

Module 1

This is a single, concatenated file, suitable for printing or saving as a PDF for offline viewing. Please note that some animations or images may not work.

It is recommended that you prioritize the readings: studying the “primary” ones first and then looking at as many of the “secondary” ones as you can. Unless otherwise noted, readings are from Systems Analysis and Design: An Object-Oriented Approach with UML, by Dennis, Wixom, & Tegarden (6th Edition, 2021). In the readings listed below, you are given a page range for the reading, but you are only required to read the subsections that are itemized. If no subsections are mentioned, you are required to read the entire page range.

Module 1 Study Guide and Deliverables

Readings:

Primary Reading for Module 1:

The following readings should be completed after reading the module parts that they pertain to.

- Pages 1-4: Introduction, Typical Systems Analyst Roles and Skills
- Pages 4-6: The Systems Development Lifecycle
- Pages 43-44: Project Identification
- Pages 53-55: Project Selection

Secondary Reading for Module 1

The following readings are not required, however they provide additional depth and examples for concepts in this module.

- Pages 424-427: Ubiquitous Computing and the Internet of Things
- Pages 138-148: Business Process Modeling with Activity Diagrams

Assignments:

- Draft Assignment 1 due Sunday, January 21 at 6:00 am ET
- Assignment 1 due Thursday, January 25 at 6:00 am ET

Assessments:

- Crediting Sources Tutorial Self-Assessment due Thursday, January 25 at 6:00 am ET

Live Classroom:

- Tuesday, January 16 from 8:00-10:00 pm ET - Class Lecture

- Wednesday, January 17 from 8:00-9:00 pm ET - Assignment Preview
- Live Office: Saturday, January 20 from 1:00-2:00 pm ET

Objectives

The specific goals of this week's lectures are to define systems analysis: its goals, its processes, and its participants. Systems analysis is sometimes referred to as system analysis (i.e., with no "s" at the end of system") or as the Analysis Phase within the software development lifecycle.

Learning Objectives

By reading the lectures and completing the assignments this week, you will be able to do the following:

- Understand Systems Analysis
- Recognize various type of Information Systems
- Recognize where projects come from
- Understand the concept of process
- Recognize system-analyst roles and skills in context
- Identify the contents of systems requirements

Learning Topics

- Types of Information Systems
- Systems Analysis
- Process
- Introduction to System-Analysis Methodology
- Participants in Systems Analysis

The Big Picture

What are information systems?

You can't do anything useful until you understand what you want or need to get done. In his essay, "No Silver Bullet—Essence and Accident in Software Engineering," Fred Brooks stated the following, which is no less true today than it was in 1986:

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

First let's understand what a "system" is. A **system** is a group of interrelated components that function together to achieve a desired result (Whitten, 2007). As an example, a course includes components such as lectures, assignments, quizzes, and tests.

An **information system** (IS) is an organization of people, data, processes, and information technology that interact to collect, process, store, and provide as output the information needed to support an organization (Whitten, 2007). An example of an information system is the one the reader is using now. This course runs on the Blackboard Learn learning management system and includes lectures (modules), assignments, assessments, discussions, internal messages, users, etc. The final key point is that information systems automate and support a business's or an individual's process.

Information systems range from the global scale to the very small scale. At the larger end of the scale, international projects are common, involving people at various sites in the U.S. and internationally. At the smaller end of the scale, mobile and wearable computing platforms and devices are being integrated into the daily fabric of life at an increasing rate. Information systems are built not only for business, education, and government but also for individuals to augment their daily lives. Modern developments in hardware and software affect our decisions about what systems to create and use. Computing is continuing to take place everywhere and in everything; it dissolves into the background and is always connected and aware, and thus it becomes ubiquitous. Our text by Dennis, Wixom and Tegarden discusses Ubiquitous Computing and the Internet of Things on pages 424 through 427 and categorizes computing into the following two categories:

- **General Computing**—Refers to smartphones and tablets with integrated components and apps that include but are not limited to cameras, GPS, sensors, media streaming and communication capability. Ubiquitous computing however is limited here as we need to use the specific device.
- **Specialized Computing**—Refers to enhanced everyday objects that can interact with each other. For example, a coffee cup that reminds you that the beverage is getting cold, or a lightbulb that turns off when you fall asleep. Computing is becoming ubiquitous.

