State changes are reflected by the behavior, i.e. how an object acts and reacts determined by the set of operations the object can perform on itself, and also

knowing its interface, functions and methods. A set of similar objects is called class. Finally, messages are requests for the receiver objects to carry out the indicated method or behavior and return the result of that action to the sender objects. States change through behavior when the object receives a message. There are many different techniques based on OO. Unified Modelling Language (UML) is considered the standard OO modelling language. Coad and Yourdon's method precedes UML.

Workflow Technique

Workflow is a flow of tasks between computer applications or people in an organization. Two or more members of a workgroup to reach a common goal can define workflow as well as any task performed in series or in parallel. Work flow is more than a technique to model a process. It is a method to analyze and improve a process, including its modelling. The workflow development process uses work flow models to capture the relevant information of the processes. This process comprises four stages: Information Gathering, Business Process Modelling, Work flow Modelling, Implementation and Verification & Execution.

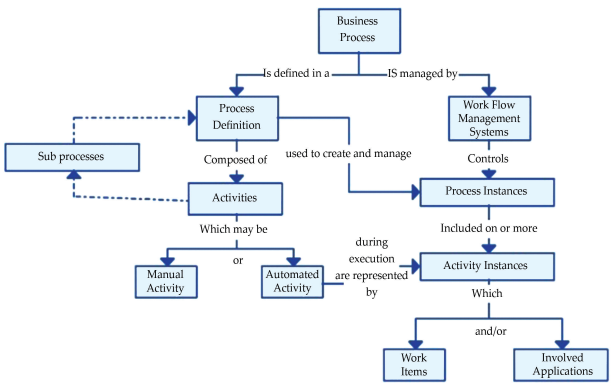


Figure 11: Work flow concept

Simulation

Simulation model comes in handy when you want to study a complex real-world system. You want to learn more about the system to make an informed decision but the complexity of the system prevents you from doing that directly.

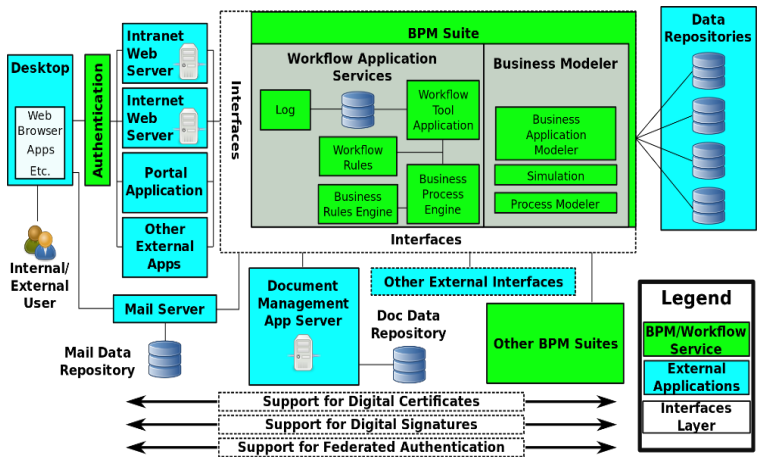
Therefore you proceed indirectly by creating and studying another entity (the simulation model), which is sufficiently similar to the real-world system. Simulation can have many forms (for example, discrete-event simulation, continuous simulation, system dynamics, Monte-Carlo simulation, qualitative simulation, etc.).

"The secret of getting ahead is getting started. The secret of getting started is breaking your complex overwhelming tasks into small manageable tasks, and then starting on the first one."

—Mark Twain

BUSINESS PROCESS MODELLING AND WORKFLOW MANAGEMENT

The terms "business processes" and "workflow management" have received a lot of attention. They are both used when talking about the automation of complex processes in office or industry. The term "business process modelling" is preferred when people concentrate on the analysis and design of processes, while "workflow management" is generally used when people also strive for supporting the execution of the formerly analyzed processes through information technology. In this paper, we propose a novel and integrated approach to the problem of workflow management. Up to now, existing proposals have concentrated either exclusively on one part (modelling or execution) of workflow management or have introduced some new language constructs for business process modelling.



The approach allows to reuse an established language for business process modelling and combines it with a verification and validation of the modelled processes and with a reliable execution of these processes on database-centered systems. Hence, we are able to integrate "legacy" business process descriptions into a workflow management system and to implement the modelled processes in a distributed environment. This evolutionary approach is a critical feature for the successful application of workflow management systems. The EPC business process modelling language we (re)use forms the basis for the description of the SAP R/3 system. Enterprises using the SAP system are forced to model their business processes using EPCs in order to customize SAP Ft/3 (Keller 1995). Hence, there already exists a lot of knowledge and a lot of business process descriptions in these enterprises. To implement the execution of workflows, we have to construct an engine which is capable of supporting the reliable execution of processes, handling multiple concurrently acting users, and which allows them to use a distributed system. The basic differences between the languages used for business process modelling and for the workflow engine are that they are used

- By different people (business process versus workflow designers)
- For different purposes (business administration versus workflow automation), and that
- The later one has to be executable.

Ideally, we should be able to implement a compiler from the higher level BP modelling language to the language of the workflow engine. However, this is not possible in the general case because at the execution level we need additional information about the information technology infrastructure which can only be supplied by external sources and which the business process modeler should not be aware of. Hence, we argue that the mapping of business processes to a workflow engine can only be done semi-automatically and will always require some interaction with a workflow designer supplying additional information.

Preliminaries

In order to be able to describe business process and workflow management issues, we need a common understanding of what

these and some related basic terms mean. Unfortunately, they are often used in the literature in different contexts with a different meaning. In this paper, we use the term business processes when we talk about high-level descriptions of organization's activities. These are used to describe market-oriented aspects like the satisfaction of customer needs. They received attention through business process reengineering, which is a method for the optimization of the business processes by explicit redesign. Through the analysis of business processes, you obtain information where an optimization of the processes is useful. Workflows describe the processes of an enterprise on a lower level with focus on the execution of the processes. Thus, workflows are a collection of tasks organized to accomplish some business process.



Workflows are divided in ad-hoc and administrative workflows. Ad-hoc workflows perform processes whose structure cannot be predicted. The order of activities forming ad-hoc process is defined only at runtime. These processes are usually supported by groupware systems. Administrative workflows have a predefined structure which is instantiated each time the workflow is performed. Thus, it is possible to support the process execution by informing the workflow participants about tasks to be done and by providing the information needed for the execution of a task. Workflow management involves the (re) design and the (re)implementation of workflows as the needs and the goals of an enterprise change. A workflow management system supports the computer based workflow management. That part of the WFMS providing the run time execution environment is called the workflow engine. This engine has to know the workflow specifications, the current instances of workflows, the information and organizational structure, and the available technology.

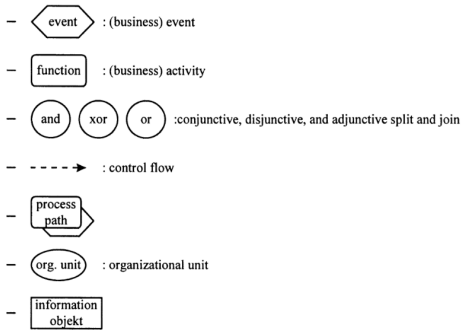
Workflow Design

In the area of business process modelling, most enterprises document their processes in an informal way only. This satisfies the need for documentation, but there is a big gap to workflow description languages if the computer based execution of the processes is required, too. Thus, if you want to support the business processes through a WFMS, you have to redefine the workflows from scratch and lose the connection to your previously described business processes. To prevent this, we follow another approach by reusing an established, but informal model for the modelling of business processes, which is afterwards formalized and extended to fulfil the requirements for the description of workflows. Depending on the detail of description, we talk about business processes and workflows, respectively. For both, we offer different methods for analyzing the models.

The EPC-model

The architecture of integrated information systems splits the total view of an enterprise into four different views: data view, function view, organization view, and process view. The process view integrates some parts of the other views because it describes the dynamics of an enterprise using information from the static views. This view documents the business processes of an enterprise. For the description of the static views, established models can be used, e.g. the ER-model for the data view. Because of the requirement to reflect organizational, functional, data oriented and dynamic aspects in the process view, the event-driven process chain model (EPCM) was developed.

The EPCM uses the following graphical symbols



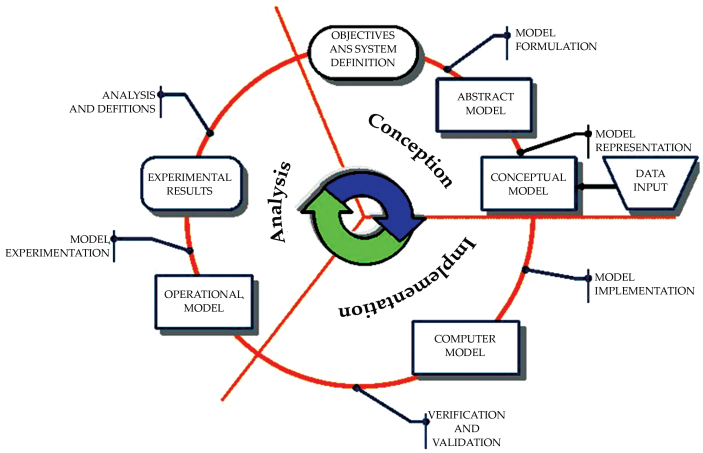
- *Event*: An event reflects a signal in a business environment which triggers the execution of a function. Thus, it is a passive element in the process.
- *Function*: A function reflects the active elements of a process in the EPC-model and describes an activity. It is executed either by a person or automatically. Linked with it is information describing which input data is used and which output data is produced. Furthermore, you can connect a function with an organizational unit to define the responsibility for the execution of the function.
- *Split/join*: A split or join describes the logic of the control flow. The AND-split executes parallel paths, which can be synchronized by an AND-join. Of an XOR-split exact one following activity is started. Thus, a synchronization of an X0R-join is not needed. The OR-split corresponds to a "don't care" split, thus all combinations of the following paths are possible. Depending on the specific case, an OR-join must synchronize the incoming paths.
- *Process path*: Process paths are connections to other EPCs to divide business processes into various EPCs.
- *Organizational unit*: This is connected with a function to define the responsibility for the execution.
- *Information object*: This describes which data is needed for the execution of a function and which data is produced by the function. The initial goal of the EPCM was to create a simple documentation of the business processes of an enterprise. It is used to inform the persons involved in the process about their tasks. An execution of the workflows based on the defined schemes is not intended. Thus, the EPCM contains some shortcomings, which must be removed to enable the execution of EPCs.

The weakest point of the EPC-model is the missing formal syntax and semantics. A formally defined syntax and semantics is not required for documentation of the processes, but is essential for analysis and execution.

Furthermore, the EPC-model does not offer a possibility to specify the data flow, the conditions of the control flow and the functions more detailed.

MODELING AND ANALYSIS USING DISCRETE-EVENT SIMULATION

Business process analysis is used by companies and organizations that want to improve their overall efficiency and operations. The end result of such analysis is typically some type of organizational restructuring or changes in the basic procedures and operations of the company. Various types of business process analysis procedures exist. One of these, discrete-event simulation, allows companies to examine large amounts of data and explore hypothetical scenarios to determine the best courses of action.



Method

Discrete-event simulation to conduct business process analysis generally involves the use of sophisticated software programs that require business analysts to provide large amounts of data to create an accurate analysis. Discrete-event simulation involves the analysis of various business practices or events that occur over the course of predetermined period of time. Various discrete events are included in the sequence to determine what the effects of each will be on the overall productivity and efficiency of the business. For instance, a company might simulate the hiring of new employees every year to determine the impact on the company's output. Each year, a different number of employees are hired and each hiring serves as a discrete event in its own right. The software-simulation program accounts for the changes as they occur.

Approaches

One of the difficulties in using business process modeling for simulation purposes is the proliferation of software programs that all claim to offer business owners and managers with the necessary tools to complete their analysis. Some of the software programs offer those conducting the analysis an opportunity to examine the business as a whole, on the macro level. Others are more narrowly focused in their approach and look at the various processes of the business from within, at the micro level. One major difficulty is sorting through the different approaches to find the one that best suits your business.

Advantages

The advantages to the business process modeling approach is that it allows the business owner and managers an opportunity to visualize what the outcome of various discrete events will be over the course of time. Some software programs use graphics and symbols in addition to providing raw data. These help to enhance the visual representation of data to the person or persons using the programs. One of the most important advantages to modeling software is the fact that it lets the business owner avoid sinking large amounts of money and resources into projects. Successful models, on the other hand, can be implemented based on the findings of the software.

Effectiveness

Business process modeling using discrete-event simulation has some promise to provide businesses with a tool for improvement. However, that tool is effective primarily in terms of quantifiable data and does not necessarily provide business owners and managers with a way of applying the data. For that reason, its effectiveness is primarily limited to its simulation of the impact of discrete events, while leaving the application to the user. The use of this type of modeling can help business owners from making mistakes and having to correct those mistakes once they occur. Overall, this process has potential as an effective analytical tool but not necessarily as a tool that provides a straightforward answer to specific problem

Business Process Integration

A business model, which may be considered an elaboration of a business process model, typically shows business data and business organizations as well as business processes. By showing business processes and their information flows a business model allows business stakeholders to define, understand, and validate their business enterprise.

The data model part of the business model shows how business information is stored, which is useful for developing software code.

Usually a business model is created after conducting an interview, which is part of the business analysis process. The interview consists of a facilitator asking a series of questions to extract information about the subject business process. The interviewer is referred to as a facilitator to emphasize that it is the participants, not the facilitator, who provide the business process information.

Although the facilitator should have some knowledge of the subject business process, but this is not as important as the mastery of a pragmatic and rigorous method interviewing business experts.

The method is important because for most enterprises a team of facilitators is needed to collect information across the enterprise, and the findings of all the interviewers must be compiled and integrated once completed.

Business models are developed as defining either the current state of the process, in which case the final product is called the "as is" snapshot model, or a concept of what the process should become, resulting in a "to be" model.

By comparing and contrasting "as is" and "to be" models the business analysts can determine if the existing business processes and information systems are sound and only need minor modifications, or if reengineering is required to correct problems or improve efficiency. Consequently, business process modeling and subsequent analysis can be used to fundamentally reshape the way an enterprise conducts its operations.

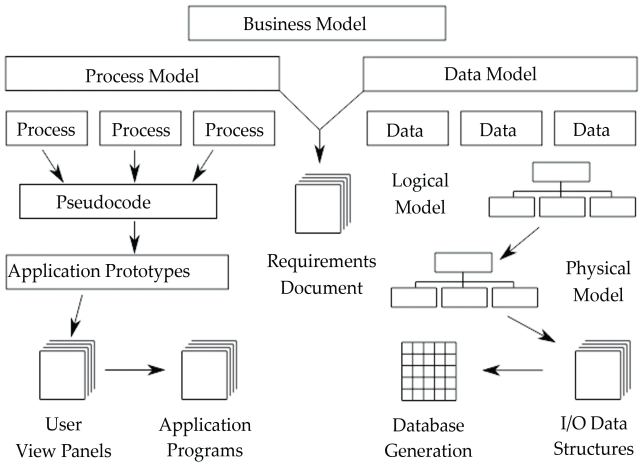


Figure 12: Example of the interaction between business process and data models.

"Quality is never an accident; it is always the result of high intention, sincere effort, intelligent direction and skillful execution; it represents the wise choice of many alternatives."

—William A. Foster

Business process integration, also known as BPI, is the strategy of finding ways to connect individual technology systems so that the flow of information between those systems is efficient and seamless. The idea of this type of integration is to eliminate the need to duplicate data in several different systems, and increase the overall efficiency of the business operation. When structured to best advantage, business process integration can save the business a great deal of time and money.

One example of business process integration involves establishing a relationship between the sales and billing systems within a company. In some cases, the connection is reciprocal, in that specific data within the sales database can be accessed from the billing system, and vice versa. This means that if the integration is configured with the right protocols, a salesperson can access current billing period figures for a given client by going to that client's profile in the sales database and executing a request. At the same time, a billing specialist could initiate a query from the billing system to the sales database and download information regarding the newest contract rates extended to a customer.

The concept of business process integration can aid in making sure that customer orders are processed, executed, and billed with relative ease, simply by establishing connections between all relevant systems. For example, if a customer schedules a conference call with a service provider, the details regarding that order are entered in a reservations department, and the information is downloaded to the conference bridge where the call will actually take place.

This means there is no need for the conference operator to enter the data manually, and possibly make an entry error that negatively impacts the conference call itself. Once the call is complete, the data for the call is automatically downloaded from the conference bridge to the billing system, where the correct rates are applied and the invoice automatically prepared by the system, without the need for a billing specialist to manually enter data from a printout of conference bridge activity.

By effectively using business process integration to establish connections between internal systems, businesses can often manage tasks with a greater degree of speed as well as accuracy. The ability to retrieve customer names and addresses for a mass mailing is simplified when an integrated system can combine a letter template with that contact information, download the created documents to a printer, while at the same time preparing an email listing that makes it possible to notify those same customers to expect the mailing in a few days. A project of this type can be managed by a single employee in a matter of a couple of hours, rather than requiring several employees to devote time to the project over one or two days.

Importance of Business Reference Model

Based on the functionality of an organization, a business reference model (BRM) depicts the organizational structure and operations of the business in a hierarchical manner. Typically, the reference model will demonstrate the functioning of operations, independent of division of labor in the organization that carries out the actual functions. Some parts of the business model, however, are designed specifically to make room for other aspects of the organization to include specific business processes and functions. The purpose of these emphases in the construction

of a BRM allows for analysis of those specific components of a business, such as information technology or data flow. In addition, such focus also allows for the alignment of new projects to specific functional areas of the business or organization.

Serving as a foundational conception, the business reference model allows for deeper understanding of how the business or organization functions. Designed purposefully for easy understanding, governments will typically use the model so citizens can better navigate how the government functions. Leveraging three main components to outline the model to include business areas, business lines and business capabilities, the objective is to demonstrate the operations and functions of the entire organization and its parts. Using the BRM, both internal and external actors can align initiatives with specific areas of the business or organization to gain a better understanding of parts potentially involved in that initiative.

With the outline of the business reference model, business areas will usually describe all activities and functions with which the business or organization is involved. Lines of business, on the other hand, will elaborate on specific functions within each individual business area. Business capabilities will expand on the lines of business to explain how the objectives of the lines of business are accomplished.

Aside from these core components to the BRM, the business areas section will also branch out to describe the common attributes often shared by each business area. For example, all components might share in the management of the organization's resources.

Covering the entire organization is the specific purpose of the business reference model. There are additional models meant to serve various components though, which are usually meant for internal use. Such models may include technical reference models, service reference models, or performance reference models.

Each model, however, ties into the business reference model, where the process for designing the target model begins. Therefore, once a business reference model is designed, all functional aspects tend to flow from and revert back to it.

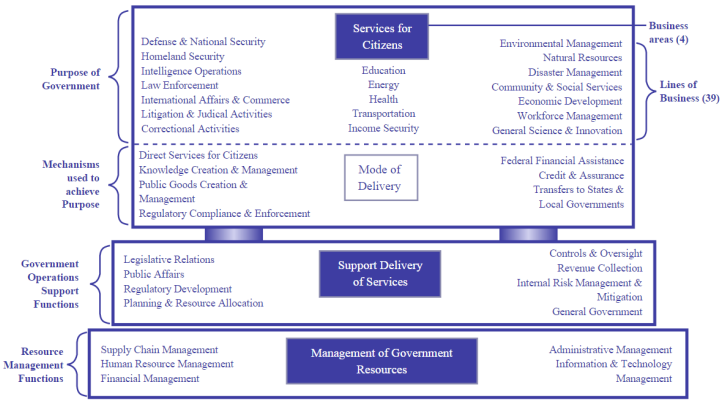


Figure 13: Example of the US Federal Government Business Reference Model.

A business reference model is a reference model, concentrating on the functional and organizational aspects of an enterprise, service organization or government agency. In general a reference model is a model of something that embodies the basic goal or idea of something and can then be looked at as a reference for various purposes. A business reference model is a means to describe the business operations of an organization, independent of the organizational structure that perform them. Other types of business reference model can also depict the relationship between the business processes, business functions, and the business area's business reference model. These reference models can be constructed in layers, and offer a foundation for the analysis of service components, technology, data, and performance.

The most familiar business reference model is the Business Reference Model of the US federal government. That model is a function-driven framework for describing the business operations of the federal government independent of the agencies that perform them.

Business Process Reengineering

Business process reengineering is an approach to the positioning of a business enterprise by essentially redesigning the structure of the business from the ground up. This radical approach seeks to interpret the standard business model in a new way, making more efficient use of available resources by seeing the function and purpose of those resources in new ways. The working

structure for Business Process Reengineering is examined in great deal in a 1993 work entitled Reengineering. These process allow businesses to depart from using a time-honored but possibly no longer timely model, and build something new and radical. This approach did not necessarily call for the complete abandonment of all aspects of the standard business model. However, the approach did call for redefining each component in the model and altering the function in a manner that would produce a business structure relevant for a new age.

Over the years, Business Process Reengineering has been known by many different titles. In some instances, the process is known simply as BPR. At other times, the approach has been called Business Process Redesign, Business Transformation, and Business Process Change Management. All these titles do speak to the foundational tenet of the process, in that the idea is to free a business from following the same old structure, simply because that is the way it has always been done. Instead, Business Process Reengineering, under all its different names, supports tearing down the business structure to the foundation and building it anew. One of the main tools that Business Process Reengineering identifies as an agent for change is the technology of the new millennium. This means that much of the computer technology that became readily available to even small businesses during the 1990s would help business owners to rethink how to structure their businesses. Some examples of how this has proven to be true include shared databases, communication networks that allow real-time interaction with multiple company sites, and wireless devices that allow work to take place outside the office.

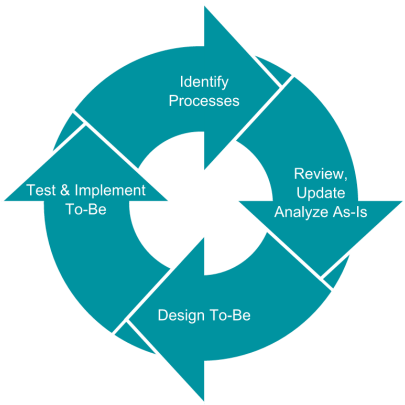


Figure 14: Business process reengineering cycle.

Business process reengineering (BPR) aims to improve the efficiency and effectiveness of the processes that exist within and across organizations. It examines business processes from a "clean slate" perspective to determine how best to construct them. Business process reengineering (BPR) began as a private sector technique to help organizations fundamentally rethink how they do their work. A key stimulus for reengineering has been the development and deployment of sophisticated information systems and networks. Leading organizations use this technology to support innovative business processes, rather than refining current ways of doing work.

Activity Flow–Oriented Approaches

Activity flow–oriented approaches are the most commonly used in business process redesign. Activity flow–oriented representations are also frequently referred to as "workflow" representations of business processes. Business process redesign approaches that build on activity flow representations are often referred to as workflow-oriented approaches to business process redesign. Because of their popularity, many different activity flow–oriented business process representation conventions exist, each with its own set of standard symbols. A particularly popular set of business process representation conventions is that discussed by J. Harrington in his book Business Process Improvement. There, several different activity flow diagrams are discussed, among which is one of the most characteristic of the activity flow orientation in business process modeling: the functional flowchart.

Activity flow diagrams are aimed at clearly showing the chronological and sequential order of execution of the activities that make up a business process. Human beings tend to perceive business processes as representations of "action" in a physical sense. Therefore, activity flow diagrams are often perceived as being among the easiest to generate and understand, and certainly easier to generate and understand than diagrams placing emphasis on communication flow elements. A recent doctoral dissertation by Azim Danesh drives this point home, providing some empirical evidence in support of this perception. The dissertation is titled IT-Enabled Process Redesign: Using Communication Flow Optimization Theory in an Information

Intensive Environment and was completed at Temple University. This perception is not really caused by a higher complexity, in terms of number of symbols used in the diagram, of activity flow diagrams. The most likely reason for the perception is that activity flow diagrams are better aligned with the way human beings envision "action," particularly in the physical sense. The problem with this perception is that most business processes operate on intangible elements, particularly information, and also have intangible elements as their outputs.

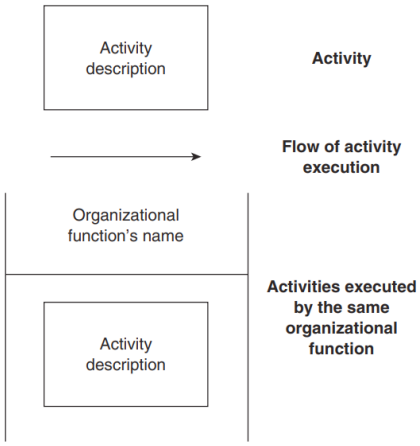


Figure 5.1. Activity Flow Representation Symbols.

"The range of what we think and do is limited by what we fail to notice. And because we fail to notice that we fail to notice there is little we can do to change until we notice how failing to notice shapes our thoughts and deeds."

—R. D. Laing

Among activity flow–oriented business process representations, one of the most popular is the functional flowchart. Representing business processes through functional flowcharts generally involves using the set of symbols shown in Figure 5.1 to show how activities are carried out chronologically. The serial sequence of activities is highlighted in the representation, and so is parallelism among activities, when sets of serially linked activities are carried out at the same time and more or less independently. In Figure 5.1, a rectangular shape represents an activity. Arrows indicate the flow of execution of activities. As mentioned before, when seen as a group, several arrows connecting a set of activities also serve to indicate the

chronological order of execution of the set of activities. This is a distinguishing characteristic of activity flow–oriented business process representations that serves to differentiate them from other types of representations of business processes (e.g., data flow diagrams, discussed later in this chapter). In Figure 5.1, the organizational function that executes an activity or set of activities is described at the top of a column containing the activity or activities. In addition to organizational functions (e.g., warehouse manager, quality control inspector), activities may also be executed by groups of organizational functions (e.g., warehousing department, quality control committee) or organizations external to the organization that houses the business process (e.g., a supplier or customer organization).

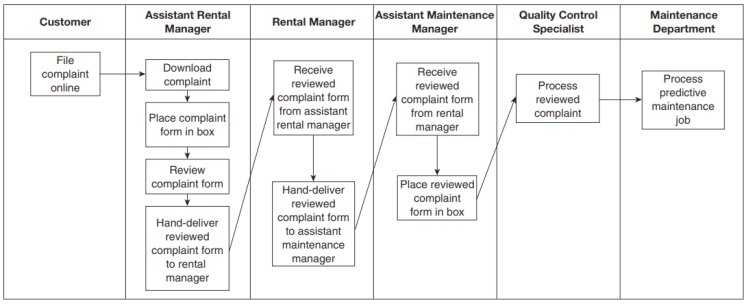


Figure 5.2: Activity Flow Representation Example.

The aforementioned symbols are used in Figure 5.2 to represent the business process whose example was discussed earlier, which refers to a car rental company's predictive maintenance process. It is important to stress a point that will become clear as we progress through this chapter: While the diagram in Figure 5.2 may be seen as a good representation of how communication takes place in the predictive maintenance process, arguably that is not the case.

In fact, one of the key omissions of activity flow diagrams is that they do not provide much detail about business process data, neither in terms of flow nor in terms of storage. This makes an activity flow diagram a somewhat incomplete representation of the car rental company's predictive maintenance process, especially when compared with the communication flow–oriented representation approaches discussed later in this chapter. Nevertheless, many would probably view the activity flow representation shown in Figure 5.2 as a fairly clear and intuitively

appealing representation of the car rental company's predictive maintenance process. Activity flow–oriented representations of business processes, such as the one shown in Figure 5.2, are among the most popular in business process redesign circles, even though their use poses some serious obstacles to business process redesign practitioners. For example, they show practically no detail in connection with the web of communication interactions that make up most business processes today. And, as previously demonstrated in this book, most business process problems seem to stem from communication configurations that are suboptimal. In the past, some adaptations of activity flow–oriented representations have been proposed with the goal of allowing for the representation of communication flow–related elements, such as data flows. However, because of the different nature of activity flow representations, those adaptations can sometimes be a little confusing from a business process representation standpoint. For example, one could try to represent the flow of data by associating it with the arrows shown in an activity flow representation (e.g., the one in Figure 5.2). However, strictly speaking, data do not flow from one activity to another. They flow from one data source to a data destination, for example, from one organizational function to another. Because of that, representing data as flowing between activities may be somewhat misleading when the time comes for business process redesign.

There are a number of variations of activity flow representations similar to the one shown in Figure 5.2. The workflow in Figure 5.2 is itself based on an adaptation of the American National Standards Institute's (ANSI) standard flowchart and has been extensively used in projects involving business process redesign groups. Flowchart variations are the block diagram, functional timeline flowchart, and geographic flowchart; see, for example, Harrington's book Business Process Improvement and the follow-up book Business Process Improvement Workbook, by Harrington and colleagues, for a more detailed discussion of those flowchart variations. It has been argued that the use of activity flow representations of business processes is generally less advisable than the use of communication flow representations, particularly when the goal is business process redesign. However, there is at least one notable exception to this rule. Activity flow representations are probably better than communication flow

3G E-LEARNING

representations when the business processes being modeled are manufacturing processes. This is particularly true for business processes that involve the assembly of a complex product (e.g., a car engine) based on a number of parts that are sequentially added to the unfinished product in an assembly line (until it becomes a finished product, of course). In the vast majority of other types of business processes, however, it is reasonable to argue that communication flow representations provide a better and more complete view of the elements that should be considered in a business process when the process is being redesigned.

ROLE OF DATA ANALYTICS IN MANUFACTURING

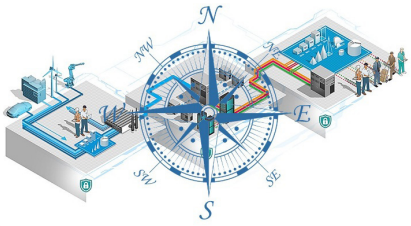
Today's manufacturing executives face a new landscape, with broad implications for profitability. From raw material supply constraints to the increasing number and complexity of production activities involved in the manufacturing process, manufacturers could benefit from a more granular approach to diagnosing and correcting process issues.

As products are usually core to a manufacturing business, data and analytics is often targeted at refining product development. However, more and more manufacturers are applying data analytics across their processes to optimize their supply chains, improve product scheduling and sales forecasting, reduce costs, develop new propositions and monitor machine usage and reliability.

The importance of Data Analytics has grown multiple fold across industries in the past few years because of a highly dynamic market. However, not all industries are making the most of it. Manufacturers have for unknown reasons shied away from analytics. As big data and analytics converge, the speed and scale of analytics changes drastically, moving beyond dashboards to intuitive action based indicators. As a result, better insights are uncovered, allowing organizations to make more accurate predictions that can lead to a competitive advantage.

Demand Planning is usually done using excel in most organizations. This process typically involves big tables of data. To understand the demand and requirement of a product, you

need to look into its historical data. Analytics with an order history of over a year or more can have 1000's of records. The results can be extracted using excel, but it will take up hours and hours of drilling into data to point out outliers. Technology now provides us ways to do this job in a more efficient and smart way.



Data Analytics and the use of analytical tools have helped prevent situations such as supply chain bottlenecks. The on time sharing of manufacturing information across an organization must be looked at from a strategic point of view for companies to get the best benefit.

Manufacturers have not yet automated data integration and analytical operations to the optimum extent in order to generate desired results. Although most organizations know that successful Analytics implementation can provide an edge in the market, the management of data is the primary area that concerns them. Business analytics can provide a clearer picture of operations, faster alerts regarding product tracking, better insights into market demand and performance.

In order to realize the advantages of business analytics', manufacturing companies must overcome certain obstacles.

Managing data is a huge task and most companies do not have the capability to gather all the necessary data from multiple divisions within the organization as they are usually lodged in different systems. To start off manufacturers must understand the business need for investing in a BI and Data Analytics solution. It will help them identify the necessary tools that need to be deployed in order to realize the full value of their business.

Below are some of the main issues faced by companies that prevent them from using analytics to its full potential:

- Data and more Data: Manufacturing units generate gigabytes of data that also has a lot of variety. The data flow into a manufacturing system is ever increasing be it information about quality parameters, process trends, maintenance, spares, inventory and a lot more.
- Data in, but no value out: Companies that collect data are unable to understand and analyze it. Businesses need to get insights into the data that is in their systems to gain insights into their business.
- Multiple systems that capture information: Companies in the process of growing adopt many systems to take care of their growing need. This leads to a complex scenario where data is available but is fragmented, this make it further difficult for companies to get value out of their data.
- Multiple Units, different locations: Consolidation and comparison of data becomes a time-consuming and manual process because when a large manufacturer has multiple units out at different locations, each of them acts as an individual business and define their data and rules differently.

Manufacturing Analytics gives companies the potential to operate with increased efficiency, improve business visibility, and support effective decision-making. Value has to be extracted from the massive amounts of information within any organization. As more and more information is collected thanks to newer technologies, the question of how to obtain value from this information remains.

The data analytics offerings for manufacturing, include:

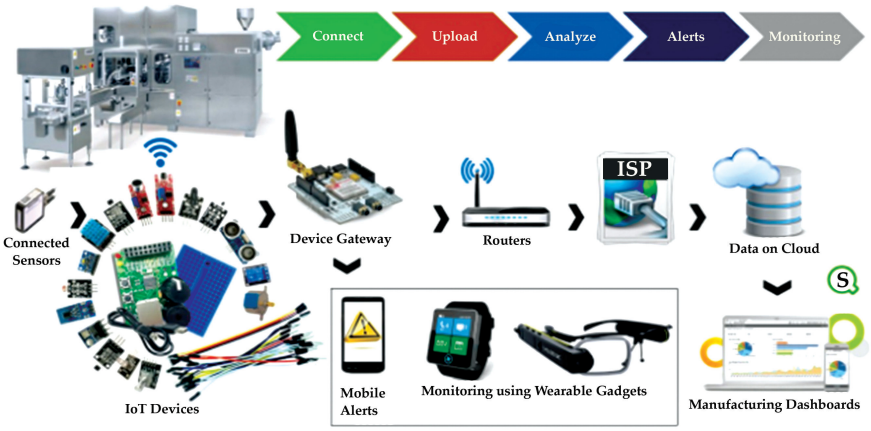
- Customer Analytics: It helps build a 360 degree view of the customer and aims to leverage consumer sentiments and usage patterns to develop better products, cross sell, reduce customer churn, drive value added services and earn better ROI on sales and marketing spend
- Manufacturing Analytics: It helps improve product quality and yield, reduces product test cycle time and improves first pass yield during new product rollouts. This ensures improved product availability and avoids losses due to stock outs

- Supply Chain Analytics: It integrates data across the value chain to provide critical information and visibility in demand-driven supply networks. Demand sensing and forecasting improves forecast accuracy even as channel fusion is taking place between brick and mortar retail and online stores, and product cycles are getting shorter and shorter. Analytics ensures that the right product is available at the right place at the right time to maximize revenue and profit
- Spend Analytics: It pinpoints inefficient spending practices and enables you better price negotiation capabilities that take advantage of volume discounts
- Service Analytics: It helps monitor and track after sales and service related processes to generate insights into spare parts supply management, proactive predictive maintenances and the discovery of new service revenue opportunities
- Financial Analytics: It enables organizations to streamline and integrate financial planning and budgeting processes, helps them close their books faster, improves visibility and predictability of business outcomes
- Data & Information Management: It manages enterprise-wide data flows including Master Data Management (MDM), data migration from legacy to ERP to Big Data platforms as part of transformational engagements, data quality assessment and cleansing and data archival. This is a range of solutions meant to address the varying needs of businesses
- Channel Spend Optimization: It enables promotion managers to model, optimize, forecast, budget, execute and manage customer promotions driven by the retail channel. It also enables you to measure product and customer performance within a country or across the globe.

Big Data Analytics in Manufacturing

Big Data Analytics in manufacturing is about more than the amount of data, Big Data analytics in manufacturing the following way: Big Data Analytics in manufacturing is about using a common data model to combine structured business system

data like inventory transactions and financial transactions with structured operational system data like alarms, process parameters, and quality events, with unstructured internal and external data like customer, supplier, Web, and machine data to uncover new insights through advanced analytical tools.



This definition of Big Data Analytics differs from the traditional approach most manufacturers and vendors have taken to dealing with manufacturing data. In most cases, manufacturers have invested heavily in data collection and visibility, mainly through legacy MES, EMI, and Data Historians.

In particular, EMI has largely been understood as a two-fold integration and dashboard tool where many vendors have invested heavily in both proprietary and open integration with **ERP** and Automation systems as well as in dashboard and mobile technologies to bring metrics to decision makers when and where they need the right information.

Therefore, EMI offerings today need to transform in three distinct ways to be truly considered Big Data Analytics in Manufacturing

- Move out of the MOM space to be able to move data from anywhere to anywhere with open technologies
- Invest in a data model that can handle structured and unstructured data from anywhere in the system architecture stack inside or outside the four walls of the factory

- Apply new analytical tools to this new data model to enable never before possible insights. Examples of these analytical tools would be: Image, Video, Geospatial, Time Series, Predictive Modeling, Machine Learning, Optimization, Simulation, and Statistical Process Control.

"Big data is at the foundation of all of the megatrends that are happening today, from social to mobile to the cloud to gaming."

—Chris Lynch

Big Data: How it is Affecting Manufacturing

Big data needs software that is capable of handling large amounts of information. This is something that at least 17% of manufacturers are taking full of advantage of. Big data analytics has given manufacturers an opportunity to serve clients in a more effective way. This is possible through data extracted from sensors that are implanted in the products they produce. The results have included after-sales services like proactive maintenance, which is rendered to clients and has proved to be invaluable.

It is clear that the use of big data and capable software is already giving to manufacturers an edge, which is necessary to compete in the global economy. Experts have also noted that the combination of big data and a manufacturer's ability to be flexible can actually boost productivity by at least 30%. Not only that, but it can also decrease cost for product development and assembly cost by up to 50%.

Experts also noted that serious implementation of big data could also decrease 7% of manufacturer's total capital costs. This definitely will give manufacturing great flexibility and the ability to improve other aspects of production. Big data is also having a great impact on the way manufacturers produce a specific item. Take, for example, a vehicle that needs to go through several steps before a manufacturing company begins assembly work. A particular model needs to produce several types of prototypes to test the design. Prototypes take time, effort, and money to make. Big data has been effectively helping some automobile manufacturing companies to reduce the need for prototypes.



Big data has helped car manufacturers perform simulated performance on a particular design before spending more time or resources on the vehicle. Volvo, in particular, started using big data to see if a particular feature or design will be accepted by consumers.

Merck–a vaccine production company–used big data to conduct 15 billion calculations and 5.5 million batch-to-batch comparisons. Through this analysis, they were able to create a more efficient vaccine without the need to invest in detailed product testing. There is an obvious trend that many manufacturers are noticing, which is big data analytics' ability to cut costs and make manufacturing more efficient. This could be the drive that is encouraging many manufacturers to use big data to convert their companies into a lean manufacturing production model.

Manufacturers can use big data to forecast demands well in advance, which could help 20 to 30% of their inventory. A manufacturer can also expect to increase its fill rate by three to seven points. Of course, manufacturers are still adapting to big data analytics and software. Some may have a bit of an issue identifying what data would be relevant to their companies and what data is not that useful. There may also be a problem in regards to asking the right kinds of questions. But these issues will be clearer the more big data analytics is implemented. The change that big data and software is having on manufacturing is an exciting step in American ingenuity, and one that will hopefully take the country into the next era of economic strength.

Big Data Examples in the Manufacturing Industry

Big data isn't a software application, technology tool or something you install. Sure there are big data tools but putting technology

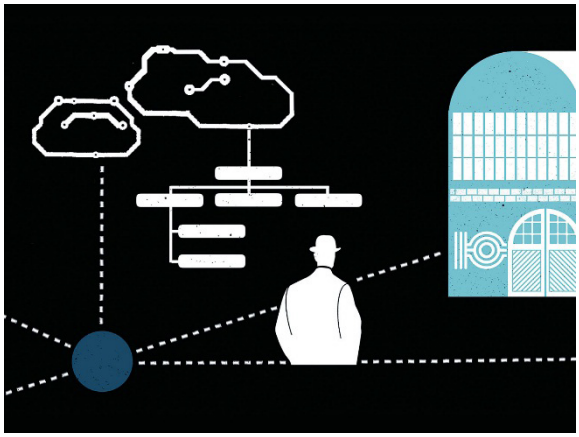
ahead of strategy is a recipe for disappointing results. Instead begin with strategy, and to spur your strategy you'll need some creating thinking. Here some examples to stimulate that thinking in a way that can dovetail into designing a big data strategy.

