1ST Class – Tuesday Evening, September 8, 2015

Text: Systems Analysis & Design, In a Changing World, 5th Edition, John Satzinger, Robert Jackson, Stephen Burd.

Instructor: Dr. LaDon Jones – ljones@coleman.edu

(Group 2) Team Name: Krazy Insane Flying Monkey Space Rangers

Team Members: Michael Fetick, Soohoon Lee, Kuldeepkumar Patel , Rahul Dev Kumar

Team Norms:

1. I keep communication channel open

2. I respect other member’s opinions/views

3. I will complete agreed tasks

4. I will do my best effort

Chapter 1 Reading: The World of the Information Systems Analyst

The key to successful **system development** is thorough systems analysis and design to understand what the business requires from the information system (page 4).

**System analysis** means understanding and specifying in detail what the information system should accomplish.

**System design** means understanding and specifying in detail how the components of the information system should be physically implemented.

A **system analyst** is a business professional who uses techniques of system analysis and design to develop information systems. They require extensive knowledge about technology, business, and people skills.

System analysis and design is a practical field grounded in time tested and rapidly evolving knowledge and techniques. It is oriented towards the system analyst but it also provides a good foundation for others who deal with business problems that could be solved with the help of an information system.

Information systems are developed to solve problems for organizations and a system analyst is often thought of as a **problem solver** rather than a programmer.

The analyst’s approach to problem solving (figure 1-1, page 5):

1. Research and understand the problem. Learn everything about it, who is involved, what business processes are included, what other systems would be affected by solving the problem.
2. Verify that the benefits of solving the problem outweigh the costs. Confirm this for management. If feasible, then move on.
3. Define the requirements for solving the problem. What specific objectives must be satisfied? What data is needed to be stored and used? What processing must be done to the data? What outputs must be produced? All this is what **needs** to be done and is defined first, not how yet.
4. Develop a set of possible solutions (alternatives). Each solution is carefully thought out and shows **how** it is done.
5. Decide which solution is best, and make a recommendation, in consultation with management. There may be more than one good solution. The best is the solution with the fewest risks and most benefits. Are they cost effective? Consistent with the corporate strategic plan? Will it integrate with other existing and planned systems? Does it use technology that fits the strategic direction that management has defined? Will end users be receptive to it?
6. Define the details of the chosen solution. Create a design specification (blueprint) for how the new system will work. It will include databases, user interfaces, networks, operating procedures, conversion plans, and program modules.
7. Implement the solution.
8. Monitor to make sure that you obtain the desired results.

A **system** is a collection of interrelated components that function together to achieve some outcome.

An **information system** is a collection of interrelated components that collect, process, store, and provide as output the information needed to complete a business task.

A **subsystem** is a system that is part of another system. A system can be divided, or decomposed, into subsystems. This approach to dividing a system into components is referred to as **functional decomposition**.

For example, a Customer Relationship Management (**CRM**) system may have thee following subsystems:

* Sales management system
* Account management system
* Product information system
* Technical support system

Components of a system can be listed as the parts that **interact**, e.g. hardware, software, inputs, outputs, data, people, and procedures.

Inputs and outputs must cross the **system boundary** that is between the system and its environment.

Another type of boundary is the **automation boundary** that separates the automated part that is work done by the computer and the manual part that is work done by people (page 8).

**Enterprise resource planning (ERP)** is a process in which an organization commits to using an integrated set of software packages for key information systems. An ERP system incorporates most or all of the following system types:

* Customer Relationship Management (CRM) system (interacts with customers). Incorporates processes that support marketing, sales, and service operations.
* Supply Chain Management (SCM) system (interacts with suppliers). Incorporates processes that integrate product development, product acquisition, manufacturing, and inventory management.
* Human Resource Management (HRM) system (interacts with employees). Incorporates processes for concerns of payroll, health insurance, pensions, hiring, and training.
* Accounting and Financial Management (AFM) system (interacts with investors). Records accounting information to produce financial statements.

(HRM and AFM systems are partly governed by external regulations and must frequently interact with regulatory authorities in areas of taxes, public financial markets, occupational health and safety, industry standards and practices.)