The systems-analysis and design approach that we will emphasize in the course concentrates on business and general computing; however, it can be adapted to specialized and ubiquitous computing.

Types of Information Systems

The following is a list of common, business-focused information systems. It is not an exhaustive list; it changes as technology, business, and consumer needs change over time. Often in the enterprise we find systems that incorporate these various systems into a single or integrated solution.

A. Transaction Processing Systems

- Process information about specific business operations

B. Management Information Systems

- Report generalized results of business operations

C. Decision Support Systems and Artificial Intelligence

- Evaluate data to help people decide among alternatives
- In some cases Artificial Intelligence systems will make their own decisions based on a pre-existing goal

D. Big Data and Data Analytics

- Manage a large amount of semi-structured and unstructured data and derive insights from it

E. Executive Information Systems

- Report strategic information specific to senior managers

F. Expert Systems

- Use techniques of artificial intelligence to capture specialized expertise

G. Communications and Collaboration Systems

- Facilitate communication between users

H. Office Automation Systems

- Replace human actions and improve workflow

I. Enterprise Systems

- Manage organization-wide system

Increasingly, systems are built to connect and automate our lives, introducing the use of artificial intelligence; below are a few examples:

J. Social Networking & Media Services

- Connect people and allow them to share interests, opinions and ideas through various mediums

K. Content Streaming Services

- Distribute media content

L. Smart Home and Vehicular Automation Systems

- Automate and control home and vehicle systems

This categorization is liable to change, as many modern systems exhibit a combination of the characteristics of these types. For example, Blackboard Learn, which is a learning management system, has components of an

office automation system (i.e. uploading assignments and taking quizzes), as well as of a communication and collaboration system (i.e. discussion boards and internal messages).

As extra reading, review the Appendix A at the end of this module for additional details on many of these information systems.

Test Yourself 1.1

A system that outputs a report of how many students are enrolled in CS682, CS684, CS669 and CS779 for Semester 2 is an example of what type of system?

Management Information System (MIS)

This is correct.

Decision Support System (DSS)

This is incorrect.

Executive Information System (EIS)

This is incorrect.

MIS is the most appropriate choice here as the above system outputs a report summary for specific courses within a specific time frame, making this targeted towards middle-management. Statistics and analysis are not performed by the system; thus it is not a DSS. Since this is more of an operational rather than a strategic report, this is not an Executive Information System.

Test Yourself 1.2

A system that displays student enrollment trends for a university for the last three years is an example of what type of system?

Management Information System (MIS)

This is incorrect.

Decision Support System (DSS)

This is incorrect.

Executive Information System (EIS)

This is correct.

EIS is the most appropriate choice here as the above system outputs a report summary of enrollment trends for an entire university, making this targeted towards executives who will use this to make

strategic decisions. Statistics and analysis are not performed by the system; thus it is not a DSS. Since this is not considered an operational output, this is not an MIS.

Systems Analysis

Systems Analysis is concerned with satisfying business goals with information technology. A **systems analyst** is a key member of a multidisciplinary team assembled to identify ways information technology can be used to realize opportunities and add business value. Systems Analysis is performed to support a business initiative (such as “We want a system that will turn off lights in our house with voice commands”) or to improve a process (such as, “We want to improve the speech recognition of the voice-controlled intelligent personal assistant”).

System analysis is defined as the study of a business problem domain to recommend improvements and specify the business requirements and priorities for the solution that information technology can satisfy (Whitten, 2007).

Project Identification

Where do Systems Development projects come from? Projects are initiated to fulfill a business need, to realize opportunities and to align with company strategy and directives. The following categories are a sample of influences for new projects, though this is not an exhaustive list:

- Business need: respond to emerging technology and market trends in order to decrease cost or to increase profit.
- Support business operations.
- Improve process efficiency and add additional features for usability.
- Improve system performance or update technology of existing systems.
- Introduce additional control and security to comply with business and government regulations.