Improved Demand Generation. Small improvements in demand gen and product forecasting models mean fewer stock outs, less idle inventory, decreased cash flow requirements and most importantly, satisfying more customers that want your product – when and where they want it.

However, manufacturing demand gen models that only look at purchase history are of course not considering fluid market conditions and real-time customer demand. Big data offers a unique opportunity to look forward when estimating product demand.

For example, record label EMI has turned their business around by shifting demand generation and product forecasting from tenured executives applying gut feel experience to big data analysts tapping into listening patterns on online music streaming services to understand which music is resonating by region and with which subcultures. They are also teaming with music web-based and mobile apps providers, online "second screen" collators and music aggregators to acquire the data and create models which show consumption trends and measure demand in real-time. Sometimes called mood mapping, manufacturers and retailers can buy or acquire consumer reactions which correlate to market demand.

Manufacturers and retailers can also use social listening tools or acquire feeds from social networks to measure consumer feedback volume, reach and sentiment for their products. With live data in hand, they can then interrogate the data by geography, consumer profile and other characteristics that permit modeled interventions. The 24 by 7 constant streams of opinions, likes, dislikes, sharing and social propagation offer manufacturers and retailers a real-time audience to test variations in promotion, pricing elasticity, content, packaging and other variables in a way that correlates testing conditions with consumption and builds a predictable demand generation model.



More New Product Successes. Introducing new products is a high risk venture for manufacturers. Combining online ideation and big data reduces this risk, accelerates production innovation, lowers R&D expense and grows revenues by delivering products that customers most want.

Using online ideation to engage consumers for their ideas and critique the ideas of others, manufacturers are able to test R&D concepts faster and with substantially larger samples of customer feedback. Real-time access to large volumes of customer opinions also allows one-off test conditions and accelerates successive iterations so that new product innovation cycles can be substantially reduced. When consumer feedback can be acquired in minutes instead of weeks, manufacturers are afforded more testing scenarios and valuable What-If analysis.

Again referencing the music industry, music labels routinely arrange for leaked songs in order to test market acceptance and determine whether the launch of a new tune is ready for prime time or needs some more refinement. While this concept has always been replicable to other industries, the addition of big data allows manufacturers to tap into social media and other online channels for real-time analysis, increase the data population for improved confidence levels and integrate unstructured data in order to bring measurability to new types of information. These collective capabilities accelerate market analysis, enable measurement specificity by key variables (geography, demographics, consumer segment, etc.) and help manufacturers deliver new products which are more eagerly embraced by customers.

More Reliable Supply Chain. Supply chain interruptions are costly. It only takes one downstream supply chain partner to stumble and thereby impact revenues and damage the manufacturer's relationships with retailers and customers. To mitigate this complex threat, some innovative manufacturers are using big data to cull information that may suggest a supply chain partner is at risk.

Manufacturers have long used structured data such as on-time deliveries, payment cycles or the use of factored receivables to gauge partner health. However, when also including semi-structured or unstructured data from external and online sources such as published financial analyst recommendations, media reviews, filed litigation or various forms of insolvency protection to name a few manufacturers can be afforded valuable time for advanced planning and contingency measures when a partner faces financial distress.

This technique can also be used to avert public relations nightmares brought on by partners using illegal or culturally unacceptable business practices. *For example*, consider recent consumer backlashes directed toward garment makers who (allegedly) unknowingly had partners using child labor or dreadful wages in distant lands. In the tech industry, consider Apple's missteps by not identifying early or responding timely to the Foxconn fiasco born from subpar labor conditions in China. Big data offers a new opportunity to keep close tabs on your suppliers and creditors.

CASE STUDY

QANTAS AIRLINES: BUSINESS MODEL AND STRATEGIES

The Aviation Industry has been one of the most dynamic industries in our history. From the development of the machines to the formation of a viable business model, the trials and road to success has been filled with little success and numerous examples of failure. One of the best examples of success in this volatile industry would be Qantas Airline. For nearly one hundred years, Qantas is one of the world's oldest, most successful air carrier operations. Invoking a business model that has proven constant transformation and change is inherent to their success, Qantas has established itself as a global provider of commercial air services. The Qantas story is of success and adaptation to the ever-changing dynamics of the aviation industry. Many airlines have come and gone throughout the history of aviation, yet Qantas has endured, this case study will explain how they achieved sustainability when so many others have failed.

Business Model and Strategies

Qantas Airlines, throughout its 100-year history has continuously changed and evolved its business model to suit current needs and have an unprecedented preparedness for future operations. The airline is continuously upgrading its fleet and has not shied away from bold fleet changes. For example, QANTAS had at one time in their history been the world's only operator with an all Boeing 747 fleet conducting international world-wide flights. QANTAS is currently branded as QANTAS – for international and domestic services, and JetStar, for Australian domestic services and has a mixed fleet of Boeing 747's and Airbus models including the A320, A330, and the A380.

The ability to reinvent themselves and remain a world class provider of international flights services is because of the ever-developing business model and the company's willingness to make change. These transformation programs are inherent to the successful business model and why QANTAS has survived such a volatile aviation industry.

Leigh Clifford, Chairman for QANTAS Airlines has stated in the company's annual performance report for FY15/16 that "The Group has increased net free cash flow, grown return on invested capital and further strengthened its balance sheet, remaining in an optimal capital position throughout 2015/16. I'm especially pleased that we have been able to return more than $1 billion U.S. dollars in cash to shareholders over the past 12 months. Over the same period, earnings per share have almost doubled to reach 49 cents." (2016 -QANTAS Performance review)

Embracing transformation has been the mantra of QANTAS since its infancy. This mentality and ability to changes is fortified through the strong leadership of the company. CEO Alan Joyce exhibits the traits of the QANTAS pioneers. He stated in his 2016 address on performance that "Transformation is making Qantas' cost base competitive. Just as importantly, it's changed the way we work. We're a far more agile company. We've accelerated our adoption of new technology, digital platforms and data analytics. And we've fostered a culture that encourages diversity, inclusion and innovation. Qantas is a very different company from just a few years ago. Together with our well-balanced Group strategy, transformation means we're resilient enough to perform in all market conditions and outperform many of our peers."

The Qantas "Group" is the totality of all the endeavors, including QANTAS International and Domestic, JetStar Domestic, QANTAS Loyalty, and both domestic and international cargo operations. The expenditures total enormous amount of outgoing cash. The aviation industry is basically in the business of selling seats, QANTAS sells the entire experience and profits for its efforts. Expenses for catering, fuel, and maintenance are all costs associated and take away from that seat sale. The reduction of these costs are top priorities to the financial leadership and QANTAS has succeeded. A total reduction in 3% of the total fuel costs over one years is outstanding. With newer, more fuel-efficient aircraft like the A380, seat capacity and sales exceed the fuel costs and profits are made, if seats are filled. QANTAS as increased passenger revenue steadily for the past two years along with the reduced fuel costs and prices of oil world-wide. The QANTAS team has been able to not only turn a profit in an industry where most others attempt to "break-even", they have made huge sums of cash. The company is thriving, the stockholders receiving a return on their investment, and employees reinvesting in the company. They seem to be doing everything right in all aspects of the business. The international division of Qantas recorded $512 million dollars in profits FY 2016 and Qantas Domestic did even better with a record $578 million. These profits are in addition to the $452 million and the $346 million recorded by JetStar and Qantas Loyalty.

As stated in the Annual Performance Report for FY 2016, "Operating cash flows of $2.8 billion saw a strong increase from the prior year, reflecting cost and revenue benefits realized through the Qantas Transformation Program, and lower Australian fuel prices. When adjusted for the principal portion of operating rental payments, Funds from Operations were $3.1 billion. Net capital expenditure of $1 billion included investment in replacement fleet such as the Boeing 787 for Jetstar International and customer experience initiatives including airport lounges and the continuation of Airbus A330 and Boeing 737 cabin reconfigurations. Qantas generated $1.7 billion of net free cash flow in the period facilitating net debt reduction and returns to shareholders of $1 billion in the financial year." (2016 – QANTAS Performance Report)

What this all means for the financial stability, is a record underlying profit of $1.53 billion dollars for FY15/16. For QANTAS that is a 57% increase since FY 2014 and is expected to increase over the next fiscal year. In addition to record profits, a debt reduction plan has been set in motion. Expenses and operating costs continue to rise, yet the aggressive approach to reducing fuel costs has also paid its dividend for the airline. Fuel consumption costs have been reduced by 3% in just one year.

The Chairman, CEO, and Board of Directors statements are best summarized using the introduction published in the Company's 2016 Annual Performance Report, it reads as "This exceptional performance reflects the strength of our Qantas Group strategy, with record results and increased margins for Qantas Domestic, Qantas International, the Jetstar Group and Qantas Loyalty, and Group-wide return on invested capital of 23 per cent. Total underlying earnings before interest and tax (EBIT) in the domestic market – across both Qantas and Jetstar – rose 30 per cent to $820 million, and total underlying EBIT from the Group's international operations was $722 million, up 107 per cent." (2016 Qantas-Performance Report)

The modern age aviation company template lies in the metrics and performance standards of QANTAS airline. Remarkably in an era of failing and defuncted air carriers, they have managed to not just make a few dollars, but made record profits in a world-wide recession. They have accomplished this through two Australian domestic air carriers offering the same destinations and different services. One with traditional services and the other a low cost no frills carrier. This has given name recognition and brand loyalty throughout the population. QANTAS has also established their brand as the industry leader in long haul international travel around the globe. Top tier services, impeccable safety records, and a command of the passenger experiences has made their brand world renowned and associated with aviation excellence. Industry acclaims and record setting financial performances are the benchmark for quality and profits. Every airline, both domestic and international carriers, should take a hard look at the metrics and performance standards established and follow their lead. It does go to show that a deregulated airline can make a profit and not just strive to meet the break-even mark. QANTAS and their transformation mentality that has proven time and again to work. The company will continue to thrive and be the example and standard bearer for all air carriers. The transformation process into the next iteration of aviation excellence will continue as their gold standard and operations way of being an aviation leader.

Fleet Management and Route Structure

The Qantas fleet structure is one of the most dynamic in the industry. Historically Qantas is known for frequent and dramatic changes to their fleet mix, type, and

designs. At one point Qantas was the world's only sole service provider using the Boeing 747. A bold move that met the demands of the time and the decision served them well. According to Alan Joyce, the companies continued transformation is an ongoing process and will continue to evolve as the market and demands change. On February 23, 2017 Mr. Joyce briefed the company's first quarter performance and highlighted the importance of his commitment to transformation. "Three years ago, when we launched our $2 billion Qantas Transformation program, we wanted to make the Qantas Group a strong, sustainable business. A business that could generate returns through the cycle – in good times, but also in tougher conditions. Our first half performance shows that we continue to deliver on that promise."

The current Qantas fleet is a diverse mix of airplane capabilities solely produced by the Boeing Corporation and the Airbus Corporation. Having limited themselves to two manufactures, repair costs, maintenance, and product support per model has a reduced overall cost. These consolidated and reduced costs allow Qantas to procure and operate the higher end qualities that come in these models. These are not bare bones transport airplanes like South West. Their fleet is a top tier luxury airline passenger experience. The overall average age of their fleet is approximately 10.8 years, which is young in aviation terms. They are also one of the first airlines to offer services in the Airbus A-380 and Boeing 747-800 which are current production models. They are the front runners at providing services with these monumental aircraft.

Their oldest models are 737-300's with an average age of 30 years amongst that type. Qantas also does not lease their airframes, they are the owner/operators of their fleet. The constant upgrading and new purchases keep them as a world leader in international services.

Qantas has established routes from Australia to the rest of the world, including destinations like New York, Los Angles, Singapore, Japan, and China. These routine and scheduled routes meet current demands as well as do the regularly scheduled routes operated domestically by Qantas and JetStar. The future schedule is much harder to predict and Qantas is always on the watch for new routes and new service locations. Both international brands, Qantas and JetStar are being marketed to emerging markets around the world. Alan Joyce provides his insight into the international markets and the future routinely. Recently he was quoted "Where Qantas International has added capacity, it's been directed to Asian markets where demand is strong. Asia was also a bright spot for Jetstar. Jetstar's international operations had a strong result off the back of growing demand between Australia and places like Indonesia and Thailand. And the Jetstar airlines in Japan and Singapore improved their performance again. The Jetstar Group overall reported another record half year profit, helped by a strong domestic performance." These emerging international

destinations are being addressed and services and fleet adjustments are being made to service these destinations.

The Qantas scheduling situation is unique due to the many destinations serviced and what airframes service certain places. There are passengers that want to fly on the A380 for the novelty, not just the destination. Qantas has recognized this and promotes these flights on this special airplane. The scheduling and marketing of these flights aids in the filling these enormous airplanes.

The Qantas route structure is also unique and has been tailored to the capabilities and location of their operation. Sydney, Australia is most the HUB for all domestic and International flights. The spokes are each point A to B destination in both modes. However, with the longevity and distance of each flight to reach its destination and return almost make the route structure like a point to point operation. They are servicing their entire continent and worldly destinations from Sydney. There are very few flights that continue to other destinations after their initial transoceanic trip. Location has dictated the HUB and the spokes have become the single point destination and return. Their maintenance and overhaul facilities are also located in Sydney with some locations throughout the world for ease of maintaining a flying fleet. Having an airplane grounded in China with little to no maintenance support does not work out, but major work and services are planned for the home location.

The Qantas overall operation provides a unique set of challenges and demands from fleet choice, scheduling long haul flights, and their location and service destinations. The company has continually met these challenges head on with the ability and willingness to be fluid and prepared for change. It is the "transformation" mentality echoed from the leadership down that facilitates the success experienced year after year. Being on the of the world's most profitable and oldest airlines providing both domestic and international services under numerous brands, Qantas has set a benchmark for success and service.

Marketing

The Queensland and Northern Territory Aerial Services (QANTAS) has a rich and storied history which started close to one hundred years ago. Throughout their history, Qantas has transformed their business model several times. They have even changed their entire fleet on several occasions. However, one thing has remained constant throughout, which has been to provide a top tier world class flying experience. Their marketing strategy has evolved over the years and kept pace with the times, but pure customer satisfaction for each passenger remains paramount to the company. In the early years, word of mouth was the only way to advertise, and satisfied customers provided the positive communications needed to expand the business. Fast forward

to today and that word of mouth advertising still has its place and it works. Qantas is known as that top tier international carrier throughout the world, and the bottom line remains customer satisfaction and a positive flight experience.

Qantas' product has remained the same since their inception, the flight experience. It boils down to the selling of seats on an airplane in most cases, as it should. Qantas has the same result in mind, selling seats, yet tackles the problems in different ways. One example would be their airplanes and the products to whom they are very specifically marketed. Instead of marketing the airline directly, Qantas has marketed the experience and considers each paying customer's price point. They have accomplished this through a multi-carrier plan, meaning two separate airlines providing domestic services at different price points. According to CEO, Alan Joyce, "The Qantas Group's main business is the transportation of customers using two complementary airline brands – Qantas and Jetstar. We have built a reputation for excellence in safety, operational reliability, engineering and maintenance, and customer service." Qantas Domestic and JetStar are both airlines operating domestic services in Australia and are both owned by Qantas. JetStar offers a no-frills approach to travel and offers significantly lower prices to these locations. Qantas Domestic however, offers the traditional upscale full service flight experience that they are known for to the same destinations at a higher price. This has enable the customer to choose either cost over services or vice versa. This marketing method still addresses the product, which is selling seats, yet provides the customer with options based on their budgets and needs.

Qantas international's product marketing is handled slightly different. The product here is still the selling of the seat, but when the Qantas method is applied, things change. Again, it is the overall experience that is the focus for this airline. Qantas' fleet choice is number one on the marketing list, with ships like the Boeing 747-800 and the Airbus A-380. People travel and choose destinations, just to fly on these machines, according to Qantas leadership. Comfort and ease of travel are essential to Qantas. Wider seats and inflight entertainment along with more non-stop international destinations than any other airline make their product more enticing to travelers. Qantas relies heavily on the established "name and reputation" of their airline for repeat customers. Brand recognition and loyalty maintain their market share bring in additional customers over time. Convenience and ease of travel are the standards established for the international traveler by Qantas. This unmatched service and available destinations set them apart from the other carriers.

The second "P' in this marketing discussion must be price. The cost of a ticket is the key to this business. Priced too high, customers will not fly. Priced too low, the airline cannot sustain operations. Most fares charged by Qantas are determined by the market, and the market fluctuates. Qantas additionally determines their price

points based on what other airlines are charging. This pricing strategy keeps Qantas competitive and in-line with the market shares. For their domestic/international no frills airline brand, Jetstar, Qantas has employed a price penetration model which offers the lowest possible price for every flight. Qantas International and Domestic also offers a full fare prices which offers customers flexibility. This flexibility for paying full fare allows the customers to change, modify, and/or cancel their reservation at no additional costs. This incentive for the business traveler is an incentive most will not overlook. The multiple pricing methods used by Qantas offer a wide range of options for customers through both their domestic and international brands. One additional option employed by Qantas is a cost margin method, where the price set for fare is figured by the cost of the conducting the flight with a full complement and an additional add-on strictly determined as profit.

For the 3rd "P" in this discussion, Promotion, is an essential part of any business. Word of mouth may have been sufficient 80 years ago, but in today's day and age it is simply not enough. With technological advances, internet availability, and mass media, exposure to advertising is plentiful. Qantas currently has three major marketing campaigns underway, mostly limited to the Australian region. Their current "Welcome Home" campaign is aimed at domestic travel and bringing people closer together using the Jetstar brand. This campaign has been hugely successful and is expected to continue through the summer 2017. Additional promotions, including web advertising and search promotions are also used by Qantas. This being fees are paid to search engine facilitator ensuring the Qantas Brand is located first on searches and adds are associated in side-bar advertising on web pages. Since the advent of social media, Qantas has established accounts on all of the most popular platforms like Twitter and Facebook to keep their customers informed and feeling as they are part of the team. Along with traditional advertising such as newspapers, magazines, radio, and television, Qantas has embraced the handheld generation. They also announce changes and improvements through the social media outlets. On February 23rd, Qantas Tweeted "Qantas has revealed its next generation Premium Economy seat, which will debut on the airline's fleet of Boeing 787-800 Dreamliner's from October. Wider and with more functional space overall, the new seat has a unique recline motion that provides a class-leading level of comfort." This type of advertisement reaches each subscriber instantly and allows for real time feedback from its customers.

The final "P" in the puzzle is place. In the Qantas story, the place is arguably the most important and the roots to its success. The place can mean several things, such as the place it all started, the place where flights originate, and lastly the place where flights go. These places are equally important. Qantas is a world leader in International Travel as well as Australian first and largest Domestic provider through

the Qantas and JetStar. JetStar also provides a low cost International option for customers on a budget. The Qantas brand is truly global and amongst few airlines of such scale and worldwide influence. Qantas' international routes cover the globe and offer services to destinations including Dubai and the Middle East, Europe, South Africa, South America, Asia, and North America. Qantas conducts business all over the world and code shares a portion of the business, except in Australia.

This concludes the four "P's" of marketing and how Qantas Airline applies the principles needed to be successful. The Qantas product is top tier, the prices are competitive, the promotions are realistic, relevant, and current, and the places speak for themselves. Qantas is known for transforming their fleet and business model to keep ahead of the industry pace and be that trend setter they have always been.

International Operations

Qantas is considered one of the world's top tier international air carriers. They have perfected international operations out of necessity due to the isolated location. Qantas does provide domestic services throughout the Australian continent, however, the main source of revenue and the bulk of their operations are geared towards providing the world class flying experience their customers have become accustomed to. Qantas' base of operations is Sydney, Australia and they have been servicing international destinations since pre-WWI.

Their current fleet for their primary international long haul flights consists of the two of the world's premier largest aircraft. The Boeing 747-800 has been in Qantas service since its introduction, as well has the Airbus A-380. The international services provided by Qantas has been recognized and awarded in numerous forums. They were awarded the Best First Class at the 2012 Australian Business Traveler Awards. Additionally, their fleet boasts Cleaner and more efficient than ever before, "burning 17% less fuel per seat than today's largest jets and producing around 60% less carbon dioxide than the average family car, per passenger kilometer."

Qantas is one of the world's most recognized airlines and a founding member of the OneWorld® alliance. It serves almost 80 destinations in 20 countries. As the only Australian airline in any global airline alliance, it offers an extensive domestic network besides serving points in Asia, the South Pacific, Europe, North and South America and Africa. As a participating member in the OneWorld alliance, Qantas customers can reach just about anywhere in the world. The code sharing and affiliations has made Qantas on of the only airline truly capable of sustaining long haul operations on a regularly scheduled basis. With destinations in the U.S., Europe, Asia, and the Middle East, Qantas' has the global reach. Routes extend from Las Angles to London to Dubai to Sydney. "Qantas code-shares with OneWorld partners American Airlines,

British Airways, Finnair, Japan Airlines and LAN, and has additional commercial agreements with Aer Lingus, Air Niugini, Air North, Air Tahiti Nui, Air Vanuatu, Alaska Airlines, Alitalia, Asiana Airlines, Bangkok Airways, China Eastern, China Southern, El Al, Emirates, Fiji Airways, Jet Airways, Jetstar, and Vietnam Airlines". (OneWorld 2017)

The relationships developed through the OneWorld Alliance has elevated the level of service expected and greatly enhanced the capabilities of Qantas. The benefits of such an alliance has increased sales, filled seats, and ensures more return customers. The level of service for international flights is a tiered system for both models of airplanes flown. Qantas offers four cabin classes on most of its long-haul international flights, including First Class, International Business, International Premium Economy (on flights operated by Boeing 747-400s and Airbus A380s) and International Economy. Frequent flyer programs and the benefits shared amongst alliance partners streamlines the system and enhances customer satisfaction.

Qantas' primary business is providing the pinnacle of the international flight experience. This is echoed throughout the company and reinforced from the top down. Affiliations are chosen specifically to extend the range and service of the Qantas brand. The code shares, the One World alliance, as well as the Qantas brand are closely monitored by the leadership and Qantas believes in remaining in a transition and transformation phase of their business model, according to the company CEO. They are constantly reevaluating the processes, services, routes, and destinations to remain a world leader. Their fleet is constantly being upgraded to a premier level and operations refined to provide the customer the best possible experience. Without the OneWorld Alliance and the code sharing agreements, Qantas would still provide the long-haul operation, however it would be point to point, most originating the Sydney.

Questions

1. What are the business model and strategies of Qantas Airlines?
2. Prepare a project report on fleet management and route structure.

KEY TERMS

Analytical representation
Software systems
Flowcharts
Data flow diagrams
real-world entity
Legacy systems
Visualization
Business process management
Gantt charts

DISCUSSION QUESTIONS

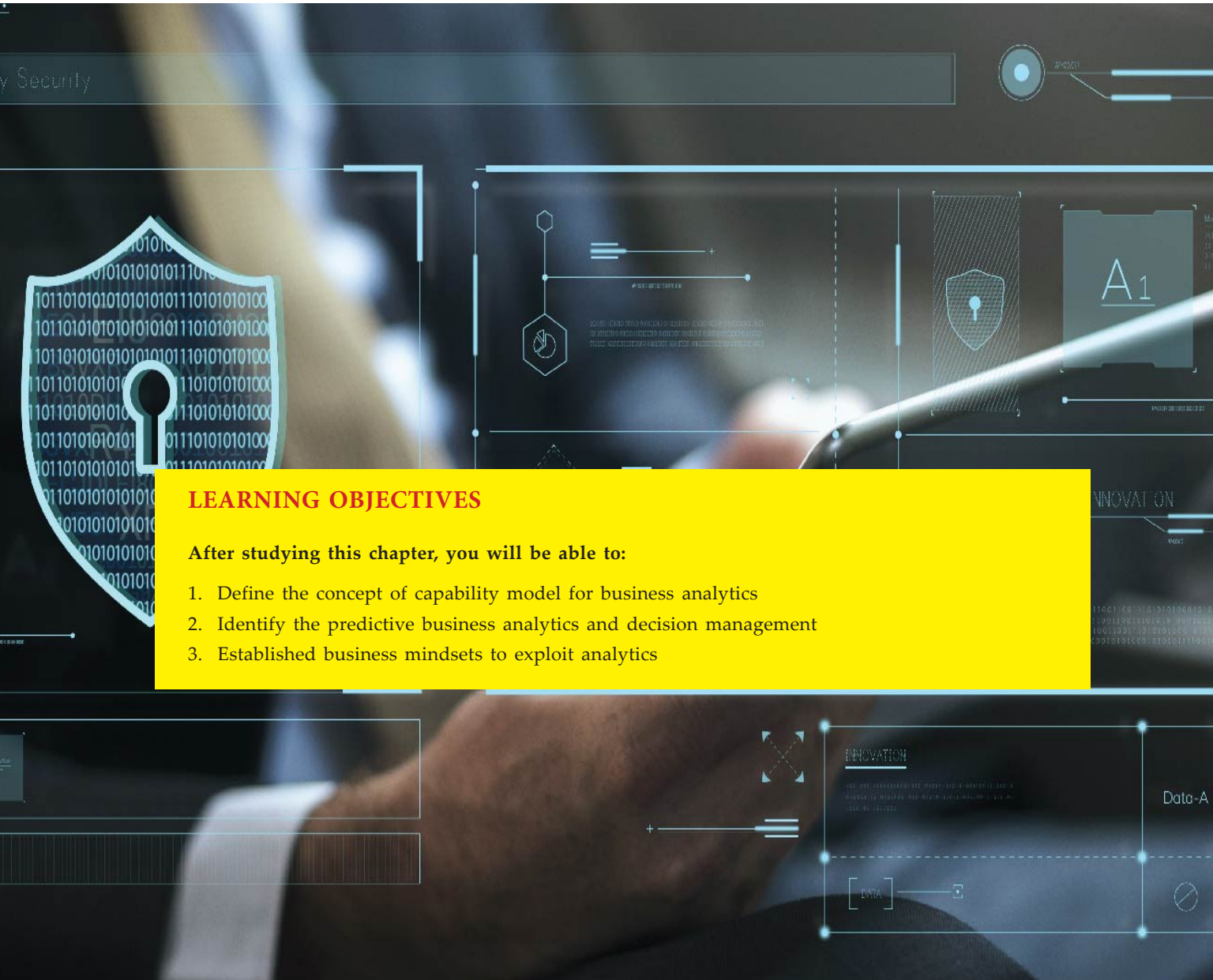
1. Describe the business process modeling notation (BPMN).
2. What do you understand by data flow diagrams Yourdon's technique?
3. Explain the integrated definition for function modeling (IDEF).
4. Define the object oriented methods.
5. What do you mean by EPC-model?
6. Discuss on importance of business reference model.
7. Define the business process reengineering.
8. What is big data analytics in manufacturing?

REFERENCES

1. Barjis, Joseph (2008). "The importance of business process modeling in software systems design". Science of Computer Programming. 71: 73–87.
2. Lasek M. and Otmanowski B. (2007). BPMN® – the standard for describing business processes. Building BPMN process models in iGrafx™ (in Polish), Warsaw: Applied Informatics and Management College Printing House.
3. Manuel Laguna, Johan Marklund (2004). Business Process Modeling, Simulation, and Design. Pearson/Prentice Hall, 2004.
4. Marcinkowski B. (2008). Business Modeling with UML and BPMN: Features and Comparison, BIR2008 – The 7th International Conference on Perspectives inBusiness Informatics Research; Gdansk: University of Gdansk Press.
5. Object Management Group (2009). Business Process Modeling Notation. Version 1.2, http://www.omg.org/spec/BPMN/1.2/PDF, downloaded 22.03.2010.
6. Ryan K. L. Ko, Stephen S. G. Lee, Eng Wah Lee (2009) Business Process Management (BPM) Standards: A Survey. In: Business Process Management Journal, Emerald Group Publishing Limited. Volume 15 Issue 5. ISSN 1463-7154.

CHAPTER 5

CAPABILITY MODEL FOR BUSINESS ANALYTICS



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Define the concept of capability model for business analytics
2. Identify the predictive business analytics and decision management
3. Established business mindsets to exploit analytics

INTRODUCTION

A Capability Model or a Business Capabilities Model is an integrated and comprehensive set of business capabilities. The capability model is a decomposition of what a business does and can do in a logical and granular grouping. A business capability model is one of the critical business architecture deliverables and a bridge between business and IT, and a foundation for enterprise transformation.

However, if you are looking for a "Competency Model" – which is about soft skills or competencies, or a "Capability Maturity Model (CMM)" – which is a way to measure the maturity of the processes, those are distinct and different from a capability model in the business architecture realm.

Business architects and enterprise architects build business capability models as a part of the overall business architecture and enterprise architecture mandate.

A capability model is an integral part of the business architecture deliverables and is a significant component of BizBok (Business Architecture Book of Knowledge). A capability model is an essential part of the TOGAF framework (The Open Group Architecture Framework).

A business capability is an elemental building block (or a Lego block) of what a business does or can do. At its core, it is an abstraction of the underlying functionality and flows expressed as a noun form (some business architects use Gerunds as well, but we are not the grammar police.) An agglomeration of a cluster of underlying business capabilities can manifest a product, a service, a platform, a business unit, a department, and of course, an enterprise.

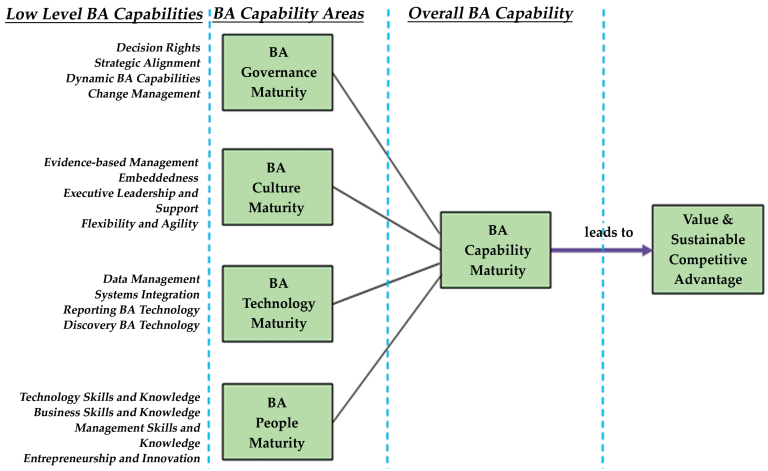
CONCEPT OF CAPABILITY MODEL FOR BUSINESS ANALYTICS

The intent of the model is to provide organizations with a means to quantify analytic capability. The model encompasses three dimensions.

Vertically analytic capability is stratified into six levels which describe the degree of discipline with which analytic processes

are managed. The second dimension – analytic roles – has only two members: analytic providers and analytic consumers. The third dimension – analytic usage – describes seven applications of analytics ranging from performance to geospatial analysis.

Together these dimensions produce a model with 84 cells. Now let us take the model from theoretical to something for practical application. The goal of the model is to quantify the level of an organization's analytic capability level with sufficient indicators to provide a measure upon which decisions can be made and actions taken – to form the basis of analytic capability assessment. Ideally that assessment should be fast and light something that is easy to perform yet provides informative results.



Every business uses analytics in some way and many have thoroughly integrated analytics into their decision and management processes. Creation and use of analytics has migrated from being primarily business intelligence (BI) and IT activities to something that permeates every part of the business.

Capabilities are generated taking into consideration various dimensions that straddle the corporate functional portfolios. Every organization has a different but similar set of dimensions.

The Capability Maturity Model

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software

development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.



Figure 5.1: Capability maturity model.

Level One: Initial: The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.

Level Two: Repeatable: This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.

Level Three: Defined: The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an approved, tailored version of the organization's standard software process for developing, testing and maintaining the application.

Level Four: Managed: Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance. At this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.

Level Five: Optimizing: The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

The Analytic Capability Model

The CMM described above is a good starting place, but not sufficiently specific to be applied to analytic processes. To build an analytic capability model, we need to add the dimensions that are specific to analytics: analytic roles and analytic uses. Figure 5.2 adds roles to the capability model using the two primary roles that occur in analytic practice: using analytics and creating analytics. It is important to realize that the same individual may perform both roles – as an analytic consumer and an analytic developer.

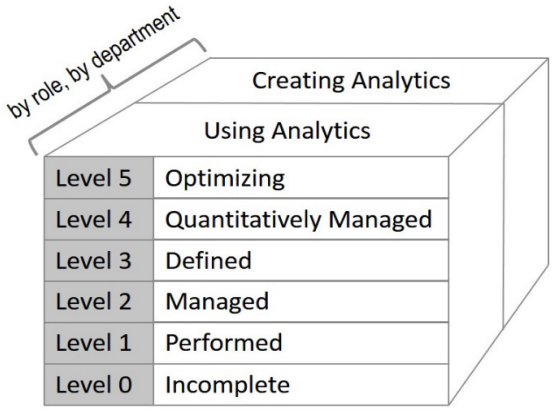


Figure 5.2: Analytics roles and capabilities.

At the intersection of each role with each level, it now becomes practical to identify specific characteristics that are indicative of that intersection. As you consider the descriptions of the capability levels above, read the word "process" in all of its occurrences as "analytic process" – the processes by which we create and use analytics.

Using analytics at Level 0 (incomplete), for example, might involve drawing conclusions and making decisions based on

a partially finished spreadsheet that may never be completed. Similarly, creating analytics at Level 0 might entail building that unfinished spreadsheet that was perhaps "good enough" at the time. At the opposite extreme, Level 5 analytic usage could be application of a performance scorecard for a BI program to continuously align business intelligence with the business. And analytic creation at Level 5 might be use of metrics to monitor and improve data and information quality in a BI program.

Adding the analytic usage dimension produces the capability model shown in Figure 5.3. The usage dimension looks at the purpose for analytics – the kinds of analyses that are performed. These include performance management, behavioral analysis, prediction and forecasting, cause-and-effect analysis, exploratory analysis and discovery, and text and spatial analytics. Each of these is an analytic process, thus a subject of capability assessment.

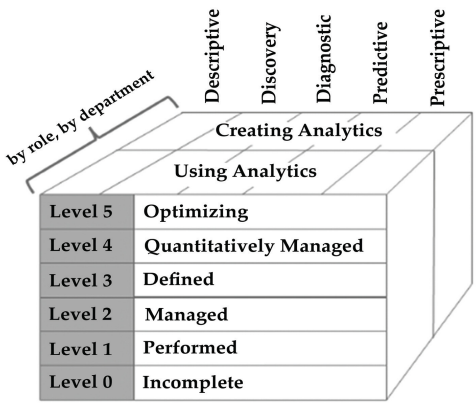


Figure 5.3: Capability, roles and analytic usage.

"God does not begin by asking our ability, but more of our availability. When we prove our dependability, He will increase our capability."

—Neal A. Maxwell

Analytic Capability Assessment

The model that is the foundation for assessment is based upon the SEI model from Carnegie-Mellon, with six levels of process

capability ranging from incomplete to optimized. Applying the model to analytics looks at two kinds of analytic processes those to create analytics and those to consume analytics.

Capability assessment uses the model as a structure for data collection. Ten data points make a tool that is fast and light, yet effective for subjective self-assessment. Selecting the best-fit response from a set of descriptive phrases determines the response value for each data point. An example of assessment results is shown in Figure 5.4.

Analytic Capability Assessment	Self Evaluation		Scores		
	Creating Analytics	Using Analytics	Create	Use	Both
Descriptive Analytics	evolving portfolio	decision support	5.0	3.0	4.0
Discovery Analytics	development projects	measure & report	2.0	1.0	1.5
Diagnostic Analytics	pockets of competency	performance management	1.0	2.0	1.5
Predictive Analytics	development projects	strategy & planning	2.0	5.0	3.5
Prescriptive Analytics	pockets of competency	rare or none	1.0	0.0	0.5
*Select a value from the dropdown list for each of the yellow shaded cells.that best decribes your **EXISTING** anallytic capabilities*			2.2	2.2	**2.2**

Figure 5.4: Analytics Capability Assessment.

It is important to recognize that this is subjective self-assessment. As is true for all self-assessment processes, quality and accuracy of the assessment are strongly influenced by the participants in the process and by the care and consideration given to each response. It is essential to get the right people together and have the right conversations to arrive at a well-reasoned consensus response.

The first interpretive error occurs by using the top of the five-point scale as the basis for comparison, and by assuming that 5 is the target value. Not every organization needs to have level 5 analytic capabilities. Thus the comparative basis should not be what is possible – the top of the scale – but what is needed.

The second error occurs by looking only at the aggregate score. Not every organization needs to have top-of-scale analytic capabilities, and very few need to be at level 5 for every category of assessment. When examined on a row-by-row basis the assessment shows a substantial score of 4.0 for descriptive analytics. With an evolving portfolio applied for decision support this organization's descriptive analytics processes are quantitatively managed and well positioned on the capability scale.

Quantitatively managed certainly seems like good news. But is it really? Once again, it is difficult to know because the basis for comparison is an external scale with no direct connection to the needs of the organization. In the case of descriptive analytics, it is entirely possible that the organization needs to go beyond quantitatively managed, striving to optimize because strategic and operational uses are missed opportunities. In this case there is a gap between capabilities and needs indicating that capabilities should be strengthened. The opposite is also possible. When capabilities exceed needs (a negative gap) you may be incurring analytics cost from which no value is derived.

Another cell-by-cell look highlights some apparent bad news. The low 1.0 scores for using discovery analytics, creating diagnostic analytics, and creating prescriptive analytics all look particularly weak.

All of these observations are interesting numbers, but they lack context to be informative a scenario similar to the vehicle traveling at 60 mph because we've only quantified capabilities and know nothing about needs.

Analytic Capability Assessment	Self Evaluation		Scores		
	Creating Analytics	Using Analytics	Create	Use	Both
Descriptive Analytics	integrated & reused	operational analytics	4.0	4.0	4.0
Discovery Analytics	repeatable processes	decision support	3.0	3.0	3.0
Diagnostic Analytics	repeatable processes	decision support	3.0	3.0	3.0
Predictive Analytics	repeatable processes	strategy & planning	3.0	5.0	4.0
Prescriptive Analytics	development projects	operational analytics	2.0	4.0	3.0
*Select a value from the dropdown list for each of the yellow shaded cells.that best decribes your **NEEDED** anallytic capabilities*			3.0	3.8	**3.4**

Figure 5.5: Analytics Needs Assessment.

To find real meaning in this assessment we must know what is needed. A complete analytic capability assessment must collect data about both current capabilities and needed capabilities, and must provide the basis to perform capability gap analysis. The spreadsheet tool for assessment includes a sheet for needs assessment (see Figure 5.5) as well as gap analysis functions. Gap analysis is generated without direct input when both capabilities and needs assessments are completed.

Assessment with Purpose

With all of the assessment functions –current capabilities, needed capabilities, and gap analysis – we now have the essential

elements to measure and evaluate with purpose. The purposes for which this tool is designed are:

- To confidently know your current level of analytic capabilities.
- Develop a well-reasoned consensus view of your need for analytic capabilities.
- Understand the gap between analytic capabilities and needs.
- Gain insights that help you make plans to close the gap.

The assessment data shown in figures 5.4 and 5.5 – responses and scoring of both capabilities and needs – is the input data to gap analysis. But the data isn't easily analyzed because it is not organized for comparison. The "gap analysis" tab of the spreadsheet reorganizes the data to support visual comparison of capabilities and needs. Figure 5.6 shows a tabular view that helps to compare the numbers.

Analytic Capability Assessment - Gap Analysis

	Current Capability			Needed Capability			Capabilities Gap		
	Creating Analytics	Using Analytics	Creating & Using	Creating Analytics	Using Analytics	Creating & Using	Creating Analytics	Using Analytics	Creating & Using
Descriptive Analytics	5.0	3.0	4.0	4.0	4.0	4.0	-1.0	1.0	0.0
Discovery Analytics	2.0	1.0	1.5	3.0	3.0	3.0	1.0	2.0	1.5
Diagnostic Analytics	1.0	2.0	1.5	3.0	3.0	3.0	2.0	1.0	1.5
Predictive Analytics	2.0	5.0	3.5	3.0	5.0	4.0	1.0	0.0	0.5
Prescriptive Analytics	1.0	0.0	0.5	2.0	4.0	3.0	1.0	4.0	2.5

Figure 5.6: Gap Analysis Table.