* Collaboration Support System (CSS). It enables geographically distributed personnel to collaborate on projects and tasks. It includes various technologies such as voice, communications, video conferencing, project management, scheduling tools, and Wiki documents.
* Knowledge Management System (KMS). Incorporates processes that store and provide access to documents from all parts of the organization, i.e. policies, procedures, technical and business data. It helps ensure a continuity of knowledge despite changes in personnel assignments.
* Business Intelligence System (BIS). Incorporates processes that support strategic planning and executive decision making. It enables users to organize internal and external data about customers, suppliers, competitors, and economic conditions for use in statistical analysis, simulations, and other forms of planning.
* Manufacturing Management System (MMS). Incorporates processes that control internal production processes that turn raw materials into finished goods.

Data sharing among all the systems is made possible by databases. A database is a centrally managed collection of data that can store large amounts of information and make it accessible to many users and systems at the same time (page 10).

Many companies use ERP systems from software vendors such as SAP, Oracle, and IBM that offer comprehensive packages in specific industries.

Chapter 2 Reading: Approaches to System Development

The **system development life cycle (SDLC)** usually has both a predictive approach and an adaptive approach.

The **predictive approach** assumes the project can be planned and organized in advance and the new information system can be developed according to the plan. Requirements are understood and well defined. There is low technical risk (page 39).

The **adaptive approach** is more flexible, assuming that the project cannot be planned out completely in advance but must be modified as it progresses. Requirements and needs are uncertain. There is high technical risk (page 39).

Traditional, predictive approach:

**Project planning** Phase – Activities: plan, organize, and schedule the project. Identify the scope of the new system. Ensure that the project is feasible. Develop a schedule, a resource plan, and a budget

**Analysis** Phase – Activities: Focus on understanding the business problem that needs to be solved and defining the business requirements.

**Design** Phase – Activities: use the defined business requirements to develop the program structure and algorithms for the new system.

**Implementation** Phase – Build the new system which includes programming, testing, and installation.

**Support** Phase – Activities needed after the system is deployed: upgrade and maintain the system.

The **waterfall model** is the most predictive approach (figure 2-4, page 41). It is rare because it requires rigid planning and final decision making at each step.

Newer adaptive approaches:

The **spiral model** is the first adaptive approach and is very popular because activities – including plans and models – are adjusted as the project progresses. The life cycle is shown as a spiral (figure 2-6, page 43) that starts in the center with an initial prototype, working its way outward through **iterations** (any number necessary), until the project is complete. The purpose of each iteration is to gather just enough information to begin developing the next prototype (page 42).

A **prototype** is a preliminary working model of a larger system. For each prototype, the development process follows a sequential path through analysis, design, construction, testing, integration with components of the previous prototype, and planning for the next prototype.

A key concept of the spiral approach is the focus on **risk**. Risk factors must be identified and studied. The greatest risks must be mitigated at the earliest (lowest) iteration.

Some examples that pose great risk are:

* Technological feasibility of new technology – Prototype to prove the technology will work as planned
* User acceptance of change – Prototype showing users how their lives are enriched by the new system.
* Risks associated with system requirements – Prototype to address issues.

The **iterative approach** divides a very large, complex problem into smaller, more easily managed problems. Each small problem is solved until they are all solved. Each iteration involves analysis, design, and implementation; then more iterations involve more analysis, more design, and more implementation.

You can organize iterations in several ways:

* Define key functions in the first iteration. After they are completed, define less crucial, system functions. Finally, define optional (nice to have) functions.
* Another way, focus on ‘dependency’ and do one subsystem at a time. The first iteration would contain the core functions and data that other subsystems depend on, and so forth.
* Sometimes, focus is on the complexity or risk because of the probability of change that affects other subsystems.
* Other times, the simplest parts are handled first to get as much of the system finished as quickly as possible.

**Incremental development** is a related approach. You complete parts of the system in a few iterations and then put the system into operation for the users. The users get the system as soon as possible and the subsequent iterations are integrated into the deployed system. Usually they’re **integrated seamlessly** and without the users’ knowledge.

Today, much of system development uses varying degrees of iteration and the **object-oriented approach** is always highly iterative.