Organizations have various ways of permitting projects to be proposed and selected, but most are initiated by existing system users and owners, customers, a leadership team, project owners, or sponsors. Organizations may have a formal system-request and feasibility-analysis process for deciding whether a project should be undertaken or placed in a backlog to be considered at a later time when the need and/or resources are more appropriate. Trade-offs such as schedule, funding, staffing and return on investment (ROI) are considered.

The decision to build versus buy is also considered here. For example, our course is organized within a Learning Management System. As an organization, we can review existing Learning Management Systems and determine

whether they will meet our needs or if we should build a brand-new system. Often, commercial off-the-shelf systems (COTS) do not perfectly align with business needs and processes. Internal process adjustments have to be made when implementing a COTS type of system, and change is not easy to introduce. However, if the business problem is unique enough and the existing COTS systems do not meet this need, then an organization may choose to build the system, which is the focus of our course.

Build vs. Buy Decision

To Buy: Is there a potential off-the-shelf system (COTS) close enough to match our enterprise needs?

- Is the system flexible enough to adopt our processes?
- Can we adapt and shift our processes to this system?
- Is the provider of the system stable? (We don't want to be forced into changing the system outside of our control.)
- Is there a single system that can perform all the functionality needed, or do we need multiple specialized and integrated systems?

To Build: Should we build a system that is customized to our enterprise needs and has the potential to evolve with our business?

- Do we have enough resources to build and maintain a system that's customized to our business?
- Are we prepared if the resources who created the original system are no longer available?

No matter the decision, once the system is acquired, there are additional tasks to complete, such as the following:

- Configure the new system
- Convert data from legacy systems
- Train users
- Manage change and transition to the new system
- Potentially maintain legacy systems that were replaced

Process

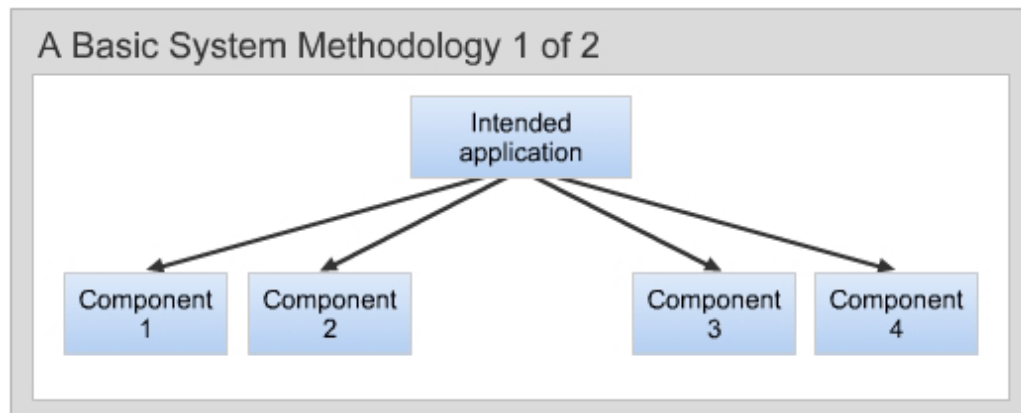
During every software process, we develop an understanding of and write down requirements. We also develop designs that accommodate the requirements. Similarly, we write code that implements our designs and satisfies the requirements. We don't cook a meal without first deciding on what we should make, finding the recipe to prepare the meal, determining and assembling the ingredients needed, and, finally, cooking the meal. We can

call this principle *orderliness*. The principle of orderliness is a common engineering guideline. We will explore phases of IT process in depth later in this course; for now, the following is a basic outline:

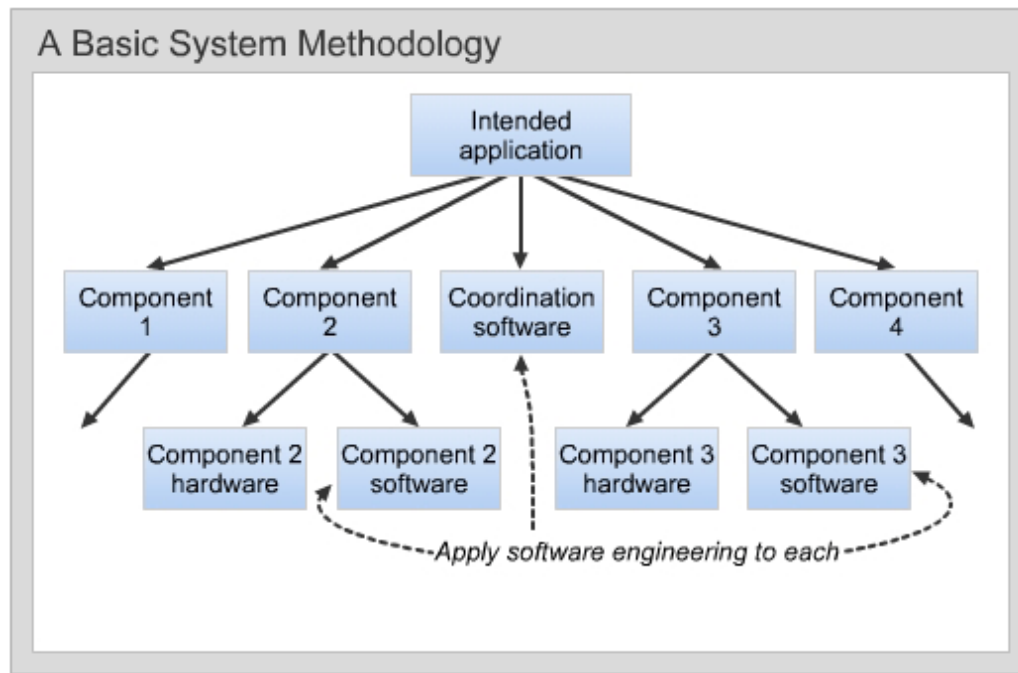
Main Phases of Software Process

- Requirements Analysis
 - Specification of application
- Design
 - Specification, Design of parts
- Implementation
 - Coding, Installation, Integration
- Testing
 - Does the application do what it's supposed to
- Maintenance
 - Defect repair, minor modifications/extensions

When we take on a project, we decompose it into smaller manageable parts. We might begin by decomposing jobs into software and hardware components, and then decompose them further into manageable parts and tasks. It is equally logical to decompose the job into components, *each* of which consists of hardware and software. A typical methodology is shown next. First, the intended application is decomposed into parts. The parts typically consist of a computer and the software running on it. This is shown in the following diagram:



Each component is then divided into its software part and its hardware part. Sometimes, separate software is required to coordinate the whole system. Each software part can be considered a separate project, but with dependencies on the other parts.



Introduction to Systems Analysis Methodology - Part 1

Answering the question of how to do systems analysis is the goal of this course. However, to give you a feel for what's involved, we will preview a common and simple set of steps for performing systems analysis.

1. Write a Mission Statement/System Overview
2. Develop Functional System Requirements (**WHAT** the system is meant to do)
3. Write System Level Use Cases (typical interactions between the system and users)
4. Create an Activity Diagram for the system-level use case (visualize the interaction between user and system)
5. Specify System Level Quality Requirements (measurable, acceptable quality of the system)
6. Write System Level Constraints (**HOW** the system is implemented, conditions that should not be violated)

Introduction to Systems Analysis Methodology - Part 2

Example of System Analysis Methodology

This example provides a systems analysis task sequence using the basic methodology. This is a rough, first cut at the process. A full system analysis contains far more detail. We will use an example to illustrate the rough process.

The *quickMessage* Example

Step 1: Mission Statement/System Overview

The **mission statement** is a high-level scope of the system that outlines the purpose of the application (perhaps in the context of its intended market), as well as its key users.

quickMessage is a browser-based chat platform for enterprise users. *quickMessage* allows users to review and manage their contact list and message contacts within the organization. *quickMessage* uses an intuitive and minimalist design interface and integrates with the organization's Active Directory listing for prepopulated contact lists.