This table takes attention away from the aggregate scores by not displaying them. Too much attention to aggregate scores introduces bias when interpreting the results. The table is organized to suggest row-by-row and column-by-column analysis. Nothing new is found by examining the first two sets of columns. They simply restate the capabilities and needs numbers shown in the earlier tables. The interesting data appears in the third set of columns where gaps are quantified for each row:

- For all of the five types of analytics the common theme is that needs exceed capabilities.
- In one instance – creating descriptive analytics – capabilities exceed needs and the gap is negative.
- In two instances – creating and using descriptive analytics, and using predictive analytics – the gap is zero.

- For predictive analytics zero gap means that the needs are being met.
- In the case of descriptive analytics the zero aggregate of creating and using is a result of +1 gap for using analytics and -1 gap for creating analytics.
- The largest gaps are in the area of prescriptive analytics where needs to use far exceed capabilities to use.

The logical conclusion from these observations is that some action should be taken to close the gap between needs and capabilities. Obviously the answer is to increase capabilities, not to decrease needs. The hard questions for planning are "Which capabilities to increase?" and "Where to begin closing the gap?" To answer these questions, it helps to visualize the gap. The gap analysis charts shown in Figure 5.7 help with visualization.

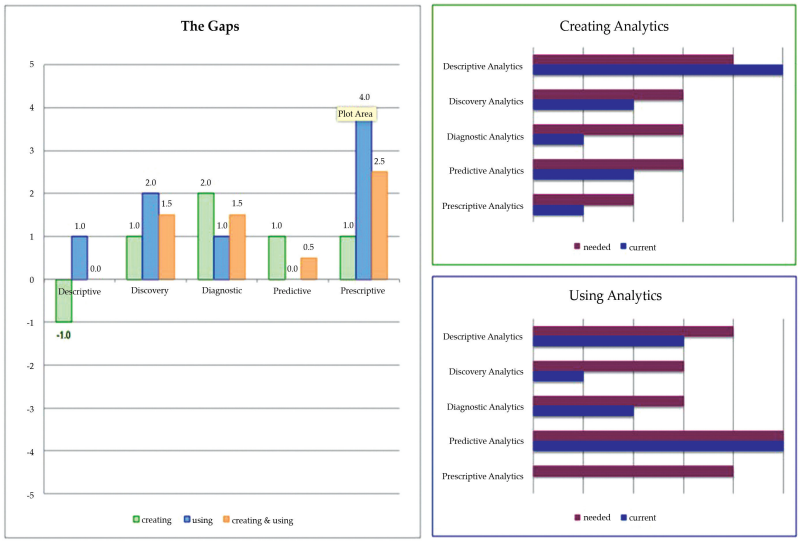


Figure 5.7: Gap Analysis Graphs.

The visual perspective brings new observations, adding to the understanding gained by looking at the gap analysis table.

- Prescriptive analytics with a disturbingly wide capabilities-to-needs gap is a high priority.
- Although the gap for descriptive analytics is not especially wide – in fact the combined creating and using gap is zero – the disparity of creating and using indicates an area of concern. Creation and use processes

for descriptive analytics seem to be out of sync with underutilized capabilities and unmet needs.

- Discovery and diagnostic analytics both need some attention. Discovery attention should focus primarily on using analytics. Diagnostic action should focus mostly on creating analytics.

Capability Assessment Process

To assess we must first collect data, and the matrix model provides the structure for data collection. Fortunately it isn't necessary to examine all 60 cells of the model to assess analytic capability. The capability levels are, in fact, a product of the assessment, not specific variables for data collection. The roles dimension and the usage dimension intersect to create ten data points as illustrated in Figure 5.8.

	Creating Analytics	Using Analytics
Descriptive Analytics		
Discovery Analytics		
Diagnostic Analytics		
Predictive Analytics		
Prescriptive Analytics		

Figure 5.8: Data Points for Capability Assessment.

A total of ten data points fits the "fast and light" criteria, but a small number of data points means that each must be carefully considered to avoid extreme bias that might be introduced by a single value. Ideally, the assessment process involves a group of stakeholders who represent a broad cross-section of the business analytics community – those who create analytics and use analytics throughout the organization - ranging from data scientists with advanced analytics technologies, to business managers using spreadsheets and/or self-service tools.

The purpose of the group is to arrive at consensus responses for each of the data points. Consensus, of course, begins with discussion. A simple spreadsheet provides the structure to drive discussion and constrains the set of responses such that they are quantifiable. Figure 5.9 illustrates the assessment spreadsheet.

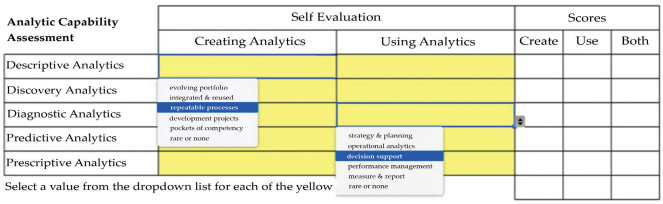


Figure 5.9: Analytics Capability Assessment Spreadsheet.

The yellow shaded areas are the spaces in which responses are collected – one column of responses for creating analytics and one for using analytics. The dropdown lists limit the set of responses in each column. Behind the scenes, the spreadsheet converts text responses to numerical values and derives capability assessment scores in the columns on the right.

Purpose

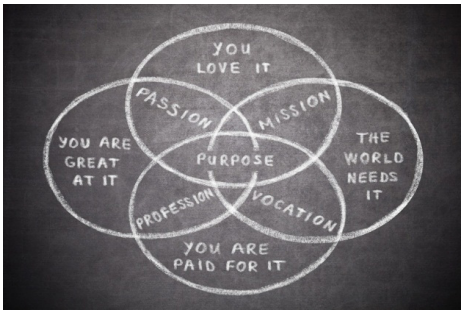
Descriptive Analytics provides information about past actions, events, and outcomes: What happened? How much? When did it happen? The purpose is to provide quantitative descriptions of systems, processes, activities, events, etc. Descriptive analytics is even applied to data with the goal of quantitatively describing the data itself, providing knowledge needed to use the data for other types of analytics.

Diagnostic Analytics also looks at past events, but from a different perspective than descriptive analytics. Diagnostics focus on cause and effect: Why do things happen? Why do outcome measures go up or down? What are the influences on business outcomes? Why do things happen at specific points in time or on particular time cycles?

Discovery Analytics uncover interesting, previously unknown facts, trends, and patterns. Think of discovery as learning analytics – seeking to find new and useful knowledge. Bridging from data to business, discovery analytics may pursue data insights, business insights, or both.

Predictive Analytics uses data to forecast probabilities of future events and conditions. The purpose is to answer questions about what is likely to happen, and forecasts are typically related to expected behaviors of individuals (customers, employees, machines, parts, etc.). Prediction may focus on specific individuals

or may categorize individuals by probability – Which customers are likely to churn? Who has high probability to respond to a cross sell offer? Who is likely to miss scheduled loan payments? Predictions are valuable in a variety of planning and decision-making processes.



Prescriptive Analytics recommends and/or automates decisions with the goal of optimizing future outcomes. Prescriptions as recommendations help decision makers to choose among alternative responses to a set of circumstances. When prescription is used to automate decision-making, the analytic model determines a single best response to a set of conditions.

Responses Creating Analytics

Evolving Portfolio: Analytics development is based upon a systematically managed collection of analytic capabilities and systems that is aligned with business processes and information needs, and that continuously adapts to business change.

Integrated and Reused: Analytic development processes have sufficient discipline and governance to ensure that measures and metrics are consistently defined, non-conflicting, non-redundant, and reused across business functions and analytic applications.

Repeatable Processes: Analytic development processes have a methodological foundation. They are defined, documented, repeatable, and teachable processes that are used consistently across the community of analytics creators.

Development Project: Analytics development activities are formalized as projects with all of the key project management elements of planning, execution, monitoring, control, and closure.

Pockets of Competency: Some business units and individuals have the qualities described for development projects, repeatable processes, and integration and reuse. The skills of these groups are local, not coordinated across groups, and not generally available to the broader organization.

Responses Using Analytics

Strategy and Planning: Analytics are viewed and applied as valuable strategy management tools that have important roles both in setting and achieving business goals. Analytics help to shape strategy, inform strategic planning with understanding and insight, and monitor to know the degree to which strategic goals are being achieved.

Operational Analytics: Analytics enable and support applications, services, and technologies to monitor, analyze, and manage performance of daily business operations. Business effectiveness and efficiency are improved through analytics that are focused on specific operational activities and workflows.

Decision Support: Analytics are applied to inform decision-making and problem-solving processes, with particular attention problems of uncertainty, ambiguity, and elusive problem definition. Iterative analysis where each answer brings new questions is a common occurrence.

Performance Management: Analytics are used to measure and monitor business results as compared to pre-defined tactical and operational goals. Directional trends and variance between goals and actual results are used to identify areas where management attention is needed.

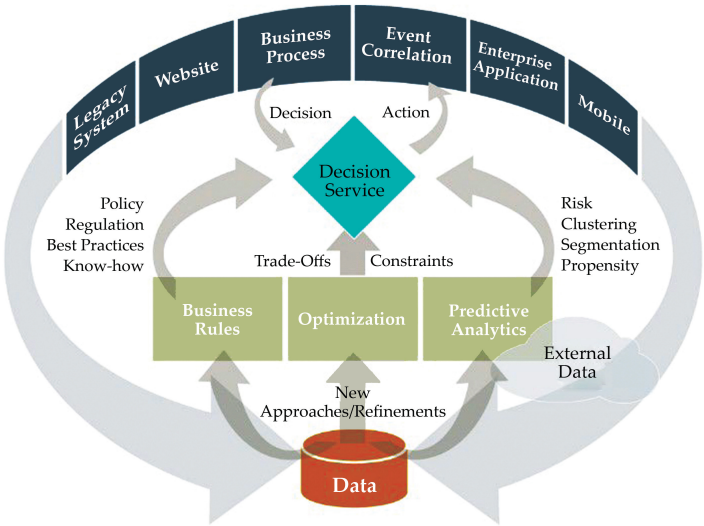
Measurement and Reporting: Business outcomes are quantified as metrics and performance indicators, and are routinely published to a community of interested business stakeholders.

"When you begin to entertain the more positive possibilities, hope for the best, plan for the best and expect the best, your eventual attraction of the best possible circumstances is almost guaranteed."

—Stephen Richards

PREDICTIVE BUSINESS ANALYTICS AND DECISION MANAGEMENT

Big data has been defined as a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, validation, storage, search, sharing, analysis, and visualization. What is needed is to shift the discussion from big data to big value. Business analytics and its amplifier, predictive business analytics, serve as a means to an end, and that end is faster, smarter decisions. Many may assume that this implies executive decisions, but the higher value for and benefit from applying analytics is arguably for daily operational decisions. Here is why.



Decisions can be segmented in three layers:

- Strategic decisions are few in number but can have large impacts. For example, should we acquire a company or exit a market?
- Tactical decisions involve controlling with moderate impacts. For example, should we modify our supply chain?
- Operational decisions occur daily, even hourly, and often affect a single transaction or customer. For example, what deal should I offer to this customer or should I accept making this bank loan?

There are several reasons that operational decisions are arguably most important for embracing analytics. First, executing the executive team's strategy is not accomplished solely with strategy maps and their resulting key performance indicators (KPIs) in a performance scorecard and dashboards. The daily decisions are what actually move the dials. Next, although much is now written about enterprise risk management, the reality is that an organization's exposure to risk does not come in big chunks. Enterprise risk management deals more with reporting. Risk is incurred one event or transaction at a time. Finally, in the sales and marketing functions, operational decisions maximize customer value much more than do policies. *For example*, what should a frontline customer-facing worker do or say to a customer to gain profit lift? Operational decisions scale from the bottom up, and in the aggregate they can collectively exceed the impact of a few strategic decisions.

Predictive Modeling

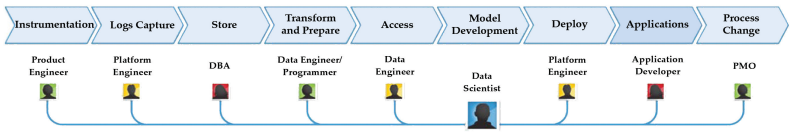
Predictive modeling mathematically represents underlying relationships in historical data in order to explain the data and make predictions, forecasts or classifications about future events. Example of predictive modeling, next best action (or offer) in e-commerce was covered in this post.

Predictive models analyze current and historical data on individuals to produce metrics such as scores. These scores rank-order individuals by likely future performance, e.g., their likelihood of making credit payments on time, or of responding to a particular offer for services. Predictive models can also detect the likelihood of a transaction being fraudulent (Risk Detection). Predictive models are frequently operationalized in mission-critical transaction systems and drive decisions and actions in near real time. A number of analytic methodologies underlie solutions in this area including:

- Applications of both linear and nonlinear mathematical programming algorithms, in which one objective is optimized within a set of constraints,
- Advanced "neural" systems, which learn complex patterns from large data sets to predict the probability that a new individual will exhibit certain behaviors

of business interest. Neural Networks are biologically inspired machine learning models that are being used to achieve the recent record-breaking performance on speech recognition and visual object recognition.

- Statistical techniques for analysis and pattern detection within large datasets.
- Predictive models summarize large quantities of data to amplify its value. The value chain for predictive modeling in a M2M scenario is shown below. It's all about having the right people and right models.



Decision Analysis and Optimization

Decision analysis refers to the broad quantitative field that deals with modeling, analyzing and optimizing decisions made by individuals, groups and organizations. Some applications include optimizing supply chain management, tracking key performance indicators, uncovering hidden sales opportunities and determining runaway operating costs.

While predictive models analyze multiple aspects of individual behavior to forecast future behavior, decision analysis analyzes multiple aspects of a given decision to identify the most effective action to take to reach a desired result.

Most consulting firms leverage decision analysis to provide bespoke data-driven solutions to a variety of business applications. Apart from statistical modeling and data analysis, the focus is also on understanding business challenges and delivering solutions. Integrated approaches to decision analysis incorporate the development of a decision model that mathematically maps the entire decision structure; proprietary optimization technology that identifies the most effective strategies, given both the performance objective and constraints; the development of designed testing required for active, continuous learning; and the robust extrapolation of an optimized strategy to a wider set of scenarios than historically encountered.

Optimization capabilities also include a proprietary mathematical modeling and programming language, an easy-to-use development and visualization environment, and a state-of-the-art set of optimization algorithms.

Transaction Profiling

Transaction profiling is a technique used to extract meaningful information and reduce the complexity of transaction data used in modeling.

Many solutions operate using transactional data, such as credit card purchase transactions, or other types of data that change over time. In its raw form, this data is very difficult to use in predictive models for several reasons. First, an isolated transaction contains very little information about the behavior of the individual who generated the transaction. In addition, transaction patterns change rapidly over time. Finally, this type of data can often be highly complex.

To overcome these issues, a set of proprietary techniques are used to transform raw data into a mathematical representation that reveals latent information, and which make the data more usable by predictive models. This profiling technology accumulates data across multiple transactions of many types to create and update profiles of transaction patterns. These profiles enable the neural network models to efficiently and effectively make accurate assessments of, for example, fraud risk and credit risk within real-time transaction streams.

Increasingly, teams are pushing the envelope of how to use information retrieval, machine learning, computational linguistics, matrix and graph algorithms, unsupervised clustering & data mining to solve profiling problems.

CHALLENGING ESTABLISHED BUSINESS MINDSETS TO EXPLOIT ANALYTICS

As one of the premier indicators of the flourishing self-service movement within the datasphere, semantically enriched smart data lakes provide a means for business users to access and analyze big data on an unprecedented, enterprise-wide scale.

The enduring relevance of these platforms is almost entirely based on the value that end-users derive from them. Maximizing that utility necessitates a set of skills both familiar and foreign to the business.



Abandoned Technological Skills

Deployments of non-semantic data lakes and relational means of ingesting, transforming and analyzing data have always required an inordinate focus on technological competence. It's a recurring paradox: Business users were unversed in integration requirements, ETL (extract, transform and load) processes and the intricacies of data modeling, yet their abilities to fulfill business objectives with data were inherently circumscribed by them. Utilizing these traditional means of accessing business intelligence and analytics required end users to be responsible – or depend on someone else – for facility in the finer points of SQL, writing code, data modeling, Hadoop skills and MapReduce jobs.



Invariably, the dearth of such intricacies on the part of the business resulted in an over-reliance on IT departments for analytics and business intelligence (BI). More recently, data scientists have been deployed to account for this skills shortage in source integration, loading and analytics results. With these professionals tasked with retooling models and processes every time business requirements or data sources changed, deployments of analytics became less frequent, the esteem for data-driven processes decreased, and tension between the business and IT fomented.

Refined Business Skills

Enriching data lakes with semantic technologies obsoletes the reliance on technological competencies and provides the business self-service data access and analytics. Consequentially, there is a greater emphasis on business skills for utilizing these options, which never expose nor require any end-user cognizance of underlying semantic components such as RDF (Resource Description Framework), OWL (Web Ontology Language), SPARQL (SPARQL Protocol and RDF Query Language), taxonomies or vocabularies. Initial configuration of smart data lakes necessitates IT or data scientist involvement as do continued efforts to load new data sources. However, most critical facets of linking, contextualizing and data preparation is done upfront and incorporated into an evolving semantic model that facilitates self-service analytics at unparalleled speeds, emphasizing the following business-oriented competencies:

Relating Domain Expertise to Data

Although it combines aspects of data-driven processes and IT involvement, this particular skill is unique to the business and typifies the way that smart data lakes enhance its ability to perform. Most business users are proficient in their domains. Maximizing smart data lake utility, however, requires pairing that expertise with an ability to use data to improve it and business objectives. Specifically, the business needs to understand what types of information relate to objectives, how it is linked to additional information, and what effect synthesizing that data creates for achieving outcomes. These users need to identify which particular attributes and properties of data benefit their

business processes most, and formulate questions around them resulting in decisive action.

Tool Manipulation: Dashboards and BI

This chapter represents the extent of the technological involvement on behalf of the business in optimizing smart data lakes. Its prominence is somewhat tempered by the self-service movement's simplification of visualizations for end users prior to the popularity of these hubs. Nonetheless, business users need proficiency in manipulating the various forms of "publishing" and viewing the results of analytics endeavors facilitated by data lakes, since they no longer have to request them from IT. Competencies in creating and tailoring dashboards, visualization tools or even previously used BI and analytics platforms are needed to determine the impact of analytics on business processes. These skills can be as basic as looking at a web browser interface for results; any variety of platforms works with these repositories.

Embracing Exploratory Analytics

This is partly based on analytics adroitness and partly predicated on the business user's choice. Perhaps even more than skill in relating domain expertise to data, it represents the full extent to which end-user ability can influence the unparalleled analytics potential smart data lakes provide. Simply understanding that newfound scope of analytics – and exploiting it – requires business user skill, particularly for those that are accustomed to traditional limitations of non-semantic data lakes and relational methods. Smart data lakes enable the business to traverse all of its organization's data, not just those in their particular domain. The expanded scope of possibilities that such data yields for analytics is only restrained by the ingenuity and drive of the user. The encompassing nature of such analytics, and the expedience at which questions are answered only to beget more questions, entails a different conception of the possibilities and relevance of analytics itself. The business must adopt this exploratory analytics mindset to approach the true yield that the data lake concept offers. All of the skills business users need to optimize smart data lake deployments pertain to analytics. This fact is largely entrenched in the reality that most of the other facets of data lakes

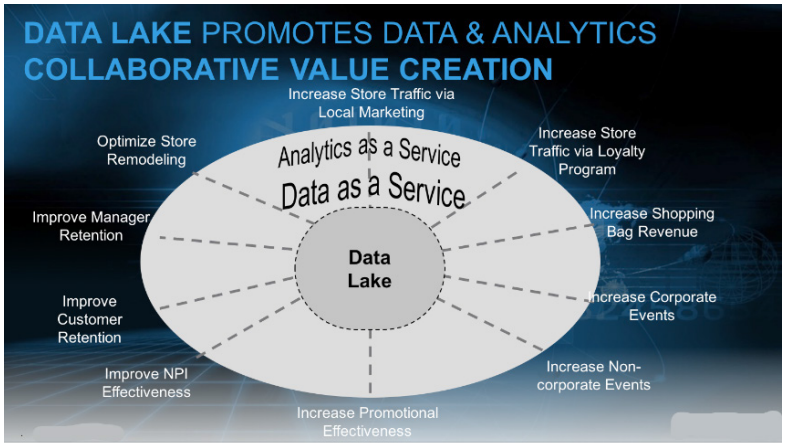
(integrating sources, contextualizing and linking data, adhering to governance practices) are automated via the incorporation of the semantic model at the heart of these platforms. Those that are not, such as loading new sources and types of data, are done by IT and are easily added to the semantic model without the typical delay associated with this process. Consequentially, business users have the luxury of concentrating on analytics to achieve departmental objectives.

"When the vision is clear, the results will appear. Keep your mindset positive as you work your plan, flourish, and always remember why you started."

—Germany Kent

Maximizing Data Lakes: Exploiting Analytics

The majority of the skills requisite for business users of smart data lakes revolve around business itself. This notion especially applies to the integration of domain knowledge with data and the basic dexterity required for tool manipulation. The lack of technological skills needed on the part of the business, however, should not be mistaken for some sort of sleight of hand. The reason these users can now traverse their organization's entire information assets with an exploratory analytics mindset that may prove revolutionary is that the underlying semantics technologies perceive the context and relationships between data elements – the end user does not. Instead, the business user merely reaps the benefits.

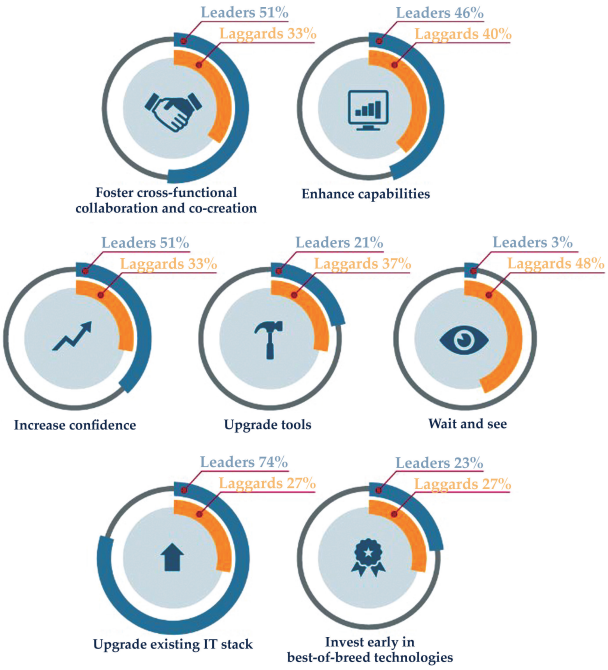


Similarly, it's the technology – graph models based on the semantic technology standards, not the business user – that is responsible for linking data sources based on those relationships for integration efforts, thus allowing the business to reap the benefits again. The same concept applies to the ability to parse through those data elements in their native forms at speeds that empower those users to leverage more of their data quicker than previously possible. The business user is not responsible for any of those processes, the technology is. Nevertheless, when equipped with the aforementioned skills he or she can readily monetize them.

Building an Analytical Organization

Analytics is often viewed primarily as an exercise in technology and data, but success is just as often a matter of leadership, culture, and human capabilities. The best analytical competitors are invariably replete with leaders who are passionate about analytical decisions, analysts who understand their business areas as well as esoteric algorithms, and managers and front-line employees who know the analytical underpinnings of their business processes. The logical response for aspiring analytical firms would simply be, "Get me some of those people!" However, there are several challenges in doing so:

- Analytical people are not easy to hire or to develop – and perhaps the most difficult role to fill is that of the analytically-driven senior executive.
- Highly analytical people are not like other employees, and their "care and feeding" is difficult.
- There are a variety of ways to organize analytical functions within an organization, each with pluses and minuses.
- Anyone wanting to build an analytical culture – as with any other new cultural emphasis – will encounter a severe shortage of explicit directions for doing so.



Despite these challenges, organizations need to build analytical capability quickly. They need to get the right people in place, and the people need to be guided by a strong and persistent analytical orientation in the cultural fabric. They need to build a critical mass of analysts who can share ideas and learn from each other, but the analysts also need to be closely attached to business units and their analytical problems and issues. The analysts must include not only seasoned professionals, but also "analytical amateurs," who take analytical decisions and actions daily, are just as important in making business analytics pay off. The organization needs the discipline of an enterprise-wide approach to analytics, but also sufficient innovative and entrepreneurial spirit to create new initiatives at every level of the business.

CONCLUSION

Business Capability is the expression or the articulation of the capacity, materials, and expertise an organization needs in order to perform core functions. Enterprise Architects use Business Capabilities to illustrate the over-arching needs of the business in order to better strategize IT solutions that meet those business needs.

Business capability models are often used for strategic planning up to 3-5 years ahead in the future. Business executives and architects usually discuss the relative importance of different business capabilities from the strategic perspective, identify the capabilities requiring the most significant enhancements in the long run and then employ the so-called technique of heat mapping, i.e. explicitly highlighting or color-coding the business capabilities that should become the primary focus of future IT investments. Often the necessary strategic capability improvements can be also classified into different types reflecting the nature of these improvements, e.g. fundamentally new business operations should be added to the capability, the quality of current operations should be raised or the existing capability should be performed at lower cost, and then color-coded appropriately in business capability models. In some cases completely new business capabilities that need to be developed from scratch may be added to business capability models and heat mapped accordingly, while some existing capabilities that lost their importance may be removed as irrelevant.

KEY TERMS

Capability maturity model
Business intelligence (BI)
Performance management
Spatial analytics
Gap analysis
Analytic capability
Software development
Behavioral analysis
Capability assessment

DISCUSSION QUESTIONS

1. Discuss about capability maturity model.
2. How can you assess analytic capabilities? Explain.
3. Define the capability assessment process.
4. Explain the building an analytical organization.
5. Describe the predictive modeling.
6. Define the refined business skills.

REFERENCES

1. Burton, B. (2012) "Eight Business Capability Modeling Best Practices Enhance Business and IT Collaboration" (#G00245455), Stamford, CT: Gartner.
2. Cantara, M., Burton, B. and Scheibenreif, D. (2016) "Eight Best Practices for Creating High-Impact Business Capability Models" (#G00314568), Stamford, CT: Gartner.
3. Greski, L. (2009) "Business Capability Modeling: Theory & Practice", Architecture and Governance Magazine, Vol. 5, No. 7, pp. 1-4.
4. Keller, W. (2015) "Using Capability Models for Strategic Alignment", In: Simon, D. and Schmidt, C. (eds.) Business Architecture Management: Architecting the Business for Consistency and Alignment, Berlin: Springer, pp. 107-122.
5. Khosroshahi, P. A., Hauder, M., Volkert, S., Matthes, F. and Gernegross, M. (2018) "Business Capability Maps: Current Practices and Use Cases for Enterprise Architecture Management", In: Bui, T. X. (ed.) Proceedings of the 51st Hawaii International Conference on System Sciences, Big Island, HI: IEEE, pp. 4603-4612.
6. Scott, J. (2009) "Business Capability Maps: The Missing Link Between Business Strategy and IT Action", Architecture and Governance Magazine, Vol. 5, No. 9, pp. 1-4.
7. Swindell, A. (2014) "Business Capability Models: Why You Might Be Missing Out on Better Business Outcomes", Architecture and Governance Magazine, Vol. 10, No. 2, pp. 3-7.

CHAPTER 6

OBJECT–ORIENTED SYSTEM DEVELOPMENT



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Learn about the procedure-oriented paradigms
2. Discuss on object–oriented analysis
3. Illustrate the object-oriented design

INTRODUCTION

Object-Oriented Development uses "objects" to model real world objects. A car or a laptop can be considered as object. While traditional programming views software as a collection of functions, an object oriented system concentrates on the objects that combines data and functionality together.

An object is something which has its own identity and can be easily compared to a real world object like a car or a laptop. An object contains a state and some behavior. The state of an object is the properties of the object at a particular time, and behavior is the functions it will perform. The behavior of an object is usually described using methods, and these methods will be part of the object itself. So you don't have to refer anywhere else for object's functionality, whereas in function based traditional approach you need to remember all the methods and their location. For instance, in java, the state of an object is the set of values of an object's variables at any particular time and the behavior is implemented as methods.

In Object-Oriented Development, we apply object orientation across every system development activity such as requirement analysis, design, programming, testing, and maintenance. For instance, an object oriented analysis (OOA) will usually have the following steps:

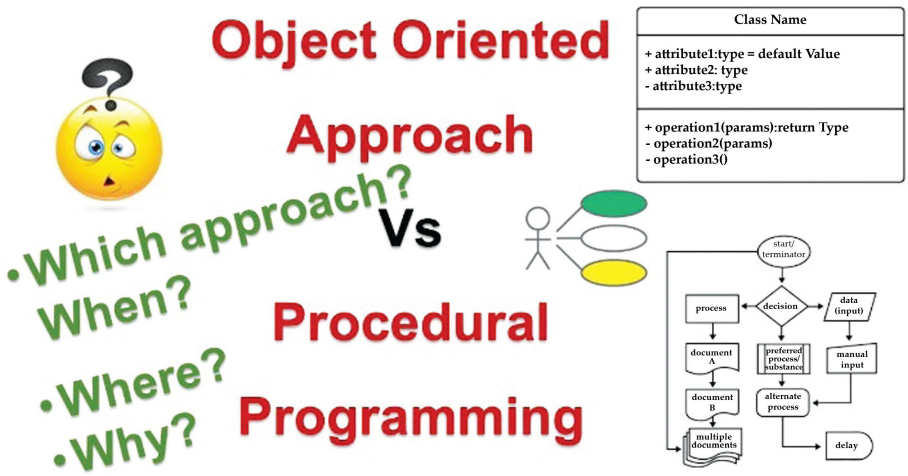
- Identifying the system functionality
- Identifying the involved objects
- Recognizing the object classes
- Analyzing the objects to fulfil the system functionality

Object-Oriented Programming (OOP) is based on object oriented features like Encapsulation, Polymorphism, Inheritance and Abstraction. These features are usually referred to as the OOPs concepts. We will see more about analysis, design, testing and maintenance later.

PROCEDURE-ORIENTED PARADIGMS

Software development is generally characterized by a chain of stages depicting the various inquires involved in the development procedure. The classic life cycle is based on an underlying model,

commonly referred to as the "water fall" model. This model tries to break up the activities that are identifiable into series of actions, each of which must be completed before the next begins. The activities include maintenance, demand analysis, design, coding, testing, and problem definition. In order to incorporate any changes or missing links further refinements to this model include iteration back to the before phases.



Problem Definition: This task needs a precise definition of the problem in user terms. A clear statement of the problem is critical to the success of the software. It helps not only the development but also the user to understand the problem better.

Evaluation: this covers a comprehensive study of the requirements of both the user and the software.

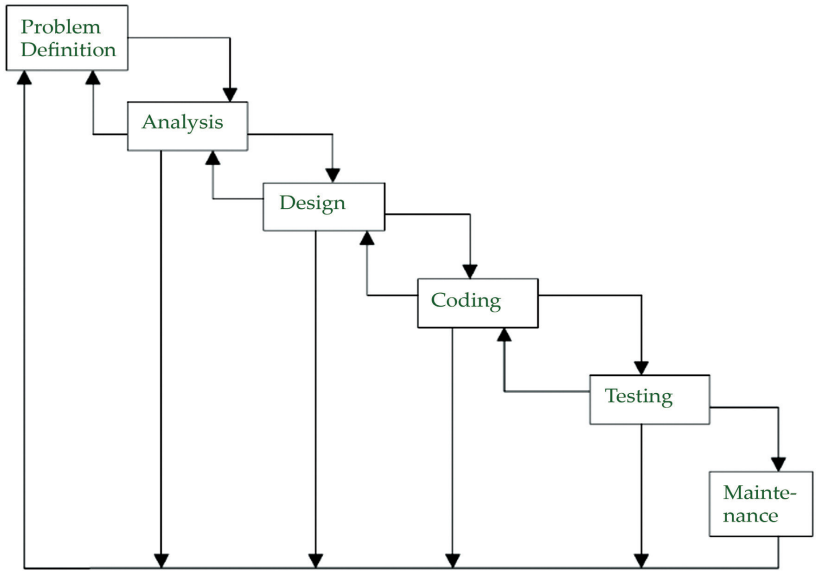
Design: The design phase deals with various theories of system design for example software architecture, data structure, and algorithms. This period answers the questions of how.

Coding: Coding refers to the translation of the design into machine-readable form. The more detailed the design, the easier is its dependability, and better the coding.

Testing: It should be tested rigorously for correctness of the code and results, once the code is written. Testing may call for the systems that are whole and the individual units. It demands a detailed plan as to what, when and how to test.

Maintenance: After the software has been installed, it may experience some changes. This may happen due to a change in the user's condition, a change in the operating environment, or an error in the software that is been mended during the testing. Care ensures that these changes are included wherever required.

- Object–Oriented Approach
  - System is defined as a collection of objects that work together to accomplish tasks
    1. Objects carry out actions when asked
    2. Each object maintains its own data



Classic software development life cycle

Each stages of the life cycle has its own goals and outputs. The output of one stage acts as an input to another period.

The software life cycle, as described below, is regularly implemented using the functional decomposition technique, popularly called top-down, modular strategy. The functional decomposition technique is based on the interpretation of the problem space and its translation as an inter-dependent set of functional into the solution space. The closing system is seen as a set of functions that are organized in a top-down hierarchal structure.

There are several flaws in the top-down, functional decomposition strategy. They include:

1. It will not permit evolutionary changes in the software.
2. Just one function characterizes the system at the top which is not always accurate. Actually many systems have no top.
3. Data is not given the importance it deserves.
4. Reusability of the code does not support.

Development Tools

A high number of tools are used in the analysis and design of the systems. It is important to note the process of systems development has been undergoing changes over the years due to continuous changes in the computer technology. Hence, there has been an evolution of new system development tools and techniques. These tools and techniques provide responses to how questions of the system development.

The development tools available today may be classified as the first generation, second generation, and third generation tools. The first generation tools developed in 1970's and the 1960's are called the conventional tools. The second generation tools introduced in the late 1970's and early 1980's are meant for the structured systems analysis and design and therefore they are known as the tools that were structured. The recent tools are the third generation ones evolved since late 1980's to satisfy the object-oriented analysis and design.

Table shows some of the popular tools used for various development processes under the three groups. It gives a fair idea if the advancement of the tools during the last three decades, although this categorization is questionable.

This section gives an overview of some of the most often used first and second-generation tools. Object oriented development software will be later in this chapter (as and when they are required).

Process	First generation	Second generation	Third generation
Physical processes	System flowcharts	Context diagrams	Inheritance graphs Object-relationship charts
Data representation	Layout forms Grid charts	Data dictionary	Objects object dictionary
Logical processes	Playscript English narrative	Decision tables &trees Data flow diagrams	Inheritance graphs Data flow diagrams
Program representation	Program flowcharts I/O layouts	Structure charts Warnier /Orr diagrams	State change diagrams Ptech diagrams Coad/Yourdon charts

Data flow diagrams: They describe the stream of data between various parts of a system. It is a network representation of the system including data and procedures files.

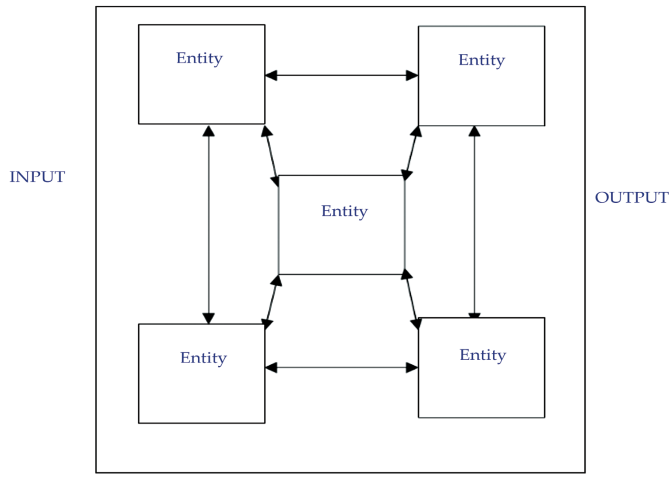
Data dictionary: A structured repository of data about data. It includes a list of terms and their definitions for save and all the data items.

Structure chart: A graphical rendering of the control logic of functions (modules) representing a system.

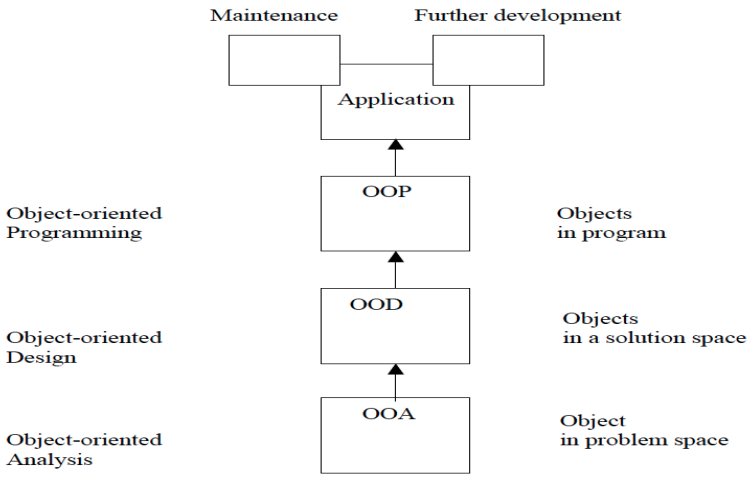
Decision table: A table for identifying a problem and the activities to be taken of settings. It presents the sense that tells us what actions to take when a specified state is true or otherwise.

Decision tree: A graphical portrayal of results that resemble the branches of a tree and the state.

Warnier / Orr diagrams: A flat hierarchy graph using nested sets of reasoning symbols, and braces, pseudo codes to show the application construction.



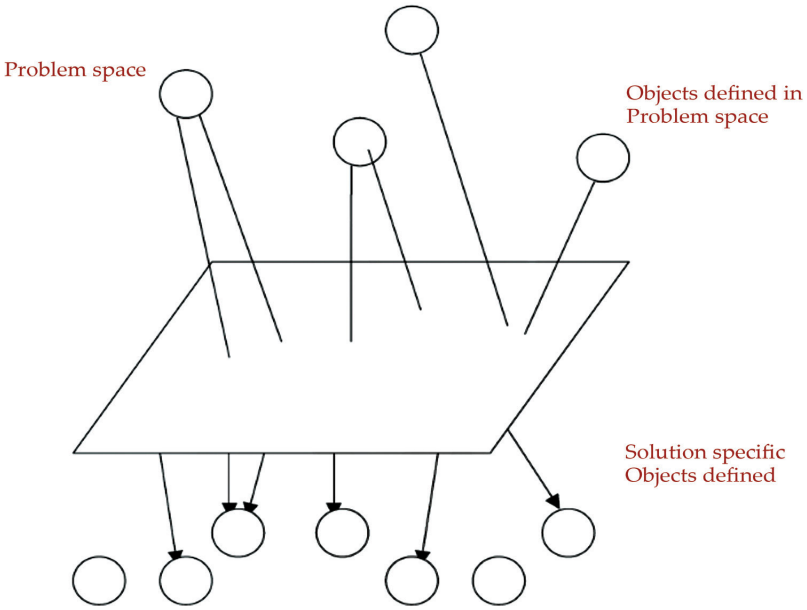
As the name indicates, the object oriented paradigm places greater emphasis on the objects that encapsulate data and procedure. They play the fundamental role in all the stages of the software development and, consequently, there exists a high degree of iteration and overlap between the periods. The whole development process becomes evolutionary in nature. Any graphical representation of the object-oriented version of the software development life cycle must, consequently, take into account these two facets of overlap and iteration. This model depicts that the development reaches a higher amount just to fall back to a previous degree and then again climbing up.



Object oriented analysis (OOA) refers to the methods of specifying requirements of the software in the terms of real-world objects, their conduct, and their interactions. Object-oriented design (OOD), on the other hand, turns the software requirements into specifications for objects and derives class hierarchies from which the objects can be created. Lastly, object-oriented programming (OOP) refers to the enactment of the program using objects, in an object-oriented programming language such as C.