Models

A **model** is a representation of an important aspect of the real world. Sometimes the term ‘abstraction’ is used because we abstract (separate out) an aspect of particular importance to us. The models used in the system development of an information system can seem much less tangible. Some are: the inputs, outputs, processes, data, objects, object interactions, events, locations, networks, devices, etc. Most are graphical models and are called diagrams and charts. One kind is a project planning model called a Gantt chart, which represents the system development project itself, highlighting its tasks and task completion dates. Some models of system components:

* Flowchart
* Data flow diagram (DFD)
* Entity-Relationship Diagram (ERD)
* Structure chart
* Use case diagram
* Class diagram
* Sequence diagram

Some models used for managing the development process:

* Gantt chart
* Organizational hierarchy chart
* Financial analysis models – NPV, ROI

Project Planning

* **Define the problem** (and the scope of the required solution). Identify the major uses of the new system.
* **Produce the project schedule.** List tasks, activities, and required staff. Identify milestones and control procedures. Plan to acquire the necessary human resources.
* **Confirm project feasibility.** Investigate economic, organizational, technical, resource, and schedule feasibilities.
* **Staff the project.** Acquire the necessary human resources.
* **Launch the project**. Reviewed with management and entails allocating funds, assigning project members, and obtaining other necessary resources, i.e. office equipment, lab time, development tools. An official announcement often commences the project launch.

Analysis Activities

* **Gather information.** Meet with users to learn as much as possible about the **problem domain**. Observe users working, interview them and ask questions about their work. Read existing documentation and review existing automated systems. Consult interested parties, i.e. middle management, senior executives, and sometimes external customers.
* **Define system requirements.** Develop an overall understanding of what is needed.
* **Build prototypes for discovery of requirements.** Users can review them and often can express their needs easier after looking at the prototype of an alternative.
* **Prioritize requirements.** The most important needs must be identified because there is always more requests than budget.
* **Generate and evaluate alternatives.** Usually there are implementation alternatives: building the system in-house, buying a software package, or contracting a third-party to develop a new system.
* **Review recommendations with management.** This recap the results of the analysis phase.

Design Activities (High Level)

* **Design and integrate the network.** Computer equipment, network, and operating systems.
* **Design the application architecture.** Provides the processing functions for the business requirements.
* **Design the user interfaces.** During the analysis, prototyping may have defined some elements that are combined here into forms, reports, screens, and sequences of interactions.
* **Design the system interfaces.** These are the precisely defined communication links between program modules.
* **Design and integrate the database.** Diagrams of the data storage requirements and may involve access to databases of other systems.
* **Prototype for design details.** Verify the correctness or the workability of the design.
* **Design and integrate the system controls.** Needed to protect the integrity of the database and the application program. These are integrated into the system as it is being designed.

Implementation Activities

* **Construct software components.** Conventionally, write computer programs using a language such as C#, Java, or Visual Basic. Other languages can used to make programs that are scripts that run in web-browsers.
* **Verify and test.** Test-driven software development is very popular. Unit testing ensures each component works. System testing ensures the system is what the user wanted.
* **Convert data.** Often, the data from an old system needs to be converted to the data format required by the new system.
* **Train users and document the system.** A critical activity so the users are able to use the new information system as soon as possible.
* **Install the system.** The new equipment must be in place and functioning, the software must be installed and working, and the database must be populated and available. Frequently, this installation is done in many places and must integrate throughout the organization.

Support Activities

* **Maintain the system.** Usually a system support team is assigned to apply patches and updates. These program updates are built to fix bugs and implement minor design changes. Many new hire, program analysts are assigned tasks: change information in a report, add an attribute to a table in a database, or change the design of Windows or a browser form.
* **Enhance the system.** As the system use gains popularity there are usually requests for design enhancements and are delivered with software upgrades. Software upgrades may be needed to address change in government regulations or change to the business environment.
* **Support the users.** A help desk, consisting of knowledgeable technicians, should be available to answer questions quickly and assist users to help increase their productivity. Training users and maintaining documentation is important.

Tools

A **tool** in the context of system development is software support that helps create models or other components required in the project. A word processing program can be used to write program specifications, data descriptions and process descriptions. A drawing program can be used to make diagrams, e.g. MS Visio. A database application may create and manage tables that store data, e.g. MS SQL Server or MYSQL Server. A project management software tool can create a model of the project task and task dependencies to keep track of the development progress, e.g. MS Project or [www.freedcamp.com](http://www.freedcamp.com). Software programmers use **integrated development environment (IDEs)** with many tools to help with programming tasks – smart editors (language specific), context-sensitive help (understands the language syntax), and debugging tools (to tell what is wrong with the code). Visual modeling tools to help create system models and sometimes generate code. Some tools are:

* Project management application
* Drawing/graphics application
* Word processor/text editor
* Visual modeling tool
* Integrated development environment (IDEs)
* Database management application
* Reverse-engineering tool
* Code generator tool

Techniques

A technique in system development is a collection of guidelines that help an analyst compete an activity or task.