A clearly stated mission statement outlining the context of the system should be about a paragraph in length, no more than 4 sentences. When we begin to think about systems, it is hard to envision every requirement.

Step 2: Functional Requirements

Functional requirements describe **WHAT** the system should do, consistent with its mission. Requirements may have sub-requirement sections.

1. *quickMessage* shall allow a *Sender* to send a message to another user or a group of users within the organization.
2. *quickMessage* contact list:
 - a. *quickMessage* shall allow *Sender* to search for another user
 - b. *quickMessage* shall allow *Sender* to review a directory contact list by department "groups."
 - c. *quickMessage* shall allow *Sender* to create a favorite contact list.
3. *quickMessage* shall allow *Sender* to set status, such as available or busy.
4.

Requirements do not describe **how** the system is implemented—this is a task that system designers will perform, and it is outlined in non-functional quality and constraints. Note that our system may have various users and components, so that functional requirements are often organized around functional areas and/or users. Consistency is important when writing out requirements; for example, the *quickMessage* user is referenced as "Sender" at all times.

Step 3: System Level Use Cases

This refers to typical anticipated usage as a sequence of user actions and the system's responses.

| | |
|---------------------|--|
| Actor: | Sender |
| Description: | This use case shows steps of how <i>quickMessage</i> shall allow a <i>Sender</i> to send a message to another user or a group of users |

| | within the organization. | |
|---------------------------|---|---|
| Step # | Actor | System |
| 1 | Sender opens the <i>quickMessage</i> . | <i>quickMessage</i> displays contact list. |
| 2 | Sender selects a contact. | <i>quickMessage</i> displays entered text by Sender. |
| 3 | Sender enters text into the chat window and clicks send. | <i>quickMessage</i> displays entered text by Sender. |
| 4 | | If contact replied, <i>quickMessage</i> displays reply text from the contact. |
| Alternate Courses: | (System Alt 4) If contact did not reply, display nothing. | |

System level use cases describe the user actions and the corresponding responses of the system. Use cases are story-like scenarios of specific usages of a proposed system. They are a means for specifying the desired system behavior which will be discussed later in detail. A system level use case encompasses a key scenario for the application. Secondary functionality might not be covered. As a use case is developed, additional functional requirements may be uncovered. Developing use cases and functional requirements is often an iterative process.





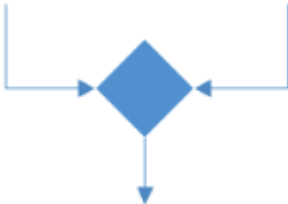
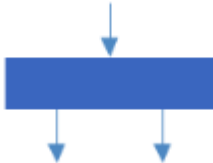
In the use case above is outlined using a template which organizes its various elements such as **Actor**, **Description**, **Steps** (Actor and System) and finally **Alternate Courses**. There are other ways to organize a use case, and we will look at a more detailed approach later in the course. For the Alternate Courses, consider the most important alternate courses which may occur, and keep the branching minimal focusing just on the immediate alternate step and not turning it into another series of steps.

Step 4: Activity Diagrams

Flowcharts are among the oldest graphical methods for depicting activities or actions and their relationships. The Unified Modeling Language (UML) which we will explore in more detail later in the courses uses an extended form of flowcharts called Activity Diagrams. Modeling can be used to visually understand and make sense of the requirements adding another perspective which may not be apparent from a textual representation and can be used by an analyst to communicate their understanding of requirements to the end-user. We will explore modeling later in this course.