By developing specifications of the objects discovered in the problem space, a clear and well-organized statement of the problem is really assembled into application. These objects form a high-level layer of definitions that are written in terms of the problem space. During the refinement of the definitions and the execution of the application objects, other objects and identified.

All the periods in the object oriented approach work more closely together because of the commonality of the object model. While in the next phase additional objects needed for a special solution are defined, in one stage, the problem domain objects are identified. The design process is duplicated for these implementation-level objects.



In contrast to the conventional top down functional decomposition approach, the object oriented strategy has many aspect of both the top-down and bottom-up design. The top practical decomposition techniques can be applied to the design of individual classes, while the final system can be constructed with the help of type modules using the bottom-up approach.

- OO Analysis and Design
  - Prototyping
    - Creating a working model of one or more parts of the system for user evaluation and feedback
- Joint Application Development (JAD)
    - Key system stakeholders and decision makers work together to rapidly define system requirements and designs

"The problem with object-oriented languages is they've got all this implicit environment that they carry around with them. You wanted a banana but what you got was a gorilla holding the banana and the entire jungle."

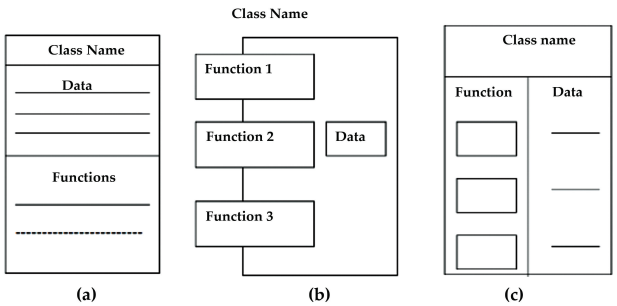
—Joe Armstrong

Object-Oriented Notation and Graphs

Graphical notations are an essential part of any development and design procedure, and object-oriented design is no exception. We need notations to signify subclasses, classes, objects, and their inter relationships. Regrettably, there are not any standard notations for representing their interactions and the object. Researchers have used their own notations. Some of them are used more frequently while others are not. Show some of the commonly used notations to represent the following:

1. Classes and objects.
2. Instances of objects.
3. Message communication between and objects.
4. Inheritance relationship.
5. Classification relationship.
6. Composition relationship.
7. Hierarchical chart.
8. Client-Server relationship.
9. Process layering.

Wherever possible we must use graphs and these notations. They improve not only the clarity of the processes but also the productivity of the software developers.

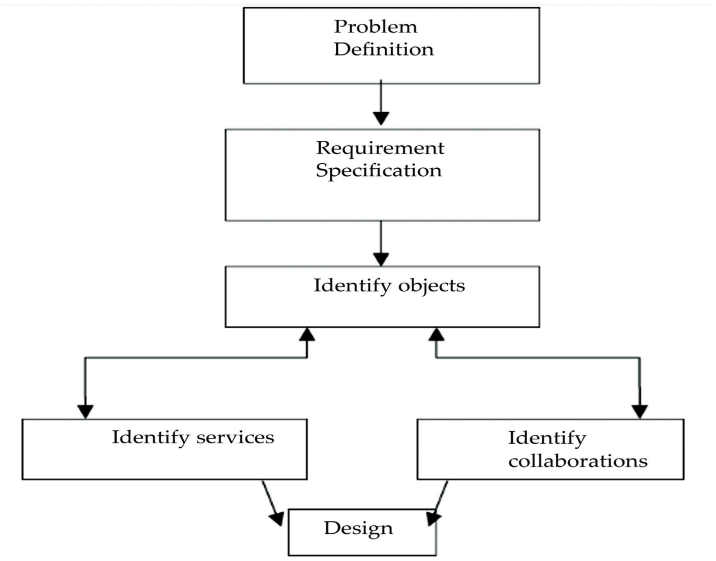


OBJECT–ORIENTED ANALYSIS

Object oriented analysis provides us with simple, yet powerful, mechanism for identifying objects, the building block of the software to be developed. The analysis is creating a reasonable model to describe the system functions and basically concerned with the decomposition of a problem into its component parts.

The object oriented analysis (OOA) strategy consists of the following steps:

1. Understanding the problem.
2. Drawing the specification of requirement of the software and the user.
3. Identifying the objects and their attributes.
4. Identifying the services that each object is expected to provide (interface).
5. Establishing interconnections (cooperation) between the objects in terms of services demanded and services left.



Problem Understanding

The first step in the analysis process is to understand the problem of the user. The problem statement should be refined and redefined in terms of computer system engineering that could indicate a computer-based solution. The problem statement

should be stated, as far as possible, in a single, grammatically correct sentence. This will enable the software engineer to have a highly focused attention on the solution of the problem.

The problem statement provides the basis for drawing the requirements specification of both the user and the software.



Requirements Specification

The next step is to comprehend what the planned system is required to do, once the problem is clearly identified. It really is important at this phase to generate a list of user requirements. A clear understanding should exist between user and the programmer of what is needed. Based on the user requirements, the specification for the software should be drawn. The developer should state clearly:

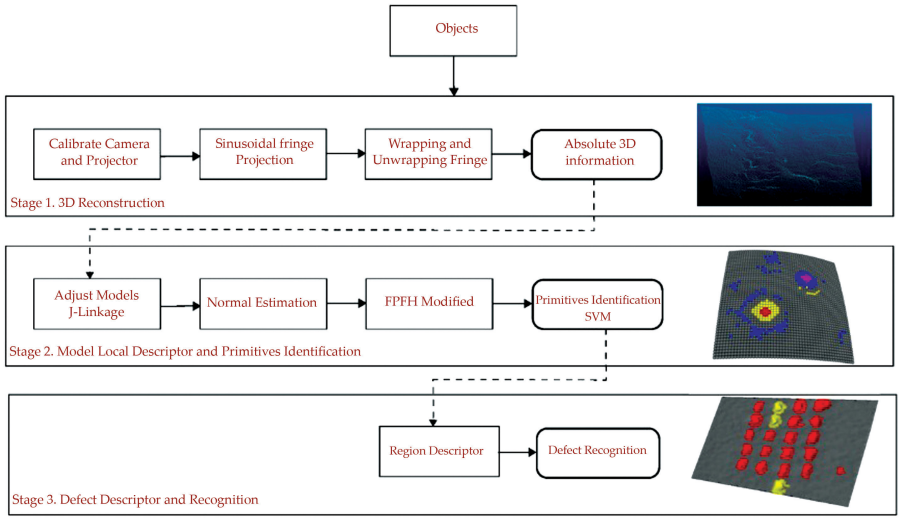
- What output signals are demanded?
- What procedures are required to generate these output signals?
- What inputs are needed?
- What resources are needed?

These specifications often server as the ultimate product to be tested by a reference for its functionality of the jobs that are intended.

Identification of Objects

Things can often be identified in terms of the abstract objects as well as the real world objects. Hence, the best place to look for object is the application itself. The programs may be assessed by using one of the following two approaches:

1. Data flow diagrams (DFD)
2. Textual analysis (TA)



Data flow diagram

The application can be represented in the form of a data flow diagram indicating how the data moves from one point to another in the system, the boxes and data store in the data flow diagram are good candidate for the objects. The process bubbles correspond to the procedures.

Textual Analysis

This approach is based on the textual description of the problem or proposed solution. The description may be of one or two sentences or one or two paragraphs depending on the type and complexity of the problem. The nouns are good indicators of the objects. The names can further the classified as proper nouns, common nouns, and mass or abstract nouns.



Establishing Interconnections

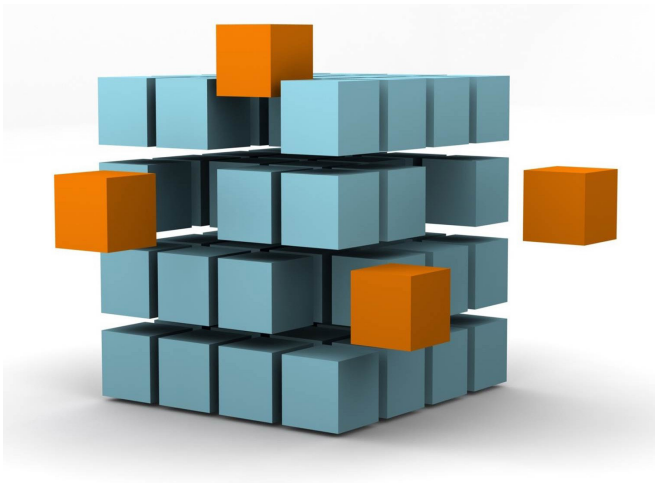
This step identifies the services that objects provide and receive. We may use an information flow diagram (IFD) or an entity-relationship(ER) diagram to enlist this information. Here, we must establish a correspondence between the services and the actual information that are being communicated.

"The object-oriented model makes it easy to build up programs by accretion. What this often means, in practice, is that it provides a structured way to write spaghetti code."

—Paul Graham

OBJECT-ORIENTED DESIGN

Design creating computational models of the system and an overall structure, and is concerned with a mapping of objects in the problem space into objects in the solution space. This period usually uses the bottom up strategy to build the structure of the system and the top down functional decomposition strategy to design the class member function that provides services. It is particularly significant to assemble hierarchies that are structured, to identify abstract classes, and to simplify the inter-object communications. Reusability of classification of the objects into subsystems, classes from the preceding designs and determination of proper protocols are some of the factors of the design phase.



The object oriented design (OOD) strategy may involve these steps:

1. Review of objects created in the analysis phase.
2. Specification of class dependencies.
3. Organization of class hierarchies.
4. Design of classes.

"Object Oriented" has clearly become the buzzword of choice in the industry. Almost everyone talks about it. Almost everyone claims to be doing it, and almost everyone says it is better than traditional function oriented design. Object oriented design is the result of focusing attention not on the function performed by the program, but instead on the data that are to be manipulated by the program. Thus, it is orthogonal to function oriented design.

Object Oriented Design begins with an examination of the real world "things" that are part of the problem to be solved. These things (which we will call objects) are characterized individually in terms of their attributes (transient state information) and behavior (functional process information). Each object maintains its own state, and offers a set of services to other objects. Shared data areas are eliminated and objects communicate by message passing (e.g. parameters). Objects are independent entities that may readily be changed because all state and representation information is held within the object itself. Objects may be distributed and may execute either sequentially or in parallel.

Object Oriented Design is not dependent on any specific implementation language. Problems are modelled using objects. Objects have

- Behavior (they do things)
- State (which changes when they do things)

The various terms related to object oriented design are Objects, Classes, Messages, Abstraction, Inheritance and Polymorphism.

Objects - An object is an entity able to save a state (information) and which offers a number of operations (behavior) to either examine or affect this state. Hence an object is characterized by number of operations and a state which remembers the effect of these operations. All objects have unique identification and are distinguishable.

Classes - In any system, there shall be number of objects. Some of the objects may have common characteristics and we can group the objects according to these characteristics. This type of grouping is known as a class. Hence, a class is a set of objects that share a common structure and a common behavior.

Messages - Conceptually, objects communicate by message passing. Messages consists of the identity of the target object, the name of the requested operation and any other operation needed to perform the function.

Abstraction-In object oriented design, complexity is managed using abstraction. Abstraction is the elimination of the irrelevant and the amplification of essentials. We can teach someone to drive any car using an abstraction. We amplify the essentials: we teach about the ignition and steering wheel, and we eliminate the details, such as details of the particular engine in this car or the way fuel is pumped to the engine

Inheritance - Inheritance is the technique that is used to build new classes from existing ones and to build object oriented class hierarchies. You can build layers of classes derived from other classes.

Polymorphism - When we abstract just the interface of an operation and leave the implementation to subclasses it is called a polymorphic operation and process is called polymorphism

Class Dependencies

Evaluation of relationships between the classes is central to the construction of a system. So, it is important to identify proper classes create their relationships and to represent the objects in the solution space. The major relationships that are significant in the context of design are:

- Inheritance relationships.
- Containment relationship.
- Use relationships.

Inheritance relationship is the highest relationship that can be represented in C. It is a powerful way of representing a hierarchical relationship directly. The real appeal and power of the inheritance mechanism is that it enables us to reuse a

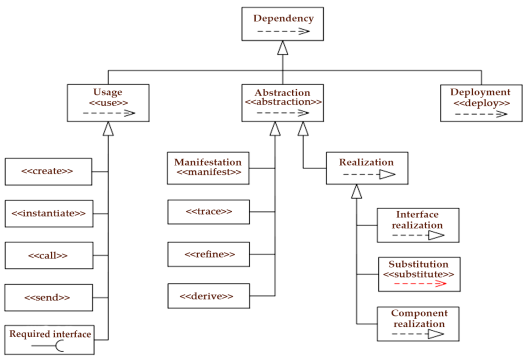
category that is almost, but not exactly, what we desire and to tailor the category in a way that is does not introduce any unwanted side effects into the rest of the class. We must review the attributes and operations of the classes and prepare an inheritance relationship table as shown in Table.

Class	Depends on
A	.....
B	A
C	A
D	B
B1	B
B2	B

Containment relationship means the use of an object of a group as an associate of another class. This is a complimentary and option technique to use the class inheritance. But, it is often a tricky problem to pick between both techniques. Typically, if there is need to override aspects or functions, then the inheritance is the best choice. On the other hand, if we need to represent a property by an assortment of types, then the containment relationship is the correct approach to follow. Another place where we need to use an object as a member is when we need to pass an aspect of a group as an argument to the constructor of another category. The "another" type must have a member object that represents the argument. The inheritance represents is a relationship and the containment represents has a relationship.

Use relationship gives a course uses and the way it uses them to information including the various classes. For example, a class A can use classes B and C in several ways:

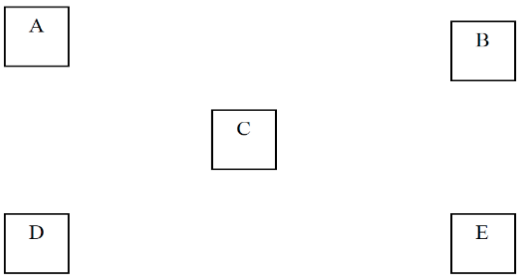
1. A reads member of B
2. A calls an associate of C
3. A creates new operator being used by B



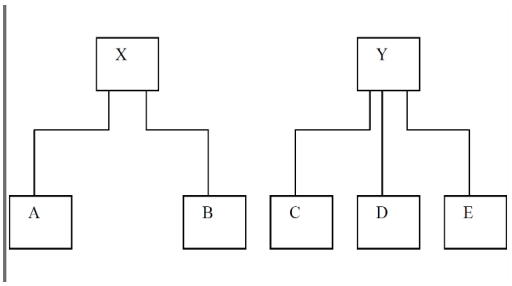
Organization of Class Hierarchies

In the preceding step, we examined the inheritance relationships. We must re-examine create a type hierarchy show that we can reuse as much data and/or functions which have been designed already and them. Organization of the category hierassssrchies includes identification of functions and common characteristics among a group of related classes and then combining them to form a new class. The new class will function as subordinate classes as the super group and the others. The new category might or might not have the meaning of an object by itself. If the object is created solely to combine the common traits, it is called an abstract class.

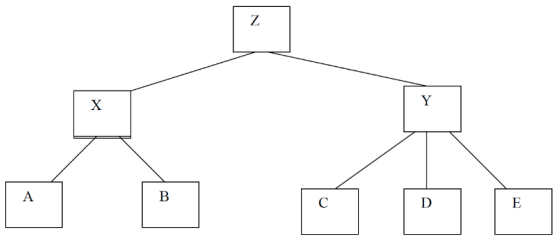
This process may be replicated at different degree of abstraction with the only aim of expanding the classes. As hierarchy construction becomes increasingly higher, the amount of implementation and specification inherited by the lower level classes increase. We way repeat the process until we are confident that no new group can be formed.



Objects in solution space



First level of hierarchy



Second level of hierarchy

Design of Classes

We have identified minimal set of operations needed by the concept a course is signifying, their attributes, and classes. Now we must take a look at the complete details that each group represents. The issues that are significant is to determine what function are to be supplied. For a category to be useful, it must contain these functions, in addition to service functions:

1. Class management functions.

- How an object is created?
- How an object is ruined?

2. Category enactment functions.

   What operations are performed on the data type of a class?

3. Classes access functions.

   How do we get information about the internal variables of the category?

4. Type utility functions.

How do we handle errors?

Other issues that are to be considered are:

- What types of access controls are needed for the base classes?
- Which function can be made virtual?
- What library classes are expected to be used in a class?
- The design of the classes that are individual has a major impact on the overall quality of the software/

Given below are some guidelines, which should be considered while designing a class:

- The public interface of a class should have only functions of the type.
- An object of one type must not send a message directly to a member of another class.
- A function should be declared public only when it really is required to be used by the objects of the group.
- A group should be dependent on as few classes as possible.
- Interaction between two classes must be explicit.
- The top type of a construction should represent the theoretical model of the goal concept.

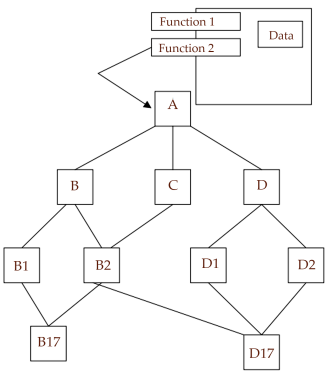
"Object-oriented programming is an exceptionally bad idea which could only have originated in California."

—Edsger Dijkstra

Design of Member Functions

We have so far identified Classes and objects, Data members, Interfaces, Dependencies, and Class hierarchy (structure).

It is time now to consider the design of the member functions. The member functions define the operations that are performed on the object's data. These functions behave like any other C function and therefore we can use the top-down functional decomposition technique to design them as shown in below figure.



Design of the Driver Program

Every C++ program must contain a main () function code known as the driver application. The executions of the program begin and end here. The driver program is primarily responsible for:

- Receiving data values from the user,
- Creating objects from the group definitions,
- For invoking the member Ordering communicating between the objects as a sequence of message functions, and
- Displaying output results in the form needed by the user.

All activities, including processing during the execution of the program, result from the reciprocal interactions of the objects. One important design choice made is the rational arrangement of the messaging passing.

Implementation

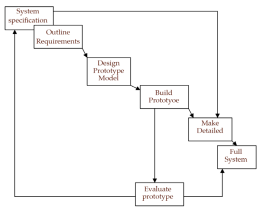
Implementation contains testing and coding. Coding include writing codes for the main program, member function and classes that acts as a driver in the program. Coding becomes easy once a detailed design has been done with attention. No program works correctly the first time. So before using testing the program is an essential part of the software development process. A comprehensive test plan should be drawn as to what, when and how to test. Class dependencies and the group interfaces are important features for testing. The final aim of testing is to see that the system performs its intended job satisfactorily.



Prototyping Paradigm

Most frequently the real world application problems are complicated in nature and therefore the arrangement of the system becomes too large to work out the exact requirements at the start. After the big system is finished, incorporation of any attributes that has been identified as "missing" at the testing or program period might be too expensive and time consuming. One way of understanding the system design and its ramifications before an entire system is assembled is to build and test a working model of the proposed system. A prototype is scaled down version of the system and may not have resource requirements and stringent performance criteria. Developer and customer agree upon particular "Outline specifications" of the system and a prototype design is proposed with available resources and the outline conditions. The leading interest is not in the prototype itself but in its performance, which is used to refine the demand specifications. Prototype analyzes various aspects of the system for example system structure, internal design, hardware demands and the final system demands and provides an opportunity to experiment. The benefits of using the prototype strategy are:

- We can make clear specifications, which are, complete and correct as far as possible.
- The user can realize what is being offered.
- Care changes that are needed when a system is installed are minimized.
- Development engineers can work from a set of specifications, which have been analyzed and approved.



Prototype is meant for experimenting. Most often it cannot be tuned into a product. However, occasionally, it may be possible to tune a prototype into a final product if proper care is taken in redesigning the prototype.

KEY TERMS

Traditional programming
Polymorphism
Data dictionary
Decision tree
Software development life cycle
Encapsulation
Software development
Decision table
Graphical representation

DISCUSSION QUESTIONS

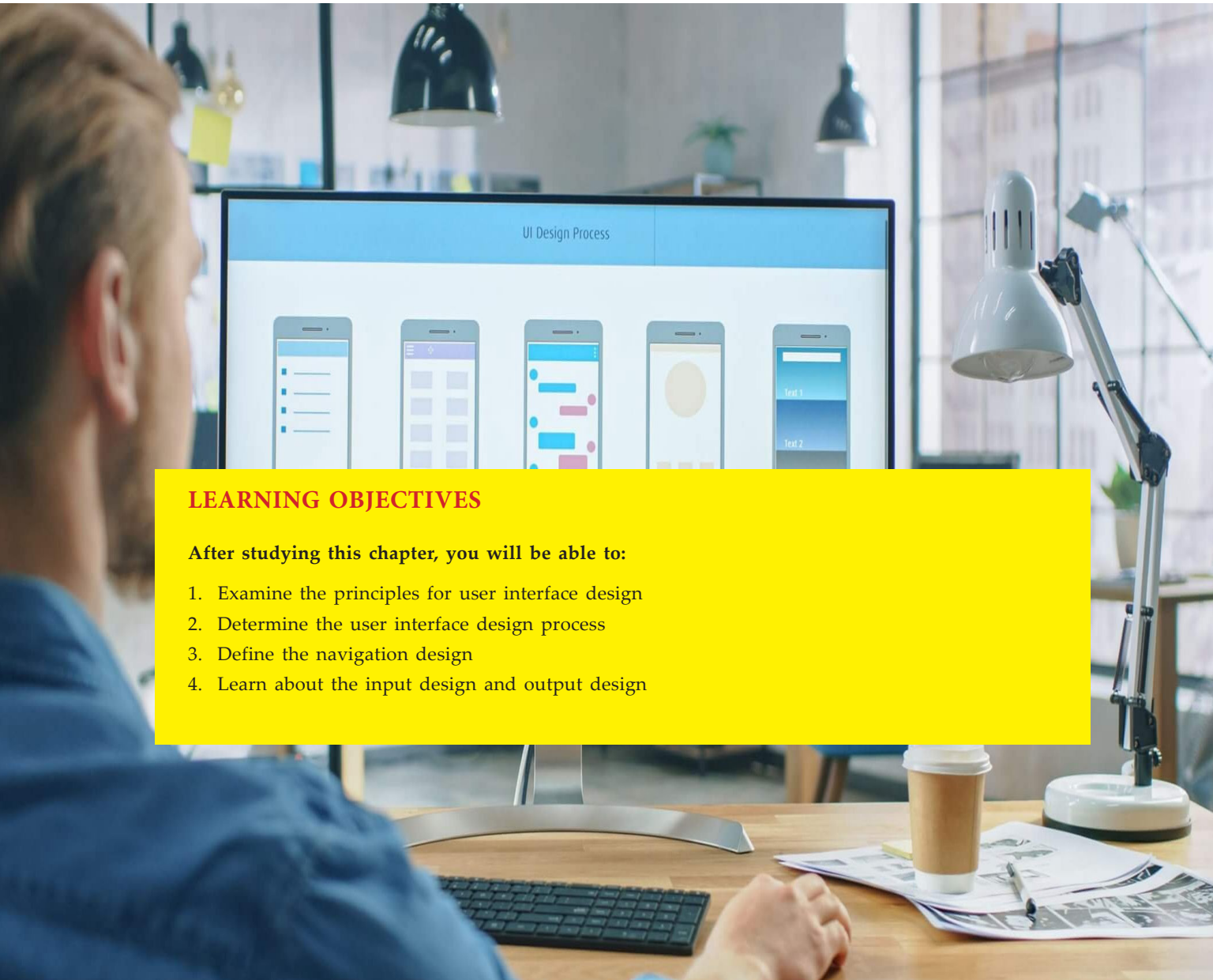
1. What do you understand by procedure-oriented paradigms?
2. What are procedure-oriented development tools?
3. Define the object-oriented notation and graphs?
4. What is object–oriented analysis?
5. How can you design classes?

REFERENCES

1. Giudice, M.D.; Soto-Acosta, P.; Carayannis, E.; Scuotto, V. Emerging perspectives on business process management (BPM): IT-based processes and ambidextrous organizations, theory and practice. Bus. Process
2. Knaepen, K.; Brooms, D. A Complete and Consistent Business: Introduction to the COSTA Model for Business Architects; Lannoo Publishers/Racine, Business & Economics: Worcester, MA, USA, 2013; pp. 1–186.
3. Meidan, A.; García-García, J.A.; Escalona, M.J.; Ramos, I. A survey on business processes management suites. Comput. Stand. Interfaces 2017, 51, 71–86.
4. Osis, J.; Asnina, E.; Grave, A. MDA Oriented Computation Independent Modeling of the Problem Domain. ENASE. In Proceedings of the 2nd International Conference on Evaluation of Novel Approaches to Software Engineering, Barcelona, Spain, 23–25 July 2007.
5. Pattanayak, S.; Roy, S. Synergizing Business Process Reengineering with Enterprise Resource Planning System in Capital Goods Industry. Procedia—Soc. Behav. Sci. 2015, 189, 471–487.
6. Podeswa, H. UML for the IT Business Analyst: A Practical Guide to Object-Oriented Requirements Gathering, 2nd ed.; Cengage Learning: Scarborough, ON, Canada, 2009.
7. Thiede, M.; Fuerstenau, D.; Barquet, A.P.B. How is process mining technology used by organizations? A systematic literature review of empirical studies. Bus. Process Manag. J. 2018, 24, 900–922.

CHAPTER 7

USER INTERFACE DESIGN



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Examine the principles for user interface design
2. Determine the user interface design process
3. Define the navigation design
4. Learn about the input design and output design

INTRODUCTION

The user interface (UI) is the space where interactions between humans and machines occur. UI is an integral aspect of user experience (UX) that consists of two major parts: visual design, which conveys the look and feel of a product; and interaction design, which is the functional and logical organization of elements. The goal of user interface design is to create a user interface that makes it easy, efficient, and enjoyable for users to interact with a product. Learn about the different types of UI as well as fundamental design requirements for each type.

UI design prioritizes the user's visual experience. A good user interface is functional, reliable, and enjoyable to use. User interface design should minimize the effort that the user has to invest interacting with a product and help users accomplish their goals with ease.

Designers use methods of user-centered design (user interviews, direct observations, etc.) to learn about their target audience and ensure that the visual language they introduce in the UI is well-tailored to them. It's important to make the UI aesthetically pleasing because interfaces that have great aesthetics are perceived as more usable by end-users, according to the aesthetic–usability effect.

PRINCIPLES FOR USER INTERFACE DESIGN

In many ways, user interface design is an art. The goal is to make the interface pleasing to the eye and simple to use, while minimizing the effort users expend to accomplish their work. The system is never an end in itself; it is merely a means to accomplish the business of the organization. We have found that the greatest problem facing experienced designers is using space effectively. Simply put, there is more information to present than room to present it. Analysts must balance the need for simplicity and pleasant appearance against the need to present the information across multiple pages or screens, which decreases simplicity. In this section, we discuss some fundamental interface design principles, which are common for navigation design, input design, and output design.

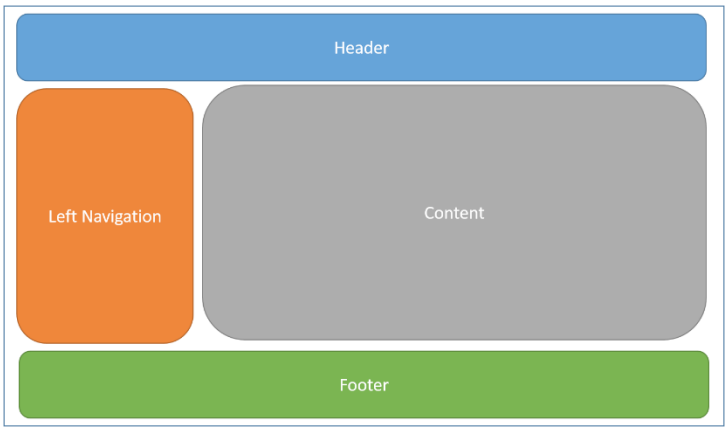
User interface design principles

Principle	Description
User familiarity	The interface should use terms and concepts which are drawn from the experience of the people who will make most use of the system.
Consistency	The interface should be consistent in that, wherever possible, comparable operations should be activated in the same way.
Minimal surprise	Users should never be surprised by the behaviour of a system.
Recoverability	The interface should include mechanisms to allow users to recover from errors.
User guidance	The interface should provide meaningful feedback when errors occur and provide context-sensitive user help facilities.
User diversity	The interface should provide appropriate interaction facilities for different types of system user.

Layout

The first principle of user interface design deals with the layout of the screen, form, or report. Layout refers to organizing areas of the screen or document for different purposes and using those areas consistently throughout the user interface. Most software designed for personal computers follows the standard Windows or Macintosh approach for screen layout. This approach divides the screen into three main areas: The top area provides the user with ways to navigate through the system; the middle (and largest) area is for display of the user's work; and the bottom area contains status information about what the user is doing.

In many cases (particularly on the Web), multiple layout areas are used



The top area provides the standard web browser navigation and command controls that change the contents of the entire system. The navigation area on the left edge maneuvers between sections and changes all content to its right. The other two section navigation areas at the top and bottom of the page provide other ways to navigate between sections. The content in the middle of the page displays the results (i.e., software review articles) and provides additional navigation within the page about these reviews. This use of multiple layout areas for navigation also applies to inputs and outputs. Data areas on reports and forms are often subdivided into subareas, each of which is used for different types of information. These areas are almost always rectangular in shape, although sometimes space constraints will require odd shapes.

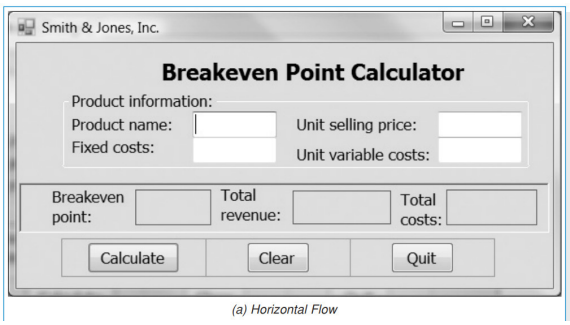
Nonetheless, the margins on the edges of the screen should be consistent. Each of the areas within the report or form is designed to hold different information. For example, on an order form (or order report), one part may be used for customer information (e.g., name, address), one part for information about the order in general (e.g., date, payment information), and one part for the order details (e.g., how many units of which items at what price each). Each area is self-contained so that information in one area does not run into another. The areas and information within areas should have a natural intuitive flow to minimize users' movement from one area to the next. People in Western nations (e.g., Europe, North America) tend to read top to bottom, left to right, so that related information should be placed so that it is used in this order (e.g., address lines, followed by city, state/province, and then ZIP code/postal code.)

Sometimes, the sequence is in chronological order, or from the general to the specific, or from most frequently to least frequently use.

In any event, before the areas are placed on a form or report, the analyst should have a clear understanding of what arrangement makes the most sense for how the form or report will be used. The flow between sections should also be consistent, whether horizontal or vertical. Ideally, the areas will remain consistent in size, shape, and placement for the forms used to enter information (whether on paper or on a screen) and the reports used to present it.

Content Awareness

Content awareness refers to the ability of an interface to make the user aware of the information it contains with the least amount of effort by the user. All parts of the interface, whether navigation, input, or output, should provide as much content awareness as possible, but it is particularly important for forms or reports that are used quickly or irregularly (e.g., a Web site). Content awareness applies to the interface in general. All interfaces should have titles (on the screen frame, for example). Menus should show where the user is and, if possible, where the user came from to get there. All areas should be clear and well defined (with titles if space permits) to reduce the chances that users become confused about the information in any area. Then users can quickly locate the part of the form or report that is likely to contain the information they need. Sometimes the areas are marked by lines, colors, or headings; in other cases, the areas are only implied. Content awareness also applies to the fields within each area. Fields are the individual elements of data that are input or output. The field labels that identify the fields on the interface should be short and specific—objectives that often conflict. There should be no uncertainty about the format of information within fields, whether for entry or display. For example, a date of 10/5/12 means different things, depending on whether you are in the United States (October 5, 2012) or in Canada (May 10, 2012). Any fields for which there is the possibility of uncertainty or multiple interpretations should provide explicit explanations. Content awareness also applies to the information that a form or report contains. In general, all forms and reports should contain a preparation date (i.e., the date printed or the date data were completed) so that the age of the information is obvious.



(a) Horizontal Flow

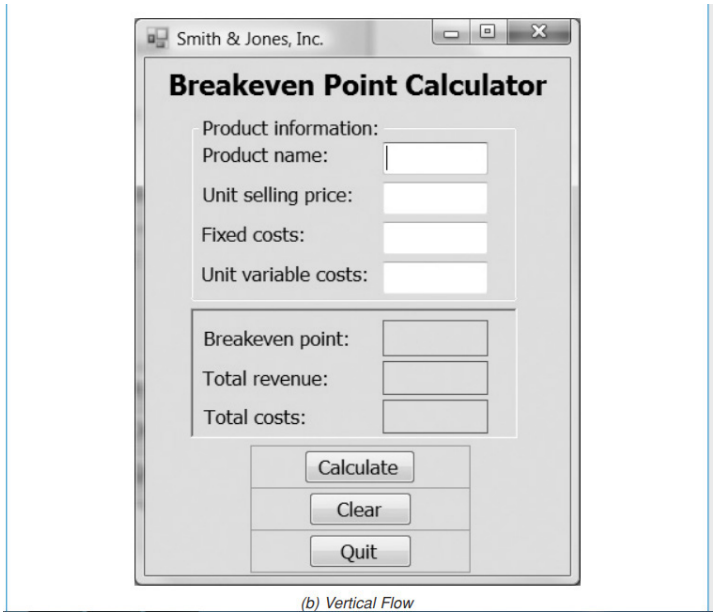


Figure 7.1: Interface Flow between Sections.

Likewise, all printed forms and software should provide version numbers so that users, analysts, and programmers can identify outdated materials.

"Design is the fundamental soul of a human-made creation that ends up expressing itself in successive outer layers of the product or service."

—Steve Jobs

USER INTERFACE DESIGN PROCESS

User interface design is a five-step process that is iterative—analysts often move back and forth between steps rather than proceed sequentially from step 1 to step 5 (Figure 7.2). First, the analysts examine the DFDs and use cases developed in the analysis phase and interview users to develop use scenarios that describe users' commonly employed patterns of actions so that the interface can enable users to quickly and smoothly perform these scenarios. Second, the analysts develop the interface structure diagram (ISD) that defines the basic structure of the interface. This diagram (or set of diagrams) shows all the interfaces (e.g., screens, forms, and reports) in the system and how they are connected. Third, the analysts design interface standards,

which are the basic design elements on which interfaces in the system are based. Fourth, the analysts create an interface design prototype for each of the individual interfaces in the system, such as navigation controls, input screens, output screens, forms (including preprinted paper forms), and reports. Finally, the individual interfaces are subjected to interface evaluation to determine whether they are satisfactory and how they can be improved. Interface evaluations almost always identify improvements, so the interface design process is repeated in a cyclical process until no new improvements are identified. In practice, most analysts interact closely with the users during the interface design process, so that users have many chances to see the interface as it evolves rather than waiting for one overall interface evaluation at the end of the interface design process. It is better for all concerned (both analysts and users) if changes are identified sooner rather than later.

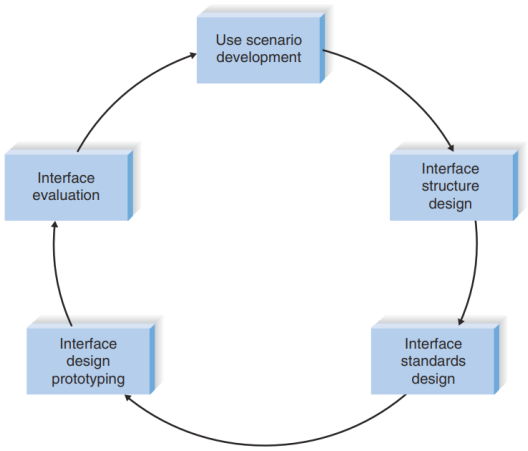


Figure 7.2: User Interface Design Process

For example, if the interface structure or standards need improvement, it is better to identify changes before most of the screens that use the standards have been designed.

Use Scenario Development

A use scenario is an outline of the steps that the users perform to accomplish some part of their work. A use scenario is one commonly used path through a use case. We model the two processes separately and write the programs separately because

they are separate processes within process 1 (Search and Browse Tunes). The DFD was designed to model all possible uses of the system—that is, its complete functionality or all possible paths through the use case. But use scenarios are just one path through the use case. In one use scenario, for example, a user will browse through many tunes, much like someone browsing through a real music store looking for interesting music. He or she will search for a tune, listen to a sample, perhaps add it to the shopping cart, browse for more, and so on. Eventually, the user will want to purchase the download(s), perhaps removing some selections from the shopping cart beforehand. In another use scenario, a user will want to buy one specific tune. He or she will go directly to the tune, price it, and buy it immediately, much like someone running into a store, making a beeline for the one item he or she wants, and immediately paying and leaving the store. This user will enter the tune information in the search portion of the system, look at the resulting cost information, and immediately buy the download or leave. Anything that slows him or her down will risk losing the sale. For this use scenario, we need to ensure that the path through the DFD as presented by the interface is short and simple, with very few menus and mouse clicks. Use scenarios are presented in a simple narrative description that is tied to the DFD. Figure 7.3 shows the two use scenarios just described. The key point in using use scenarios for interface design is not to document all possible use scenarios within a use case, because then you end up just repeating the DFD in a different form. The goal is to describe the handful of most commonly occurring use scenarios so that the interface can be designed to enable the most common uses to be performed simply and easily.

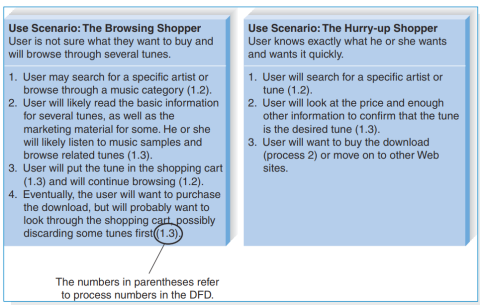


Figure 7.3: Two Use Scenarios for the Search and Browse Tunes Use Case.

Interface Structure Design

The interface structure design defines the basic components of the interface and how they work together to provide functionality to users. An interface structure diagram (ISD) is used to show how all the screens, forms, and reports used by the system are related and how the user moves from one to another. Most systems have several ISDs, one for each major part of the system. An ISD is somewhat similar to a DFD in that it uses boxes and lines to show structure. However, unlike DFDs, there are no commonly used rules or standards for their development. With one approach, each interface element (e.g., screen, form, report) on an ISD is drawn as a box and is given a unique number (at the top) and a unique name (in the middle). The numbers usually follow a tree-type structure, although this is not always done. Unlike the DFDs, however, the numbers do not mean that all the screens belong to "parents" higher in the tree; instead, they usually imply relationships between a menu and a submenu. The lines denote the ability to navigate from one menu to another. Each box on the ISD also shows (at the bottom) the DFD process that is supported by the interface (Figure 7.4). Sometimes, there is more than one interface for a given process (e.g., in Figure 7.4, interfaces 1.1 through 1.3 support process 1.1.1); in other cases, there is only one interface for each process (e.g., interfaces 3.1 through 3.3 support processes 1.1.3.1 through 1.1.3

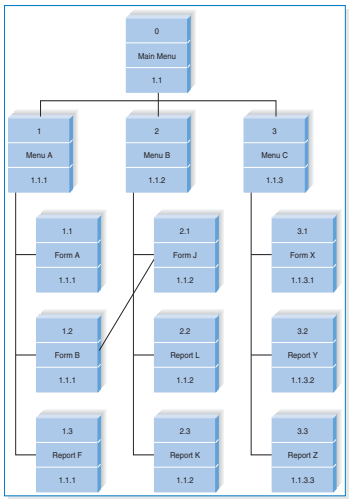


Figure 7.4: Example Interface Structure Diagram.

Each interface is linked to other interfaces by lines that show how users can transition from one interface to the next. In most cases, the interfaces form a hierarchy, or a tree; but sometimes, an interface is linked to one outside of the hierarchy, as shown by the link from Form J to Form B (e.g., the ability to update customer information, such as address, while entering a new order). The basic structure of the interface follows the basic structure of the business process itself as defined in the process model. The analyst starts with the DFD and develops the fundamental flow of control of the system as it moves from process to process.