* Strategic planning techniques
* Project management techniques
* User interviewing techniques
* Data-modeling techniques
* Relational database design techniques
* Structured analysis techniques
* Structured design techniques
* Structured programming techniques
* Software testing techniques
* Object-oriented analysis and design techniques

The Traditional Approach

Used to develop information systems with structured and modular programming, often called ‘structured system development.’ It has many variations, such as one popular variation, information engineering (IE).

Structured system development (**structured approach**), also called the **structured analysis and design technique (SADT)**:

* Structured analysis techniques
* Structured design techniques
* Structured programming techniques

**Structured programming** has one beginning and one end. Each step consists of one of the following three programming constructs:

* A sequence of program statements
* A decision where one set of statements or another is executed
* A repetition of set of statements (with an entry point and an exit point)

**Top down programming** divides more complex programs into a hierarchy of program modules with the top module ‘calling’ lower modules to do subtasks and then returning.

Structured design is a technique, following guidelines, on how the program is divided by deciding what the modules (set of programs) should do, specifically what each should be, what each should accomplish, and how they are organized into a hierarchy. Program modules should be:

* **Loosely coupled**. Each module is independent and modifications do not affect the others.
* **Highly cohesive**. Each module accomplishes one clear task. This is for clarity and independence.

Modern Structured Analysis

System requirements define what the system must do in great detail, but without committing to one specific technology. The **structured analysis** technique helps define:

* what it needs to do (processing requirements),
* what data it needs to store and use (data requirements),
* what inputs and outputs are needed,
* how the functions work together, as a whole, to accomplish tasks.

The key graphical model of the system requirements used with structured analysis is called the **data flow diagram (DFD)**. It shows inputs, processes, storage, and outputs, and the way they function together. It can identify all the events that will cause a system activity.

A model of the needed data, which is information based on the type of things involved, is called an **entity-relationship diagram (ERD)**. The data entities on the ERD correspond to the data stored on the DFDs.

Weaknesses of the structured approach

* many considered it to be weak because the techniques do not address all the activities of analysis and design
* many desire a more comprehensive and rigorous set of techniques to make it more like an engineering discipline, instead of an art
* many feel the transition did not work well between the structured analysis (DFD) and the structured design (structure chart)
* many think data modeling and the ERD are more important than modeling processes with the DFD
* it makes the central focus of the system on processes rather than data
* many thought the development should begin only after the organization conducts an overall strategic system planning effort. To include this effort leads into ‘Information Engineering.’

**Information engineering** is a refinement to structured development that begins with overall strategic planning to define all the information systems that the organization needs to conduct its business (the **application architecture plan**), which includes (page 58):

* A definition of the business functions and activities that the systems need to support,
* The data entities about which the systems need to store information,
* The technological infrastructure that the organization plans to use to support the information systems.

As the project progresses, the activities and data are refined. At each step, the project team creates models of the processes, the data, and the ways they are integrated. The data types don’t change much but the processes to collect the data change frequently. The information engineering approach focuses more on the data than the structured approach and it includes processing requirements. The processing model of information engineering is the **process dependency diagram**. It is like a DFD but it focuses on processes dependent on other processes, with events triggering the processes.

The information engineering approach provides a more complete life cycle support, through the use of an integrated tool for automation and enforcing rigid use by the analyst. It is credited to James Martin who developed the methodology in the 1980’s.

The **Object-Oriented approach** views an information system as a collection of interacting objects that work together to accomplish tasks. There are no processes or programs; no data entities or files. The system consist of objects. An object is a thing in the computer system that is capable of responding to messages. The Object-Oriented approach began with the development of the Simula programming language in Norway in the 1960s. Simula was used to create computer simulations involving objects such as ships, buoys, tide, etc., to study ship movement. In the 1970s, the Smalltalk language solved problems creating graphical user interfaces (GUI). Later other languages were used: C++, Java, C# which focus on writing definitions of the types of objects needed in the system, and not just for the GUI.

**Object-oriented analysis (OOA)** defines all of the types of objects that do the work in the system. It shows what user interactions, called ‘use cases,’ are required to complete tasks.

**Object-oriented design (OOD)** technique:

* defines all of the additional types of objects necessary to communicate with people and devices in the system,
* shows how the objects interact to complete tasks,
* refines the definition of each object so it can be implemented with a specific language or environment

**Object-oriented programming (OOP)** technique:

* consists of writing statements in a programming language to define what each type of object does.

