Data Science Joint Education Program

INFO 200
Information Systems Analysis

Chapter #1

Systems, Roles, and Development Methodologies

Keywords and Phrases

agent of change	open source communities
agile approach	open source software (OSS)
agile methods	planning game
Ajax	planning phase
bespoke software	productionizing phase
computer-assisted software engineering (CASE)	prototyping
CASE tools	Scrum
exploration phase	security
human–computer interaction (HCI)	systems analysis and design
iterations to the first release phase	systems analyst
Linux kernel	systems consultant
maintenance phase	systems development life cycle (SDLC)
object-oriented systems analysis and design	Unified Modelling Language (UML)

Overview

- Chapter #1 provides:
  - An understanding of the motivation for and the need for systems analysis and design in organizations
  - An overview of the many roles a systems analysis may be required to occupy
  - A realization of the nature and requirements of the role(s) of a systems analysist
  - An understanding of the fundamentals of three principal development methodologies:
    - The systems development life cycle (SDLC)
    - The agile approach (including Scrum)
    - Object-oriented systems analysis and design

Major Topics Introduced

- The fundamentals of different kinds of information systems
- The roles of systems analysts
- Phases in the systems development life cycle as they relate to *Human-Computer Interaction* (HCI) factors
- *Computer-assisted software engineering* (CASE) tools
- Developing *Open Source Software* (OSS)
    - Also termed *Free Open Source Software* (FOSS)
    - However: open source software is governed by license conditions which can be variable:
        - OSS can be free for all users
        - OSS may be restricted to personal use with license fees payable for use by organizations

Information as a Resource

Information and Organizations

- Organizations have long recognized the importance of managing key resources such as people and raw materials.
- We are in the *information age* where:
  - Decision makers understand that information is not just a by-product of conducting business
  - Information is a key resource to organizations
  - Information fuels business and can be the critical factor in determining the success or failure of a business
  - Information is power (and the lack of information is a threat and a weakness)
  - Information has now moved to its rightful place as a key resource

Information Management

- To maximize the usefulness of information a business must:
  - Manage data and information effectively and correctly (just as it manages other resources)
- Managers must understand that costs are associated with:
  - Production
  - Distribution
  - Security
  - Storage (*and importantly*)
  - Retrieval of all information
  - Cloud-based approaches, methods, and systems
- It is important to note that data and information can be in either traditional formats of in digital (computerized) formats
  - Managing computer-generated information differs from handling manually produced data

Information

- Data and information (and information systems) is all around us:
  - It is not free (although to many users it may appear free)
  - The use of data and information has a strategic use for positioning a business competitively should not be taken for granted
- The ready availability of networked computers, along with access to the Internet and the Web:
  - Has created an information explosion throughout society in general and business in particular
- Moreover: there is a greater quantity of computer information to administer
  - The cost of organizing and maintaining it can increase at alarming rates
  - Users often treat it less sceptically than information obtained in different ways

The Need for Systems Analysis and Design

Why?

- Why is systems analysis and design important?
- Building and installing a system without proper planning:
  - Leads to great user dissatisfaction
  - Frequently causes systems to fall into disuse
  - Systems may not be accepted or used by users within an organization
- It lends structure to the analysis and design of information systems
- The process is:
  - A series of processes systematically undertaken to improve a business through the use of computerized information systems

Systems Analysis and Design

- Systems analysis and design (performed by systems analysts) attempts to understand:
  - What people need to systematically analyse data input or data flows
  - The processing (or) transformation of data
  - The storage (temporary and persistent) storage of data
  - The recording and output information in the context of a particular organization or enterprise
- The systems analysts seeks to:
  - Identify and solve the right problems by performing a thorough analysis of the client's system
  - Systems analysis and design is used to:
    - Analyse, design, and implement improvements to the computerized information systems that support users and the functions of businesses

Security

- Security is critical to the functioning of organizational information systems
- Security is a challenge for everyone involved in systems development
- Information systems present multiple vulnerabilities
  - Perfect and unbreakable security (for computerised systems) is not practical
  - Organizations make ‘trade-off’s’ (security *vs* usability)
  - Organizations must ‘weigh’ the value of the data they are storing with the risk that they will experience a security breach

Security

- As a systems designer and developer:
  - It is important that in the you design new system() there is an awareness of the need to design security into a system from the very beginning
- Developing privacy controls and security by design:
  - from the outset of systems design (of new systems) it is desirable and effective than adding it to older legacy systems
- For older (legacy) systems:
  - Always examine such systems when updating or improving for ways to address vulnerabilities and improve training for security issues
  - Addressing such security requirements is a feature of the maintenance phase of the software life cycle

Planning

- System planning is essential as:
  - Developing a system without proper planning leads to:
    - User dissatisfaction
    - Frequently causes the system to fall into disuse
- Systems analysis and design:
  - Lends structure to the analysis and design of information systems
  - Lack of a suitable structure is a costly endeavour
    - Avoid disorganised and haphazard design and development
  - Analysis and design can be thought of:
    - A series of processes systematically undertaken to improve a business through the use of computerized information systems

User Involvement

- User involvement throughout a systems project is critical to the successful development of computerized information systems.
- Systems analysts (we consider the roles in the organization later):
  - Are the other essential component in developing useful information systems
- Users are moving to the forefront as software development teams become more international and collaborative in their composition
- This means that there is more emphasis on working with software users in the design phase when performing:
  - The analysis of the business
  - Identifying and analysing the business problems
  - Identifying the business objectives
  - Communicating the analysis and design of the planned system to all involved (stakeholders)

Roles of a Systems Analyst

A Systems Analyst

- A systems analyst systematically assesses:
  - How users interact with technology
  - How businesses function
- A systems analyst investigates and examines:
  - The collection and inputting of data
  - The processing of data
  - The outputting of information (with the goal of improving organization processes)
- Many improvements involve better support of users' work tasks and business functions through the use of computerized information systems:
  - This definition emphasizes a systematic, methodical approach to analysing (and potentially improving) what is occurring in the specific context experienced by users and created by a business

A Systems Analyst

- The definition of a systems analyst is necessarily broad
- An analyst must
  - Be able to work with people at all levels of management and all descriptions
  - He/she must be an experienced knowledge engineer (human interactions and information elicitation)
  - Must be experienced in working with computers and computer systems
- An analyst:
  - Plays many roles
  - Must be able to balancing several roles simultaneously (at the same time)
- The three primary roles of a systems analyst are:
  - Consultant
  - supporting expert
  - An agent of change

Consultant

- A systems analyst frequently acts as a systems consultant to humans and their businesses
  - He/she may be hired specifically to address IS issues in a business
- An advantage is a fresh perspective a consultant provides (a view not possessed by an organization or the management)
- There are however disadvantages: a consultant will be an outsider with limited (or no) prior knowledge related to the true organization culture.
- As an outside consultant:
  - There is a heavy reliance on the systematic analysis methods discussed to design appropriate information systems for users working in a particular business
  - There will be a reliance on information systems users to aid an understanding the organizational culture from a range of opinions and viewpoints

Systems Analyst as Supporting Expert

- Another role that you may be required to play is that of supporting expert within a business for which you are regularly employed in some systems capacity
- In this role:
  - An analyst draws on professional expertise concerning computer hardware and software and their uses in the business.
  - This work is often not a full-blown systems project and may entail only a small modification or a decision affecting a single department.
- As the supporting expert:
  - You are not managing the project
  - You are merely serving as a resource for those who are managing it.
- If you are a systems analyst employed by a manufacturing or service organization, many of your daily activities may be encompassed by this role.

Required Qualities for a Systems Analyst

- A systems analyst must display a number of personality traits:
- He/she must be:
  - A problem solver capable of innovative thinking
  - Be aware of current social and technical developments
  - Be a good communicator
  - Self-disciplined and self motivated
- A systems analyst must have strong personal and professional ethics
  - Systems must comply with evolving legal, regulatory, and social constraints

Required Qualities for a Systems Analyst

- Problem solving: he/she views the analysis of problems as a challenge and who enjoys devising workable solutions
- When necessary, an analyst must be able to systematically tackle the situation at hand through a skilful application of tools, techniques, and experience.
- An analyst must also be a communicator capable of relating meaningfully to other people (on all levels within an organisation) over extended periods of time.
- Systems analysts need to be able to:
  - Understand humans' needs when interacting with technology
  - Enough computer experience to write and understand computer code
  - To understand the capabilities of computers and computer systems
  - To glean information requirements from users
  - To communicate what is needed to computer programmers

Systems Analyst and Ethics

- Analysts must display:
  - Strong personal ethics
  - Strong professional ethics
  - An ethical approach to systems analysis and design is important in shaping client relationships and compliance
- A systems analyst must be:
  - self-disciplined
  - Self-motivated
  - Capable of managing and coordinating Other people (teams) and the multiple project resources.
- Systems analysis is demanding as it is ever changing and challenging