There are usually several major parts to an information system, each of them distinct, in the same way that there are several high-level processes in a DFD. In general—but not always—there is one ISD for each process on the level 0 DFD. The analyst then examines the use scenarios to see how well the ISD supports them. Quite often, the use scenarios identify paths through the ISD that are more complicated than they should be. The analyst then reworks the ISD to simplify the ability of the interface to support the use scenarios, sometimes by making major changes to the menu structure, sometimes by adding shortcuts.

Interface Standards Design

The interface standards are the basic design elements that are common across the individual screens, forms, and reports within the system. Depending on the application, there may be several sets of interface standards for different parts of the system (e.g., one for Web screens, one for paper reports, one for input forms). For example, the part of the system used by data-entry operators may mirror other data-entry applications in the company, whereas a Web interface for displaying information from the same system may adhere to some standardized Web format. Likewise, each individual interface may not contain all the elements in the standards (e.g., a report screen may not have an "edit" capability), and they may contain additional characteristics beyond the standard ones, but the standards serve as the touchstone which ensures that the interfaces are consistent across the system

Interface Metaphor

First of all, the analysts must develop the fundamental interface metaphor(s) that defines how the interface will work. An interface metaphor is a concept from the real world that is used as a model for the computer system. The metaphor helps the user to understand the system and enables the user to predict what features the interface might provide, even without actually using the system. Sometimes systems have one metaphor, whereas in other cases there are several metaphors in different parts of the system. In many cases, the metaphor is explicit. Quicken, for example, uses a checkbook metaphor for its interface, even to the point of having the users type information into an on-screen form that looks like a real check register. In other cases, the metaphor is implicit, or unstated, but it is there nonetheless. Many Windows systems use the paper form or table as a metaphor. In some cases, the metaphor is so obvious that it requires no thought.

The Tune Source Digital Music Download system, for example, will use the retail music store as the metaphor (e.g., shopping cart). In other cases, a metaphor is hard to identify. In general, it is better not to force a metaphor that really doesn't fit a system, because an ill-fitting metaphor will confuse users by promoting incorrect assumptions.

Interface Templates

The interface template defines the general appearance of all screens in the information system and the paper-based forms and reports that are used. The template design, for example, specifies the basic layout of the screens (e.g., where the navigation area[s], status area, and form/report area[s] will be placed) and the color scheme(s) that will be applied. It defines whether windows will replace one another on the screen or will cascade on top of each other.

The template defines a standard placement and order for common interface actions (e.g., "File, Edit, View" rather than "File, View, Edit"). In short, the template draws together the other major interface design elements: metaphors, objects, actions, and icons.

Interface Objects

The template specifies the names that the interface will use for the major interface objects, the fundamental building blocks of the system such as the entities and data stores. In many cases, the object names are straightforward, such as calling the shopping cart the "shopping cart." In other cases, it is not simple. For example, Tune Source has chosen to call its digital music downloads "tunes." Some people may refer to individual music selections as "tracks" or "cuts." Obviously, the object names should be easily understood and should help promote the interface metaphor. In general, in cases of disagreements between the users and the analysts over names, whether for objects or actions (discussed later), the users should win. A more understandable name always beats a more precise or more accurate name.

Interface Actions

The template also specifies the navigation and command language style (e.g., menus) and grammar (e.g., object–action order; see "Navigation Design"). The template gives names to the most commonly used interface actions in the navigation design (e.g., "buy" versus "purchase," or "exit" versus "quit").

Interface Icons

The interface objects and actions, and also their status (e.g., deleted, error), may be represented by interface icons. Icons are pictures that will appear on command buttons as well as in reports and forms to highlight important information. Icon design is very challenging because it means developing a simple picture less than half the size of a postage stamp that needs to convey an often-complex meaning. The simplest and best approach is to adopt icons developed by others (e.g., a blank page to indicate "create a new file," a diskette to indicate "save"). This has the advantage of quick icon development, and the icons may already be well understood by users because users have seen them in other software. Commands are actions that are especially difficult to represent with icons because they are in motion, not static. Many icons have become well known from widespread use, but icons are not as well understood as it was at first believed

that they would be. The use of icons can sometimes cause more confusion than insight. (For example, did you know that a picture of a sweeping paintbrush in Microsoft Word means "format painter"?) Icon meanings become clearer with use, but because they are often cryptic, many applications now provide text tool tips that appear when the pointer hovers over an icon. This feature explains the purpose of the icon in words.

"Questions about whether design is necessary or affordable are quite beside the point: design is inevitable. The alternative to good design is bad design, not no design at all."

—Douglas Martin

INTERFACE DESIGN PROTOTYPING

An interface design prototype is a mock-up or a simulation of a computer screen, form, or report. A prototype is prepared for each interface in the system to show the users and the programmers how the system will perform.

In the "old days," an interface design prototype was usually specified on a paper form that showed what would be displayed on each part of the screen. Paper forms are still used today, but more and more interface design prototypes are being built with computer tools instead of on paper. The three most common approaches to interface design prototyping are storyboards, HTML prototypes, and language prototypes.

Storyboard

At its simplest, an interface design prototype is a paper-based storyboard. The storyboard shows hand-drawn pictures of what the screens will look like and how they will flow from one screen to another, in the same way that a storyboard for a cartoon shows how the action will flow from one scene to the next (Fig. 7.5). Storyboards are the simplest technique because all they require is paper (often on a flip chart) and a pen—and someone with some artistic ability

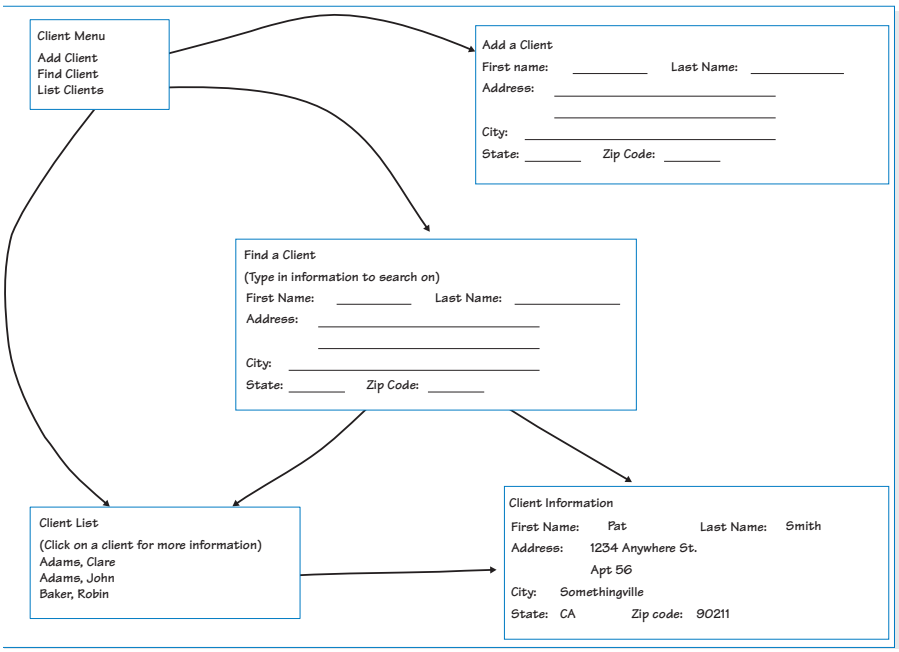


Figure 7.5: An Example Storyboard.

HTML Prototype

One of the most common types of interface design prototypes used today is the HTML prototype. As the name suggests, an HTML prototype is built with the use of Web pages created in HTML (hypertext mark-up language). The designer uses HTML to create a series of Web pages that show the fundamental parts of the system.

The users can interact with the pages by clicking on buttons and entering pretend data into forms (but because there is no system behind the pages, the data are never processed). The pages are linked together so that, as the user clicks on buttons, the requested part of the system appears.

HTML prototypes are superior to storyboards in that they enable users to interact with the system and gain a better sense of how to navigate among the different screens. However, HTML has limitations—the screens shown in HTML will never appear exactly like the real screens in the system (unless, of course, the real system will be a Web system in HTML).

Language Prototype

A language prototype is an interface design prototype built in the actual language or by the actual tool that will be used to build the system. Language prototypes are designed in the same ways as HTML prototypes. (They enable the user to move from screen to screen, but they perform no real processing). For example, in Visual Basic, it is possible to create and view screens without actually attaching program code to the screens. See Figure 7.6 for a sample Visual Basic Language prototype. Language prototypes take longer to develop than do storyboards or HTML prototypes, but they have the distinct advantage of showing exactly what the screens will look like. The user does not have to guess about the shape or position of the elements on the screen.

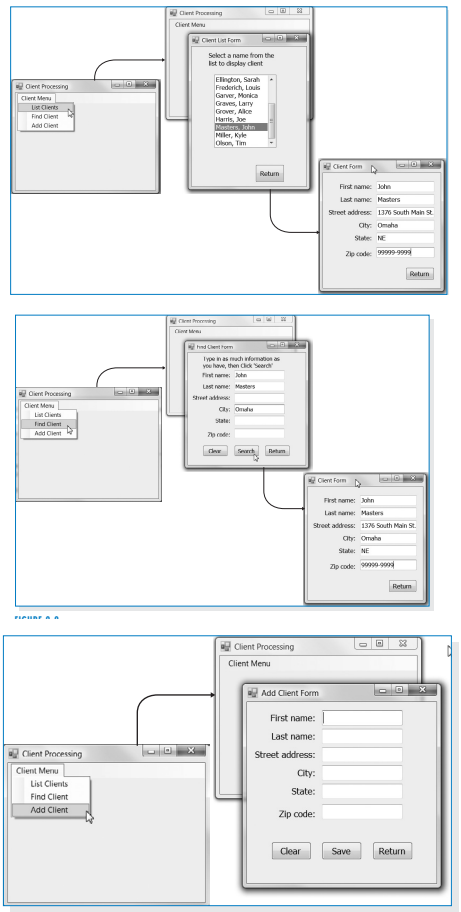


Figure 7.6: Sample Language Prototype.

Selecting the Appropriate

Techniques Projects often use a combination of different interface design prototyping techniques for different parts of the system. Storyboarding is the fastest and least expensive, but provides the least amount of detail. Language prototyping is the slowest, most expensive, and most detailed approach. HTML prototyping falls between the two extremes. Therefore, storyboarding is used for parts of the system in which the interface is well understood and when more expensive prototypes are thought to be unnecessary. HTML prototypes and language prototypes are used for parts of the system that are critical, yet not well understood.

Interface Evaluation

The objective of interface evaluation is to understand how to improve the interface design. Interface design is subjective; there are no formulas that guarantee a great user interface. Most interface designers intentionally or unintentionally design an interface that meets their personal preferences, which may or may not match the preferences of the users. The key message, therefore, is to have as many people as possible evaluate the interface—and the more users, the better. Most experts recommend involving at least 10 potential users in the evaluation process. Many organizations save interface evaluation for the very last step in the SDLC before the system is installed. Ideally, however, interface evaluation should be performed while the system is being designed—before it is built—so that any major design problems can be identified and corrected before the time and cost of programming have been spent on a weak design. It is not uncommon for the system to undergo one or two major changes after the users see the first interface design prototype, because they identify problems that are overlooked by the project team.

As with interface design prototyping, interface evaluation can take many different forms, each with different costs and different levels of detail. Four common approaches are heuristic evaluation, walk-through evaluation, interactive evaluation, and formal usability testing. As with interface design prototyping, the different parts of a system can be evaluated by different techniques.

Heuristic Evaluation

A heuristic evaluation examines the interface by comparing it to a set of heuristics, or principles, for interface design. The project team develops a checklist of interface design principles—from the list at the start, for example, as well as the lists of principles in the navigation, input, and output design sections. At least three members of the project team then individually work through the interface design prototype, examining each interface to ensure that it satisfies each design principle on the formal checklist. After each member has gone through the prototype separately, they all meet as a team to discuss their evaluation and identify specific improvements that are required. Because this technique does not involve the users, it is considered the weakest type of evaluation.

Walk-through

Evaluation An interface design walk-through evaluation is a meeting conducted with the users who will ultimately have to operate the system. The project team presents the prototype to the users and walks them through the various parts of the interface. The project team shows the storyboard or actually demonstrates the HTML or language prototype and explains how the interface will be used. The users identify improvements to each of the interfaces that are presented.

Interactive Evaluation

With an interactive evaluation, the users themselves actually work with the HTML or language prototype in one-on-one sessions with members of the project team. (An interactive evaluation cannot be used with a storyboard.) As the user works with the prototype (often by going through the use scenarios or just navigating at will through the system), he or she tells the project team members what he or she likes and doesn't like and what additional information or functionality is needed. As the user interacts with the prototype, team members record the situation when the user appears to be unsure of what to do, makes mistakes, or misinterprets the meaning of an interface component. If the pattern of uncertainty, mistakes, or misinterpretations recurs across several evaluation sessions with several users, it is a clear indication that those parts of the interface need improvement.

3G E-LEARNING

Formal Usability

Testing Formal usability testing is commonly done with commercial software products and products developed by large organizations that will be widely used through the organization. As the name suggests, it a very formal— almost scientific—process that can be used only with language prototypes (and systems that have been completely built and are awaiting installation or shipping). As with interactive evaluation, usability testing is done in one-on-one sessions in which a user works directly with the software. However, it is typically done in a special lab equipped with video cameras and special software that records each and every keystroke and mouse operation so that they can be replayed to help in understanding exactly what the user did.

The user is given a specific set of tasks to accomplish (usually the use scenarios), and after some initial instructions the project teams member(s) are not permitted to interact with the user to provide assistance. The user must work with the software without help, which can be hard on the user if he or she becomes confused with the system. It is critical that users understand that the goal is to test the interface, not their abilities, and if they are unable to complete the task, the interface— not the user—has failed the test. Several performance measures are used, such as time to complete the task, error rate, and user satisfaction. Formal usability testing is very expensive. Each one-user session (which typically lasts one to two hours) can take one to two days to analyze due to the volume of detail collected in the computer logs and videotapes. Most usability testing involves 5 to 10 users. Fewer than 5 users makes the results depend too much on the specific individual users who participated, but more than 10 users is often too expensive to justify (unless you work for a large commercial software developer).

NAVIGATION DESIGN

The navigation component of the interface enables the user to enter commands to navigate through the system and perform actions to enter and review information it contains. The navigation component also presents messages to the user about the success or failure of his or her actions. The goal of the navigation system is to make the system as simple as possible to use. A good

navigation component is one that the user never really notices. It simply functions the way the user expects, and thus the user gives it little thought.

Basic Principles

One of the hardest things about using a computer system is learning how to manipulate the navigation controls to make the system do what you want. Analysts usually must assume that users have not read the manual, have not attended training, and do not have external help readily at hand. All controls should be clear and understandable and placed in an intuitive location on the screen. Ideally, the controls should anticipate what the user will do and simplify his or her efforts. For example, many set-up programs are designed so that, for a typical installation, the user can simply keep pressing the "Next" button

Prevent Mistakes

The first principle of designing navigation controls is to prevent the user from making mistakes. A mistake costs time and creates frustration. Worse still, a series of mistakes can cause the user to discard the system. Mistakes can be reduced by labeling commands and actions appropriately and by limiting choices. Too many choices can confuse the user, particularly when they are similar and hard to describe in the short space available on the screen. When there are many similar choices on a menu, consider creating a second level of menu or a series of options for basic commands. Never display a command that cannot be used. For example, many Windows applications gray-out commands that cannot be used; they are displayed on pull-down menus in a very light-colored font, but they cannot be selected. This shows that they are available, but that they cannot be used in the current context. It also keeps all menu items in the same place. When the user is about to perform a critical function that is difficult or impossible to undo (e.g., deleting a file), it is important to confirm the action with the user (and make sure the selection was not made by mistake). This is usually done by having the user respond to a confirmation message that explains what the user has requested and asks the user to confirm that this action is correct.

Simplify Recovery from

Mistakes No matter what the system designer does, users will make mistakes. The system should make it as easy as possible to correct these errors. Ideally, the system will have an "Undo" button that makes mistakes easy to override; however, writing the software for such buttons can be very complicated.

Use Consistent Grammar

Order one of the most fundamental decisions is the grammar order. Most commands require the user to specify an object (e.g., file, record, word) and the action to be performed on that object (e.g., copy, delete). The interface can require the user to first choose the object and then the action (an object–action order) or first choose the action and then the object (an action–object order). Most Windows applications use an object–action grammar order (e.g., think about copying a block of text in your word processor). The grammar order should be consistent throughout the system, both at the data element level and at the overall menu level. Experts debate about the advantages of one approach over the other, but because most users are familiar with the object–action order, most systems today are designed with that approach.

Types of Navigation Controls

There are two traditional hardware devices that can be used to control the user interface: the keyboard and a pointing device, such as a mouse, trackball, or touch screen. In recent years, voice recognition systems have made an appearance, but they are not yet common. There are three basic software approaches for defining user commands: languages, menus, and direct manipulation.

Languages

With a command language, the user enters commands in a special language developed for the computer system (e.g., UNIX and SQL both use command languages). Command languages sometimes provide greater flexibility than do other approaches, because the user can combine language elements in ways not

predetermined by developers. However, they put a greater burden on users because users must learn syntax and type commands rather than select from a well-defined, limited number of choices. Systems today use command languages sparingly, except in cases in which there are an extremely large number of command combinations, making it impractical to try to build all combinations into a menu (e.g., SQL queries for databases).

Natural language interfaces are designed to understand the user's own language (e.g., English, French, and Spanish). These interfaces attempt to interpret what the user means, and often they present back to the user a list of interpretations from which to choose. Many "help" systems today enable the user to ask free-form questions to get help.

Menus

The most common type of navigation system today is the menu. A menu presents the user with a list of choices, each of which can be selected. Menus are easier to learn than languages because the user sees an organized, but limited, set of choices. Clicking on an item with a pointing device or pressing a key that matches the menu choice (e.g., a function key) takes very little effort. Therefore, menus are usually preferred to languages. Menus should be designed with care, because the submenus behind a main menu are hidden from users until they click on the menu item. It is better to make menus broad and shallow (i.e., with each menu containing many items and each item containing only one or two layers of menus) rather than narrow and deep (i.e., with each menu containing only a few items, but each item leading to three or more layers of menus). A broad and shallow menu presents the user with the most information initially, so that he or she can see many options, and requires only a few mouse clicks or keystrokes to perform an action. A narrow and deep menu makes users hunt and seek for items hidden behind menu items and requires many more clicks or keystrokes to perform an action. Research suggests that in an ideal world, any one menu should contain no more than eight items and it should take no more than two mouse clicks or keystrokes from any menu to perform an action (or three from the main menu that starts a system).7 However, analysts sometimes must break this guideline in the design of complex systems. In this

case, menu items are often grouped together and separated by a horizontal line (Fig. 7.7). Often, menu items have hot keys that enable experienced users to quickly invoke a command with keystrokes in lieu of a menu choice (e.g., Control-F in Word invokes the Find command, whereas Alt-F opens the File menu). Menus should put together similar categories of items so that the user can intuitively guess what each menu contains. Most designers recommend grouping menu items by interface objects (e.g., Customers, Purchase Orders, Inventory) rather than by interaction actions (e.g., New, Update, Format), so that all actions pertaining to one object are in one menu, all actions for another object are in a different menu, and so on. However, this is highly dependent on the specific interface. As Figure 7.7 shows, Microsoft Visual Studio groups menu items by interface objects (e.g., File, Project, and Window) and by interface actions (e.g., Edit, View, and Build) on the same menu. Some of the more common types of menus include menu bars, drop-down menus, pop-up menus, tab menus, tool bars, and image maps.

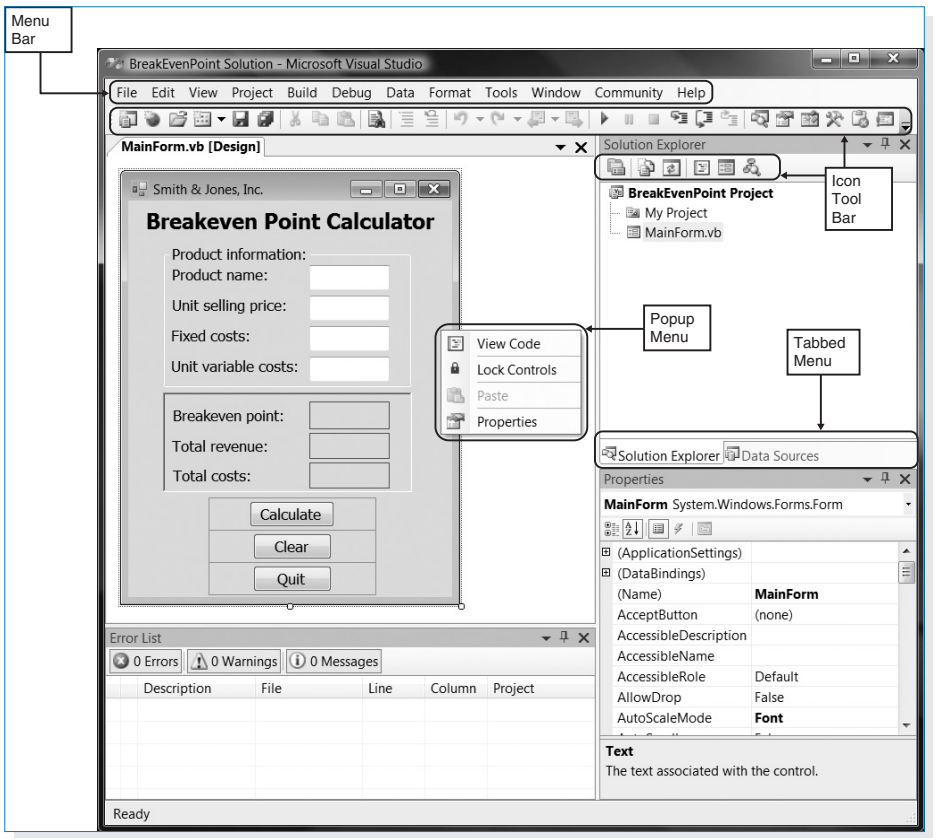


Figure 7.7: Common Types of Menus.

Direct Manipulation

With direct manipulation, the user enters commands by working directly with interface objects. For example, users can change the size of objects in Microsoft PowerPoint by clicking on objects and moving the sides, or they can move files in Windows Explorer by dragging the file names from one folder to another. Direct manipulation can be simple, but it suffers from two problems. First, users familiar with language- or menu-based interfaces don't always expect it. Second, not all commands are intuitive. (For example, how do you copy [not move] files in Windows Explorer?)

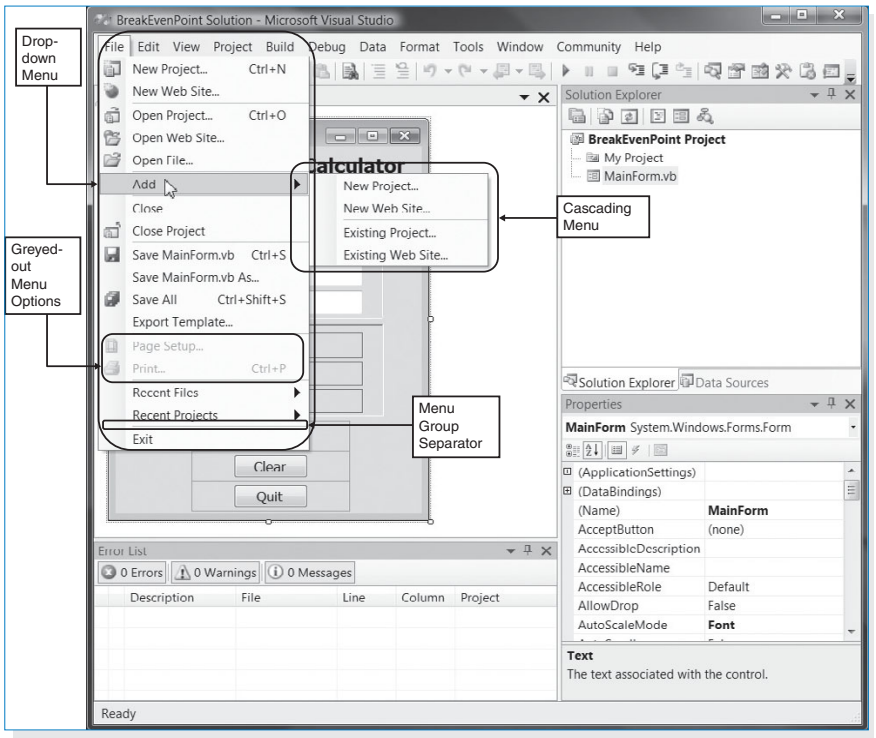


Figure 7.8: Common Types of Menus.

"Even the best designers produce successful products only if their designs solve the right problems. A wonderful interface to the wrong features will fail."

—Jakob Nielsen

INPUT DESIGN

Input mechanisms facilitate the entry of data into the computer system, whether highly structured data, such as order information (e.g., item numbers, quantities, costs), or unstructured information (e.g., comments). Input design means designing the screens used to enter the information, as well as any forms on which users write or type information (e.g., time cards, expense claims).

Basic Principles

The goal of input design is to capture accurate information for the system simply and easily. The fundamental principles for input design reflect the nature of the inputs (whether batch or online) and ways to simplify their collection.

Use Online and Batch Processing Appropriately

There are two general approaches for entering inputs into a computer system: online processing and batch processing. With online processing (sometimes called transaction processing), each input item (e.g., a customer order, a purchase order) is entered into the system individually, usually at the same time as the event or transaction prompting the input. For example, when you borrow a book from the library, buy an item at the store, or make an airline reservation, the computer system that supports each process uses online processing to immediately record the transaction in the appropriate database(s). Online processing is most commonly used when it is important to have real-time information about the business process. For example, when you reserve an airline seat, the seat is no longer available for someone else to use, so that piece of information must be recorded immediately. With batch processing, all the inputs collected over some period are gathered together and entered into the system at one time in a batch. Some business processes naturally generate information in batches. For example, most hourly payrolls are done by batch processing because time cards are gathered together in batches and processed at once. Batch processing also is used for transaction processing systems that do not require real-time information. For example, most stores send sales information to district offices so that new replacement inventory can be ordered. This information could be sent in real

time as it is captured in the store, so that the district offices are aware within a second or two that a product is sold. If stores do not need up-to-the-second real-time data, they will collect sales data throughout the day and transmit the data every evening in a batch to the district office. This batching simplifies the data communications process and often cuts communications costs. It does mean, however, that inventories are not accurate in real time, but rather are accurate only at the end of the day after the batch has been processed.

Capture Data at the Source

Perhaps the most important principle of input design is to capture the data in an electronic format at the original source or as close to the original source as possible. In the early days of computing, computer systems replaced traditional manual systems that were based on paper forms. As these business processes were automated, many of the original paper forms remained, either because no one thought to replace them or because it was too expensive to do so. Instead, the business process continued to contain manual forms that were taken to the computer center in batches to be typed into the computer system by a data-entry operator. Many business processes still operate this way today. For example, many organizations have expense claim forms that are completed by hand and submitted to an accounting department, which approves them and enters them into the system in batches. There are three problems with this approach. First, it is expensive because it duplicates work. (The form is filled out twice, once by hand and once by keyboard.) Second, it increases processing time because the paper forms must be physically moved through the process. Third, it increases the cost and the probability of error, because it separates the entry from the processing of information; someone may misread the handwriting on the input form, data could be entered incorrectly, or the original input may contain an error that invalidates the information.

Most transaction processing systems today are designed to capture data at its source. Source data automation refers to using special hardware devices to automatically capture data without requiring anyone to type it. Stores commonly use bar code readers that automatically scan products and enter data

directly into the computer system. No intermediate formats, such as paper forms, are used. Similar technologies include optical character recognition, which can read printed numbers and text (e.g., on checks); magnetic stripe readers, which can read information encoded on a stripe of magnetic material similar to a diskette (e.g., credit cards); and smart cards that contain microprocessors, memory chips, and batteries (much like credit card-size calculators). A recent development is the RFID (radio frequency identification) tag, combining a microprocessor chip with an antenna to broadcast its information to electronic readers. Information can be read from or written to the RFID tag. As well as reducing the time and cost of data entry, these systems reduce errors because they are far less likely to capture data incorrectly. Today, portable computers and scanners allow data to be captured at the source even in mobile settings (e.g., air courier deliveries, use of rental cars). A lot of information, however, cannot be collected by these automatic systems. Today, with the widespread use of the Web, much data is captured directly from the customer. Consequently, the forms for capturing information on-screen should provide a logical flow and should allow the user to easily complete the forms and check their entries before submitting them. Since data entered by the user is prone to inaccuracies, validation checks should be used whenever possible.

Minimize Keystrokes

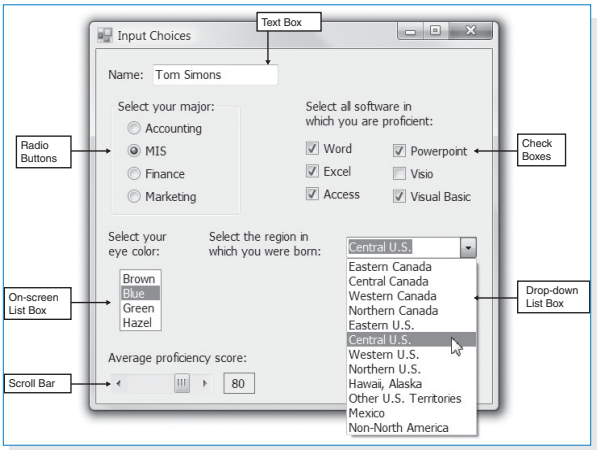
Another important principle is to minimize keystrokes. Keystrokes cost time and money, whether they are performed by a customer, user, or trained data-entry operator. The system should never ask for information that can be obtained in another way (e.g., by retrieving it from a database or by performing a calculation). Likewise, a system should not require a user to type information that can be selected from a list; selecting reduces errors and speeds entry. In many cases, data have values that often recur. These frequent values should be used as the default value for the data so that the user can simply accept the value and not have to retype it time and time again. Examples of default values are the current date, the area code held by the majority of a company's customers, and a billing address that is based on the customer's residence. Most systems permit changes to default values to handle data entry exceptions as they occur.

Types of Inputs

Each data item that has to be input is linked to a field, on the form into which its value is typed. Each field also has a field label, which is the text beside, above, or below the field, that tells the user what type of information belongs in the field. Often, the field label is similar to the name of the data element, but the two do not have to have identical wording. In some cases, a field will display a template over the entry box to show the user exactly how data should be typed.

Text

As the name suggests, a text box is used to enter text. Text boxes can be defined to have a fixed length or can be scrollable and accept a virtually unlimited amount of text. In either case, boxes can contain single or multiple lines of textual information. Never use a text box if you can use a selection box. Text boxes should have field labels placed to the left of the entry area, with their size clearly delimited by a box (or a set of underlines in a non-GUI interface). If there are multiple text boxes, their field labels and the left edges of their entry boxes should be aligned. Text boxes should permit standard GUI functions such as cut, copy, and paste.



Numbers

A number box is used to enter numbers. Some software can automatically format numbers as they are entered, so that 3452478

becomes $34,524.78. Dates are a special form of numbers that sometimes have their own type of number box. Never use a number box if you can use a selection box.

Selection Box

A selection box enables the user to select a value from a predefined list. The items in the list should be arranged in some meaningful order, such as alphabetical for long lists, or in order of most frequently used. The default selection value should be chosen with care. A selection box can be initialized as "unselected" or, better still, start with the most commonly used item already selected. The choice among the types of text selection boxes generally comes down to one of screen space and the number of choices the user can select. If screen space is limited and only one item can be selected, then a drop-down list box is the best choice, because not all list items need to be displayed on the screen. If screen space is limited, but the user can select multiple items, an on-screen list box that displays only a few items can be used. Check boxes (for multiple selections) and radio buttons (for single selections) both require all list items to be displayed at all times, thus requiring more screen space, but since they display all choices, they are often simpler for novice users.

"UI is the saddle, the stirrups and the reigns. UX is the feeling you get being able to ride the horse, and rope your cattle."

—Dain Miller

Input Validation

All data entered into the system must be validated in order to ensure accuracy. Input validation (also called edit checks) can take many forms. Ideally, to prevent invalid information from entering the system, computer systems should not accept data that fail any important validation check. However, this can be very difficult, and invalid data often slip by data-entry operators and the users providing the information. It is up the system to identify invalid data and either make changes or notify someone who can resolve the information problem.

OUTPUT DESIGN

Outputs are the reports that the system produces, whether on the screen, on paper, or in other media, such as the Web. Outputs are perhaps the most visible part of any system, because a primary reason for using an information system is to access the information that it produces.

Basic Principles

The goal of the output mechanism is to present information to users so that they can accurately understand it with the least effort. The fundamental principles for output design reflect how the outputs are used and ways to make it simpler for users to understand them.

Understand Report Usage

The first principle in designing reports is to understand how they are used. Reports can be used for many different purposes. In some cases—but not very often—reports are read cover to cover because all information is needed. In most cases, reports are used to identify specific items or are used as references to find information, so the order in which items are sorted on the report or grouped within categories is critical. This is particularly important for the design of electronic or Web-based reports. Web reports that are intended to be read end to end should be presented in one long scrollable page, whereas reports that are primarily used to find specific information should be broken into multiple pages, each with a separate link. Page numbers and the date on which the report was prepared also are important for reference reports. The frequency of the report may also play an important role in its design and distribution. Real-time reports provide data that are accurate to the second or minute at which they were produced (e.g., stock market quotes). Batch reports are those that report historical information that may be months, days, or hours old, and they often provide additional information beyond the reported information (e.g., totals, summaries, historical averages). There are no inherent advantages to real-time reports over batch reports. The only advantages lie in the time value of the information. If the information in a report is time critical (e.g., stock prices, air

traffic control information), then real-time reports have value. This is particularly important because real-time reports often are expensive to produce; unless they offer some clear business value, they may not be worth the extra cost.

Manage Information

Load Most managers get too much information, not too little (i.e., the information load confronting the manager is too great). The goal of a well-designed report is to provide all the information needed to support the task for which it was designed. This does not mean that the report should provide all the information available on the subject—just what the users decide they need to perform their jobs. In some cases, this may result in the production of several different reports on the same topics for the same users, because they are used in different ways. This is not bad design. For users in Westernized countries, the most important information generally should be presented first, in the top left corner of the screen or paper report. Information should be provided in a format that is usable without modification. The user should not need to re-sort the report's information, highlight critical information to find it more easily amid a mass of data, or perform additional mathematical calculations.

Minimize Bias

No analyst sets out to design a biased report. The problem with bias is that it can be very subtle; analysts can introduce it unintentionally. Bias can be introduced by the way in which lists of data are sorted, because entries that appear first in a list may receive more attention than those appearing later in the list. Data often are sorted in alphabetic order, making those entries starting with the letter a more prominent. Data can be sorted in chronological order (or reverse chronological order), placing more emphasis on older (or most recent) entries. Data may be sorted by numeric value, placing more emphasis on higher or lower values. For example, consider a monthly sales report by state. Should the report be listed in alphabetic order by state name, in descending order by the amount sold, or in some other order (e.g., geographic region)? There are no easy answers to this, except to say that the order of presentation should match the way in which the information is used.

Types of Outputs

There are many different types of reports, such as detail reports, summary reports, exception reports, turnaround documents, and graphs (Fig. 7.9). Classifying reports is challenging because many reports have characteristics of several different types. For example, some detail reports also produce summary totals, making them summary reports.

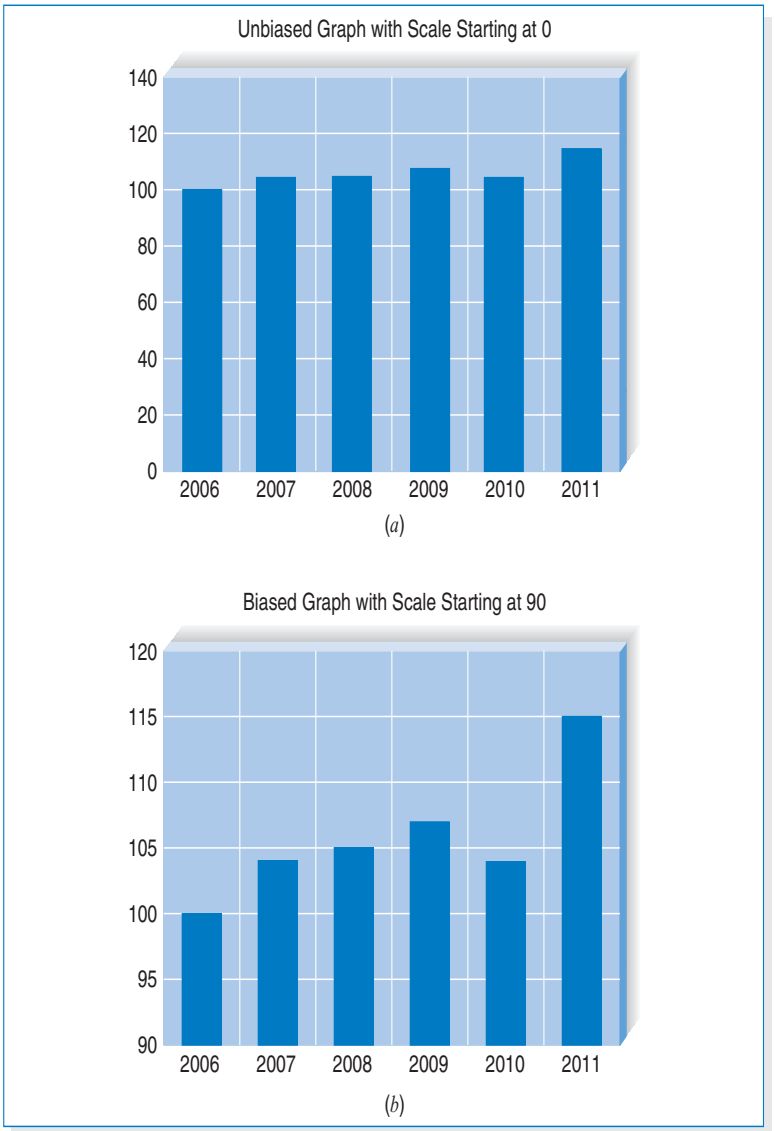


Figure 7.9

Media

There are many different types of media used to produce reports. The two dominant media today are paper and electronic. Paper is the more traditional medium and is relatively permanent, easy to use, and accessible in most situations. It also is highly portable, at least for short reports. Paper also has several rather significant drawbacks. It is inflexible. Once the report is printed, it cannot be sorted or reformatted to present a different view of the information. Likewise, if the information on the report changes, the entire report must be reprinted. Paper reports are expensive, are hard to duplicate, and require considerable supplies (paper, ink) and storage space. Paper reports are also hard to quickly move long distances (e.g., from a head office in Toronto to a regional office in Bermuda).