An object is a ‘type’ of thing and identifying types of objects means classifying things. A classification or “class” represents a collection of similar objects; so object-oriented development uses a **class diagram** to show all the classes of objects in a system. There may be subclasses of a class that “inherit” characteristics from the class above it. The OOD approach yields several benefits, mainly naturalness (people are accustomed to tangible objects) and reuse (other systems use the same objects). Many systems today combine both traditional and object-oriented technology. Some IDEs also combine these technologies in the same tool. Often, information systems have use OOP for the GUI and procedural programming for the rest.

The **Unified Process (UP)** is an object-oriented system development methodology offered by IBM’s Rational Software. Originated by the three proponents of Unified Modeling Language (UML): Grady Booch, James Rumbaugh, and Ivan Jacobson. It uses UML for system models. The UML is a standard modeling notation for the object-oriented approach. But the UP is not a standard OO development methodology. The UP is designed to reinforce the following six “best practices” of system development that are common to many system development methodologies:

* Develop iteratively
* Define and manage system requirements
* Use component architectures
* Create visual models
* Verify quality
* Control Changes

The UP defines four life cycles:

* **Inception** (Defines the scope of the project by specifying use cases. Do a feasibility study),
* **Elaboration** (Several iterations that takes part of the system and define the requirements, design the solution, and implement the solution. The requirements and design are defined by creating use case diagrams, class diagrams, sequence diagrams, state diagrams, and other UML diagrams. Final cost and benefit estimates are completed by the end of this phase.),
* **Construction** (Continue to build the system with additional iterations that include design, implementation, and testing, possibly creating multiple releases of the system.),
* **Transition** (You turn the system over to the end-users and focus on end-users training, installation, and initial support.).

More about UP in Chapter 17.

**Extreme Programming (XP)** is a system development approach that adapts techniques from many sources and adds new ideas. Popularized by Kent Beck, it is sometimes referred to as a “lightweight” system development methodology because it is simple and focuses on making the development process more efficient. The developers begin planning the system by having the users describe ‘user stories,’ like use cases. Users describe their needs and also describe acceptable test to prove it works. The developers plan a series of releases with incremental development. XP is much like other methodologies but it has popular features: It requires continual testing, continuous integration, and heavy user involvement. Two-man teams are required for software coding and it limits work to a 40 hour-week. More about XP in Chapter 17.

**Scrum** is another new adaptive development methodology. The term refers to rugby’s system for getting an out-of-play ball back into play. The basic idea behind Scrum is to respond to a current situation as rapidly and positively as possible. Scrum is good in a dynamic environment when changes are so numerous that projects can bog down and never reach completion; it focuses on the development team and their work. It emphasizes individuals more than processes and describes how teams of developers can work together to build software in a series of short mini-projects. Software is developed incrementally and controls are imposed to focus on the things that can be accomplished.

More about Scrum in Chapter 17.

|  |
| --- |
|  |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | **From :** | Michael Fetick <mfetick-84270@student.coleman.edu> | | **Subject :** | Possible topic | | **To :** | Rahul Dev-Kumar <rdev-kumar-660251681@student.coleman.edu>, kpatel-660252388@student.coleman.edu, slee-660252451@student.coleman.edu | | |  | | --- | | Fri, Sep 18, 2015 01:16 PM | | | | Team,  I watched a video on SCRUM and it looks so easy:  NEW Intro to Agile Scrum in Under 10 Minutes - What is Scrum? Download from: <https://www.youtube.com/watch?v=XU0llRltyFM>  And then another: Introduction to Scrum - 7 Minutes Download from: <https://www.youtube.com/watch?v=9TycLR0TqFA>  It's a possible topic for ... Michael | |

Tools to Support System Development (page 63)

Use automated tools to improve speed and quality of system development work.

Visual modeling tool – It is more than a drawing tool for making graphical models. It contains a database of information about the system, called a **repository**. It includes the models, descriptions, and references that link the various models together. Some are Microsoft Visio, Visible Analyst from Visible Systems Corp, and IBM’s Rational Software Development platform.