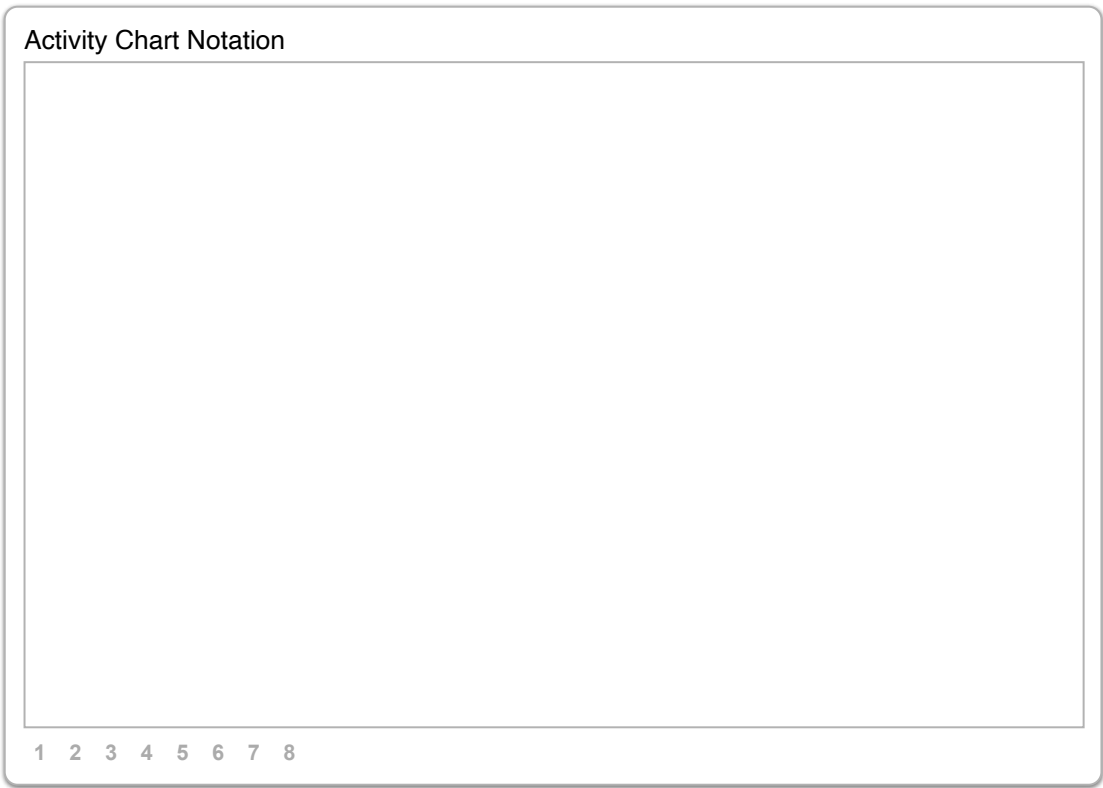
Activity Diagram Components

| Term and Definition | Symbol |
|---------------------|--------|
|---------------------|--------|

| | |
|--|---|
| Activity <ul style="list-style-type: none"> A step by the end user or system |  |
| Control flow <ul style="list-style-type: none"> Shows the sequence of execution |  |
| Initial Node <ul style="list-style-type: none"> Beginning of the activities |  |
| Final Activity Node <ul style="list-style-type: none"> End of partitivities |  |
| Decision Node <ul style="list-style-type: none"> Logical condition to depict the flow of activities based on decision Labeled with decision criteria (i.e. True) to continue down a specific path | |
| Merge Node <ul style="list-style-type: none"> Bring back together different decision paths that where created using a decision node |  |
| Fork Node <ul style="list-style-type: none"> Split flow into parallel or concurrent flows or activities |  |
| Join Node <ul style="list-style-type: none"> Bring back flow from parallel or concurrent flows or activities | |
| Swimlanes | |

| | |
|---|--|
| <ul style="list-style-type: none">Organize the activity diagram into columns or rows based on who is responsible in performing the activity | |
| Adapted and expanded from Whitten and Dennis, A., Wixom, B. H., Tegarden, 2021 (pp. 140) | |

The following provides another example of Activity Diagram Notation



Let’s depict the previously outlined system level use case for *quickMessage* with its activity diagram, side by side.