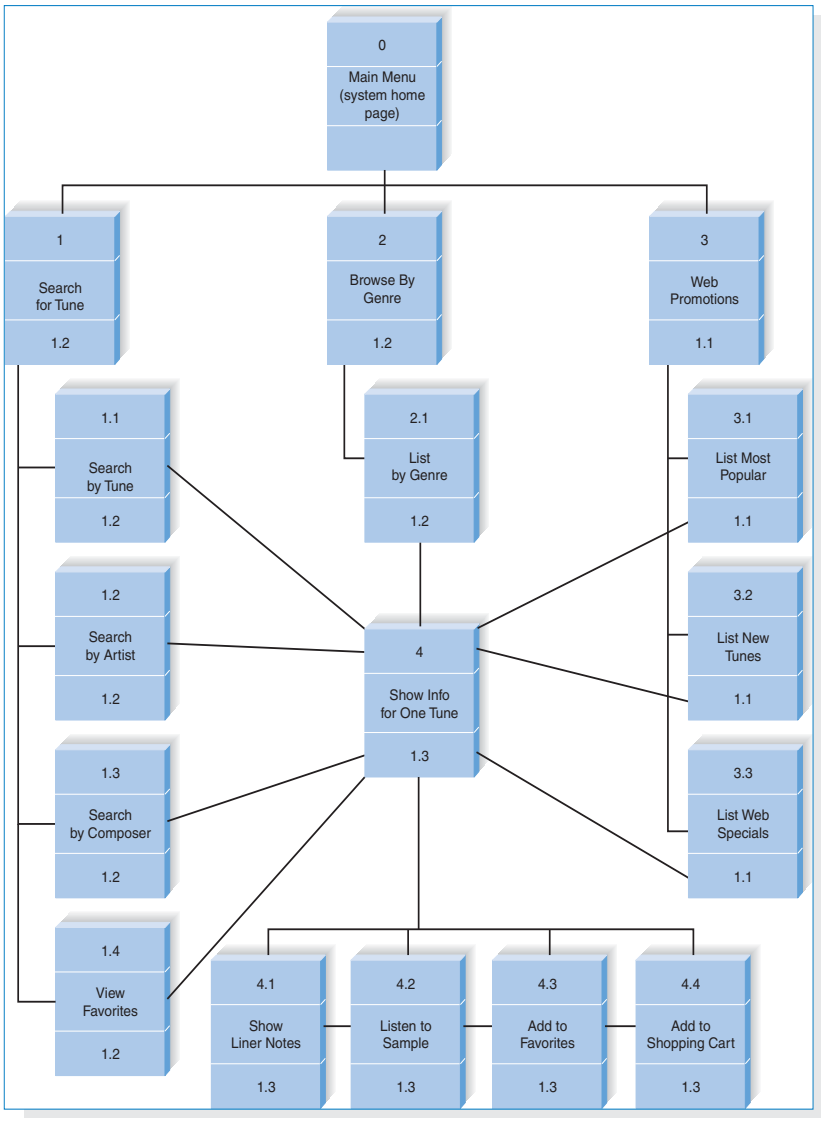
Many organizations are therefore moving to electronic production of reports, whereby reports are "printed," but stored in electronic format on file servers or Web servers so that users can easily access them. Often, the reports are available in more predesigned formats than are their paper-based counterparts, because the cost of producing and storing different formats is minimal. Electronic reports also can be produced on demand as needed, and they enable the user to search more easily for certain words. Furthermore, electronic reports can provide a means to support ad hoc reports when users customize the contents of the report at the time the report is generated. Some users may still print the electronic report on their own printers. The reduced cost of electronic delivery over distance and improved user access to the reports usually offsets the cost of local printing.

Use Scenario Development

The first step in the interface design process was to develop the key use scenarios for the Digital Music Download system. Jason Wells, senior systems analyst at Tune Source and project manager for the Digital Music Download system, began by examining the DFD and thinking about the types of users and how they would interact with the system. Jason identified two use scenarios: the browsing shopper and the hurry-up shopper.

Interface Structure Design

Next, Jason created an ISD for the Web system. He began with the DFDs to ensure that all functionality defined for the system was included in the ISD the ISD for the Web portion of process 1.



Reviewing the Interface Structure Design Jason then examined the use scenarios to see how well the ISD enabled different types of users to work through the system. He started with the "browsing shopper" scenario and followed it through the ISD, imagining what would appear on each screen and pretending

to navigate through the system. He found that the ISD worked well. Jason next explored the "hurry-up shopper" scenario. In this case, the ISD did not work as well. Moving from the home page to the search page to the list of matching tunes to the tune info page with price and other information takes three mouse clicks. This falls within the three-click rule, but for someone in a hurry, this may be too many. Jason decided to add a "quick search" option to the home page (interface 0) that would enable the user to enter one search criterion (e.g., searching by just artist name or title, rather than doing a more detailed search, as would be possible on the search page) that, with one click, would take the user to the one tune that matched the criterion (interface 4) or to a list of tunes if there were more than one (interfaces 1.1, 1.2, 1.3). This would enable an impatient user to get to the tune of interest in one or two clicks.

"In many ways design is akin to gardening: get rid of the weeds. If data is showing a feature has poor usage or unintentional consequences, kill it."

—Didier Hilhorst

Interface Standards Design

Once the ISD was complete, Jason moved on to develop the interface standards for the system. The interface metaphor was straightforward: a Tune Source retail music store.

Interface Template Design

For the interface template, Jason decided on a simple, clean design that had the Tune Source logo in the upper left corner. The template had a navigation menu across the top for navigation within the entire Web site (e.g., overall Web site home page, store locations) and one menu down the side of the page for navigation within the Digital Music Download system.

The left-edge menu contained the links to the three top-level screens, as well as the "quick search" option. The center area of the screen is used for displaying forms and reports when the appropriate link is clicked.

Interface Metaphor: A Tune Source Music Store.

Interface Objects

- **Tune:** All items, whether single track or longer composition, unless it is important to distinguish among them.
- **Artist:** Person or group who records the tune.
- **Title:** Title or name of tune.
- **Composer:** Person or group who wrote the music for the tune (primarily used for classical music).
- **Music Genre:** Type of music. Current categories include Rock, Jazz, Classical, Country, Alternative, Soundtracks, Rap, Folk, Gospel.
- **Tune List:** List of tune(s) that match the specified criteria.
- **Favorites List:** A place for the customer to store tunes of interest.
- **Shopping Cart:** Place to store tunes until they are purchased.

Interface Actions

- **Search for:** Display a tune list that matches specified criteria.
- **Browse:** Display a tune list sorted in order by some criteria.
- **Buy:** Buy permission to download tune(s).
- **Download:** Transfer file touser's computer after purchase.

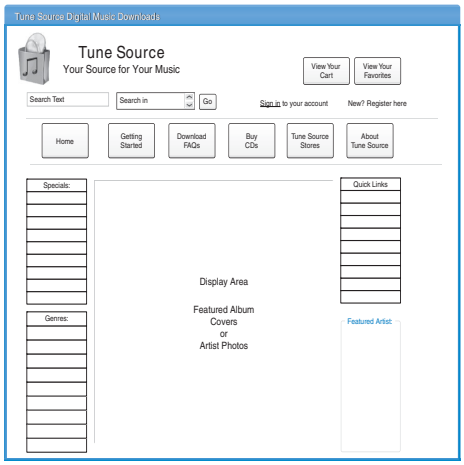
Interface Icons

- **Tune Source Logo** will be used on all screens.

At this point, Jason decided to seek some quick feedback on the interface structure and standards before investing time in prototyping the interface designs. Therefore, he met with Carly Edwards, the project sponsor, to discuss the emerging design. Making changes at this point would be much simpler than waiting until after doing the prototype. Carly had a few suggestions, so, after the meeting, Jason made the changes and moved into the design prototyping step.

Design Prototyping

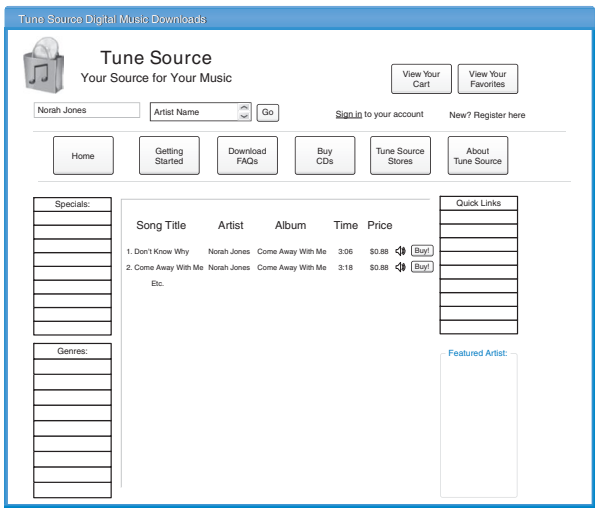
Jason decided to use Visio to create prototypes of the system. The Digital Music Download system was new territory for Tune Source and a strategic investment in a new business model, so it was important to make sure that no key issues were overlooked. The prototypes would provide the most detailed information and could be developed rapidly. Once satisfied with the layout, Jason planned to create HTML prototypes that could be used to test the interface interactively. Jason began designing the prototype, starting with the home screen, and gradually worked his way through all the screens. The process was very iterative, and he made many changes to the screens as he worked.



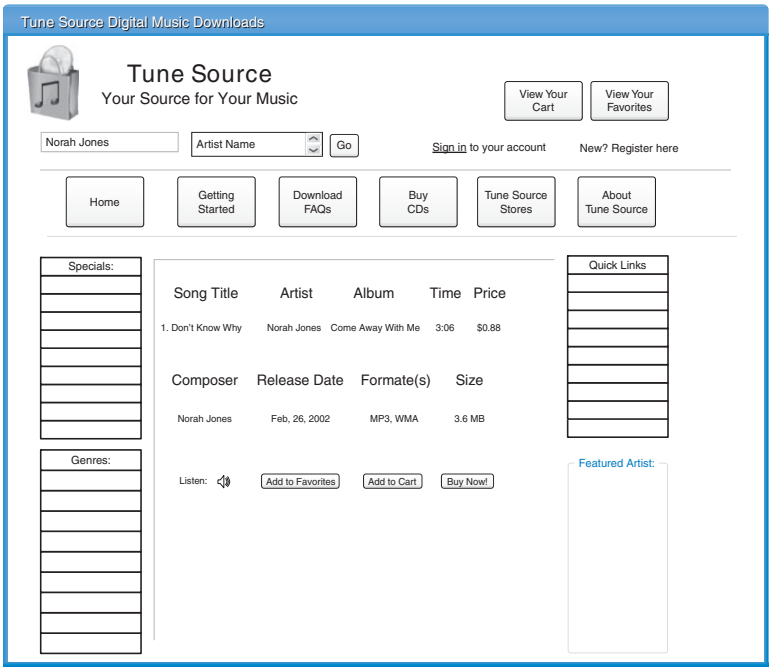
Once he had an initial prototype designed, he posted it on Tune Source's internal intranet and solicited comments from several friends with lots of Web experience. He revised it according to the comments he received.

Interface Evaluation

The next step was interface evaluation. Jason decided on a two-phase evaluation. The first evaluation was to be an interactive one conducted by Carly Edwards, her marketing managers, selected staff members, and selected store managers. They worked hands-on with the prototype and identified several ways to improve it. Jason modified the HTML prototype to reflect the changes suggested by the group and asked Carly to review it again.



(a)



(b)

The second evaluation was again interactive, this time performed by a series of two focus groups of potential customers, one with little Internet experience, the other with extensive Internet experience. Once again, several minor changes were identified. Jason again modified the HTML prototype and asked Carly to review it once more. When she was satisfied, the interface design was complete.

CONCLUSION

User interface design has been a topic of considerable research, including on its aesthetics. Standards have been developed as far back as the 1980s for defining the usability of software products. One of the structural bases has become the IFIP user interface reference model.

The model proposes four dimensions to structure the user interface:

- The input/output dimension (the look)
- The dialogue dimension (the feel)

- The technical or functional dimension (the access to tools and services)
- The organizational dimension (the communication and co-operation support)

This model has greatly influenced the development of the international standard ISO 9241 describing the interface design requirements for usability. The desire to understand application-specific UI issues early in software development, even as an application was being developed, led to research on GUI rapid prototyping tools that might offer convincing simulations of how an actual application might behave in production use. Some of this research has shown that a wide variety of programming tasks for GUI-based software can, in fact, be specified through means other than writing program code.

KEY TERMS

User interface (UI)	User experience (UX)	Macintosh
Customer information	Payment information	Content awareness
Interface structure diagram	Web interface	Interface metaphor

DISCUSSION QUESTIONS

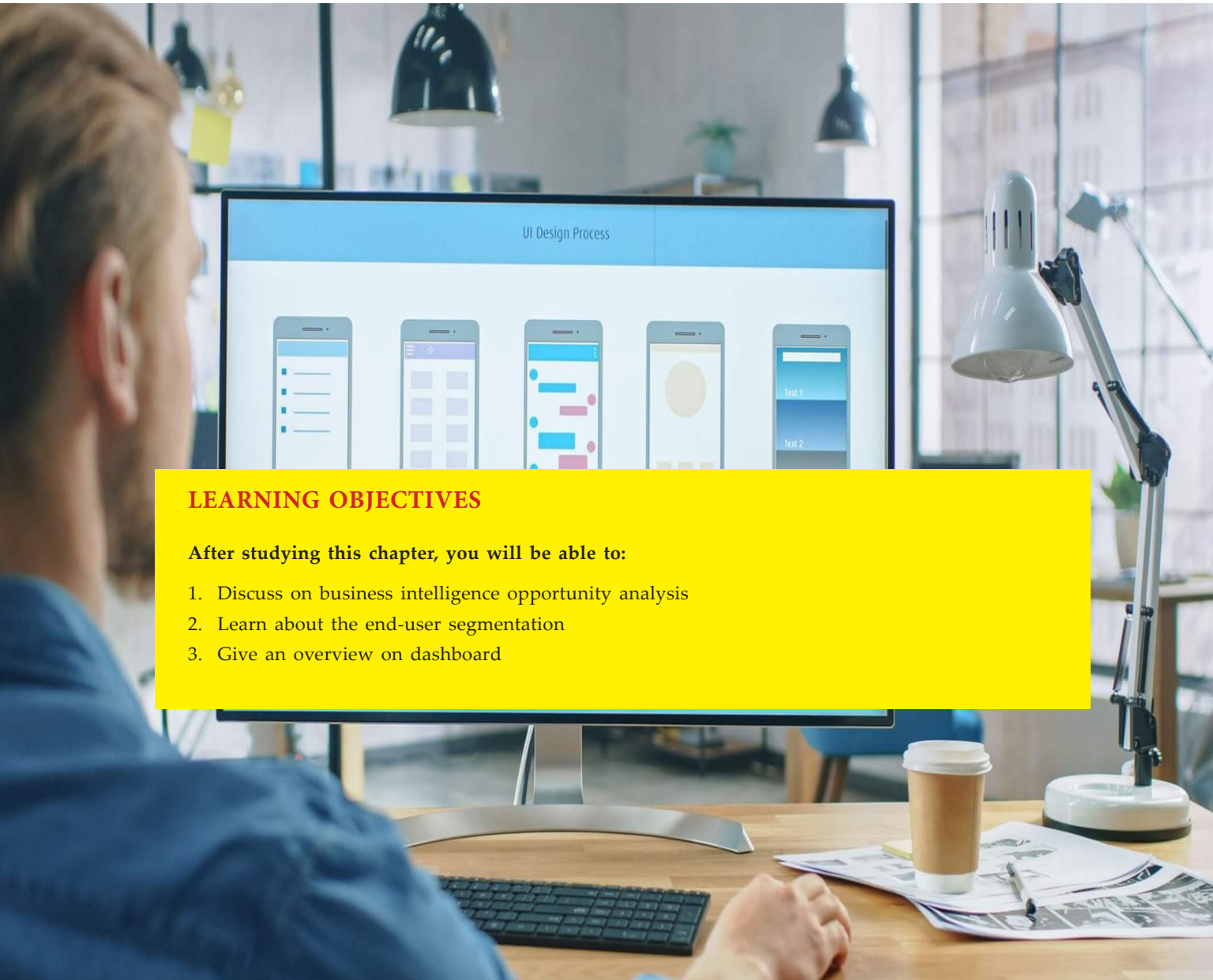
1. Describe the process of user interface design.
2. Discuss about interface structure diagram.
3. Focus on interface standards design.
4. Explain the navigation design.
5. Write short notes on interface evaluation.

REFERENCES

1. Norman, D. A. (2002). "Emotion & Design: Attractive things work better". Interactions Magazine, ix (4). pp. 36–42. Retrieved 20 April 2014.
2. Ann Blandford. "Semi-structured qualitative studies". The Encyclopedia of Human-Computer Interaction, 2nd Ed. Interaction Design Foundation. Retrieved 20 April 2014.
3. Karen Holtzblatt and Hugh R. Beyer. "Contextual design". The Encyclopedia of Human-Computer Interaction, 2nd Ed. Interaction Design Foundation. Retrieved 20 April 2014. Martin Fowler. "Forms and control". GUI architecture. Thought works publication. Retrieved 20 February 2017.
4. The role of context in perceptions of the aesthetics of web pages over time". International Journal of Human–Computer Studies. 2009-01-05. Retrieved 2009-04-02.
5. Creating user interfaces using programming by example, visual programming, and constraints". ACM. 1990-04-11. Retrieved 2009-04-02.
6. Wolf, Lauren (23 May 2012). "6 Tips for Designing an Optimal User Interface for Your Digital Event". INXPO. Archived from the original on 16 June 2013. Retrieved 22 May 2013.

CHAPTER 8

BUSINESS INTELLIGENCE USER MODEL



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Discuss on business intelligence opportunity analysis
2. Learn about the end-user segmentation
3. Give an overview on dashboard

INTRODUCTION

Business intelligence combines business analytics, data mining, data visualization, data tools and infrastructure, and best practices to help organizations make more data-driven decisions. In practice, you know you've got modern business intelligence when you have a comprehensive view of your organization's data and use that data to drive change, eliminate inefficiencies, and quickly adapt to market or supply changes. Modern BI solutions prioritize flexible self-service analysis, governed data on trusted platforms, empowered business users, and speed to insight.

Business intelligence (BI) comprises the strategies and technologies used by enterprises for the data analysis and management of business information. Common functions of business intelligence technologies include reporting, online analytical processing, analytics, dashboard development, data mining, process mining, complex event processing, business performance management, benchmarking, text mining, predictive analytics, and prescriptive analytics.

BI tools can handle large amounts of structured and sometimes unstructured data to help identify, develop, and otherwise create new strategic business opportunities. They aim to allow for the easy interpretation of these big data. Identifying new opportunities and implementing an effective strategy based on insights can provide businesses with a competitive market advantage and long-term stability, and help them take strategic decisions.

Business intelligence can be used by enterprises to support a wide range of business decisions ranging from operational to strategic. Basic operating decisions include product positioning or pricing. Strategic business decisions involve priorities, goals, and directions at the broadest level. In all cases, BI is most effective when it combines data derived from the market in which a company operates (external data) with data from company sources internal to the business such as financial and operations data (internal data). When combined, external and internal data can provide a complete picture which, in effect, creates an "intelligence" that cannot be derived from any singular set of data.

BUSINESS INTELLIGENCE OPPORTUNITY ANALYSIS

The business intelligence model (BIM) is a model that offers opportunities for businesses to transform raw data into meaningful and useful information, in order to build an effective strategic plan, as well as create tactical and operational insights for decision-making within a given timeframe. In the end, this knowledge needs to get to the right people, at the right moment, and via the right channel. Organizations collect large quantities of information. These are often raw data, such as facts and large data chains. This information, data, needs to be processed and interpreted because it opens up new opportunities within the organization, which may result in a competitive advantage. The purpose of Business Intelligence (BI) depends on the strategy of the organization. This is often derived from the business goal or mission statement. The prime goals of using BI are to increase revenues and/or reduce costs, thereby increasing profits and creating business value. Identifying and prioritizing opportunities to use business information, business analysis, and structured decisions are the main methods of BI opportunity analysis. A structured approach that aligns two key factors is needed to achieve those goals:

- Business drivers, business strategies, goals and objectives, and the core business processes that drive profits
- BI applications that will improve the effectiveness of those core business processes

The figure shows the Business Intelligence opportunity analysis. This flow of the opportunity analysis from top to down is from strategic to operational and is used to get the idea about return on investment (ROI) in BI whereas a flow from bottom to top is used to determine how the information and associated BI applications that have been proposed by business users can be used to further organizational performance.

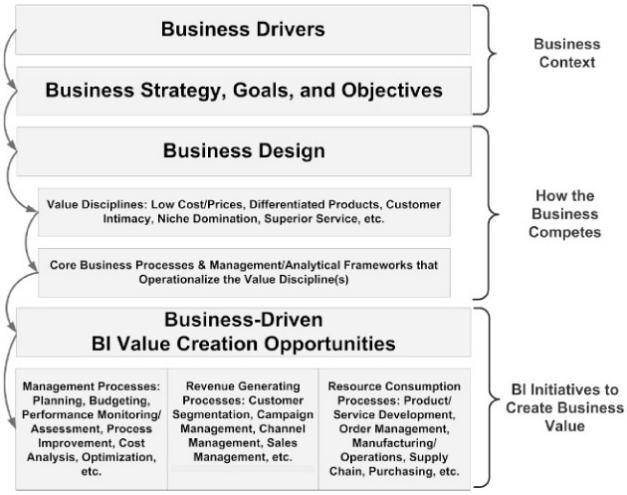


Figure 8.1: Business Intelligence opportunity analysis.

Business Intelligence Opportunity Analysis: Practical Considerations

There are different type of contexts in which the opportunity analysis might be used.

- Enterprise BI opportunity analysis for a company that is just getting started with BI.
- BI opportunity analysis for a division or function that has experience with BI but has not achieved the ROI it sought.

Because of these practical differences, and because of the potential impact on ROI, it is important to identify some key practical considerations that affect the cost and quality of the BI opportunity analysis.

Art versus Science in Business Intelligence Opportunity Analysis

The art lies in picking the suitable frameworks that can fit the BI opportunity analysis and can bring the appropriate people from business, IT and management into the conversation. Art comes into play in determining how BI can be used within core processes to impact profit. The main task always is to imagine how the core business processes could be improved by leveraging business

information, business analyses, and potentially structured business decisions. Two key components to for understanding BI value creation opportunities requires: Knowledge of a company's core business processes and knowledge of how BI has been used in different contexts to improve profits.

There are different way to stimulate the imagination to use analytical framework:

- To read BI case studies. Such case studies are available from TDWI and other sources.
- To use generic analytical frameworks.
- To use "balanced scorecards" at strategic level.

BI is highly useful in reporting results and improving the efficiency of the underlying processes that drive results. In the business process improvement arena, a wide range of published process performance metrics can be used as a starting point for BI opportunity analysis.

"BI investments will generate incremental profit in excess of the amounts invested" - the validity of this premise can be done by analyzing two factors: Whether the prerequisites for leveraging BI have been met and whether the BI opportunity analysis has correctly identified proper BI investments. The BI opportunity analysis and the creation of the BI opportunity map set the stage for a sequence of investments in BI. Low end BI annual investments range from $1 million and $5 million for departmental or functional BI investments to around $5 million or more at enterprise level. From these magnitudes it is clear that the BI opportunity analysis be conducted with appropriately that is drilling down and closely examining each potential BI investment.



Business-centric methods go beyond traditional approaches into defining the business value capture mechanism for each BI project. This includes determining and specifying a priori *how business processes and key decision processes must change* in order to leverage BI investments. Business-centric BI methods extend the technical methods of BI pioneers such as Inmon, Kimball, and Imhoff by designing ROI into BI initiatives from the outset and by systematically driving the use of BI into the core business processes and decisions that determine business results. The BI opportunity analysis team must rigorously examine the business value capture mechanism for each identified opportunity and the important point is to effectively involve business executives and managers so as to main high quality of business map and business analysis so as to support a multi-million dollar investment. This will involve developing a "as-is" and "to-be" process models as a means of understanding Questions such as How does a process works , how it may work after BI implementations, and How much change in profits and costs will occur in BI affected processes.

"I believe Business Objects is on the cusp of becoming a multi-billion-dollar sales company. There is tremendous growth potential for business intelligence.."

—Bernard Liautaud

Risk-Reward Tradeoffs and the Business Intelligence Opportunity Map

It is imperative to evaluate the business risks and the technical risks associated with the proposed BI investments on the BI opportunity map as many BI initiatives though been successful from a technical perspective but had been failures from the business perspective. BI readiness assessment framework should be used to tackle with this. It can be used to gauge a company's prerequisites to leverage BI successfully and thus create incremental profits.

- *Strategic alignment.* The BI investment must be used within a core business process that makes a difference in profits.
- *Business and IT partnership*. Business must bear responsibility for the process changes required to leverage BI investments.

- *Continuous process improvement culture*. Necessary to be positioned to leverage BI-driven process changes that impact profits.
- *Culture that supports the use of information and analytical tools.* Well positioned to leverage BI-driven process changes that impact profits.
- *Decision process engineering culture*. If decision making style is structured better opportunity to insert BI into decision processes.
- *BI and DW technical readiness*. Inappropriate experience with the technical aspects of BI and DW can lead to failures
- *BI portfolio management*. Manage BI as a portfolio to leverage BI investments and technical experience reducing costs and technical risk.

BI opportunity analysis explicitly considers the business value capture mechanism of each BI project, manages BI risks, and manages the process changes required to deliver the promised business value. It is part of broader business-centric BI method that seeks to ensure that BI investments deliver business value.

Content Management System

A content management system (CMS) is a system used to manage the content of a Web site. Typically, a CMS consists of two elements: the content management application (CMA) and the content delivery application (CDA). The CMA element allows the content manager or author, who may not know Hypertext Markup Language (HTML), to manage the creation, modification, and removal of content from a Web site without needing the expertise of a Webmaster. The CDA element uses and compiles that information to update the Web site. The features of a CMS system vary, but most include Web-based publishing, format management, revision control, and indexing, search, and retrieval.

The Web-based publishing feature allows individuals to use a template or a set of templates approved by the organization, as well as wizards and other tools to create or modify Web content. The format management feature allows documents including legacy electronic documents and scanned paper documents to be formatted into HTML or Portable Document Format (PDF)

for the Web site. The revision control feature allows content to be updated to a newer version or restored to a previous version. Revision control also tracks any changes made to files by individuals. An additional feature is indexing, search, and retrieval. A CMS system indexes all data within an organization. Individuals can then search for data using keywords, which the CMS system retrieves.

A CMS system may also provide tools for one-to-one marketing. One-to-one marketing is the ability of a Web site to tailor its content and advertising to a user's specific characteristics using information provided by the user or gathered by the site (for example, a particular user's page sequence pattern). For example, if you visit a search engine and search for "digital camera," the advertising banners will advertise businesses that sell digital cameras instead of businesses that sell garden products.

Two factors must be considered before an organization decides to invest in a CMS. First, an organization's size and geographic dispersion must be considered especially if an organization is spread out over several countries. For these organizations, the transition to CMS is more difficult. Secondly, the diversity of the electronic data forms used within an organization must be considered. If an organization uses text documents, graphics, video, audio, and diagrams to convey information, the content will be more difficult to manage.

Advantages of Content Management System

Content management systems make your content smarter and more powerful. Because content is stored only one time no matter how many times it is used, the system can track everything that happens to it. And editors only have to handle the content one time while the changes are made globally within and across all documents. Some other advantages are mentioned below:

- *Quick and easy page management* – Any approved user can quickly and easily publish online without complicated software or programming.
- *Workflow management* – An integrated workflow process facilitates better content management.

- *Flexibility for developers* – Because the CMS enables non-technical users to easily publish content, this frees up technical developers to focus on functionality and enhanced features.
- *Design is separate from content* – You can manipulate content without fear of accidentally changing the design.
- *Database-driven* – You only need to change data once for it to be updated throughout your site.
- *Shared resources* – Website managers will have access to shared resources, such as modules, images, audio and video files, etc.
- *Approval systems* – You can give different levels of access to different users, and the CMS has mechanisms to ensure content is approved before going live.
- *Mobile ready* – The CMS automatically scales your site to fit tablets, mobile devices and smaller browser windows.
- *Archive capabilities* –You can track who has made changes to your page and archive previous versions of your page.
- *Remote access* – You can access and update your site from anywhere with an Internet connection.
- *Security* – Security is automatic.
- *Search engine-friendly* – The CMS helps to optimize your website so that search engine users can easily find your information.
- *Updates*— The CMS allows alerts to be set to notify the editor when content needs to be reviewed, updated or removed. This will help prevent old data from being presented and misinforming users.

END-USER SEGMENTATION

All business people have to deal with data in some form and with computing technologies to get their jobs done. Some can work with tools such as spreadsheets, and some are quite "dangerous." The varying degree of skills and knowledge base among users can be staggering. People tend to learn from others, and they learn bad practices along with the good ones.There are user types delineated by application requirements and by skill. If you look at the users in light of their application requirements, there are three distinct types:

- Traditional query and reporting
- OLAP
- Data mining

A pyramid of users by BI application type typically looks like the one depicted in figure 8.2.

The majority of your BI users will fall in the traditional query and reporting area. This has nothing to do with their level of skill or their position within the enterprise. The most important user of all may receive one small report once a month. This report may be responsible for the majority of territory assignments and critical sales efforts throughout the company. This user may have minimal computer skills and no involvement in the creation or maintenance of the output.

A common error in the BI space is to assign business value to the end user's skills in handling a tool. No one is negating or minimizing the value that a power-user may provide, but there is no correlation between the power-user's ability to perform technical work and the impact upon the business.

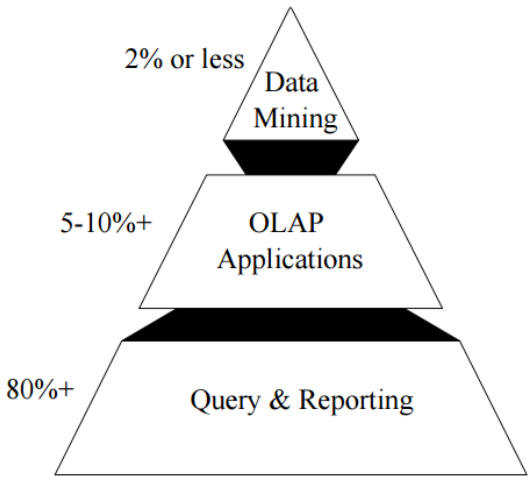


Figure 8.2: End-user segmentation by product type.

The majority of BI users identified by degree of skill and involvement in the processes will be casual users at best. The profile of the number of users by technical skills may appear as shown in figure 8.3.

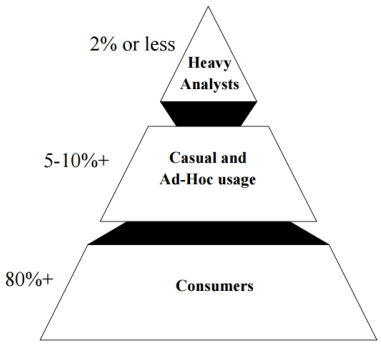


Figure 8.3: End-user segmentation by technical skills.

Note that there are large number of users who have little skill or involvement in the technical end of BI and also a large number of users who interoperate with BI applications at the query and reporting level and not in the more sophisticated types such as data mining.

User segmentation by business impact populations is shown in figure 8.4. Using business impact by user more accurately reflects the overall benefit to the enterprise than cranking out thousands of reports at all levels and assuming the tonnage will produce gold.

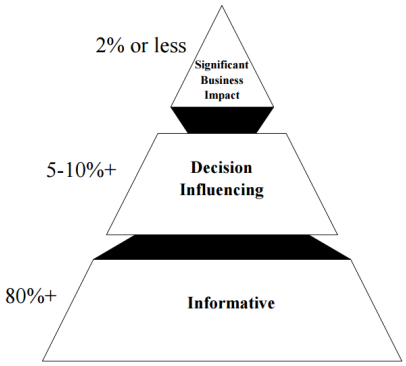


Figure 8.4: End-user segmentation by business impact.

Basic Reporting and Querying

A querying and reporting tool helps you run regular reports, create organized listings, and perform cross-tabular reporting and querying. Here are some querying and reporting tools to familiarize yourself with.

The Role of Structured Query Language (SQL)

SQL is the official database query language used to access and update the data contained within a relational database management system, or RDBMS.

The significance of SQL for querying and reporting is that the language has represented a *mostly* standard way to access multiple relational database management system (RDBMS) products.

Each RDBMS product has a slightly different SQL dialect. Although the basic syntax is the same, especially for the most commonly used commands, all SQL dialects are slightly different. In the early 1990s, despite these syntactical differences, several different efforts provided a common gateway to SQL RDBMS products.

The most successful was Microsoft Open Database Connectivity (ODBC). The phrase *ODBC-compliant* became important to RDBMS applications in the early and mid-1990s. A similar standard for the Java community also emerged in the late-1990s — Java Database Connectivity (JDBC). Virtually all major database manufacturers today have accessibility through both of these standard connectivity interfaces.

Technical Query Tools

The use of SQL as the basis for most querying and reporting tools was both good and bad for data warehousing. On the positive side, many more product-to-product matchups are possible in data warehousing environments, enabling tools to be provided both by RDBMS vendors and other third-party vendors.

On the negative side, though, SQL is a relatively complex language after you get past the basics. A series of query tools primarily allow users to type in and edit SQL queries. These tools are not really designed for end users, though it's amazing how often they are deployed in end-user organizations.

User Query Tools

Most end-user querying and reporting tools provide visually oriented, painting, environments that enable users to design

screens for report layouts, the data columns desired for the report, or the rows of data that they want to select (only salespeople who have met their quota, for example). Using all this "painted" information, most tools have increasingly taken a smart query generation approach. Instead of generating a single, overly complex SQL statement that could get you an A in database class but draw a disgusted shake of the head from someone who has done this stuff in the real world, a sequence of SQL statements (usually taking advantage of temporary tables for intermediate results) is generated. This sequence, in effect, decomposes the query into a more efficient series of steps.

Reporting Tools

When end users want more complex user interaction or sophisticated formats, a tool with more reporting features is leveraged. You can find a separation between pure query tools and pure reporting tools. The query tool provides data access, filtering, and simple formatting. If you are distributing reports across your enterprise or need to generate form-safe presentation, you use a reporting tool. Like with query tools, reporting tools provide an environment that enables you to create sophisticated layouts that focus on formatting the data retrieved by the database query.

"I think you might see us growing much deeper into banking. You might see us acquiring companies in the banking area. You might see us acquiring companies in the retail area. I think you might see us acquiring companies in the telecommunications. I think you will see us getting stronger in business intelligence.

—Larry Ellison

Online Analytical Processing (OLAP)

OLAP (online analytical processing) is computer processing that enables a user to easily and selectively extract and view data from different points of view. For example, a user can request that data be analyzed to display a spreadsheet showing all of a company's beach ball products sold in Florida in the month of July, compare revenue figures with those for the same products in September, and then see a comparison of other product sales in Florida in the same time period. To facilitate this kind of analysis, OLAP data is stored in a multidimensional

database. Whereas a relational database can be thought of as two-dimensional, a multidimensional database considers each data attribute (such as product, geographic sales region, and time period) as a separate "dimension." OLAP software can locate the intersection of dimensions (all products sold in the Eastern region above a certain price during a certain time period) and display them. Attributes such as time periods can be broken down into sub-attributes.

OLAP can be used for data mining or the discovery of previously undiscerned relationships between data items. An OLAP database does not need to be as large as a data warehouse, since not all transactional data is needed for trend analysis. Using Open Database Connectivity (ODBC), data can be imported from existing relational databases to create a multidimensional database for OLAP.

Two leading OLAP products are Hyperion Solution's Essbase and Oracle's Express Server. OLAP products are typically designed for multiple-user environments, with the cost of the software based on the number of users.

OLAP functionality is characterized by dynamic multi-dimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities including:

- calculations and modeling applied across dimensions, through hierarchies and/or across members
- trend analysis over sequential time periods
- slicing subsets for on-screen viewing
- drill-down to deeper levels of consolidation
- reach-through to underlying detail data
- rotation to new dimensional comparisons in the viewing area

OLAP is implemented in a multi-user client/server mode and offers consistently rapid response to queries, regardless of database size and complexity. OLAP helps the user synthesize enterprise information through comparative, personalized viewing, as well as through analysis of historical and projected data in various "what-if" data model scenarios. This is achieved through use of an OLAP Server.

OLAP Server

An OLAP server is a high-capacity, multi-user data manipulation engine specifically designed to support and operate on multi-dimensional data structures. A multi-dimensional structure is arranged so that every data item is located and accessed based on the intersection of the dimension members which define that item. The design of the server and the structure of the data are optimized for rapid ad-hoc information retrieval in any orientation, as well as for fast, flexible calculation and transformation of raw data based on formulaic relationships. The OLAP Server may either physically stage the processed multi-dimensional information to deliver consistent and rapid response times to end users, or it may populate its data structures in real-time from relational or other databases, or offer a choice of both. Given the current state of technology and the end user requirement for consistent and rapid response times, staging the multi-dimensional data in the OLAP server is often the preferred method.

Types of OLAP Servers

There are four types of OLAP servers:

- Relational OLAP (ROLAP)
- Multidimensional OLAP (MOLAP)
- Hybrid OLAP (HOLAP)
- Specialized SQL Servers

Relational OLAP: ROLAP servers are placed between relational back-end server and client front-end tools. To store and manage warehouse data, ROLAP uses relational or extended-relational database management system (RDBMS).

ROLAP includes the following:

- Implementation of aggregation navigation logic.
- Optimization for each DBMS back end.
- Additional tools and services.

Multidimensional OLAP: MOLAP uses array-based multidimensional storage engines for multidimensional views of data. With multidimensional data stores, the storage utilization may be low if the data set is sparse. Therefore, many MOLAP

server use two levels of data storage representation to handle dense and sparse data sets.

Hybrid OLAP (HOLAP): Hybrid OLAP is a combination of both ROLAP and MOLAP. It offers higher scalability of ROLAP and faster computation of MOLAP. HOLAP servers allows to store the large data volumes of detailed information. The aggregations are stored separately in MOLAP store.

Specialized SQL Servers: Specialized SQL servers provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.

OLAP Techniques

Online analytical processing or OLAP is implemented in many different ways. However, the commonest way is to stage information retrieved from different corporate databases, such as data warehouses, is *staged* stored temporarily into OLAP multi-dimensional databases for retrieval by the front-end systems. The multidimensional database is optimized for fast retrieval. Several techniques for the quickening of data retrieval and analysis are used on the procedural side of the database management.

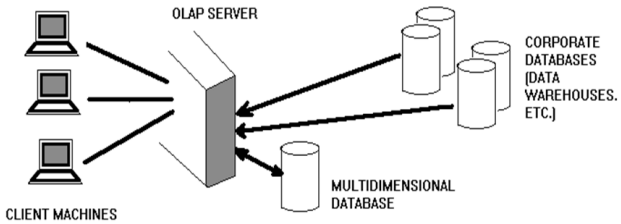


Figure 8.5: OLAP implementation.

OLAP is basically concerned with multidimensionality of data. Multidimensional models take advantage of inherent relationships in data to populate data in multi-dimensional matrices called *Data Cubes or Hypercubes* (for more than three dimensions). A standard spreadsheet, signifying a conventional database, is a two-dimensional matrix. One example would a spreadsheet of regional sales by product for a particular time period. Products could be shown as rows with sales revenues for

each region comprising the columns. Adding a time dimensional such as the organization's fiscal quarters would produce a three-dimensional matrix, which could be represented using a cube.

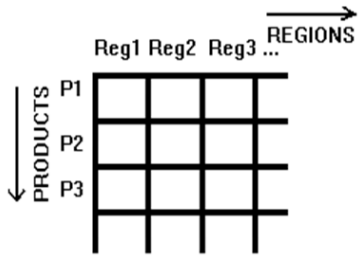


Figure 8.6: A two dimensional database matrix.

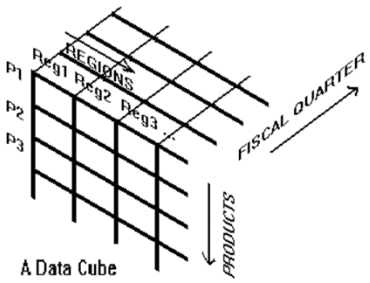


Figure 8.7: A data cube.

In figure 8.5, there is a three dimensional data cube that organizes product sales data by fiscal quarters and sales regions. Each cell could contain data for specific product, specific fiscal quarter and specific region. By including additional dimensions, a data hyper-cube could be produced, although more than three dimensions cannot be graphically visualized. The data can be queried directly in any combinations of dimensions, bypassing complex database queries.

A multidimensional model of storage involves two types of tables – *dimension tables* and *fact tables*. A dimension table simply stores the tuples of attributes of the dimension. A fact table can be thought of as having tuples, one per a recorded fact. This fact contains some measured or observed variable(s), and identifies it with pointers to dimension table. The fact table contains data and the dimensions identify each tuple in that data.

Changing from one dimension hierarchy to another is easily achieved in a data cube by a technique called *Pivoting (also called rotation)*. Several other techniques go into making a complete OLAP implementation. These vary from drilling down into finer details, from different databases, to rolling up and grouping into larger units.

OLAP provides a user-friendly environment for interactive data analysis. A number of OLAP data cube operations exist to materialize different views of data, allowing interactive querying and analysis of the data.