2ND Class – Thursday Evening, September 10, 2015

The Keirsey Temperament Sorter (70 question) online at [www.keirsey.com](http://www.keirsey.com)   
I measured myself as: **E**(7)-I(3), **S**(12)-N(8), T(9)-**F**(11), **J**(12)-P(8) = **ESFJ**

E – Extravert, I – Introvert, S – Sensing, N – Intuitive, T – Thinking, F – Feeling,

J – Judging, P – Perceptive (Possibilities)

Guardians: ESTJ, ISTJ, ESFJ, ISFJ (majority)

Idealists: ENFP, INFP, ENFJ, INFJ, ENFP, INFP

Rationalists: ENTP, INTP, ENTJ, INTJ

Artisans: ESTP, ISTP, ESFP, ISFP

DISC – Dominance, Influence, Steady, (perfectionist)

Chapter 3 Reading: The Analyst as a Project Manager

Project Management

Project Success Factors

The Role of Project Manager

Project Management throughout the SDLC

Project Management and the Level of Formality

Agile Software Development

Project Management Knowledge Areas

PMI – PMBOK

1. Project Scope Management
2. Project Time Management
3. Project Cost Management
4. Project Quality Management
5. Project Human Resource Management
6. Project Communications Management
7. Project Risk Management
8. Project Procurement Management
9. Project Integration Management

Project Initiation and Project Planning

Driving forces: 1) respond to an opportunity, 2) resolve a problem, or 3) conform to a directive.

Initiating the Customer Support System (CSS) for RMO

Project Planning Activities

1. Define the problem.
2. Produce the project schedule.
3. Confirm project feasibility.
4. Staff the project.
5. Launch the project.

Defining the Problem

Develop the System Scope Document

1. Define the problem description precisely.
2. Review the business needs to identify the expected business benefits.
3. Identify the expected (high-level) capabilities of the new system. This defines the scope.

Build a preliminary prototype as a ‘proof of concept’

Develop the Context Diagram, which is a data flow diagram (DFD) showing the scope of the system.

Producing the Project Schedule

Developing a Work Breakdown Structure (WBS)

Techniques: Top-down, Bottom-up, Template, or Analogy.

Build a schedule using a Gantt chart. And a PERT/CPM chart (MS Project calls it a Network Diagram).

Develop resource requirements and a staffing plan.

Scheduling the Entire SDLC.

Identifying Project Risks and Confirming Project Feasibility

Assessing the Risks to the Project (Risk Management)

Organizational and Cultural Feasibility

Technological Feasibility

Schedule Feasibility

Resource Feasibility

Economic Feasibility

Development Costs

* Salaries and wages
* Equipment and installation
* Software and licenses
* Consulting fees and payments to third parties
* Training
* Facilities
* Utilities and tools
* Support staff
* Travel and miscellaneous

Sources of Ongoing Costs of Operations (Not including the normal running of the business)

* Connectivity
* Equipment maintenance
* Cost to upgrade software licenses
* Computer operations
* Programming support
* Amortization of equipment
* Training and ongoing assistance (the help desk)
* Supplies

Sources of Benefits

* Reducing staff by automating manual functions or increasing efficiency
* Maintaining constant staff with increasing volumes of work
* Decreasing operating expenses such as shipping charges for ‘emergency shipments’
* Reducing error rates through automated editing and validation
* Achieving quicker processing and turnaround of documents or transactions
* Capturing lost discounts on money management
* Reducing bad accounts or bad credit losses
* Reducing inventory or merchandise losses through tighter controls
* Collecting receivables (account receivables) more rapidly
* Capturing lost income due to ‘stock-outs’ by implementing better inventory management
* Reducing cost of goods through volume discounts and purchases
* Reducing paperwork costs by implementing electronic data interchange (EDI) and other automation

Financial Calculations

* Net Present Value (NPV) (of the system)
* Payback period, Breakeven point
* Return on investment (ROI)

Intangible Benefits

* Increased levels of service
* Increased customer satisfaction
* Survival
* Develop In-house Expertise

Intangible Costs

* Reduced employee morale
* Lost productivity
* Lost customers or sales

Sources of Funds

Completing the Feasibility Analysis

Staffing and launching the Project

* Develop a resource plan for the project
* Identify and request specific technical staff
* Identify and request specific user staff
* Organize the project team into workgroups
* Conduct preliminary training and team-building exercises

Recap of Project Planning for RMO

Chapter 4 Reading: Investigating System Requirements

Analysis Activities in More Detail

Gather information

Define System Requirements

Prioritize Requirements

Prototype for Feasibility and Discovery

Generate and Evaluate Alternatives

Review Recommendations with Management

Functional and Nonfunctional System Requirements

Functional Requirements

“... are the activities (business uses) that the system must perform.”

Nonfunctional Requirements

“... are characteristics of the system, other than activities it must perform or support.”