The Systems Development Life Cycle

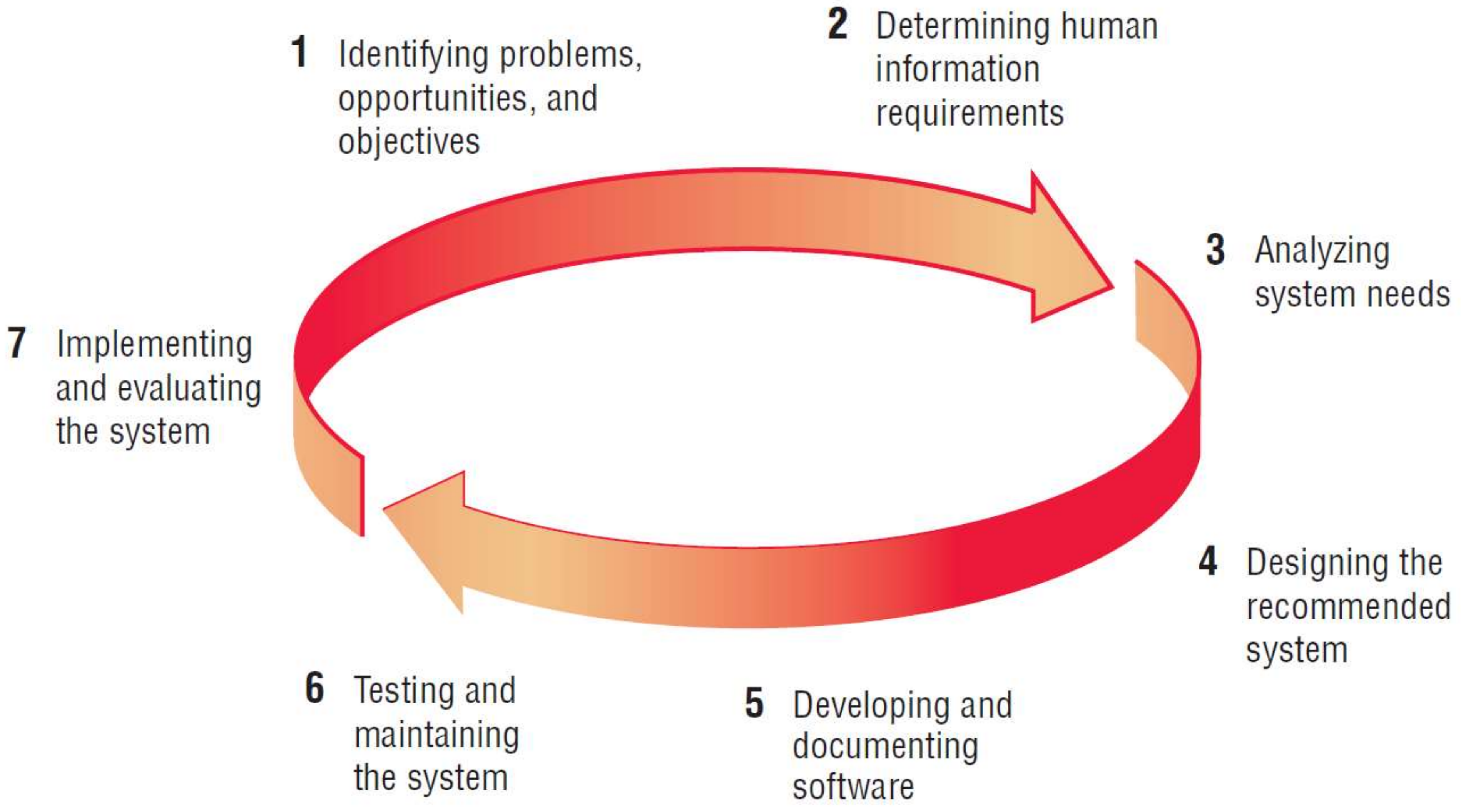
Systems Development Life Cycle (SDLC)

- The Systems Development Life Cycle (SDLC):
  - Is a systematic approach analysts take to the analysis and design of systems (including information systems)
  - It is a phased approach to analysis and design (to solving business problems)
  - It is based on the assumption that systems are developed through the use of a specific cycle of analyst and user activities
  - Each phase has unique user activities
- The SDLC:
  - Is often called the waterfall method because the system analysis completes the first phase, then moves down to the next
  - However there are iterative approaches such as the spiral model

Systems Development Life Cycle (SDLC)

- The Systems Development Life Cycle (SDLC):
  - Analysts disagree on exactly how many phases there are in the SDLC
  - However, systematic methods are generally recognized as the best and support its organized approach
- In Figure 1.1 we have divided the cycle into seven phases
  - While each phase is presented discretely phases are never completed as a separate step
  - In practice: several activities may occur simultaneously and activities may be repeated

The Seven Phases of the SDLC (Figure 1.1)


1 Identifying problems, opportunities, and objectives
2 Determining human information requirements
3 Analyzing system needs
4 Designing the recommended system
5 Developing and documenting software
6 Testing and maintaining the system
7 Implementing and evaluating the system

Identifying Problems, Opportunities, and Objectives

- In this first phase of the SDLC:
  - Analysts are concerned with correctly identifying problems, opportunities, and objectives (the requirements analysis and specification)
  - This stage is critical to the success of the project because no one wants to waste time addressing the wrong problem.
  - The first phase requires that:
    - The analyst analyse what is happening in a business
    - Then, together with other organization members, the analyst pinpoints problems
    - Other stakeholders (in the IS) often bring up these problems (the reason the analyst was initially engaged)

Opportunities, and Objectives

- Opportunities are situations are identified that can be improved through the use of computerized information systems
- Seizing opportunities may allow the business to gain a competitive edge or set an industry standard.
- Identifying objectives is also an important component of the first phase.
  - The analyst must first discover what the business is trying to do.
  - Then the analyst will be able to see whether some aspect of information systems applications can help the business reach its objectives by addressing specific problems or opportunities

The Stakeholders

- The stakeholders (the people) involved in the first phase are:
  - Users
  - Analysts
  - Systems managers coordinating the project
- . Activities in this phase consist of:
  - Interviewing user management
  - Summarizing the knowledge obtained
  - Estimating the scope of the project
  - Documenting the results.

The Goals of an Analysis

- The output from an analysis is:
  - The analyst understands how users accomplish their work when interacting with a computer
  - Begin to know how to make the new system more useful and usable
  - Know the business functions
  - Have complete information on the:
    - People
    - Goals
    - Data
    - Procedures involved
- The overall goal is to learn the who, what, where, when, how, and why of the current system

The Output from the Analysis

- The output of this phase is:
  - A feasibility report that contains a problem definition and summarizes the objectives
- Management must then make a decision on whether to proceed with the proposed project based on:
  - Adequate funding
  - If unrelated problems require addressing
- There are many problems for which a computerised system is not appropriate and:
  - A different solution may be recommended and the systems project will not proceed

Determining Human Information Requirements

- In the second phase the analyst determines the human needs of the users involved, using a variety of tools to understand how users interact in the work context with their current information systems
- The analyst uses interactive methods such as:
  - Interviewing
  - Sampling and investigating hard data
  - Using questionnaires
  - Unobtrusive methods such as:
    - Observing decision makers' behaviour and their office environments
  - All-encompassing methods, such as prototyping

Human Computer Interaction

- The demand for analysts capable of incorporating *Human–Computer Interaction* (HCI) into the systems development is increasing:
  - Companies realize that the quality of systems and the quality of work life can be improved by taking a human-centered approach at the outset of a project
- The analyst uses interactive methods to answer many questions concerning *Human–Computer Interaction* (HCI) including questions such as:
  - "What are the users' physical strengths and limitations?"
- In other words,
  - "What needs to be done to make the system audible, legible, and safe?"
  - "How can the new system be designed to be easy to use, learn, and remember?"
  - "How can the system be made pleasing or even fun to use?"
  - "How can the system support a user's individual work tasks and make them more productive in new ways?"

Determining Human Information Requirements

- In the information requirements phase of the SDLC the analyst is:
  - Trying to understand what information users need to perform their jobs.
- At this point, the analyst is examining how to
  - make the system useful to the people involved.
  - How can the system better support individual tasks that need to be done?
  - What new tasks are enabled by the new system that users were unable to do without it?
- How can the new system extend a user's capabilities beyond what the old system provided?
- How can the analyst create a system that is rewarding for workers to use?
- The people involved in this phase are the analysts and users, typically operations managers and operations workers (stakeholders)

Determining Human Information Requirements

- The systems analyst needs to know the:
    - Details of the functions of the current system
    - Stakeholders (the people who are involved with the IS and those with an interest in the system)
    - What? (the business activity)
    - Where? (the environment in which the work takes place)
    - When? (the timing)
    - How? (how the current procedures are performed) of the business under study.
- The analyst must then ask why the business uses the current system?
    - There may be good reasons for doing business using the current methods
    - These should be considered when designing any new system.