Here is the list of OLAP operations:

- Roll-up
- Drill-down
- Slice and dice
- Pivot (rotate)

Roll-up

Roll-up performs aggregation on a data cube in any of the following ways:

- By climbing up a concept hierarchy for a dimension
- By dimension reduction

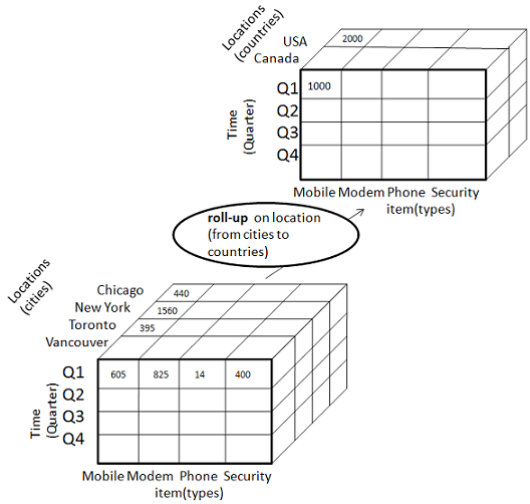


Figure 8.8: Roll-up operation.

- Roll-up is performed by climbing up a concept hierarchy for the dimension location.
- Initially the concept hierarchy was "street < city < province < country".
- On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
- The data is grouped into cities rather than countries.
- When roll-up is performed, one or more dimensions from the data cube are removed.

Drill-down

Drill-down is the reverse operation of roll-up. It is performed by either of the following ways:

- By stepping down a concept hierarchy for a dimension
- By introducing a new dimension.

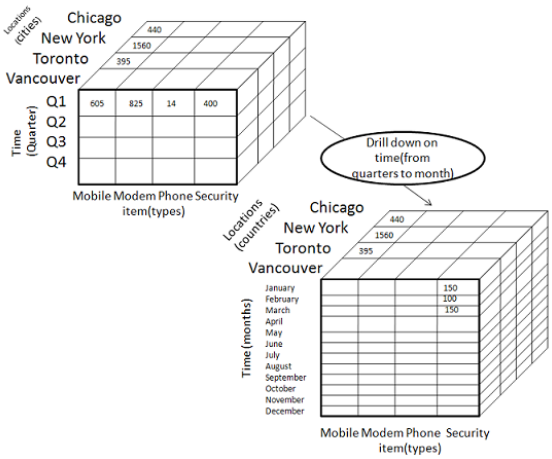


Figure 8.9: Drill-down operation.

"In an ever-faster moving world, disruption and uncertainty represent both challenges and opportunities in enterprise. Modern cloud-based tools for scenario planning, forecasting, and managing future value creation enable companies to move even faster and outperform the competition."

—Florian Winterstein

- Drill-down is performed by stepping down a concept hierarchy for the dimension time.

- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.
- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.

Slice

The slice operation selects one particular dimension from a given cube and provides a new sub-cube.

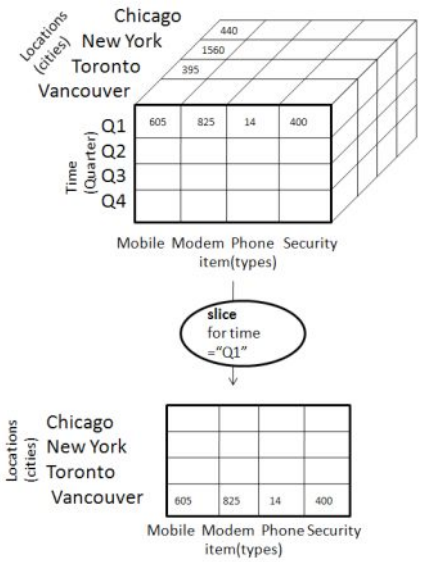


Figure 8.10: Slice operation.

- Here Slice is performed for the dimension "time" using the criterion time = "Q1".
- It will form a new sub-cube by selecting one or more dimensions.

Dice

Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following figure that shows the dice operation.

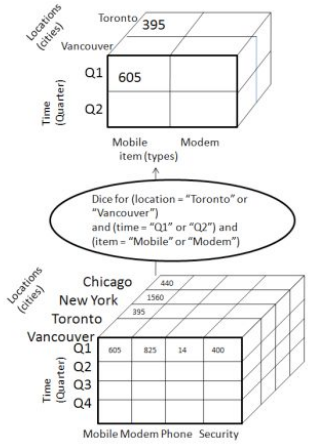


Figure 8.11: Dice operation.

The dice operation on the cube based on the following selection criteria involves three dimensions.

- (location = "Toronto" or "Vancouver")
- (time = "Q1" or "Q2")
- (item =" Mobile" or "Modem")

Pivot

The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data. Consider the following figure that shows the pivot operation.

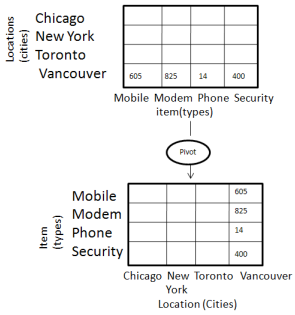


Figure 8.12: Pivot operation.

In this the item and location axes in 2-D slice are rotated.

"In the future, I think it's pretty plausible that collective intelligence tools and skills will be important in order to be a part of global dialog, global business, and global creativity. People who know how to negotiate collective intelligence networks are going to be in a good position to contribute to global society."

—Jane McGonigal

OLAP Applications

OLAP is widely used in several realms of data management. Some of these applications include: -

Financial Applications

- Activity-based costing (resource allocation)
- Budgeting

Marketing/Sales Applications

- Market Research Analysis
- Sales Forecasting
- Promotions Analysis
- Customer Analyses
- Market/Customer Segmentation

Business modeling

- Simulating business behavior
- Extensive, real-time decision support system for managers

All of the above applications need ability to provide managers with the information they need to make effective decisions about an organization's strategic directions. The key indicator of a successful OLAP application is its ability to provide information, as needed, that is, its ability to provide "just-in-time" information for effective decision-making. This requires more than a base level of detailed data.

Applying the OLAP to Data Warehousing

Data warehousing is primarily used to archive data that is of the nature of business knowledge. A primary requirement is the

fast analysis of this shared data, resulting in multidimensional views of the data, which in turn results in knowledge acquisition. OLAP comes into play when it is required to analyze data stored in a warehouse, so as to generate complex results, in a time-constrained environment that is, just-in-time information. Most data warehouses support, what is called Ad hoc querying, which implies that any combination of complex queries can be executed against the stored data.

Several tradeoffs must be made when applying OLAP to warehouses, particularly because of the bulk of data that has to be analyzed. For example, an OLAP system must be able to process a complex query in the shortest possible time. For this, it can use two approaches to have pre-calculated data, or to apply all calculations on the fly. The first approach requires huge amounts of data storage area, and hence exotic hardware support, which proves to be cost-ineffective. The second approach is to apply calculations, on the fly, which results in slower execution of the queries. Thus, an optimal approach is to pre-calculate some data, and calculate in real-time other parts of the queries. These elements need to be balanced in an optimal way for a good implementation of OLAP systems.

Benefits of using OLAP

OLAP holds several benefits for businesses: -

- OLAP helps managers in decision-making through the multidimensional data views that it is capable of providing, thus increasing their productivity.
- OLAP applications are self-sufficient owing to the inherent flexibility provided to the organized databases.
- It enables simulation of business models and problems, through extensive usage of analysis-capabilities.
- In conjunction with data warehousing, OLAP can be used to provide reduction in the application backlog, faster information retrieval and reduction in *query drag*.

The applications and benefits of online analytical processing (OLAP) indicate clearly that it is one of the revolutionary technologies that can be used extensively in data management. All businesses, big or small, can benefit from using OLAP, at some scale or other. The instant access to information and the

apparent knowledge gained from this information is invaluable considering against the cost involved in establishing this setup.

Data Mining

Data mining is sorting through data to identify patterns and establish relationships.

Data mining parameters include:

- Association - looking for patterns where one event is connected to another event
- Sequence or path analysis - looking for patterns where one event leads to another later event
- Classification - looking for new patterns (May result in a change in the way the data is organized but that's ok)
- Clustering - finding and visually documenting groups of facts not previously known
- Forecasting - discovering patterns in data that can lead to reasonable predictions about the future (This area of data mining is known as predictive analytics.)

Data mining techniques are used in a many research areas, including mathematics, cybernetics, genetics and marketing. Web mining, a type of data mining used in customer relationship management (CRM), takes advantage of the huge amount of information gathered by a Web site to look for patterns in user behavior.

Data can be mined whether it is stored in flat files, spreadsheets, database tables, or some other storage format. The important criteria for the data is not the storage format, but its applicability to the problem to be solved.

DASHBOARD

A business intelligence dashboard is a data visualization tool that displays the current status of metrics and key performance indicators (KPIs) for an enterprise. Dashboards consolidate and arrange numbers, metrics and sometimes performance scorecards on a single screen. They may be tailored for a specific role and display metrics targeted for a single point of view or department.

The essential features of a BI dashboard product include a customizable interface and the ability to pull real-time data from multiple sources.

Oracle and Microsoft are among the vendors of business intelligence dashboards. BI dashboards can also be created through other business applications, such as Excel. Business intelligence dashboards are sometimes referred to as enterprise dashboards.

The business intelligence dashboard is often confused with the performance scorecard. The main difference between the two, traditionally, is that a business intelligence dashboard, like the dashboard of a car, indicates the status at a specific point in time. A scorecard, on the other hand, displays progress over time towards specific goals. Dashboard and scorecard designs are increasingly converging. For example, some commercial dashboard products also include the ability to track progress towards a goal. A product combining elements of both dashboards and scorecards is sometimes referred to as a scoreboard.

"Web applications will become more and more ubiquitous throughout our human environment, with walls, automobile dashboards, refrigerator doors all serving as displays giving us a window onto the Web."

—Tim Berners-Lee

Characteristics of a Dashboard

A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance.

Here are the key characteristics of a dashboard:

- All the visualizations fit on a single computer screen — scrolling to see more violates the definition of a dashboard.
- It shows the most important performance indicators / performance measures to be monitored.
- Interactivity such as filtering and drill-down can be used in a dashboard; however, those types of actions should

not be required to see which performance indicators are under performing.

- It is not designed exclusively for executives but rather should be used by the general workforce as effective dashboards are easy to understand and use.
- The displayed data automatically updated without any assistance from the user. The frequency of the update will vary by organization and by purpose. The most effective dashboards have data updated at least on a daily basis.

The Benefits of Business Dashboards

A dashboard is an easy to read, one page summary of the analysis of the information. It is an overview of your system at a glance. There are many advantages that result in the utilization of this tool. Some of the most important benefits are:

- *Customizable.* - Dashboards could be customized in terms of users and expectations. Each decision level dashboard can be customized to present the most valuable and useful set of information. This allows each person to see the level of detail that they need in order to get their job done and meet their goals.
- *All-in-one*. - In the past users would spend large amount of time reviewing and analyzing different reports to end in a final conclusion. This tool allows to see, at a glance, an overall situation report of the desired information.
- *Drill into detail*. - But, having all-in-one does not means the absence of details. Dashboards are developed with the ability to get as deeper in information as required by simply selecting the desired variable or object.
- *Intuitive data presentation*. - There is no need for complicated and exhaustive training. Dashboards are design to be intuitive to any user. The graphic design allows an easy and smooth navigation throughout the information.
- *Mobile device accessible*. - Most dashboards software are programmed to suit any mobile device. The idea is to reach anywhere, to everyone, in a timely manner with the most accurate information.

The dashboard method of reviewing details and viewing the status of operations provides a significant opportunity to make your business more efficient and quick to respond to issues and opportunities.

Advanced/Emerging BI Technologies

The business intelligence continues to advance today as the vendor's offer more powerful and innovative solutions. With all the changes the idea of BI systems remains the same which is timely, accurate, high-value, and actionable insights. The new capabilities serve the simple goals in one way or the other. Some of the advances taking place today are taking the current capabilities and making them better so that more powerful graphic representations of data is available.

In a BI environment, the vendors are always coming out with new tools, but there is no guarantee that the latest thing can help the company with their business problems.

Some of the areas where the research is growing rapidly belong to the following general categories.

- *Visualization:* The BI software can make the information more usable by using visualization techniques. Visualization refers to presenting the numbers, the statistics, the metrics and others that lacks a graphical format which will be easier to understand and interpret.
- *Guided analysis:* The BI tools with the guided-analysis features may have the process wizards that guide the users from step to step, or set up event alerts that will monitor the system or the user's activity and will only offer help when it is required. When a particular trigger conditions are met, the user is alerted and the guided-analysis tools will help to direct the user on how to progress. An interactive, target-oriented BI will have to answer a particular set of questions in a structured way, telling certain actions which will be based on the user's input and analysis.
- *Data mining:* Data mining is all about examining the oceans of pail business data to find useful insights about the past that can act as a guide to the future including

the needle which is really needed from a huge stack of needles. The data that used to need an entire storage room full of documents can be now digitally given and put on a few square millimeters of magnetic tape or a hard drive.

- *Unstructured data tools*: With the unstructured data tools the irregular data can be turned into business insights with the search and indexing capabilities for the hard-to-quantity information formats.

Future of Business Intelligence

BI is at a crossroads; its place as an integral part of Data Management is secure, with everyone from the public to the private sector incorporating data into their operational processes. Yet, as the reliance on data increases, the nature of the role that BI will play in future data integration is split into two diametrically opposed possibilities that can substantially alter the form of data management.

Implications for Big Data

There's no denying the impact of Big Data on the future of BI. The sheer size and variation of Big Data requires a refined set of analytic tools, especially for real-time analysis – one of the most useful BI features. These tools will need to account for both unstructured and structured data, and expand beyond relational technologies for useful integration. Such cutting edge advancements can almost certainly be handled better by formally educated BI analysts instead of laymen.

The trouble with using conventional BI tools to analyze Big Sata is the speed – or lack thereof. However, a number of BI vendors have recently developed applications specifically designed for Big Data and Hadoop, the most commonly used open source filing system. Cloudera has created an open source query engine that operates in real time and is significantly faster than traditional Hive querying; it has garnered support from a number of BI product developers. Other vendors are projected to release Data Visualization tools and applications that can cache querying results, allowing users to sift through them at their leisure. These products will make Big Data and Hadoop a lot

more viable for BI analysis, although much of this technology is still being developed and best managed by analysts specifically trained in this field.

Simplification

Every product needs to end up in the hands of its final user. BI's real users are the decision makers in a company, statuesquely called the "Executives". BI has successfully transitioned from the obscure backrooms of IT into the realms of analysts. However, the journey will end when the Executives can use appropriate tools to aid their primary job – making decisions.

There are several developments in the world of BI that are helping to demystify its capabilities and render it more functional for the average executive. The trend to simplify its tools and market solutions towards such professionals is already underway. Conventional BI accessibility – through the IT department – is not always expedient enough to keep up with the demands for data, largely because of the need for IT "middlemen" to sort through data first before significance is attached to it. Developments in BI for Big Data that improve the speed and value of such data will also influence the simplification of tools that will be able to perform many of the same functions for little data, including deep analysis of varying sources in real time. By providing information from a variety of sources that relate to a specific transaction or customer input via streaming, BI will become considerably more viable for executives looking to utilize this software on their own.

The Mobile Movement and Analytics

There are two interrelated factors that are paving the way for the simplification of BI for business users. One is the increasing usage of mobile devices, which provide access to proprietary data remotely and within the workplace. Several vendors have developed dashboards and Data Visualization techniques specifically for these devices, which will likely continue due to the lowered cost of enterprises encouraging employees to work with their own handsets – or simply issuing them instead of traditional computers. By providing tools that work with these autonomous devices, BI vendors are subtly encouraging

executives to utilize these capabilities without third-party intervention. The second factor encouraging a future in which BI is simplified and incorporated into the daily functions of those working in business is the ubiquity of analytics. Although BI offers a variety of tools to extract meaning from reams of data, its predictive analytics capability is one of its more frequently used. Analytics are everywhere, especially in mobile devices, and are playing a growing role in the management of operations, strategies, and fundamental aspects of people's lives. The ubiquity of analytics encourages an environment in which tools such as BI are used as frequently as possible by business representatives; the fact that so many of them are available on mobile applications reinforces this trend.

In Hindsight: Looking Back to the Future

The need for formal education for BI analysts will likely continue in the coming years, although it will eventually become less necessary due to the emerging simplification of BI tools that will encourage executives to use this software more autonomously and effectively. In much the same way that technology tends to transition from being used by the elite to the general population, BI is in the process of evolving into a vital series of tools that greatly inform real-time decisions for executives, and will eventually become marketed as such.

The key to the timeframe in which this realization of BI's full potential will occur pertains to its simplification process that not only includes visualization techniques and dashboards for mobile devices, multiple source analysis streamed in real time and the prevalence of analytics, but also its increasing use for identifying diverse entities and tracing data lineage. These final two applications are projected to expand in the coming years as more sources of data and regulations regarding discoverability become prominent issues in data management. The presence of inexpensive, reliable data discovery tools for the desktop and handset will fuel BI's progression from a dedicated search for a particular piece of information to a readily accessible component of the daily elucidation of pertinent facts related to operational processes.

3G E-LEARNING

CASE STUDY

ZARA : APPLICATION OF BUSINESS INTELLIGENCE IN RETAIL INDUSTRY

ZARA is a Spanish clothing and accessories retailer based in Arteixo, Galicia. Founded in 24 May ,1975 by Amancio Ortega and RosalÃa Mera, the brand is renowned for it's ability to deliver new clothes to stores quickly and in small batches. Zara needs just two weeks to develop a new product and get it to stores, compared to the six-month industry average, and launches around 10,000 new designs each year. Zara was described by Louis Vuitton Fashion Director Daniel Piette as "possibly the most innovative and devastating retailer in the world. The company produces about 450 million items a year for its 1,770 stores in 86 countries.

The Zara has made of use of Information Systems (IS) and to advance in many areas. This has resulted in huge success for the company. This included application of Business intelligence (BI) involves technologies, practices for collection, integration and applications to analyze and present business information. The main aim of business intelligence is to promote better business decision making.

BI describes a group of information on concepts and methods to better decision making in business. This is achieved by employing a fact based support systems. The intelligence systems are data-driven and sometimes used in executive information systems. Predictive views on business operations can be provided by use of BI systems. predictive views on business operations can be provided by use of BI systems since historical and current data has been gathered into a data bank performance management benchmarking is done whereby information on other companies in the same industry is gathered.

Since the Zara's have large network and therefore dealing with large volumes of data, an enterprise information system has been employed in the firm. This is generally a type of computing systems that involves an enterprise class that is, typically offering total quality service handling large volumes of data and able to sustain a big organization. With this system, a technology platform is provided which enables the enterprise is provided to that information can be shared in all useful levels of management enterprise systems are important in removing the problem of fragmentation of information. This happens when there are numerous information systems in an enterprise. The problem is solved by developing a standard data structure.

The Zara being big organization, the enterprise systems is housed in many different data centers, and includes content management system as the main application. The

Zara team comprises technology professionals. These include content specialists, network and system engineers, flash developers, database business analysts and administrators, software developers, quality assurance managers and computer and applications support technicians. All these specialists work in tandem to bring about competitive advantage to business by allowing for quick-response capability.

The Zara is devoted to integrating information technology appropriately into all areas of its operations and activities. The range of services and resources available to its clients is attributed to the commitment of integration of IT properly in the organization. The client services group in the Zara partners with staff and clients to identify and meet each group's technological requirements. This group of technical advisors associates with departments to execute a roadmap for a team's technological vision and then defines this vision within entity projects.

Much support is required for invariable innovation. The Zara IT group is devoted to developing community through technology and operates closely with business associates. The IT group is devoted to developing community through technology and operates closely with business associates. The IT group has technical support and tools they require to come up with new ideas and spread the ideas to the wider community worldwide. This ensures that client needs are realized. A constantly growing state-of-the-art technology infrastructure has enabled the firm to develop and maintain a fully integrated organization/ enterprise. The infrastructure has enabled the firm to develop and maintain a fully integrated organization/ enterprise. The infrastructure entails a core of systems and attempts designed to produce the flexibility and capacity for innovation and growth.

The Zara thrives in an environment of change, experimentation and learning that is spreading over the boundaries for the application of IT in their business in the enterprises an easy-to-use modular tools, templates and platforms that involve all sides of life at the Zara firms including career development , administration and operations are implemented and developed.

In the Zara other web-based solutions are deployed with an advance knowledge management thereby making a big shift in the quality and speed of work in how the enterprises function. In the exploding growth of the software market a new world growth for the software market a new world of connectivity is realized in the Zara. The urgency of the business recognizing the importance of corporate portal has enabled linkage of information, data, people and knowledge to provide business solutions. The corporate portals come from consumer portals like Alta-vista, yahoo! and Lycos.

The portals (gateways) show the importance of letting clients have a wide scope of varied information on the web. This has given rise to the increase in multitasking,

receiving information and checking from varied sources and thereby getting involved in projects that cross geographical boundaries with this technology the needs of the community, employees and even the extended network that is more advanced are served.

The Zara is possibly the most devastating and innovative retailer in the world. With more than 1000 shops world wide, the Zara has turned controlled over garment factories into a competitive advantage by making and designing the garments. By making the garments itself, it can quickly react to varying market trends. The Zara has been able to succeed in building a massive brand, without promotion or advertising but through the information systems and information technology.

Questions

1. How does Zara made the use of Information Systems (IS) and to advance in many areas of retails?
2. Prepare a project report and mention content management system as the main application of Zara organization.

KEY TERMS

Data visualization	Data mining	Content management system
Decision-making	Business value	Return on investment
Business-centric methods	Business risks	Portable document format
Benchmarking	Technical risks	Business performance management

DISCUSSION QUESTIONS

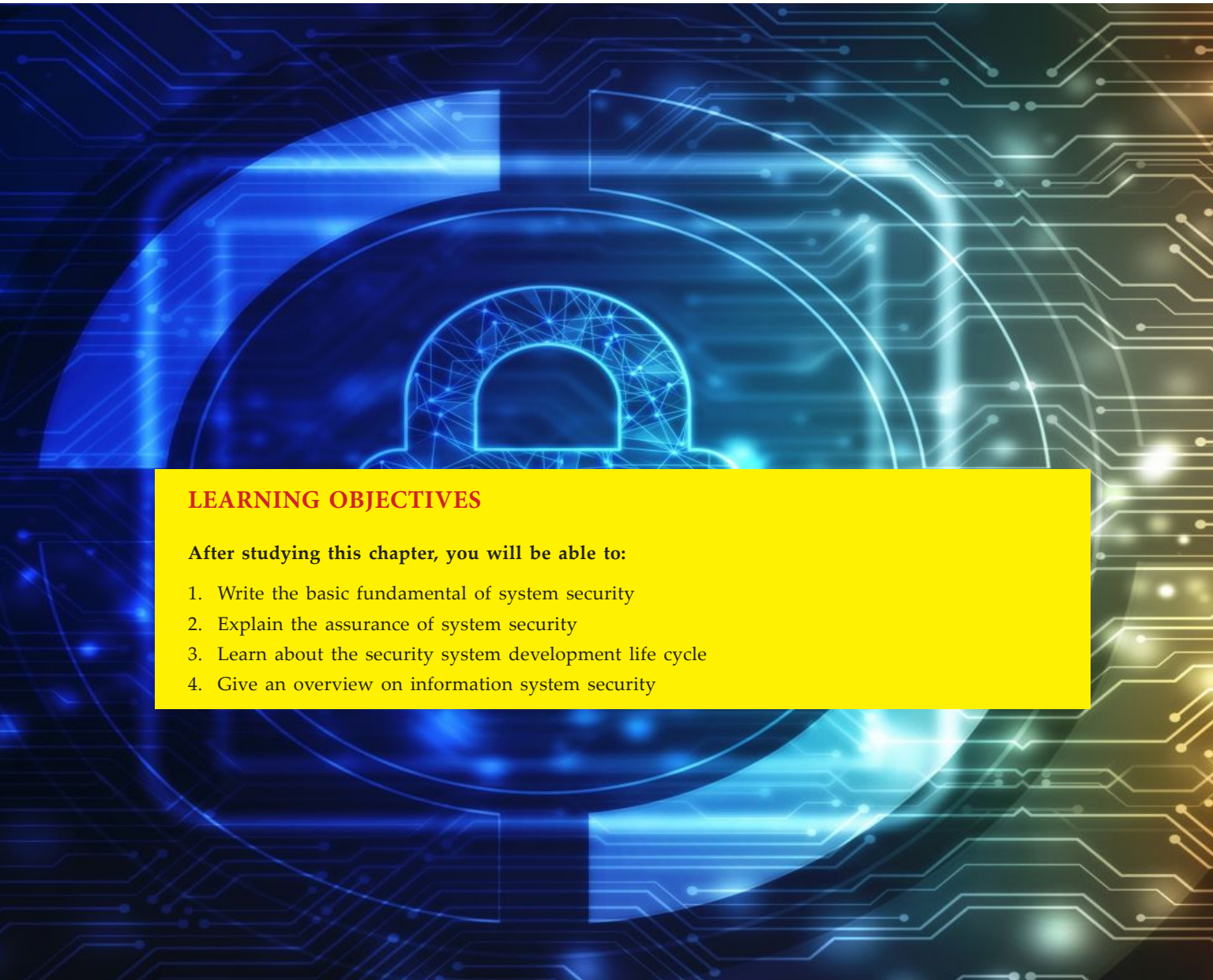
1. What is content management system (CMS)? Write the purpose of CMS.
2. Explain the purpose of cubes.
3. What is end-user segmentation? Describe.
4. Describe online analytical processing (OLAP).
5. What is business intelligence dashboard? Write its importance.

REFERENCES

1. Adomavicius, G., Tuzhilin, A.: Using Data Mining Methods to Build Customer Profiles. IEEE Computer 34(2) (2001)
2. Andrew Brust (14 February 2013). "Gartner releases 2013 BI Magic Quadrant". ZDNet. Retrieved 21 August 2013.
3. Chugh, R & Grandhi, S 2013, 'Why Business Intelligence? Significance of Business Intelligence tools and integrating BI governance with corporate governance', International Journal of E-Entrepreneurship and Innovation,
4. Fischer, Gerhard (2001), "User Modeling in Human-Computer Interaction", User Modeling and User-Adapted Interaction 11: 65–86
5. Golden, Bernard (2013). Amazon Web Services For Dummies. For dummies. John Wiley & Sons.
6. Julian, Taylor (10 January 2010). "Business intelligence implementation according to customer's needs". APRO Software. Retrieved 16 May 2016.
7. Peter Rausch, Alaa Sheta, Aladdin Ayesh : Business Intelligence and Performance Management: Theory, Systems, and Industrial Applications, Springer Verlag U.K., 2013,
8. Ross, E., Intelligent user interfaces: Survey and research directions, University of Bristol, Bristol, UK, 2000.
9. Xing, F. Z.; Cambria, E.; Welsch, R. E. (2018). "Natural language based financial forecasting: a survey" (PDF). Artificial Intelligence Review. 50 (1): 49–73.

CHAPTER 9

SYSTEM SECURITY



LEARNING OBJECTIVES

After studying this chapter, you will be able to:

1. Write the basic fundamental of system security
2. Explain the assurance of system security
3. Learn about the security system development life cycle
4. Give an overview on information system security

INTRODUCTION

System security goes hand-in-hand with data security. System security describes the controls and safeguards that an organization takes to ensure its networks and resources are safe from downtime, interference or malicious intrusion. If data security is meant to protect the information in the books in the library, then system security is what protects the library itself.

Here are some of the common techniques for cyber-attacks, and what your organization can do to mitigate the risk.

- **Backdoor attack:** Many computer networks might not be as secure as they seem. Sometimes programmers leave in code that allows them to access the network easily, usually for debugging purposes. Hackers might exploit these weak points. Be sure to review the code for any customized software used at your organization and that software-as-a-service and platform-as-a-service suppliers are not vulnerable to these kinds of attacks.
- **Denial of service (DoS) attack:** Instead of breaking into your computer network, malicious parties might try to overwhelm it by bombarding it with requests for service, slowing access and network-reliant operations to a crawl. A regular denial of service attack can be stopped by blocking the attacker's IP address. A more sophisticated type of attack, distributed denial of service (DDoS) attack, is harder to stop, as it involves many IP addresses. Several vendors sell solutions that reduce the effects of DDoS attacks.
- **Direct access attack:** When people have access to your physical assets, it's relatively easy to access your most sensitive information. Cyber criminals can simply steal laptops, hard drives and flash drives, or break into an office, and take or copy the devices that contain the information they want. The best defense against this type of attack is heightened security, worker training and information encryption.
- **Malware attack:** In a malware attack, a malicious party gains access to your computer network and then encrypts all your data. To get the encryption key, you must pay a ransom. Typically, the ransom escalates in price over

time, and payment must be made in Bitcoin. Malware usually makes its way onto your network through a virus or worm, so educate your employees about the danger of clicking on suspicious leaks or attachments.

BASIC FUNDAMENTAL OF SYSTEM SECURITY

The security of a computer system is a crucial task. It is a process of ensuring the confidentiality and integrity of the OS. Security is one of most important as well as the major task in order to keep all the threats or other malicious tasks or attacks or program away from the computer's software system. A system is said to be secure if its resources are used and accessed as intended under all the circumstances, but no system can guarantee absolute security from several of various malicious threats and unauthorized access.

The security of a system can be threatened via two violations:

- **Threat:** A program that has the potential to cause serious damage to the system.
- **Attack:** An attempt to break security and make unauthorized use of an asset.

Security violations affecting the system can be categorized as malicious and accidental threats. Malicious threats, as the name suggests are a kind of harmful computer code or web script designed to create system vulnerabilities leading to back doors and security breaches. Accidental Threats, on the other hand, are comparatively easier to be protected against. Example: Denial of Service DDoS attack. Security can be compromised via any of the breaches mentioned:

- **Breach of confidentiality:** This type of violation involves the unauthorized reading of data.
- **Breach of integrity:** This violation involves unauthorized modification of data.
- **Breach of availability:** It involves unauthorized destruction of data.
- **Theft of service:** It involves the unauthorized use of resources.

- **Denial of service:** It involves preventing legitimate use of the system. As mentioned before, such attacks can be accidental in nature.

Security System Goal

Henceforth, based on the above breaches, the following security goals are aimed:

1. **Integrity:** The objects in the system mustn't be accessed by any unauthorized user & any user not having sufficient rights should not be allowed to modify the important system files and resources.
2. **Secrecy:** The objects of the system must be accessible only to a limited number of authorized users. Not everyone should be able to view the system files.
3. **Availability:** All the resources of the system must be accessible to all the authorized users i.e. only one user/process should not have the right to hog all the system resources. If such kind of situation occurs, denial of service could happen. In this kind of situation, malware might hog the resources for itself & thus preventing the legitimate processes from accessing the system resources.

Threats can be classified into the following two categories:

1. **Program Threats:** A program was written by a cracker to hijack the security or to change the behavior of a normal process. In other words, if a user program is altered and further made to perform some malicious unwanted tasks, then it is known as Program Threats.
2. **System Threats:** These threats involve the abuse of system services. They strive to create a situation in which operating-system resources and user files are misused. They are also used as a medium to launch program threats.

Types of Program Threats

1. **Virus:** An infamous threat, known most widely. It is a self-replicating and malicious thread that attaches itself to a system file and then rapidly replicates itself, modifying and destroying essential files leading to a system breakdown.

Further, Types of computer viruses can be described briefly as follows:

- file/parasitic – appends itself to a file
- boot/memory – infects the boot sector
- macro – written in a high-level language like VB and affects MS Office files
- source code – searches and modifies source codes
- polymorphic – changes in copying each time
- encrypted – encrypted virus + decrypting code
- stealth – avoids detection by modifying parts of the system that can be used to detect it, like the read system call
- tunneling – installs itself in the interrupt service routines and device drivers
- multipartite – infects multiple parts of the system

2. **Trojan Horse:** A code segment that misuses its environment is called a Trojan Horse. They seem to be attractive and harmless cover programs but are really harmful hidden programs that can be used as the virus carrier. In one of the versions of Trojan, the User is fooled to enter confidential login details on an application. Those details are stolen by a login emulator and can be further used as a way of information breaches. One of the major as well as a serious threat or consequences of the Trojan horse is that it will actually perform proper damage once installed or run on the computer's system but at first, a glance will appear to be useful software and later turns out to be maliciously unwanted one.

Another variance is Spyware, Spyware accompanies a program that the user has chosen to install and download ads to display on the user's system, thereby creating pop-up browser windows and when certain sites are visited by the user, it captures essential information and sends it over to the remote server. Such attacks are also known as Convert Channels.

3. **Trap Door:** The designer of a program or system might leave a hole in the software that only he is capable of using, the Trap Door works on similar principles. Trap Doors are quite difficult to detect as to analyze them,

one needs to go through the source code of all the components of the system. In other words, if we may have to define a trap door then it would be like, a trap door is actually a kind of a secret entry point into a running or static program that actually allows anyone to gain access to any system without going through the usual security access procedures.

4. **Logic Bomb:** A program that initiates a security attack only under a specific situation. To be very precise, a logic bomb is actually the most malicious program which is inserted intentionally into the computer system and that is triggered or functions when specific conditions have been met for it to work.
5. **Worm:** A computer worm is a type of malware that replicates itself and infects other computers while remaining active on affected systems. A computer worm replicates itself in order to infect machines that aren't already infested. It frequently accomplishes this by taking advantage of components of an operating system that are automatic and unnoticed by the user. Worms are frequently overlooked until their uncontrolled replication depletes system resources, slowing or stopping other activities.

Types of System Threats

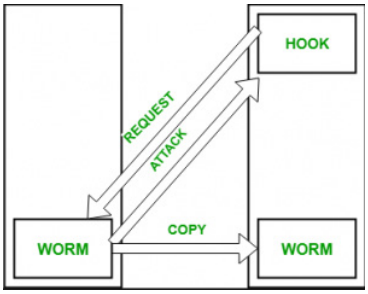
Aside from the program threats, various system threats are also endangering the security of our system:

1. Worm: An infection program that spreads through networks. Unlike a virus, they target mainly LANs. A computer affected by a worm attacks the target system and writes a small program "hook" on it. This hook is further used to copy the worm to the target computer. This process repeats recursively, and soon enough all the systems of the LAN are affected. It uses the spawn mechanism to duplicate itself. The worm spawns copies of itself, using up a majority of system resources and also locking out all other processes.

The basic functionality of the worm can be represented as:

2. Port Scanning: It is a means by which the cracker identifies the vulnerabilities of the system to attack. It is an automated

process that involves creating a TCP/IP connection to a specific port. To protect the identity of the attacker, port scanning attacks are launched from Zombie Systems that is systems that were previously independent systems that are also serving their owners while being used for such notorious purposes.



3. Denial of Service: Such attacks aren't aimed for the purpose of collecting information or destroying system files. Rather, they are used for disrupting the legitimate use of a system or facility.

These attacks are generally network-based. They fall into two categories:

- Attacks in this first category use so many system resources that no useful work can be performed.
- For example, downloading a file from a website that proceeds to use all available CPU time.
- Attacks in the second category involve disrupting the network of the facility. These attacks are a result of the abuse of some fundamental TCP/IP principles.

Security Measures Taken

To protect the system, Security measures can be taken at the following levels:

- **Physical:** The sites containing computer systems must be physically secured against armed and malicious intruders. The workstations must be carefully protected.
- **Human:** Only appropriate users must have the authorization to access the system. Phishing (collecting confidential information) and Dumpster Diving (collecting basic information so as to gain unauthorized access) must be avoided.

- **Operating system:** The system must protect itself from accidental or purposeful security breaches.
- **Networking System:** Almost all of the information is shared between different systems via a network. Intercepting these data could be just as harmful as breaking into a computer. Henceforth, Network should be properly secured against such attacks.

Usually, Anti Malware programs are used to periodically detect and remove such viruses and threats. Additionally, to protect the system from Network Threats, a Firewall is also be used.

Hash Functions in System Security

Hash Function is a function that has a huge role in making a System Secure as it converts normal data given to it as an irregular value of fixed length. We can imagine it to be a Shaker in our homes.

When we put data into this function it outputs an irregular value. The Irregular value it outputs is known as "Hash Value". Hash Values are simply numbers but are often written in Hexadecimal. Computers manage values as Binary. The hash value is also data and is often managed in Binary.

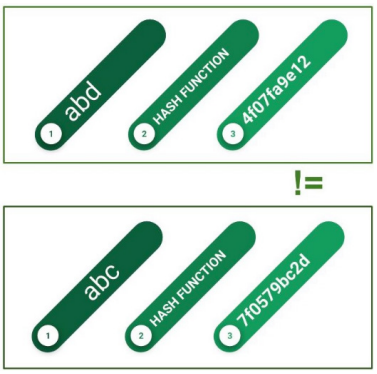


A hash function is basically performing some calculations in the computer. Data values that are its output are of fixed length. Length always varies according to the hash function. Value doesn't vary even if there is a large or small value.

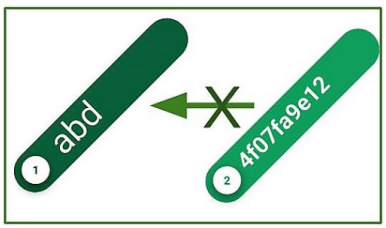


If given the same input, two hash functions will invariably produce the same output. Even if input data entered differs by

a single bit, huge change in their output values. Even if input data entered differs huge, there is a very minimal chance that the hash values produced will be identical. If they are equal it is known as "Hash Collision".



Converting Hash Codes to their original value is an impossible task to perform. This is the main difference between Encryption as Hash Function.



ASSURANCE OF SYSTEM SECURITY

Security engineering incorporates a number of cross-disciplinary skills, including cryptography, computer security, tamper-resistant hardware, applied psychology, supply chain management, and law. Security requirements differ greatly from one system to the next. System security often has many layers built on user authentication, transaction accountability, message secrecy, and fault tolerance. The challenges are protecting the right items rather than the wrong items and protecting the right items but not in the wrong way.

The purpose of system security is to help assure system value-delivery while undergoing adversity. The primary objective of System Security Engineering (SSE) is to minimize or contain

defense system vulnerabilities to known or postulated security threats and to ensure that developed systems protect against these threats. Engineering principles and practices are applied during all system development phases to identify and reduce these system vulnerabilities to the identified system threats.

The basic premise of SSE is recognition that an initial investment in "engineering out" security vulnerabilities and "designing-in" countermeasures is a long-term benefit and cost saving measure. Further, SSE provides a means to ensure adequate consideration of security requirements, and, when appropriate, that specific security-related designs are incorporated into the overall system design during the engineering development program. Security requirements include: physical; personnel; procedural; emission; transmission; cryptographic; communications; operations; and, computer security.

There may be some variation in the SSE process from program to program, due mainly to the level of design assurance—that is, ensuring that appropriate security controls have been implemented correctly as planned—required of the contractor. These assurance requirements are elicited early in the program (where they can be adequately planned), implemented, and verified in due course of the system development.

The System Security Engineering Management Plan (SSEMP) is a key document to develop for SSE. The SSEMP identifies the planned security tasks for the program and the organizations and individuals responsible for security aspects of the system. The goals of the SSEMP are to ensure that pertinent security issues are raised at the appropriate points in the program, to ensure adequate precautions are taken during design, implementation, test, and fielding, and to ensure that only an acceptable level of risk is incurred when the system is released for fielding.

The SSEMP forms the basis for an agreement with SSE representing the developer, the government program office, the certifier, the accreditor, and any additional organizations that have a stake in the security of the system. The SSEMP identifies the major tasks for certification & accreditation (C&A), document preparation, system evaluation, and engineering; identifies the responsible organizations for each task; and presents a schedule for the completion of those tasks.

SSE security planning and risk management planning includes task and event planning associated with establishing statements of work and detailed work plans as well as preparation and negotiation of SSE plans with project stakeholders. For each program, SSE provides the System Security Plan (SSP) or equivalent. An initial system security Concept of Operations (CONOPS) may also be developed. The SSP provides: the initial planning of the proposed SSE work scope; detailed descriptions of SSE activities performed throughout the system development life cycle; the operating conditions of the system; the security requirements; the initial SSE risk assessment (includes risks due to known system vulnerabilities and their potential impacts due to compromise and/or data loss); and, the expected verification approach and validation results.

These plans are submitted with the proposal and updated as required during engineering development. In the case where a formal C&A is contracted and implemented, these plans comply with the government's C&A process, certification responsibilities, and other agreement details, as appropriate. The C&A process is the documented agreement between the customer and contractor on the certification boundary. Upon agreement of the stakeholders, these plans guide SSE activities throughout the system development life cycle.

"As a matter of fact, yeah, they were foolproof. The problem is that you don't have to protect yourself against fools. You have to protect yourself against people like me."