Technical requirements

Performance requirements

Usability requirements

Reliability requirements

Security requirements

Models and Modeling

The Purpose of Models

Types of Models

Mathematical Models

Descriptive Models

Graphical Models

Overview of Models Used in Analysis and Design

Stakeholders – The Source of System Requirements

Users as Stakeholders

Business Users

Information Users

Management Users

Executive Users

External Users

Client Stakeholders

Technical Stakeholders

Techniques for Information Gathering

“Today, analysts use an accelerated approach by balancing the review of current business functions with the new system requirements.”

Question Themes – “…obtain information to build a logical model of the new business system.”

What Are the Businesses Processes?

How is the Business Processes performed?

What Information is required?

Review Existing Reports, Forms, and Procedure Descriptions

Conduct Interviews and Discussions with Users

Preparing for the Interview

Conducting the Interview

Following up the Interview

Observe and Document Business Processes

Observing

Documenting Workflows with Activity Diagrams

Build Prototypes

Characteristics of effective prototypes are operative, focused, and quick.

Distribute and Collect Questionnaires

Conduct Joint Application Design Sessions

People involved include: the JAD Session Leader, Users, Technical Staff, Project Team Members.

Research Vendor Solutions

Positives of Vendor Solutions:

1. Researching alternative solutions help users generate new ideas for how to better perform their business functions.
2. Some of these solutions are excellent and state of the art. To develop a system otherwise (from scratch), may result an obsolete system.
3. It is often cheaper, quicker, and less risky to buy a solution than to build it.

Danger of Vendor Solutions:

1. If a solution is purchased too early in the process, then the company’s needs may not be fully investigated.

Validating the Requirements

“Structure Walkthrough”

What and When

Who

How – Preparation, Execution, Follow-up

HW1B – Weekly ELP/Thesis Assignment

Submitted assignment HW1B\_Thesis\_Assignment\_MichaelFetick\_84270\_COM620.docx titled, “Risks to Systems Development Projects” on Tuesday, September 15, 2015.

**HW 7C: CPT Assignment**

ALL Students: Students must attend the Career Services’ Job Fair scheduled for Tuesday, September 15th, 2015 from 4:30pm-6:30pm – Hopper Hall… Students unable to attend the job fair due to work requirements or an excused reason must contact Julia Katawazi to receive an alternative assignment.

Email from Julia – Michael, Below is the alternative assignment.

Assignment: Student must complete the Employment Verification Packet. (See attached forms)

The student will also need to write a reflection on how the course content and their job experience has helped them become a better manager or leader. Give at least three examples. Follow APA format, 500 word count minimum, no plagiarism (the use of work “borrowed” from a third party, whether that is another student or a website/book, etc.), and due date—late policy will apply regardless of reason for not turning in the assignment on time. Upload to Webclass on or before due date.

I submitted the CPT assignment HW7C-CPT-Michael Fetick-84270\_COM620.docx, titled “How College and My Job Helped Me Be a Better Leader.” on Monday, September 21, 2015, 3:25 PM; and it is due by Saturday, October 24, 2015, 11:55 PM.

My paper reflects on how the course content of my college course: COM 620 Advanced Systems Design has helped me to become a better leader. It also reflects on how my job experience has helped me become a better leader. I gave four examples of how this happened.

1. Learning leadership by learning about Project Management

Starting within the third chapter, *‘The Analyst as a Project Manager,’* it explains all the elements and aspects of project management and the responsibilities of the project manager (Satzinger, Jackson, & Burd, 2009). It goes on to acclaim the importance of the Program Management Institute (PMI) as a worldwide professional organization. Also how PMI has defined a body of knowledge (BOK) for project management; and the PMBOK is organized into nine project management knowledge areas (Pages 82-82). I looked at the PMI.org website and they sell the PMBOK Guide (book) for $65.95 or $49.50 (member price).

1. Learning leadership from professional affiliation

Attending a local Career Fair is a step in the right direction. I reviewed the job openings of the Career Fair attendees/job-recruiters, by browsing the posted job openings on their websites. Starting with CGI-Federal, I found a job opening for a Recent Graduate/Programmer and applied for it by submitting my resume online. I was curious that CGI-Federal also had a job posted for a Project Manager but it requires a PMI-PMP certification. I looked at the PMI.org website again and I saw the annual membership cost USD $129 plus the local chapter membership fee of USD $35. But also, they offer a student membership cost USD $32 plus the local chapter membership fee of USD $35. Since I am currently a student, I would save a lot of money to join now and put can show the alliance on my resume. So I joined PMI and the San Diego Chapter at pmi-sd.org. After I became a PMI member, I reviewed the member benefits and discovered I can download electronic copies of most of their books and guides for free. So I downloaded all I could, including the PMBOK Guide (619 pages), for free.