| | | |
|--------------|---|--|
| Actor: | Sender | |
| Description: | This use case shows steps of how <i>quickMessage</i> shall allow a <i>Sender</i> to send a message to another user or a group of users within the organization. | |
| Step # | Actor | System |
| 1 | Sender opens the <i>quickMessage</i> . | <i>quickMessage</i> displays contact list. |
| 2 | Sender selects a contact. | <i>quickMessage</i> displays a chat window |

| | | |
|---------------------------|---|---|
| | | with the selected contact. |
| 3 | Sender enters text into the chat window and clicks send. | <i>quickMessage</i> displays entered text by Sender. |
| 4 | | If contact replied, <i>quickMessage</i> displays reply text from the contact. |
| Alternate Courses: | (System Alt 4) If contact did not reply, display nothing. | |

Step 5: System Level Quality Requirements

These describe measurable and acceptable non-functional quality for the system.

1. *quickMessage* messages shall take no more than one second to display in the chat window 95% of the time.
2. *quickMessage* functional areas shall be no more than 2 clicks away from the main page 95% of the time.
3.

System level quality requirements are non-functional requirements that outline the quality of the system within an acceptable range. For example, for this application, maximum acceptable display rate is important, and almost always messages need to display within one second. We will explore non-functional requirements in depth later but the idea is that while important, they are not the reason that the system is built. For example, “Displaying a text message” may be a reason an application is built but “displaying messages within one second,” even though it may be important, is not.

Step 6: System Level Constraints

These are non-functional constraints on how the system is to be implemented that the designer should not violate.

1. *quickMessage* app window shall have a minimum size of 250 pixels wide by 400 high.
2. *quickMessage* app shall be implemented using responsive design utilizing html5.
3. *quickMessage* app shall be compatible with Chrome v51 and above, Firefox v48 and above.
4. *quickMessage* app shall integrate with company Active Directory infrastructure for contact listing.
5.

The goal of the constraints is to place a bound on the number of possible implementations—for example, the implementation language of the system, or a database to be used, or what platforms it should be implemented on specifically. Research can help determine these constraints. In the example above, the decision to use specific versions of Chrome and Firefox over Internet Explorer is based on timely real-world browser usage matrices.

Notice how the mission statement, functional requirements, use cases and non-functional quality requirements have consistency and that the requirements complement each other.

Introduction to Systems Analysis Methodology - Part 3

Organizations and methodologies often use different terminology for requirements. The mission statement might be called a mission, vision, or systems context statement. Functional requirements and use cases might be lumped together into “business requirements” or organized as user stories. Constraints and quality requirements are often lumped with non-functional requirements. Requirements are typically developed in stages, with increasing levels of detail, and these stages might have different names and may focus on different audiences, such as end-users and developers.

The main points to take away here are:

1. It is important to **capture the requirement**
2. **Organize requirements** clearly and appropriately.
3. **Maintain consistency** within the requirements.

Appendices C & D at the end of this module contain additional examples.

Introduction to Systems Analysis Methodology - Part 4

Test Yourself 1.3

Select which of the following are clearly written functional requirements for an Online Course System such as Blackboard: (Check all that are true.)

Online Course System shall have an Internal Messaging feature that allows users to select recipients, read and write messages, including sending attachments.

This choice should be selected. Good choice, a clearly written functional requirement outlining WHAT the system does.

Online Course System Internal Messaging system shall notify user through email that an internal message is available.

This choice should be selected. Good choice, a clearly written functional requirement outlining WHAT the system does.

Online Course System shall have a chat feature, allowing users to type in text and submit it to other chat participants by hitting the Enter key on a keyboard or Send button [displayed] on the screen.

This choice should not be selected. This is close, however this requirement has a bit too much detail, verging on being a use case sequence. Focus on the WHAT: Online Course System shall have a chat feature, allowing users to type in text and submit it to other chat participants.

Online Course System Internal Messaging interface shall be user-friendly.

This choice should not be selected. This requirement deals with HOW the design of the system is implemented, it is non-functional. Here is a clearly defined version of the above requirement as a non-functional constraint. HOW: Online Course System buttons shall have a single word descriptive label to explain functionality.