—Jeffery Deaver

System Assurance

Engineering for System Assurance in NATO Programs, defines system assurance as:

…the justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle... This confidence is achieved by system assurance activities, which include a planned, systematic set of multi-disciplinary activities to achieve the acceptable measures of system assurance and manage the risk of exploitable

vulnerabilities. The NATO document is organized based on the life cycle processes in ISO/IEC 15288:2008 and provides process and technology guidance to improve system assurance.

One may consider a system with respect to what it is (structure), what it does (function), what it contains (contents), what it uses, (inputs, resources), or where it resides (environment). The focus of security (that which we safeguard) may be structure, state, function, functional exchange (input, output), raw materials (access, supply), fuel/energy to run the system, containing whole (system of systems), or the broader current order (ecosystem). System security focus includes but is not limited to harden, defend, protect, maximize use of renewable resources, minimize use of depletable resources, remain compatible with the current order, and remain viable and relevant.

Software Assurance

Since most modern systems derive a good portion of their functionality from software, software assurance becomes a primary consideration in systems assurance. The Committee on National Security Systems (CNSS) defines software assurance as a "level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle and that the software functions in the intended manner."

Goertzel point out that "the reason software assurance matters is that so many business activities and critical functions—from national defense to banking to healthcare to telecommunications to aviation to control of hazardous materials—depend on the on the correct, predictable operation of software."

Hardware Assurance

System hardware assurance is a set of system security engineering activities undertaken to quantify and increase the confidence that electronics function as intended and only as intended throughout their life cycle and to manage identified risks. See System Hardware Assurance for more information.

System Description

Robust security design explicitly rather than implicitly defines the protection goals. The Certified Information Systems Security Professional (CISSP) Common Body of Knowledge (CBK) partitions robust security into ten domains:

1. Information security governance and risk management addresses the framework, principles, policies, and standards that establish the criteria and then assess the effectiveness of information protection. Security risk management contains governance issues, organizational behavior, ethics, and security awareness training.
2. Access control is the procedures and mechanisms that enable system administrators to allow or restrict operation and content of a system. Access control policies determine what processes, resources, and operations users can invoke.
3. Cryptography can be defined as the principles and methods of disguising information to ensure its integrity, confidentiality, and authenticity during communications and while in storage. Type I devices are certified by the US National Security Agency (NSA) for classified information processing. Type 2 devices are certified by NSA for proprietary information processing. Type 3 devices are certified by NSA for general information processing. Type 4 devices are produced by industry or other nations without any formal certification.
4. Physical (environmental) security addresses the actual environment configuration, security procedures, countermeasures, and recovery strategies to protect the equipment and its location. These measures include separate processing facilities, restricted access into those facilities, and sweeps to detect eavesdropping devices.
5. Security architecture and design contains the concepts, processes, principles, and standards used to define, design, and implement secure applications, operating systems, networks, and equipment. The security architecture must integrate various levels of confidentiality, integrity, and availability to ensure effective operations and adherence to governance.

6. Business continuity and disaster recovery planning are the preparations and practices which ensure business survival given events, natural or man-made, which cause a major disruption in normal business operations. Processes and specific action plans must be selected to prudently protect business processes and to ensure timely restoration.
7. Telecommunications and network security are the transmission methods and security measures used to provide integrity, availability, and confidentiality of data during transfer over private and public communication networks.
8. Application development security involves the controls applied to application software in a centralized or distributed environment. Application software includes tools, operating systems, data warehouses, and knowledge systems.
9. Operations security is focused on providing system availability for end users while protecting data processing resources both in centralized data processing centers and in distributed client/server environments.
10. Legal, regulations, investigations, and compliance issues include the investigative measures to determine if an incident has occurred and the processes for responding to such incidents.

One response to the complexity and diversity of security needs and domains that contribute to system security is "defense in depth," a commonly applied architecture and design approach. Defense in depth implements multiple layers of defense and countermeasures, making maximum use of certified equipment in each layer to facilitate system accreditation.

Discipline Management

Discipline management includes the following fundamental security principles with informal descriptions. There are many formal definitions that take on different nuances depending on context; e.g., government-based, international standards, or particular legislation.

Principle	Description (Positive)	Description (Negative)
Confidentiality	Ensure only authorized disclosure	Safeguard against unauthorized disclosure
Integrity	Ensure only authorized modification	Safeguard against unauthorized modification
Availability	Ensure ready for use	Safeguard against denial of service
Possession	Ensure physical possession	Safeguard against loss or theft; minimize damage resulting from loss or theft
Utility	Ensure *fit for purpose*, or *ready for use*	Safeguard against loss of purpose
Authenticity	Ensure system conforms to reality or acts upon information that conforms to reality	Safeguard against deception
Privacy	Ensure the right to be unobserved and the right to be forgotten	Safeguard against unwanted observation and persistence of personal details (e.g., photos)
Non-Repudiation	Ensure non-deniability	Safeguard against anonymity in performing actions
Authorized Use	Ensure only authorized use of services	Safeguard against misuse of cost-incurring (resource expending) activity

Security Domains

There are many security domains, each of which may have distinct or overlapping disciplines. Some security domains for consideration:

- Orchestration Space guides workflow execution
- Workflow Space fulfills the organizational mission
- Organizational Efficacy is the capacity to fulfill mission
- Safeguards preserve organizational efficacy
- Desired Security Posture is an intentionally assumed position to define applicable safeguards
- Risk Posture is an intentionally assumed position to drive desired security posture
- Mission Risk Tolerance helps define risk posture
- Tolerance for workflow disruption drives mission risk tolerance
- Marketplace Dynamics (ecosystem dynamics) drive mission priorities
- Mission Priorities are non-static and influence mission risk tolerance

- Vulnerability Space contains weaknesses within orchestration and workflow
- Vulnerabilities influence risk posture
- Threat Space contains potential disruptions to workflow
- Compliance Space partially address the threat space
- Compliance drivers influence risk posture
- Continuous Monitoring provides snapshots of current security posture
- Gap Analysis provides difference between desired and current security posture
- Corrective Action provides gap closure within resource constraints
- Resource Constraints: people (skills, knowledge), budget, technology, etc.

A change in any one domain has cascading effects on the other domains. In this sense, they have causal loop or dynamic feedbacks resulting in infinite cycles. An emerging goal is to identify, understand, and act upon domain states and interactions within cyber-relevant time.

SECURITY SYSTEM DEVELOPMENT LIFE CYCLE

Security System Development Life Cycle (SecSDLC) is defined as the set of procedures that are executed in a sequence in the software development cycle (SDLC). It is designed such that it can help developers to create software and applications in a way that reduces the security risks at later stages significantly from the start. The Security System Development Life Cycle (SecSDLC) is similar to Software Development Life Cycle (SDLC), but they differ in terms of the activities that are carried out in each phase of the cycle. SecSDLC eliminates security vulnerabilities. Its process involves identification of certain threats and the risks they impose on a system as well as the needed implementation of security controls to counter, remove and manage the risks involved. Whereas, in the SDLC process, the focus is mainly on the designs and implementations of an information system. Phases involved in SecSDLC are:

- System Investigation: This process is started by the officials/directives working at the top level management in the organization. The objectives and goals of the project are considered priorly in order to execute this process. An Information Security Policy is defined which contains the descriptions of security applications and programs installed along with their implementations in organization's system.
- System Analysis: In this phase, detailed document analysis of the documents from the System Investigation phase are done. Already existing security policies, applications and software are analyzed in order to check for different flaws and vulnerabilities in the system. Upcoming threat possibilities are also analyzed. Risk management comes under this process only.
- Logical Design: The Logical Design phase deals with the development of tools and following blueprints that are involved in various information security policies, their applications and software. Backup and recovery policies are also drafted in order to prevent future losses. In case of any disaster, the steps to take in business are also planned. The decision to outsource the company project is decided in this phase. It is analyzed whether the project can be completed in the company itself or it needs to be sent to another company for the specific task.
- Physical Design: The technical teams acquire the tools and blueprints needed for the implementation of the software and application of the system security. During this phase, different solutions are investigated for any unforeseen issues which may be encountered in the future. They are analyzed and written down in order to cover most of the vulnerabilities that were missed during the analysis phase.
- Implementation: The solution decided in earlier phases is made final whether the project is in-house or outsourced. The proper documentation is provided of the product in order to meet the requirements specified for the project to be met. Implementation and integration process of the project are carried out with the help of various teams aggressively testing whether the product meets the system requirements specified in the system documentation.

- Maintenance: After the implementation of the security program it must be ensured that it is functioning properly and is managed accordingly. The security program must be kept up to date accordingly in order to counter new threats that can be left unseen at the time of design.

Security System Handle Cyber Security Threats and Vulnerabilities

Stories of organizations paralyzed by cyber security threats and vulnerabilities are at their peak. According to a report published by Symantec Corp, India is one of the top five countries that have become the victim of cyber-crime. Nowadays, modern technologies such as cloud computing, IoT, cognitive computing, etc. are categorized as the critical assets of any organization. With the increase in the use of advanced technology and interconnected applications, there is a rapid spike not only in businesses but also in threats and vulnerabilities as well.

In this feature, we focus on security threats, challenges faced by defenders to protect the organization from emerging threats, and how the security system should evolve to overcome the day-to-day critical security challenges. Let's dig deep into the below topics.

- Cyber Threats
- Security Challenges
- How must security system evolve?

Cyber Threats

Technology is transforming as never before. With the advancement in technology, organizations started to experience consistent business growth at a faster pace. They were able to interconnect people, robots, gadgets, contents, and more in an intelligent way that drives more business. But, at the same time, this advancement in technology opens up a center of attention for cybercrimes, targeted attacks, and corporate espionage.

A cyber threat is a malicious attack that gains unauthorized access to a system or network and thereby damages or steals confidential data. Let's go a bit further to understand the different types of cyber threats.

- Ransomware
- DDoS Attack
- Threats originated within an organization
- Data Breaches
- Advanced Persistence Threat (APT)

1. **Ransomware.** Ransomware is malware that encrypts the system data and demands payment for access permission. It prevents you from accessing the system, and it can also destroy the data if the payment is not made on time. Based on a survey conducted by Sophos, over 51% of organizations were attacked by ransomware during the year 2019. Ransomware is also available as Ransomware-as-a-service (RaaS) over the dark web marketplace. WannaCry, NotPetya, SimpleLocker, TeslaCrypt, CryptoLocker, and PC Cyborg are some of the Ransomware.
2. **Distributed denial-of-service (DDoS) attacks.** The DDoS attack is a malicious attack that increases the traffic of a server with overwhelming random traffic. In DDoS, the server is targeted by different independent networks with the help of botnet, and this is how it differs from DoS. One of the famous and highest reported impacts was against Dyn, a US-based DNS service provider. The DDoS attack against Dyn has affected many websites including Twitter, GitHub, Amazon, Netflix, and more.
3. **Threats originated within an organization.** Internal threats are malicious threats that come from people within the organization who have access to confidential information. It can be employees, former employees, partners, associates, and so on. Using these threats, the attacker can bypass security in a legalized way.
4. **Data Breaches.** Data Breaches can be defined as the leakage of confidential information that includes sensitive corporate documents, technical blueprints, trade secrets, and more. It can lead to financial loss, brand reputation loss, customer trust loss, and so on. Some of the main reasons for Data Breaches are malicious attacks, a weak security system, and human errors. As per the Verizon Data Breach report, over 88% of data breaches involve human errors.

5. **Advanced persistence threat (APT).** APT is an advanced attack threat. It uses multiple phases to break the network and thereby allow unauthorized people to stay in the organization network. APT can happen through spear-phishing or inside threats. This threat is hard to detect and can retrieve valuable information over a sustained period.

Security Challenges

Be ahead of your adversaries; if you fail to do so, soon you will become a victim. Let's discuss some of the security challenges.

1. **Slow security adaption.** One of the issues related to the cybersecurity system is that cybersecurity solutions are not advancing at an expected rate. In today's digital era, cloud technologies and other solutions are evolving at a faster pace, and the traditional network architecture has been deputized with simple and flat architecture. But, concerns cybersecurity solutions, many organizations still use traditional zone-based security solutions to prevent threats.
2. **Human Errors.** Human errors such as system misconfiguration, insufficient patch management, etc. are common in the majority of organizations. These errors resulted in numerous cyber attacks. According to the IBM security threat, over 95% of cyber-attacks are due to human errors.
3. **Third-party vendor security risk.** In today's world, everything is connected. Organizations let third parties store their information for better business operations. But, if they don't choose a trustworthy third-party vendor, then the organization is at risk. Here, an attacker can bypass the security system by initiating supply chain attacks.

How must security system evolve?

We will discuss advanced security strategies to defend against threats and strengthen the cybersecurity system. Let's take a moment to understand some of the best security practices. They are as follows:

1. **Threat Prevention Strategies.** Security researchers are researching and innovating effective solutions to prevent threats. They work around the clock aiming at zero-day vulnerabilities and are also actively involved in conducting awareness programs. Threat prevention strategies are mainly categorized into four main sections. They are as follows:

- **Reduce the attack surface**: Continuous process of vulnerability scanning practice helps to determine top risk applications, security gaps in the network, risky users and processes, and more. Relative Attack Surface Quotient (RASQ) is one such method that can keep track of every change to the attack surface.
- **Complete visibility**: End-point protection is another factor to take on board. In most cases, end-point security can be compromised by using SMB-based vulnerabilities. So, it is important to separate normal SMB behavior from strange SMB behaviors, and this categorization can be done by providing complete visibility. It is the key that can identify malicious behavior.
- **Prevent known threats**: Firewalls and anti-virus software are necessary to prevent known threats. It is the first step towards defending networks and endpoints.
- **Prevent unknown threats**: Advanced and unknown threats are evolving as never before. As a result, it is more challenging to achieve a 100% threat protection. To deal with such threats, organizations have to adopt new techniques such as dynamic and behavioral analysis, deep learning techniques, and attacker techniques, tactics, and procedures (TTPs) analysis.

2. **Zero-trust approach.** The Zero-trust approach strategy is the continuous verification of all data and assets. It helps to detect the attackers who exfiltrate sensitive information through lateral movements. Let's take a moment to understand the process of the Zero-trust approach.

- **Identify and classify sensitive data**: It is necessary to identify and classify sensitive data for data protection.
- **Map the data flow**: You have to understand the application flow across the network by collaborating

with the network team, application team, and security architect.

- **Architect the network**: Architect the network by identifying the physical and virtual configurations. It includes the communication flow between multiple networks and external data accessing procedures.
- **Create the policy base**: While creating a policy base, you should include an efficient access control mechanism, information about user identity, application behavior, and so on.
- **Continuous monitoring**: In this process, continuous monitoring of both internal and external traffic is performed. Here the network and application logs are checked frequently on a real-time basis.

3. **Assume breach approach.** The reality is that none of the security prevention technology can ensure you 100% protection against threats. As the days' pass, advance threats manage to bypass the security system. Here comes the importance of the assume breach approach. It is a way of testing the incident response force of an organization. It provides various security solutions and services. They are as follows:

- **Red-team exercise**: It is an advanced version of penetration testing, where a team of highly professional security experts not only finds vulnerabilities but also tests an organization's threat detection and response capabilities. It opens up a way for immediate as well as long-term security posture improvement.
- **Continuous monitoring**: Continuous monitoring is necessary to detect threats at an early stage, and it can be achieved by providing real-time visibility of users as well as network endpoints. An active security monitoring system can ensure cyber hygiene and compliance by actively monitoring the network, application, and user activities. Some of the common tools used for monitoring are security information and event management (SIEM) tool and endpoint detection and response (EDR) tool.

"Very smart people are often tricked by hackers, by phishing. I don't exclude myself from that. It's about being smarter than a hacker. Not about being smart."

—Harper Reed

Attackers are constantly looking for vulnerabilities to gain unauthorized access to an application or a network. With the advancement in technology, security threats and models are evolving at a faster pace. These security threats and models can stay hidden, self-destruct, by-pass traditional security systems, and so on. As a result, it can cost damage to your confidential information, corporate eavesdropping, and more. Hence, it is necessary to tailor a vulnerability management system that defends against threats and strengthens the cybersecurity system.

Port Security in Computer Network

Attackers' task is comparatively very easy when they can enter the network they want to attack. Ethernet LANs are very much vulnerable to attack as the switch ports are open to use by default. Various attacks such as Dos attack at layer 2, address spoofing can take place. If the administrator has control over the network then obviously the network is safe. To take total control over the switch ports, the user can use a feature called port-security. If somehow prevent an unauthorized user to use these ports, then the security will increase up to a great extent at layer 2.

Users can secure a port in two steps:

- Limiting the number of MAC addresses to a single switch port, i.e if more than the limit, Mac addresses are learned from a single port then appropriate action will be taken.
- If unauthorized access is observed, the traffic should be discarded by using any of the options, or more appropriately, the user should generate a log message so that unauthorized access can be easily observed.

Port security – Switches learn MAC addresses when the frame is forwarded through a switch port. By using port security, users can limit the number of MAC addresses that can be learned to a port, set static MAC addresses, and set penalties for that

port if it is used by an unauthorized user. Users can either use restrict, shut down or protect port-security commands.

Let's discuss these violation modes:

- **protect –** This mode drops the packets with unknown source mac addresses until you remove enough secure mac addresses to drop below the maximum value.
- **restrict –** This mode performs the same function as protecting, i.e drops packets until enough secure mac addresses are removed to drop below the maximum value. In addition to this, it will generate a log message, increment the counter value, and will also send an SNMP trap.
- **shut down –** This mode is mostly preferred as compared to other modes as it shut down the port immediately if unauthorized access is done. It will also generate a log, increment counter value, and send an SNMP trap. This port will remain in a shutdown state until the administrator will perform the "no shutdown" command.
- **Sticky –** This is not a violation mode. By using the sticky command, the user provides static Mac address security without typing the absolute Mac address. For example, if user provides a maximum limit of 2 then the first 2 Mac addresses learned on that port will be placed in the running configuration. After the 2nd learned Mac address, if the 3rd user wants to access then the appropriate action will be taken according to the violation mode applied.

The port security will work on access port only i.e to enable port security, the user first has to make it an access port.

Configuration – Applying port-security on fa0/1 interface of switch .first, convert the port to an access port and will enable port-security.

```
S1(config)#int fa0/1
S1(config-if)#switchport mode access
S1(config-if)#switchport port-security
```

Use sticky command so that it will learn the Mac address dynamically and will provide the limit and the appropriate action that should be taken.

```
S1(config-if)#switchport port-security mac-address sticky
S1(config-if)#switchport port-security
maximum 2
S1(config-if)#switchport port-security violation shutdown
```

If the user wants to provide a static entry, then configure that by starting its Mac address.

```
S1(config-if)#switchport port-security
S1(config-if)#switchport port-security violation shutdown
S1(config-if)#switchport port-security mac-address aa.bb.
cc.dd.ee.ff
```

Security in Distributed System

In a distributed system, one must consider many possible security risks. To mitigate these risks there are a number of strategies that can be employed:

- Encryption algorithms that protect data in transit and at rest.
- Firewalls that limit access to specific ports/cables.
- Intrusion detection systems that identify anomalous behavior among network services.
- Intrusion prevention systems (IPS) respond to attempted intrusions by initiating defensive actions like blocking suspicious IP addresses or taking down compromised servers.

These measures may be insufficient, to identify attacks at the network level without help from other sources. We cannot only prevent malicious actors from gaining access to our machines from other machines in the same firewall but can also monitor our own actions.

Reckless data sharing can significantly increase exposure to both the threats themselves and the costs entailed in defending against them.

Goals of Distributed System Security: Security in a distributed system poses unique challenges that need to be considered when designing and implementing systems. A compromised computer or network may not be the only location

where data is at risk; other systems or segments may also become infected with malicious code. Because these types of threats can occur anywhere, even across distances in networks with few connections between them, new research has been produced to help determine how well distributed security architectures are actually performing.

In the past, security was typically handled on an end-to-end basis. All the work involved in ensuring safety occurred "within" a single system and was controlled by one or two administrators. The rise of distributed systems has created a new ecosystem that brings with it unique challenges to security.

Distributed systems are made up of multiple nodes working together to achieve a common goal, these nodes are usually called peers.

Security Requirements and Attacks Related to Distributed Systems: A distributed system is composed of many independent units, each designed to run its own tasks without communicating with the rest of them except through messaging service. This means that a single point of failure can render a system completely incapable without any warning since there is no single point that can perform all necessary operations.

Attacks related to distributed systems are an area of active research. There were two main schools of thought, those who believed that network worms could be stopped by employing firewalls and those who did not.

A firewall might do nothing about worms and their ability to spread across various types of networks, especially wireless networks and the Internet. This was because although firewalls were able to stop intruders from gaining access through the firewall, they were unable to stop a worm from self-replicating.

To summarize, there are numerous attacks that can be used against a network worm that has to do with breaking functionality and altering data, or simply deleting it.

"If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology."

—Bruce Schneier

INFORMATION SYSTEM SECURITY

Information System Security or INFOSEC refers to the process of providing protection to the computers, networks and the associated data. With the advent of technology, the more the information is stored over wide networks, the more crucial it gets to protect it from the unauthorized which might misuse the same. Every organization has the data sets that contain confidential information about its activities.

The major reason of providing security to the information systems is not just one fold but 3 fold:

1. Confidentiality
2. Integrity
3. Availability

Together, these tiers form the CIA triangle that happened to be known as the foremost necessity of securing the information system. These three levels justify the **principle of information system security**.

Let us go through the same one by one:

1. **Confidentiality:** The main essence of this feature lies in the fact that only the authorized personnel should be allowed the access to the data and system. The unauthorized individuals must be kept away from the information. This is ensured by checking the authorization of every individual who tries to access the database. For e.g. An organization's administration must not be allowed to access the private information of the employees.
2. **Integrity:** Integrity is ensured when the presented data is untouched or rather, is not altered by any unauthorized power. The information thus can be referred with the eyes closed. The integrity of the information can be altered in either unintentional or intentional ways. Intentionally, information can be passed through malicious content by any individual. Rather, unintentionally, any authorized individual might himself hamper the information for example, he might delete any specific important part of information.
3. **Availability:** This feature means that the information can be accessed and modified by any authorized personnel

within a given time frame. The point here to be noted is that the accessibility of the information is limited. The time frame within which it can be accessed is different for every organization.

Balancing Information Security and Access

It is the sole purpose of the organization to protect the interests of the users and to provide them with appropriate amount of information whenever necessary. Also, at the same time, it is necessary to provide adequate security to the information so that not anyone can access it. The need for maintaining the perfect balance of information security and accessibility arises from the fact that information security can never be absolute.

It would be harmful to provide free access to a piece of information and it would be hard to restrict any accessibility. So, one needs to make sure that the exact required balance is maintained so that both the users and the security professionals are happy.

Tools of Information Security

There are various tools which are or which can be used by various organizations in order to ensure the maximum information system security. These tools however, do not guarantee the absolute security, but as stated above, helps in forming the crucial balance of information access and security.

Let us study these tools one by one:

1. **Authentication:** This is the foremost important tool that needs to be kept in mind before starting the crucial process of ensuring security. The process of authentication is when the system identifies someone with one or more than one factors. These factors must be unique for most of the users. For example, ID and password combinations, face recognition, thumb impression etc. These factors cannot always be trusted as one could lose them or it might be accessed by any outsider. For these circumstances, one can use multi factor authorization which is done by combining any two or more of the above factors.

2. **Access Control:** After ensuring that the right individual gets the access to information, one has to make sure that only the appropriate information reaches him or her. By using the tool of access control, the system judges that which user must be able to re4ad or write or modify certain piece of information. For this it generally maintains a list of all the users. One could find two type of lists:
   - **Access Control List (ACL)** – This is just the list of individuals who are eligible to access the information
   - **Role- Based access Control List (RBAC)** – This list comprises of the names of authorized personnel and their respective actions they are authorized to perform over the information.
3. **Encryption:** Sometimes the information is transmitted over the internet so the risk of anyone accessing it increases and now the tools have to be strong to avoid it. In this scenario, the information can be easily accessed and modified by anyone. To avoid this, a new tool is put to work, Encryption. Using encryption, one can put the confidential information into bits of unreadable characters that are difficult to decrypt and only the authorized receivers of the information can read it easily.

Information System and Security

The Information System is an integrated set of the component for collecting, storing, processing and communicating information. Business firm and other organization on the information system to manage their operation in the marketplace supply service and augment personals lives.

There are two types of the information system which are given below: General purpose information system, and specialized information system.

1. **General purpose information system:** There are some general types of information system .for example a database management system (DBMS) is a combination of software and data that makes it possible to organize and analyze data. Database management system software

is typically not designed to work with a specific organization or a specific type of analysis.

2. **Specialized information system:** In contrast, there is a number of a specialized information system that has been specifically designed to support a particular process within an organization or to carry out very specific analysis task.

Security Challenge

The number of smart phone devices capable of offering internet technology and experience rivaling desktop computer standards is growing at a fast pace. Security and privacy concern for mobile devices rival or go beyond similar concern for a laptop computer as mobile device are even more mobile by nature and are less likely to be managed by an organization.

Ensure Security: In order to ensure security, it is necessary to provide at least the following services, which are given below.

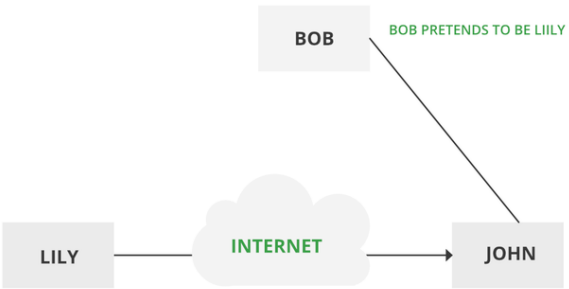
- **Authorization:** It is act of determining whether an (authenticate) entity has the right to execute action.
- **Audit:** An auditing service providing a history of action that can be used to determine what (if anything) went wrong and what caused it to go wrong.
- **Physical authentication:** Some firm of authentication such as an object (a key or a smart card) or a personal characteristic like a fingerprint, retinal pattern, and hand geometry.
- **Data Confidentiality:** It protects against disclosure of any data while in transit and is provided by encryption of data.

Active and Passive attacks in Information Security

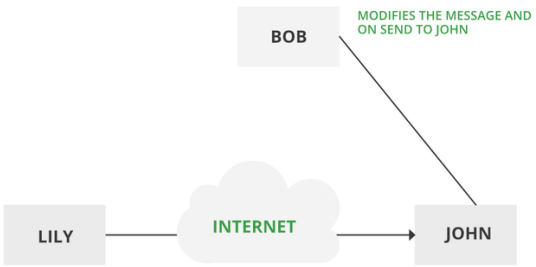
Active attacks: An Active attack attempts to alter system resources or affect their operations. Active attacks involve some modification of the data stream or the creation of false statements. Types of active attacks are as follows:

Masquerade – A masquerade attack takes place when one entity pretends to be a different entity. A Masquerade

attack involves one of the other forms of active attacks. If an authorization procedure isn't always absolutely protected, it is able to grow to be extraordinarily liable to a masquerade assault. Masquerade assaults may be performed using the stolen passwords and logins, with the aid of using finding gaps in programs, or with the aid of using locating a manner across the authentication process.



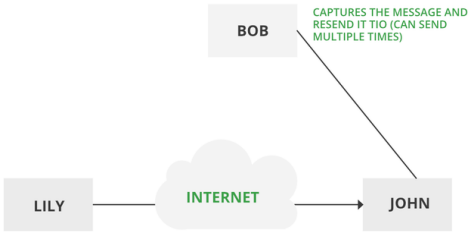
Modification of messages –It means that some portion of a message is altered or that message is delayed or reordered to produce an unauthorized effect. Modification is an attack on the integrity of the original data. It basically means that unauthorized parties not only gain access to data but also spoof the data by triggering denial-of-service attacks, such as altering transmitted data packets or flooding the network with fake data. Manufacturing is an attack on authentication. For example, a message meaning "Allow JOHN to read confidential file X" is modified as "Allow Smith to read confidential file X".



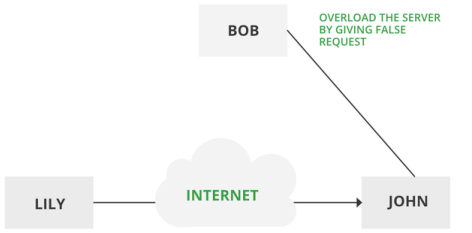
Repudiation – This attack occurs when the network is not completely secured or the login control has been tampered with. With this attack, the author's information can be changed by actions of a malicious user in order to save false data in log

files, up to the general manipulation of data on behalf of others, similar to the spoofing of e-mail messages.

Replay – It involves the passive capture of a message and its subsequent transmission to produce an authorized effect. In this attack, the basic aim of the attacker is to save a copy of the data originally present on that particular network and later on use this data for personal uses. Once the data is corrupted or leaked it is insecure and unsafe for the users.



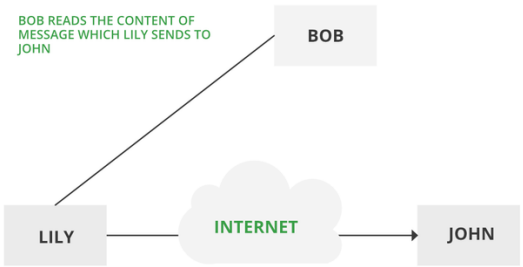
Denial of Service – It prevents the normal use of communication facilities. This attack may have a specific target. For example, an entity may suppress all messages directed to a particular destination. Another form of service denial is the disruption of an entire network either by disabling the network or by overloading it with messages so as to degrade performance.



Passive attacks: A Passive attack attempts to learn or make use of information from the system but does not affect system resources. Passive Attacks are in the nature of eavesdropping on or monitoring transmission. The goal of the opponent is to obtain information that is being transmitted. Types of Passive attacks are as follows:

- The release of message content – Telephonic conversation, an electronic mail message, or a transferred file may contain sensitive or confidential information. We would

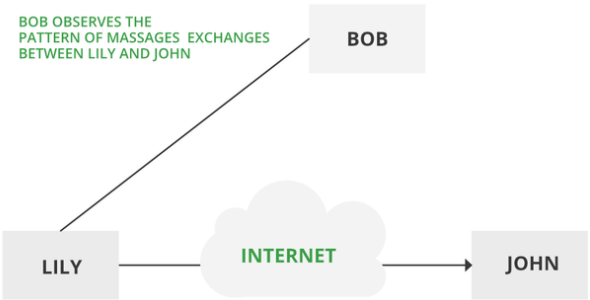
like to prevent an opponent from learning the contents of these transmissions.



- **Traffic analysis –** Suppose that we had a way of masking (encryption) information, so that the attacker even if captured the message could not extract any information from the message.

  The opponent could determine the location and identity of communicating host and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

  The most useful protection against traffic analysis is encryption of SIP traffic. To do this, an attacker would have to access the SIP proxy (or its call log) to determine who made the call.



"Security leaders are under a lot of pressure to show quick wins while knowing full well that everything they do will be heavily scrutinized and challenged, and ultimately, they will pay the price for things that are not under their control.

—Yaron Levi

Difference between Information Security and Network Security

Information Security is the measures taken to protect the information from unauthorized access and use. It provides confidentiality, integrity, and availability. It is the superset that contains cyber security and network security. It is necessary for any organization or firm that works on a large scale.

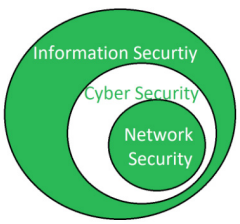
Examples and inclusion of Information Security are as follows:

- Procedural Controls
- Access Controls
- Technical Controls
- Compliance Controls

Network Security: Network Security is the measures taken by any enterprise or organization to secure its computer network and data using both hardware and software systems. This aims at securing the confidentiality and accessibility of the data and network. Every company or organization that handles a large amount of data, has a degree of solutions against many cyber threats.

Examples and inclusion of Network Security are as follows:

- Firewall
- Network Segmentation
- Remote Access VPN
- Email Security
- Intrusion Prevention Systems (IPS)
- Sandboxing
- Hyper-scale Network Security.
- Data Loss Prevention (DLP)



Difference between Information Security and Network Security:

Parameters	Information Security	Network Security
Data	It protects information from unauthorized users, access, and data modification.	It protects the data flowing over the network.
Part of	It is a superset of cyber security and network security.	It is a subset of cyber security.
Protection	Information security is for information irrespective of the realm.	It protects anything in the network realm.
Attack	It deals with the protection of data from any form of threat.	It deals with the protection from DOS attacks.
Scope	It strikes against unauthorized access, disclosure modification, and disruption.	Network Security strikes against trojans.
Usage	It provides confidentiality, integrity, and availability.	It provides security over the network only.
Ensures	Information security ensures to the protection of transit and stationary data.	Network security ensures to protect the transit data only.
Deals with	It deals with information assets and integrity, confidentiality, and availability.	It secures the data traveling across the network by terminals.

CONCLUSION

Computer security attempts to ensure the confidentiality, integrity, and availability of computing systems and their components. Three principal parts of a computing system are subject to attacks: hardware, software, and data. These three, and the communications among them, are susceptible to computer security vulnerabilities. In turn, those people and systems interested in compromising a system can devise attacks that exploit the vulnerabilities.

In this chapter we have explained the following computer security concepts:

- Security situations arise in many everyday activities, although sometimes it can be difficult to distinguish between a security attack and an ordinary human or technological breakdown. Alas, clever attackers realize this confusion, so they may make their attack seem like a simple, random failure.
- A threat is an incident that could cause harm. A vulnerability is a weakness through which harm could occur. These two problems combine: Either without the other causes no harm, but a threat exercising a vulnerability means damage. To control such a situation, we can either block or diminish the threat, or close the vulnerability (or both).

- Seldom can we achieve perfect security: no viable threats and no exercisable vulnerabilities. Sometimes we fail to recognize a threat, or other times we may be unable or unwilling to close a vulnerability. Incomplete security is not a bad situation; rather, it demonstrates a balancing act: Control certain threats and vulnerabilities, apply countermeasures that are reasonable, and accept the risk of harm from uncountered cases.
- An attacker needs three things: method—the skill and knowledge to perform a successful attack; opportunity—time and access by which to attack; and motive—a reason to want to attack. Alas, none of these three is in short supply, which means attacks are inevitable.

KEY TERMS

Data security	Cyber-attacks	Malware attack
Operating-system resources	Convert Channels	System resources
Intrusion prevention system	Information systems	Information security

DISCUSSION QUESTIONS

1. What are the security system goal?
2. Define the various types of program threats.
3. Elaborate the hash functions in system security.
4. Write the useful security domains.
5. What is the role of port security in computer network?
6. How to balancing information security and access?
7. What are the relationship between information system and security?
8. How security system should evolve to handle cyber security threats and vulnerabilities?

REFERENCES

1. Baldwin, K., J. Miller, P. Popick, and J. Goodnight. 2012. The United States Department of Defense Revitalization of System Security Engineering Through Program Protection. Proceedings of the 2012 IEEE Systems Conference, 19-22 March 2012, Vancouver, BC, Canada. Accessed 28 August 2012 at http://www.acq.osd.mil/se/docs/IEEE-SSE-Paper-02152012-Bkmarks.pdf.
2. CNSS. 2010. National Information Assurance Glossary", Committee on National Security Systems Instruction (CNSSI) no. 4009". Fort Meade, MD, USA: The Committee on National Security Systems.
3. DHS. 2010. Build Security In. Washington, DC, USA: US Department of Homeland Security (DHS). Accessed September 11, 2011. Available: https://buildsecurityin.us-cert.gov.
4. DODI5200.44, United States Department of Defense, Protection of Mission Critical Functions to Achieve Trusted Systems and Networks, Department of Defense Instruction Number 5200.44, November 2012, Accessed 3 November 2014 at Defense Technical Information Center http://www.dtic.mil/whs/directives/corres/pdf/520044p.pdf.
5. Goertzel, K., et al. 2008. Enhancing the Development Life Cycle to Produce Secure Software: A Reference Guidebook on Software Assurance. Washington, DC, USA: Data and Analysis Center for Software (DACS)/US Department of Homeland Security (DHS).
6. NATO. 2010. Engineering for System Assurance in NATO programs. Washington, DC, USA: NATO Standardization Agency. DoD 5220.22M-NISPOM-NATO-AEP-67.
7. Tipton, H.F. (ed.). 2006. Official (ISC)2 guide to the CISSP CBK, 1st ed. Boston, MA, USA: Auerbach Publications.

INDEX

A

Agglomeration 170
Analysis Model 90, 92, 93
Analytic capability 170, 171, 173, 176, 179

B

Business analytics 262
Business architecture 170
Business area analysis (BAA) 54
Business Capabilities Model 170
Business intelligence (BI) 171, 188, 262, 291
Business intelligence model (BIM) 263
Business process modeling 120, 121, 124, 137

C

Capability assessment 171, 174, 176, 180, 194
Capability Maturity Model (CMM) 170, 171
Colored Petri nets 129
Computer network 296, 328, 330
Computer system 3, 4, 5, 6, 8, 9, 297, 300
Computer technology 201
Confidentiality 297, 307, 308, 328, 329
Content awareness 225, 258
Convert Channel 299, 330
Cyber criminal 296

D

Data-entry operator 230, 248
Data flow diagrams (DFD) 125
Data mining 262, 270, 271, 284, 287, 293
Data modeling 95
Data object description 91
Data visualization 262, 284
Data warehouse 283
Decision tree 202, 218
Delivery system 3
Development process 48
Distributed denial of service (DDoS) 296
Dumpster Diving 301

E

Electrical engineer 48
Entity-relationship diagram 93, 95

F

Flowchart 124, 167

G

Gantt chart 127, 128, 167
Graphical rendering 202
Graphical representation 120, 121, 129
Graphic container 122

H

Hash Function 302, 303

I

Information engineering (IE) 52
Information strategy planning (ISP) 54, 56
Interface structure diagram (ISD) 226, 229

J

Joint Application Development (JAD) 204

K

Key performance indicators (KPIs) 284

M

Macintosh 223, 258
Malicious intrusion 296
Mathematical representation 129
Modular strategy 200

N

Navigation design 221, 222, 232, 258

O

Object oriented analysis (OOA) 198, 206
Object-Oriented Development 198
Object-Oriented Programming (OOP) 198
Online analytical processing (OLAP) 283, 294
Open Group Architecture Framework 170
Operational efficiency 2
Opportunity analysis 261, 263, 264, 265, 266, 267

P

Polymorphism 198, 210, 211, 218
Programming language 51
Prototype 217

R

Return on investment (ROI) 263

S

Self-assessment 175
Sesign modeling 90
Software development 198, 218
Software engineering 50, 55, 65, 70, 73, 76, 80
Software Engineering Institute (SEI) 171
Software life cycle 200
Software transform 90
Spyware 299
System analysis and design 2
Systems development 3
System security 296, 303, 306
Systems engineering 48

T

Technical representation 90
Traditional programming 198

U

Unified Modelling Language (UML) 130
User experience (UX) 222
User interface (UI) 222

V

Visual experience 222
Visualization 123
Visual language 222

W

Web site 225, 254
Workflow 120, 130, 131, 132, 133, 134, 144, 147

Z

Zombie System 301

Level: Beginner to Advanced
Subject: Computer and Information Science

Systems Analysis and Design

3rd Edition

The key to business success is the ability to collect, organize and interpret information. Systems analysis and design is a proven methodology that helps both large and small organizations realize the full benefits of leveraging information. As a systems analyst, the person in the organization most involved with systems analysis and design, you will enjoy a rich career path that will enhance both your computing and interpersonal skills. Organizations are complex systems made up of interconnected and interlocking subsystems. Changes in one part of the system have both expected and unanticipated consequences in other parts of the system. Systems are created to solve problems. You can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design (SAD) mainly deals with the activities of software development. System approval is a way of thinking about the analysis and design of computerized applications. It provides a framework for visualizing the organizational and environmental factors affecting a system. When a computer is introduced into an organization, various functions and malfunctions affect both the user and the organization. System design is the process of planning a new business system or a system that will replace or complement an existing system. But before this planning can be done, we must thoroughly understand the legacy system and determine how computers can best be used to make its operations more effective. In this dynamic world, the subject System Analysis and Design (SAD) mainly deals with the activities of software development. Systems analysts solve business problems by analyzing information systems requirements and designing such systems using analysis and design techniques.
The text in this edition is revised and restructured and new topics are added dealing with the concepts, skills, methodologies, techniques, tools, and perspectives essential for systems analysts.

ISBN 978-1-98468-218-5

BIBLIOTEX
Digital Library

www.3ge-learning.com
email: info@3ge-learning.com