The Analysis

- Agile development (introduced in later tutorials) is an outgrowth of the object-oriented approach (OOA) to systems development that includes:
  - a method of development (including generating information requirements) as well as software tools.
- If the reason for current operations is that “it’s always been done that way,” the analyst may wish to improve on the procedures.
- At the completion of this phase the analyst should:
  - Understand how users accomplish their work when interacting with a computer
  - Begin to know how to ANALYSIS FUNDAMENTALS make the new system more useful and usable.
- The analyst should also know how the business functions and have complete information on the people, goals, data, and procedures involved.

Determining Human Information Requirements

- The analyst uses interactive methods to answer many questions concerning *Human–Computer Interaction* (HCI) including questions such as:
  - “What are the users’ physical strengths and limitations?”
- In other words,
  - “What needs to be done to make the system audible, legible, and safe?”
  - “How can the new system be designed to be easy to use, learn, and remember?”
  - “How can the system be made pleasing or even fun to use?”
  - “How can the system support a user’s individual work tasks and make them more productive in new ways?”

Analysing System Needs

- The next phase involves an analysis of system needs
- Special tools and techniques help the analyst make requirements a determination
  - Tools such as data flow diagrams (DFDs)chart the:
    - Input
    - Processes
    - business's functions
  - Activity diagrams (or) sequence diagrams show the sequence of events showing systems in a structured graphical format
- From data flow, sequence, (or) other diagrams
  - A data dictionary is developed that lists all the data items used in the system with their specifications

Analysing System Needs

- During this phase an analysis of the structured decisions carried out
  - Structured decisions are those for which the conditions, condition alternatives, actions, and action rules can be determined.
  - Three major tools are used when analysing structured decisions:
    - Structured English
    - Decision tables
    - Decision trees.
- At this point in the SDLC a systems proposal is prepared which summarises:
  - What has been discovered about the users, usability, and usefulness of current systems
  - Provides cost-benefit analyses of alternatives
  - Makes recommendations on what (if anything) should be done.

Analysing System Needs

- If one of the recommendations is acceptable to management, the analyst proceeds along that course
- Each systems problem is unique, and there is never just one correct solution.
- The manner in which a recommendation or solution is formulated depends on
  - The individual qualities
  - The professional training of each analyst
  - The analyst's interaction with users in the context of their work environment.

Designing the Recommended System

- In the design phase of the SDLC:
  - The information collected earlier is used to create the logical design of the IS
  - Input procedures are designed for accurate data entry to ensure correct input
  - Additionally users are provided with complete and effective input to the IS using techniques of good form and web page or screen design
  - The output will be a mode of the IS and related systems
- Part of the logical design of the information system is devising the HCI
  - The interface connects the user with the system and is extremely important
  - The user interface is designed and evaluated with the help of users to make sure the system is audible, legible, and safe, as well as attractive and enjoyable to use

Designing the Recommended System

- Examples of physical user interfaces include
  - keyboards (to type in questions and answers)
  - Onscreen menus (to elicit user commands)
  - Graphical user interfaces (GUIs) that use a mouse or touch screen
- The design phase also includes
  - Designing databases and data structures to store data and information needed by decision makers in the organization.
  - Designing data backup systems and disaster recovery system
- Users benefit from a well-organized database that is logical to them and corresponds to the way they view their work – in this phase
  - The analyst also works with users to design output (either onscreen or printed) that meets their information needs

Developing and Documenting Software

- In the fifth phase of the SDLC:
  - The analyst works with computer programmers
  - The aim is the development of any original software that is needed.
- Additionally, in this phase:
  - The analyst works with users to develop effective system documentation for software
  - The documentation will include:
    - Procedure manuals
    - User manuals (graded for different levels of user)
    - An online help system
    - A website with Frequently Asked Questions (FAQ's) (or) ReadMe files system shipped with new software.

Developing and Documenting Software

- Because users are involved from the beginning:
  - Documentation should address the questions they have raised and solved jointly with the analyst.
- Documentation tells users:
  - How to use software
  - What to do if software problems occur.
- Programmers have a key role in this phase because they:
  - Design the program and write the program code
  - Test and remove syntactic and logic errors from computer programs
  - Document the program code (essential in maintenance phase of the SLC)
- To ensure quality:
  - A programmer may conduct either a design or a code walkthrough
  - This can explain complex portions of the software to a team of other programmers

Testing and Maintaining the System

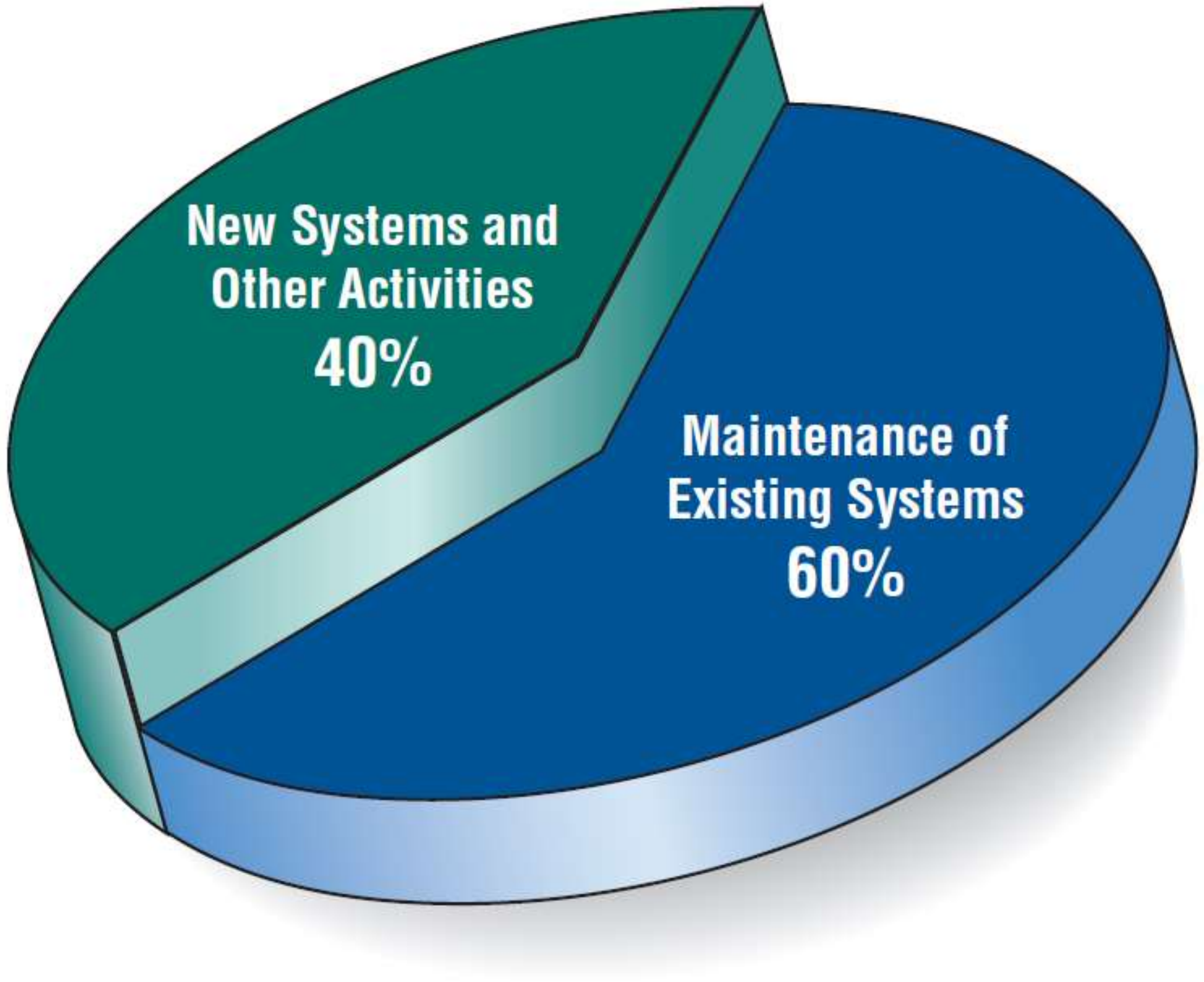
- Testing and maintaining the system is very important:
  - It is much less costly to catch errors and problems before implementation and the system is signed over to users
  - While programmers test a computer program testing by the systems analysts both alone and in conjunction with coders.
  - A series of tests to pinpoint problems is run
  - Initially with sample data and eventually with actual data from the current system
- Test plans:
  - Are generally created early in the SDLC (at the design stage) and are refined as the project progresses
  - Maintenance of the system and its documentation begins in this phase and is carried out routinely throughout the life of the IS

Testing and Maintaining the System

- Businesses spend a great deal of time and money on maintenance
- Much of a computer programmers time will be spent on routine work consisting of maintenance and updating of software
- Some maintenance such as program updates can be completed automatically via a vendor site on the Web
- Many of the systematic procedures the analyst employs throughout the SDLC can help ensure that maintenance is kept to a minimum