Online Course System Internal Messaging shall be compatible with all modern web browsers.

This choice should not be selected. This requirement deals with the HOW, an implementation constraint, so it is non-functional. Here is a clearly defined version of the above requirement as a non-functional constraint focusing on the HOW: Online Course System Internal Messaging shall be compatible with Chrome version 57 and above, Internet Explorer 11 and above, Firefox version 53 and above. A good way to support this constraint is through research. Based on gs.StatCounter.com as of May 2017, Chrome versions 57 and above accounted for just below 50% of all browser use. This provides valid insight as to why specific browser versions should be supported by the application.

Participants in System Analysis

There are many different participants within systems analysis, as well as within all phases of software development. Systems analysts play the key business and technical role that makes projects happen. They are sometimes project managers as well. They have to understand the context of the work, why it is needed or wanted, who wants it, and what it is intended to replace. Systems analysts are responsible for working with stakeholders and builders to ensure that stakeholder requirements are identified correctly and are met by the

system. It is crucial for systems analysts to understand the business process that they will be working to implement or improve through technology, in other words, systems analysts need to place themselves in the shoes of the person who will use the system. They must also learn what the constraints are on the project.

Systems Analyst Roles

The following are typical participants with System Analyst roles. System builders and stakeholders such as project sponsors and end users also participate in the process at various stages (adapted from Dennis, Wixom and Tegarden, 2021, pp. 2-4):

Business Analyst

- Responsible for identifying business value (i.e., to improve and/or support a business process).
- Represents interests of the organization, project sponsors and end users.

Systems Analyst

- Focuses on information technology (i.e., generates ideas and suggestions on how technology can improve and support business processes).
- Responsible for identifying system requirements and design and ensuring that these match business goals.
- Represents interests of system builders and stakeholders.

Infrastructure Analyst

- Responsible for technical infrastructure and design including how the system will be implemented within the infrastructure (i.e., determines how hardware, software, network, and database systems will be implemented).
- Represents interests of system builders and those who will maintain the system past implementation.

Project Manager

- Responsible for planning the project and ensuring that it is completed on time and within allocated resources (i.e., staff, funding) and that it meets the requirements specified.
- Represents interests of system builders and stakeholders.

Systems Analyst Skills

Systems analysts and system builders (typically programmers) require a wide range of skills, summarized as follows (adapted and expanded from Dennis, Wixom and Tegarden, 2021, pp. 2-4):

- Business

- Analytical
- Interpersonal
- Communication
- Management
- Technical
- Political
- Ethical

Systems Analyst's Tasks

The following summarizes the systems analyst's tasks.

- Understand and express context of job
 - Why the job is needed, what the system interfaces with
- Gather requirements
 - What the system is supposed to do for its stakeholders
- Document constraints (not an exhaustive list)
 - Existing infrastructure and databases
 - Programming languages
 - Graphical user interface specification
 - Integration points
 - Legacy applications
- Perform high level physical design
- Perform high level logical design
- Work with business analysts, designers, system builders and stakeholders

System Builders

System Builders are the key participants in software development. Whitten and Bentley have classified system builders as shown below (adapted from Whitten & Bentley, 2007, p. 10):

- Application Programmers
 - Write code that implements required system capabilities using high level programming languages or scripting languages
- System Programmers

- Handle operating system tasks that allow several programs to share computer resources (i.e., memory, disk space, or communications lines)
- Database Architects and Programmers
 - Create and implement database designs and database queries
- Database Administrators
 - Ensure security, integrity, and performance of databases
- Network and infrastructure architects and administrators, security administrators
 - Design and implement the network, servers, and storage systems
- Webmasters & content management specialists
 - Design and implement websites and/or manage website content

In addition to system builders the following additional personnel are typically involved in information technology projects.

- Technical liaison working to bridge the gap between system builders and systems users, especially if the project involves international teams.
- Change management personnel provide documentation, training and assist with transition to the new or updated system.
- Quality assurance personnel responsible for testing the system are also involved in software and systems projects.

Boston University Metropolitan College