Testing and Maintaining the System

- Some Researchers Estimate that the Amount of Time Spent on Systems Maintenance May Be as Much as 60 Percent of the Total Time Spent on Systems Projects (Figure 1.2)
- In the development and updating of software the 60% (maintenance) 40% (creation) ratio is an approximation
- Often a 70% (maintenance) 30% (creation) ratio is considered to be the general rule



Implementing and Evaluating the System

- The last phase of systems development where the analyst helps to implement the new IS
- This phase involves:
  - Training users (on many levels and roles) use and manage the system
  - Vendors may do some training but oversight of training is the responsibility of the systems analyst.
- Additionally the analyst:
  - Plans a smooth conversion (implementation) from the old system to the new one
  - This process includes converting files from old formats to new ones (or)
  - Building a database
  - installing software, hardware, and equipment
  - Bringing the new system into production

Implementing and Evaluating the System

- A critical decision when replacing an existing (legacy) system is the approach to implementation
  - There are two approaches:
    1. Straight replacement
       - Switch off the existing system and switch on the new system
    2. Piggy-back the new system on the old system the try to reduce the risk of failure
    3. Introduce the new system on a modular basis
- The first approach is fraught with potential issues
- The second and third approaches are generally preferred method as the risk of total system failure is much reduced

Implementing and Evaluating the System

- Evaluation is included as part of this final phase of the SDLC
- In the ‘real-world’ evaluation takes place during every phase of the SLC
- A key criterion that must be satisfied is whether the intended users are indeed using the system
  - We must note that systems work is often cyclical
  - An iterative approach is the general rule:
    - When an analyst finishes one phase of systems development and proceeds to the next
    - The discovery of a problem may force the analyst to return to the previous phase and modify the work done there.

The Impact of Maintenance

- Following installation a system must maintained
  - Maintenance of computer systems and application involves:
    - Updating and keeping a system up to date
    - Modifications (for new features)
    - Addressing ‘software bugs’ and security patches
- Estimates of the time spent by departments on maintenance
  - Range from 48 percent to 60 percent of the total time spent developing systems
  - Very little time remains for new systems development

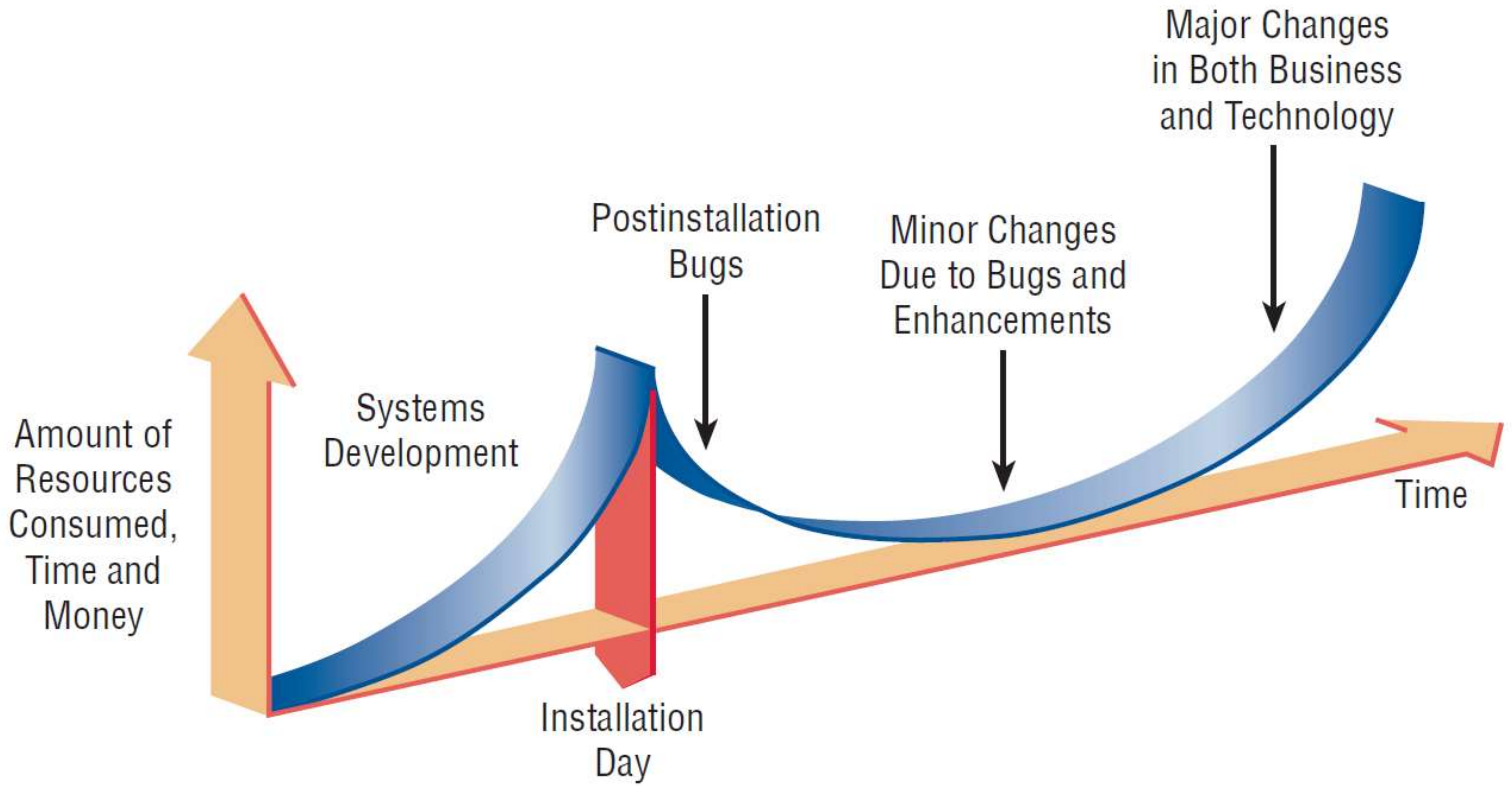
The Impact of Maintenance

- As the number of programs written increases, so does the amount of maintenance they require
- Maintenance is performed for two reasons:
  - Correction of software errors
    - No matter how thoroughly a system is tested, bugs or errors creep into computer applications.
    - Bugs in commercial PC software are often documented as "known anomalies," and they are corrected when new versions of the software are released or in an interim release.
    - In custom software (also called bespoke software), bugs must be corrected as they are detected
  - To enhance the software's capabilities in response to changing organizational needs (generally involving one of the following three situations):
    - Users often request additional features after they become familiar with the computer system and its capabilities
    - The business changes over time
    - Hardware and software are changing at an accelerated pace.

The Impact of Maintenance

- Over time the cost of continued maintenance will be greater than that of creating an entirely new system
  - At that point it becomes more feasible to perform a new systems study
  - Figure 1.2 shows the Resource Consumption over the System Life (the SLC) and illustrates the amount of resources (usually time and money) spent on systems development and maintenance.
  - The area under the curve represents the total dollar amount spent
  - It can be seen that:
    - Over time, the total cost of maintenance is likely to exceed the cost of developing a new system and systems development
    - At a certain point it becomes more feasible to perform a new systems study because the cost of continued maintenance is clearly greater than the cost of creating an entirely new information system

Resource Consumption over the SLC (Figure 1.2)


Major Changes in Both Business and Technology
Postinstallation Bugs
Minor Changes Due to Bugs and Enhancements
Systems Development
Amount of Resources Consumed, Time and Money
Time
Installation Day

Summary

- Maintenance is an ongoing process over the life cycle (SLC) of an information system
- After the information system is installed maintenance usually takes the form of correcting previously undetected software errors
- Once these are corrected the system approaches a steady state providing dependable service to its users
- Maintenance during this period may consist of removing a few previously undetected bugs and updating the system with a few minor enhancements
- As time goes on and the business and technology change
  - However the maintenance effort increases dramatically

Structured Analysis and Design

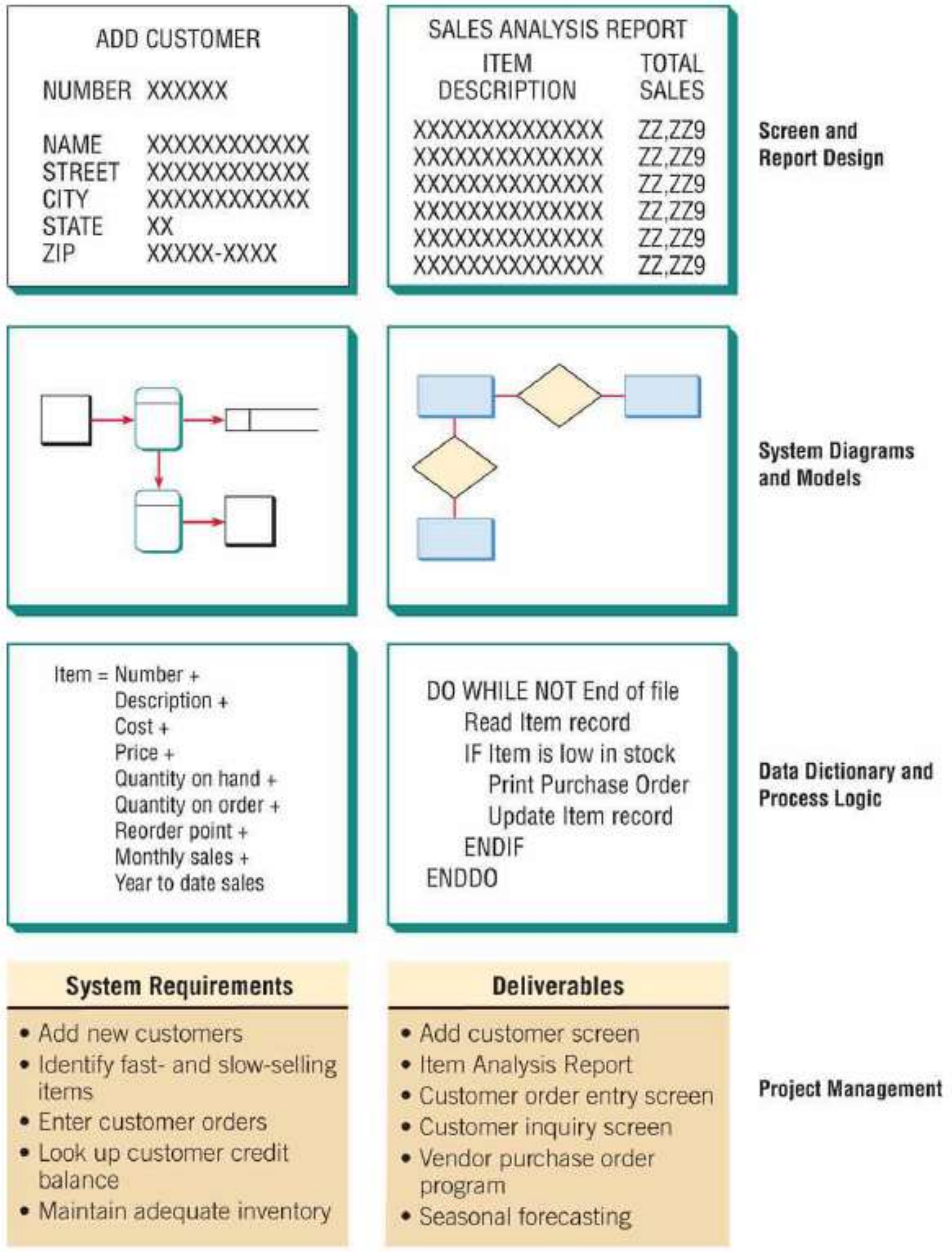
- Approaches to Structured Analysis and Design and to the Systems Development Life Cycle (SDLC):
  - Traditional systems development life cycle
  - CASE systems development life cycle
  - Object-oriented systems analysis and design
- Analysts who adopt the SDLC approach often benefit from productivity tools
  - Called computer aided software engineering (CASE) tools
  - Created explicitly to improve their routine work through the use of automated support

Using Case Tools

- Analysts rely on CASE tools to:
  - Increase productivity
  - Communicate more effectively with users and in collaborative design
  - Integrate the work that they do on the system from the beginning to the end of the life cycle.
- All the information about the project is stored in an “encyclopedia” called the CASE repository:
  - A large collection of records, elements, diagrams, screens, reports, and other information (see Figure 1.3)
- Analysis reports may be produced using the repository information to show where the design is incomplete or contains errors

The Repository Concept

- Figure 1.3
- We can see from the figure:
    - Screens and report design
    - Screen design and models
    - Data dictionary and process logic
    - Project management setting out:
- System requirements
- deliverables



```
ADD CUSTOMER

NUMBER  XXXXXX

NAME    XXXXXXXXXXX
STREET  XXXXXXXXXXX
CITY    XXXXXXXXXXX
STATE   XX
ZIP     XXXXX-XXXX
```

```
SALES ANALYSIS REPORT
    ITEM            TOTAL
 DESCRIPTION        SALES
XXXXXXXXXXXXX      ZZ,ZZ9
XXXXXXXXXXXXX      ZZ,ZZ9
XXXXXXXXXXXXX      ZZ,ZZ9
XXXXXXXXXXXXX      ZZ,ZZ9
XXXXXXXXXXXXX      ZZ,ZZ9
XXXXXXXXXXXXX      ZZ,ZZ9
```

```
Item = Number +
       Description +
       Cost +
       Price +
       Quantity on hand +
       Quantity on order +
       Reorder point +
       Monthly sales +
       Year to date sales
```

```
DO WHILE NOT End of file
   Read Item record
   IF Item is low in stock
      Print Purchase Order
      Update Item record
   ENDIF
ENDDO
```

System Requirements

- Add new customers
- Identify fast- and slow-selling items
- Enter customer orders
- Look up customer credit balance
- Maintain adequate inventory

Deliverables

- Add customer screen
- Item Analysis Report
- Customer order entry screen
- Customer inquiry screen
- Vendor purchase order program
- Seasonal forecasting

Visible Analyst

- Visible Analyst (VA) is one (of many) example of a CASE tool:
  - It enables systems analysts to achieve graphical planning, analysis, and design in order to build complex client/server applications and databases
- Visible Analyst and software products such as Microsoft Visio or OmniGraffle (ANALYSIS FUNDAMENTALS) provide a system where users can draw and modify diagrams and models easily and quickly ‘on-the-fly’ (however consider: a pretty model may not be logically correct)
- However, VA is a full-fledged CASE tool with a repository and other features, whereas the other two are not.
- Analysts and users alike report that CASE tools afford them a means of communication about the system during its conceptualization

SDLC and Case Tools

- Through the use of automated support featuring onscreen output:
  - Clients can readily see how data flows and other system concepts are depicted
  - Clients can then agree the design plan (or)
  - Clients can request corrections or changes that would have taken too much time with older tools.
- CASE tools:
  - Also help support the modelling of an organization’s functional requirements
  - Assist analysts and users in drawing the boundaries for a given project
  - Help in the visualisation of how the project integrates and meshes with other parts of the organization

The Agile Approach

The Agile Approach

- While our focus is on the SDLC (the most widely used approach in practice)
- There are scenarios where an alternative approach is more appropriate
  - Perhaps a systems project using a structured approach has recently failed
  - Perhaps the organization subcultures, composed of several different user groups, seem more in step with an alternative method
- Analysts must:
  - Be aware of the alternative methods
  - Apply the most appropriate method driven by an organisations circumstances
  - Organizations may consider an alternative or a supplement to structured analysis and design and to the SDLC

The Agile Approach

- The agile approach is a software development approach based on
    - Values / principles / core practices
    - The four values are communication / simplicity / feedback / courage
- In this approach these values must be adopted for all projects
    - It is good practice to adopt these values in all methods including the SDLC
    - In practice all projects will need revision and adjustments in project management
- In Chapter 6 we can see that a agile methods can ensure successful completion of a project by adjusting the important resources of time, cost, quality, and scope.
    - When these four control variables are properly included in the planning, there is a state of balance between the resources and the activities needed to complete the project

Development Practices

- Taking development practices to the extreme is most noticeable when one pursues practices that are unique to agile development
- There are typically four core agile practices:
  - Short release
  - The 40 hour work week
  - Hosting an outside customer
  - Using pair programming
- At first glance these practices appear extreme
- But we can learn some important lessons from incorporating many of the values and practices of the agile approach into systems analysis and design projects.
- We also explore an agile method called Scrum (see Chapter #6), which is named after a starting position in the sport of rugby.
  - Scrum's success has much to do with its extremely quick releases.

The Agile Approach

- The sprint cycle:
  - A typical sprint cycle will last between two and four weeks
  - At the end of that period the team is expected to produce a potentially releasable product
  - That means that applications or websites are constantly changing
  - Each iteration boasts a new set of features produced during the sprint cycle
- In the agile approach
  - In general activities and behaviours shape the way development team members and customers act during the development of an agile project
  - Two words that characterize a project done with an agile approach are: *Interactive* and *incremental*
- Figure 1.4 illustrates the five distinct stages of the agile approach:
  - *exploration*, *planning*, *iterations to the first release*, *productionizing*, and *maintenance*

The Agile Process

- Three red arrows loop back into the "Iterations" box showing changes created through repeated testing and feedback leading to a stable but evolving system
- That multiple looping arrows feed back into the productionizing phase showing:
  - The pace of iterations is increased after a product is released
- The red arrow is shown leaving the maintenance stage and returning to the planning stage
  - This shows a continuous feedback loop involving customers and the development team as they agree to alter the evolving system

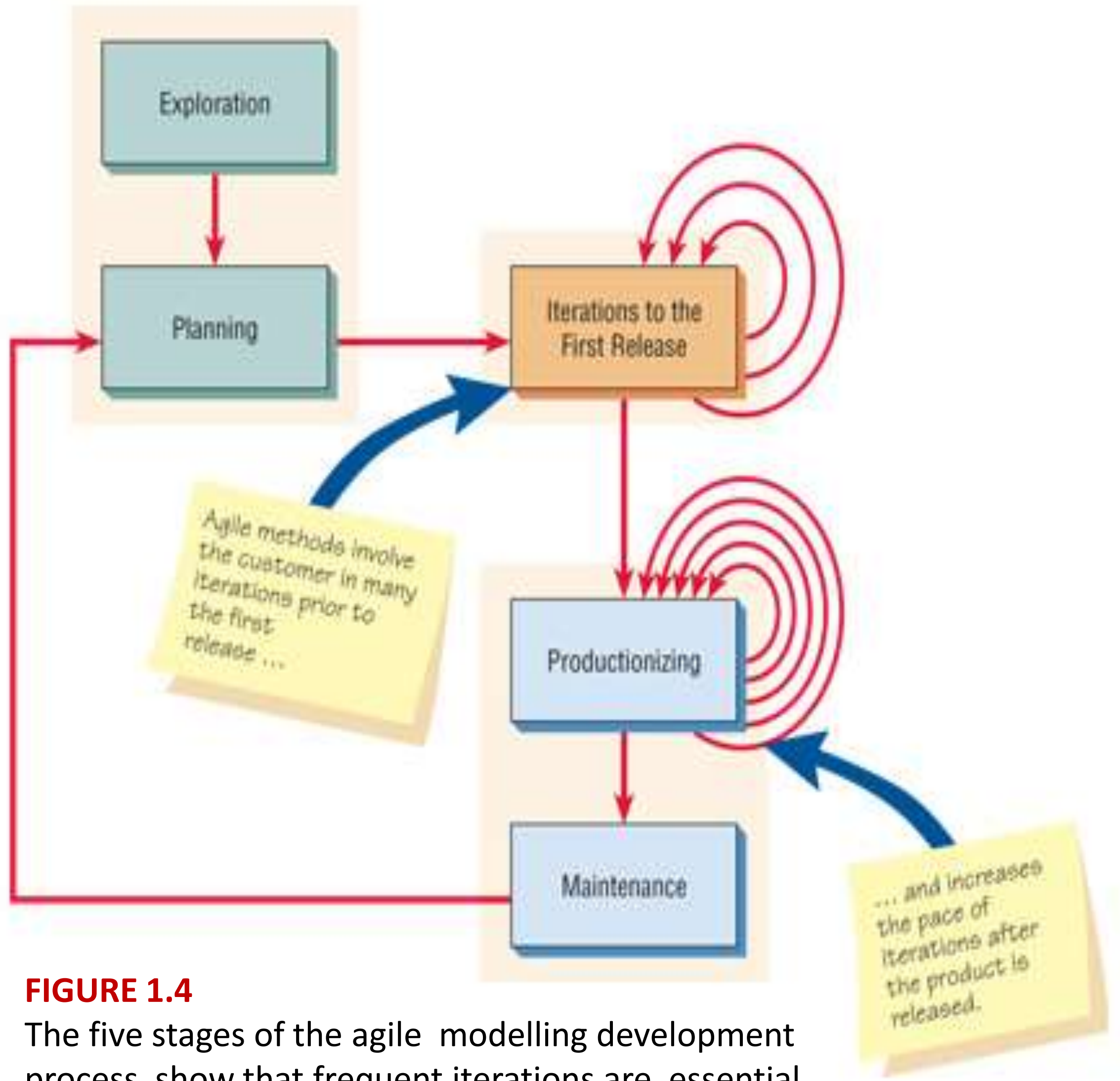

Exploration
Planning
Iterations to the First Release
Agile methods involve the customer in many iterations prior to the first release ...
Productionizing
Maintenance
... and increases the pace of iterations after the product is released.

FIGURE 1.4
The five stages of the agile modelling development process show that frequent iterations are essential for successful system development.

Five Stages of Agile Development

- The five phases (stages) that define the agile development method are:
  - Exploration
  - Planning
  - Iterations to the first release
  - Productionizing
  - Maintenance
- In the following slides we introduces these phases.

The Exploration Stage

- During exploration:
  - The environment (the domain of interest) is explored
  - The aim is to:
    - Confirm that the problem can (and should) be approached with agile development
    - Assemble the team
    - Assess team member skills
- The time scale:
  - A few weeks (if you already know your team members and technology)
  - A few months (if everything is new)
- In this stage:
  - Potential technologies will be investigated
  - The time required for completing a variety of tasks
  - Customers (the client) experiment with drafting user stories and refine the story to enable a suitable development time to be estimated
  - Adopt a playful and curious attitude toward the work environment, its problems, technologies, and people.

The Planning Stage

- The next stage of the agile development process is planning
  - In contrast to the first stage planning may take only a few days
- In this stage analysts and customers agree on a delivery schedule
  - Generally from two months to 6 months to address the most pressing business problems (you will be addressing the smallest, most valuable set of stories).
- If your exploration activities were sufficient:
  - This stage should be very short.
- The agile planning process has been characterized using the idea of a planning game (as devised by Kent Beck, the father of Extreme Programming)

The Planning Stage

- The planning game spells out rules that can help formulate the agile development team's relationship with their business customers
  - Although the rules for how each party acts during development
  - The rules are not intended to replace the relationship but are a basis for building and maintaining a relationship
- Using the metaphor of a game we talk in terms of the goal of the game and consider the:
  - *Strategy* to pursue
  - *Pieces* to move
  - *Players* involved

The Goal of the Game Scenario

1. Is to maximize the value of the system produced by the agile team
2. To arrive at the value
    1. The costs of development must be deducted
    2. The time, expense, and uncertainty taken on so that the development project can go forward

- The strategy pursued by the agile development team is always one of limiting uncertainty and risk
    - To do that they design the simplest solution possible
    - Put the system into production as soon as possible
    - Get feedback from the business customer about what's working
    - and adapt their design based on the development process and feedback

Planning and Story Cards

- Story cards become the pieces in the planning game that briefly describe:
    - The task
    - Provide notes
    - Provide an area for task tracking
- The two main players in the planning game are:
    - The development team
    - The client (the business customer)
- Deciding which business group will be the business customer may be difficult because:
    - The agile process is an unusually demanding role (the customer is a central part)
- Customers decide what the development team should tackle first:
    - Their decisions will set priorities and check functionalities throughout the process

Iterations to the First Release

- The third stage in the agile development process is composed of iterations to the first release
- Typically these iterations (cycles of testing, feedback, and change) are about three weeks in duration
- You will be pushing yourself to sketch out the entire architecture of the system (even though it is just in outline or skeletal form)
- One goal is to run customer-written function tests at the end of each iteration
- During the iterations stage you should also question whether the schedule needs to be altered or whether you are tackling too many stories

Iterations to the First Release

- The ‘fun’ element in the agile development methodology is am important component
- It is important to make small rituals out of each successful iteration
- Involve customers as well as developers
- Always celebrate your progress even if it is small because this is part of the culture of motivating everyone to commit to the project

Productionising

- Several activities occur during the productionizing phase
  - In this phase the feedback cycle speeds up so that rather than receiving feedback for an iteration every three weeks
  - Software revisions completed in around in one week
- You may hold daily briefings so everyone knows what everyone else is doing
  - The product is released in this phase
  - But it may be improved by adding other features
- Getting a system into production is an exciting event.
  - Make time to celebrate with your teammates and mark the occasion
  - One of the keys to the agile approach is that systems development is fun!

Maintenance

- Once a system has been released:
  - it must to be kept running smoothly
  - New features may be added
  - Riskier customer suggestions may be considered
  - The team members may be rotated on or off the team
- The attitude you take at this point in the development process:
  - is more conservative (careful) than at any other time
  - You are now in a “keeper of the flame” mode rather than the playful one you experienced during exploration.

Object-Oriented Systems Analysis and Design

Unified Modeling Language (UML) Phases

- The stages in UML design:
  - Define the use case model:
    - *Use case* diagram
    - *Use case* scenarios
  - Create *UML diagrams*
  - Develop *class* diagrams
  - Draw *statechart* diagrams
  - Modify the *UML* diagrams
  - *Develop* and *document* the system

Object-Oriented Analysis and Design

- Object-oriented (OO) systems analysis and design is:
  - An approach intended to facilitate the development of systems that must change rapidly in response to dynamic business environments
  - See Chapter #10 where the differences from the structured approach of the SDLC and when it may be appropriate to use an OO approach
- In summary:
  - OO techniques often work well in scenarios where complicated information systems are:
    - Undergoing continuous maintenance
    - Frequent adaptation
    - Frequent redesign

Object-Orientation

- OO approaches generally use UML which is:
  - An industry standard for modelling OO systems
  - UML is used to model scenarios and break down systems into ***Use-Case models***
- OO programming differs from traditional procedural programming in that:
  - It is predicated on the concept of "objects"
  - It examines objects that are part of a system
  - Each "object" is a computer representation of some actual thing or event.
  - Objects may be customers, items, orders etc
  - OO programming is designed to promote code reuse

Objects and Classes and Similarities to the SDLC

- Objects are represented by and grouped into classes that are optimal for reuse and maintainability
- A class:
  - Uses encapsulation and polymorphism
  - Defines a set of shared attributes and behaviours found in each object in the class
- OO similarities to the SDLC:
  - The phases in UML are similar to those in the SDLC
  - Because these two methods share rigid and exacting modelling:
    - They happen at a slower and more deliberate pace than the phases of agile modelling

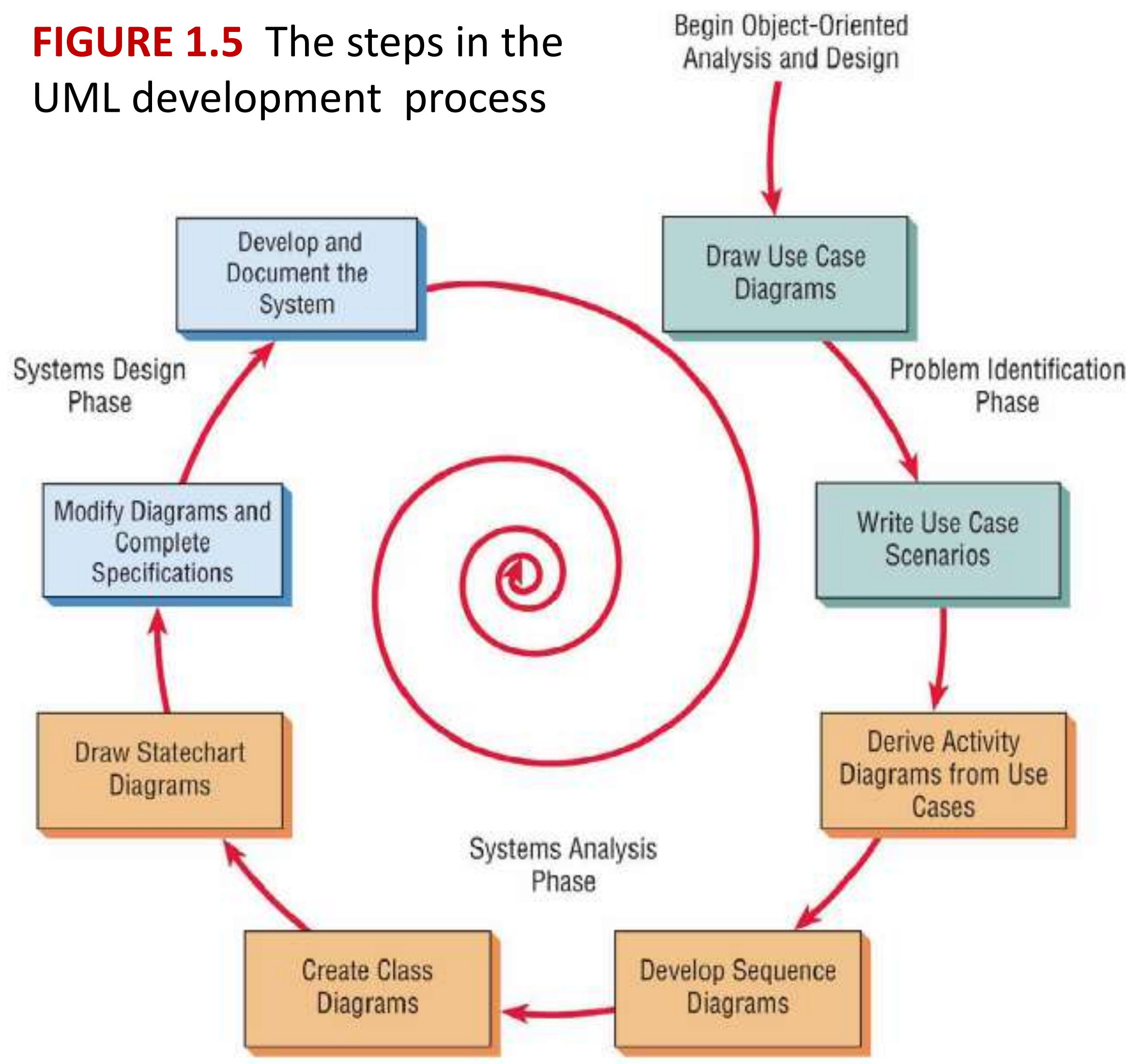
Problem Solving

- An analyst assesses the problem and identification phases (see Figure 1.5) which are:
  - An *analysis* phase, and a *design* phase
  - Chapters #2 and #10 discuss the specific details
- We set out in the following slides a brief overview of the OO UML process

Problem Assessment

- An analyst assesses the problem and identification phases (see Figure 1.5) which are:
  - An *analysis* phase, and a *design* phase
  - Chapters #2 and #10 discuss the specific details
- We set out in the following slides a brief overview of the OO UML process

FIGURE 1.5 The steps in the UML development process



Begin Object-Oriented Analysis and Design
Draw Use Case Diagrams
Problem Identification Phase
Write Use Case Scenarios
Derive Activity Diagrams from Use Cases
Systems Analysis Phase
Develop Sequence Diagrams
Create Class Diagrams
Draw Statechart Diagrams
Modify Diagrams and Complete Specifications
Systems Design Phase
Develop and Document the System

Define the Use-Case Model

- DEFINE THE USE CASE MODEL:
  - In this phase, the analyst identifies the actors and the major events initiated by the actors
  - Often the analyst starts by drawing a diagram with stick figures representing the actors and arrows showing how the actors relate.
  - This is called a use case diagram (Chapter 2):
    - It represents the standard flow of events in the system
- Then an analyst typically writes up a use case scenario (see Chapter #2):
  - This describes in words the steps that are normally performed

The Systems Analysis Phase

- DURING THE SYSTEMS ANALYSIS PHASE:
  - Begin drawing the UML diagrams
  - In the second phase (Chapter 10):
    - The analyst draws activity diagrams (to illustrate all the major activities in the use case scenario)
- Additionally:
  - The analyst creates one or more ***sequence diagrams*** for each use case that show the sequence of activities and their timing
  - This is an opportunity to return to the original analysis and:
    - Review and validate the use cases
    - Rethink them
    - Make any modifications considered necessary

Develop Class Diagrams

- CONTINUING IN THE ANALYSIS PHASE:
  - Develop class diagrams
  - In class diagrams nouns in the use cases are objects that can potentially be grouped into classes
  - For example: every automobile is an object that shares characteristics with other automobiles and together they make up a class
- STILL IN THE ANALYSIS PHASE:
  - Draw ***statechart diagrams***
  - The class diagrams are used to draw *statechart diagrams* which help in understanding complex processes that cannot be fully derived by the sequence diagrams
  - The *statechart diagrams* are extremely useful in modifying class diagrams
  - This continues the iterative process of UML modelling

Modify UML Diagrams

- BEGIN SYSTEMS DESIGN BY MODIFYING THE UML DIAGRAMS THEN COMPLETE THE SPECIFICATIONS:
  - Systems design means modifying the existing system
  - This implies modifying the diagrams drawn in the previous phase
  - These diagrams can be used to derive classes, their attributes, and methods (operations)
- An analyst will need to:
  - Write class specifications for each class
  - All attributes, methods, and their detailed descriptions must be included
- The analyst also will develop:
  - Method specifications that detail the input and output requirements for each method
  - A detailed description of the internal processing of the method

Development and Documentation

- DEVELOP AND DOCUMENT THE SYSTEM:
- UML is a modelling language
- An analyst may create wonderful model
  - But if the system isn't developed there is not much point in building models
  - Documentation is critical
  - The more complete the information you provide to the development team through documentation and UML diagrams is: the faster the development and the more solid the final production system
- Object-oriented methodologies often focus on small, quick iterations of development (often termed the *Boehm spiral model*)
  - The analysis is performed on a small part of the system
  - Usually starting with a high-priority item or perhaps the item that has the greatest risk

The Final Steps

- This is followed by design and implementation
- It is an iterative process in which the cycle is repeated with:
    - An analysis of the next part
    - Design
    - Some initial implementation
- This is repeated until the project is complete
- Constant revision and the reworking of the (related) UML diagrams and the components modelled is normal
- UML is a powerful modelling tool that is:
    - Well understood and graphically clear
    - Is supported by many software application
    - Is accepted as an industry standard

Method Selection

Choosing Which Systems Development Method to Use

- Choose either:
  - SDLC
  - Agile
  - Object-oriented methodologies
- The differences among the three approaches described here are not as large as they seem
- In all three approaches:
  - the analyst needs to understand the organization first (see #Chapter 2)
  - Then the analyst or project team needs to budget time and resources and develop a project proposal (see #Chapter 3)
  - Next, they need to interview organization members and gather detailed data using questionnaires (see #Chapter 4)
  - Sample data from existing reports and by observing how business is currently transacted (see #Chapter 5).

Choosing Which Systems Development Method to Use

- The three approaches have all of these activities in common
- Even the methods themselves have similarities
  - The SDLC and OOAs both require extensive planning and diagramming
  - The agile approach and the OOA both allow subsystems to be built one at a time until the entire system is complete
  - The agile and SDLC approaches are both concerned with the way data logically moves through the system
- Given a choice to develop a system using an SDLC approach, an agile approach, or an object-oriented approach:
  - which would you choose?
  - Figure 1.6 provides a set of guidelines to help you choose which method to use when developing your next system

Method Selection Summary

- When to use agile:
  - The problems modeled lend themselves to classes
  - An organization supports the UML learning
  - Systems can be added gradually, one subsystem at a time
  - Reuse of previously written software is a possibility
  - It is acceptable to tackle the difficult problems first
- When to Use SDLC:
  - Systems have been developed and documented using SLDC
  - It is important to document each step
  - **Upper level management feels more comfortable or safe using SDLC**
  - There are adequate resources and time to complete the full SDLC
  - Communication of how new systems work is important

Method SelectionSummary

- When to Use an Object-Oriented approach:
  - The problems modeled lend themselves to classes
  - An organization supports the UML learning
  - Systems can be added gradually, one subsystem at a time
  - Reuse of previously written software is a possibility
  - It is acceptable to tackle the difficult problems first

Open Source Software

Developing Open Source Software

- In traditional software development:
    - Proprietary program code is hidden from the users
    - An example is Microsoft Windows (in all the editions)
- As an alternative to traditional software development:
    - Open Source Software (OSS)
    - With OSS, many users and coders can study, share, and modify the code, or computer instructions (subject to license conditions)
    - Rules of this community include the idea that any program modifications must be shared with all the people on the project and that all licenses must be adhered to.
    - Development of OSS also has been characterized as a philosophy rather than simply as the process of creating new software

Developing Open Source Software

- Those involved in OSS communities often view it as a way to help societies change.
- Widely known open source projects include:
  - Apache for developing a web server
  - The browser called Mozilla Firefox
  - Linux, which is a Unix-like open source operating system
    - Linux provides full details of the kernel (Microsoft does not) where the community can re-program and re-compile (distribution is controlled) the kernel)
- However, it would be an oversimplification to think of OSS as a monolithic movement, and it does little to reveal what type of users or user analysts are developing OSS projects and on what basis

Developing Open Source Software

- To help us understand the open source movement researchers have recently categorized open source communities into four community types and six dimensions:
  - The communities:
  - ad hoc / standardized / organized / commercial
- The dimensions:
  - General structure / environment / goals / methods / user community / licensing
- Research has argued that OSS is at a crossroads and that the commercial and community open source groups need to understand where they converge and where the potential for conflict exists

Why Organizations Participate in Open Source Communities?

- Organizations participate in open source communities for a variety of reasons:
  - One is the rapidity with which new software can be developed and tested
  - It is faster to have a committed group of expert developers develop, test, and debug code than it is to have one isolated team working on software development. This cross-fertilization can be a boon to creativity.
  - Another reason to participate is the benefit of having many good minds working on innovative applications. Yet another reason for participating in an open source community might be the potential for keeping down development costs.
- Organizations might seek to participate in an open source community:
  - To a desire to bolster their own self-image and to contribute something worthwhile to the larger software development community.
  - They may want to contribute generously or altruistically to a higher good beyond developing profitable proprietary software, and doing so may make them appear as "good guys" to the external public

Why Organizations Participate in Open Source Communities?

- Organizations may ask (or even require) that their software developers become involved in one or more open source projects or communities:
  - But individual developers must interact with the community in a meaningful and knowledgeable way first to prove themselves worthy members of the group based on their merits and then to strike up and maintain relationships that are mutually beneficial.
- Organizations and the design teams within them interact with Open Communities:
  - Companies are taking advantage of an entire range of options that help them strike a harmonious equilibrium so that their contributions to the open community and their differentiation from the open community become clear strategically.
  - Reasons for contribution to and differentiation from include:
    - Cost / managing resources
    - The time it takes to bring a new product to the market

The Role of the Analyst in Open Source Software

- An analyst may be requested (by an employer) to participate in an open source community
- One widely known open source community is that surrounding the Linux kernel
  - This is a large (mostly virtual) community of developers who all have different levels or types of participation and who all have different reasons for being involved
- Other well-known open source projects include:
  - Mozilla Firefox / Android / Apache projects/ etc
- Even NASA, the U.S. National Aeronautics and Space Administration, has a lively open source community (see http://ti.arc.nasa.gov/opensource/).
- One reason your company may ask you to participate in an open source community is curiosity about what the software benefits to the organization might be

The Role of the Analyst in Open Source Software

- An organisation’s participation:
  - May be a result of a ‘bandwagon’ effect
  - When it becomes known that competitors are already participating – an organization may want to get involved
- With competitors actively participating in an open community, an organization may calculate that it is something that should at least be investigated seriously and not dismissed

The Role of the Analyst in Open Source Software

- Organisations’ may want employees to act as a developer in an open community to achieve what researchers have labelled “responsive design” which is:
  - While you are participating in the open source community
  - You are at the same time employed by an organization
- The organisation wants to leverage your participation in the open source community to incorporate OSS designs into proprietary products, processes, knowledge, and IT artefacts that it is developing
- The organizations may hope to eventually sell as a product that is differentiated from what the open source community has produced
- Through a process of responsive design the IT artefact is imbued with both community and organization structures, knowledge, and practices.

Summary Chapter #1

Chapter #1 Summary

- Systems analysis and design is a systematic approach designed to:
  - Identify problems / opportunities / objectives / threats
  - Analyse human and computer generated information flows in organizations
  - Design computerized information systems to solve problems.
- Systems analysts are required to take on many roles in the course of their work including:
  - An outside consultant to business
  - A supporting expert within a business
  - An agent of change in both internal and external situations
- It is important that you
  - Design new systems with an awareness of what you can do to design security into a system from the very beginning

Chapter #1 Summary

- Analysts possess a wide range of skills. First and foremost:
  - An analyst is a problem solver
  - Someone who enjoys the challenge of analysing a problem and devising a workable solution
- Systems analysts require:
  - Communication skills (that enable them to relate meaningfully to many different kinds of people on a daily basis)
  - Computer
  - Understanding and relating well to users is critical to their success
- Analysts proceed systematically.
- The framework for their systematic approach is provided in what is called the systems development life cycle (SDLC).

Chapter #1 Summary

- This life cycle can be divided into seven sequential phases
- Although in reality the phases are interrelated and are often accomplished simultaneously.
- The seven phases are identifying:
  - Problems / opportunities / objectives
  - Determining human information requirements
  - Analysing system needs
  - Designing the recommended system
  - Developing and documenting software
  - Testing and maintaining the system; and implementing and evaluating the system.

Chapter #1 Summary

- The agile approach is a software development approach based on:
  - Values
  - Principles
  - Core practices
- Systems that are designed using agile methods:
  - Can be developed rapidly
  - Scrum is an agile method that emphasizes short releases and allows teams to choose what they want to work on
- Stages in the agile development process are:
  - Exploration / planning / iterations to the first release / productionizing / maintenance.

Chapter #1 Summary

- A third approach to systems development is called object-oriented analysis design
  - These techniques are based on object-oriented programming concepts that have become codified in UML (a standardised modelling language)
    - In UML objects that are created which include code, data, and operations (methods) to be performed on the data.
    - In UML key diagrams help analyse, design, and communicate UML-developed systems
- These systems are usually developed as components, and reworking the components many times is typical in object-oriented analysis and design
- Analysts will have increasing opportunities to participate in open source development communities, often through their primary organization. Organizations participate in development of open source for many reasons.
- One of the well-known open source communities maintains the Linux kernel and is supported by the Linux Foundation

Overview

- We have considered Chapter #1 where we have introduced:
  - Information is a key resource
  - Integration of traditional systems with new technologies
  - Roles and qualities of the systems analyst
  - The systems development life cycle
  - CASE tools
  - Agile systems development
  - Object-oriented systems development
  - Open source systems
- In the following tutorial sessions we will address the topics introduced in Chapter #1

Case Studies

MAC APPEAL

- At home and in our visits to university campuses and businesses around the world, we've noticed that students and organizations are increasingly showing an interest in Macs.
    - Therefore, we thought it would add a little bit of interest to show some of the Mac options available to a systems designer.
- Today, about one out of seven personal computers purchased in the United States is a Mac.
    - Macs are quality Intel-based machines that run under a competent operating system and can also run Windows, so in effect, everything that can be done on a PC can be done on a Mac.
    - One way to run Windows is to boot directly into Windows (once it's installed); another is to use virtualization, using software such as Fusion by VMware, which is shown in Figure 1.MAC.
- Adopters of Macs have cited many reasons for using Macs including:
    - Better security built into the Mac operating system
    - Intelligent backup using the built-in Time Machine
    - The multitude of applications already included
    - The reliability of setup and networking
    - The ability to sync Macs with other Macs and iPhones.
- The most compelling reason may be the design of the Apple products and computers