1. Learning leadership is from networking

I attended a breakfast/meeting of the San Diego PMI Chapter on September 18th (Lai, 2015). It’s a smart resource to network with and I was able to meet four program managers from the group. At this meeting I learned about Agile Software Development from Mr. Eugene Lai, Principal Program Manager, Computer Sciences Corporation (CSC). Also, I will be attending a PMI San Diego Chapter Dinner Event on September 23rd, with the topic: Networking for Project Manager - Building Magical Relationships, presented by Ms. Abby Kohut, Author; Speaker & Career Consultant (Kohut, 2015).

1. Learning leadership from professional certification

I have decided to pursue the PMI-PMP certification. I want to put a spin on it from the perspective of a Chief Information Security Officer (CISO) and the need of a Security Plan. I don’t see any attention given to this need in the PMBOK Guide standards and I will question that.

3RD Class – Tuesday Evening, September 15, 2015

Best Ever Presentations

1. Concise
2. Visuals
3. Flow and transition
4. Know your audience
5. Energy and confidence
6. Humor
7. Stories
8. Careful of PPT slides

Our Team Leaders

- Develop a communication plan for the item: Hopping Egg

Be creative! Take turns expressing ideas and communicating.

Presentation.

4TH Class – Thursday Evening, September 17, 2015

Watched videos of Ms. Erica Olson:

* What is Strategic Planning, Really?   
  Downloaded from <https://www.youtube.com/watch?v=mLJ34L5UW4E&list=PLXPmT3UOw1bDgiH0kBwQInzlT2L4fAkSz>
* How to Write a Vision Statement that Inspires.   
  Downloaded from <https://www.youtube.com/watch?v=ioY-YSOKBtY>
* Idiot's Guide To Project Management (For Dummies As Well)  
  Downloaded from <https://www.youtube.com/watch?v=r5qFLd1u0XQ>

**What is your vision?**

Our culture can evolve and advance to an optimal level of system development so the world’s natural resources are not squandered.

**What is your mission?**

To contribute my knowledge and talents to advancing project planning for system development.

Chapter 5 Reading: Modeling System Requirements

User Goals, Events, and Use Cases

User Goal Technique, Use case

CRUD Technique (Verify we have everything we need.)

Create, Read/Report, Update and Delete

- Often introduced in database design

- Technique to validate, refine, cross-check use cases

- NOT for primarily identifying use cases

- For each Domain Class, verify there are use cases for CRDU.

Steps:

1. Identify stakeholders and users
2. For each type, determine what they need.
3. Add a new use case
4. Say which application is doing the steps. Carefully name the use case.

Use Case Diagram (Model)

* List of all the use cases
* UML to make the diagram
* Actor
* Automation boundary (system)

Elementary business process (EBP)

Event Decomposition Technique (it also captures temporal and state events)

Types of Events

* External Events, Temporal Events, State Events

Identifying Events

* Events Versus Prior Conditions and Responses
* Sequence of Events: Tracing a Transaction’s Life Cycle
* Technology-Dependent Events and System Controls
* The ‘Perfect Technology Assumption’

Events in the RMO Case

Looking at Each Event and the Resulting Use Case

* Contents of an ‘Event Table’: Event, Trigger, Source, Use Case, Response, Destination

Use Case Descriptions

Actor, scenario

Brief Description

Intermediate Description

Fully Developed Description

* Pre-conditions, post-conditions

Looking at Each Event and the Resulting Use Case

“Things” in the Problem Domain

Types of Things

* Types of things: Tangible things, Roles played, Organizational Units, Devices, Sites/Locations, and Incidents/events/interactions.

Procedure for Developing an Initial List of Things

1. Use the event table and information about each event, and identify all the nouns. Make this list of the “things” based on the nouns, as shown in figure 5-19, on page 179.
2. Use other information, from existing systems, current procedures, and current reports or forms, to add items or categories as needed.s
3. Refine the list and record assumptions or issues to explore.
   1. Include it by asking these questions:

- Is it a unique thing?

- Is it inside the scope?

- Does the system need to remember more than one of these?

* 1. Exclude it by asking these questions:

- Is it a synonym for some other thing I have identified?

- Is it an output of the system, produced from other information I have identified?

- Is it an input that results from recording some other information I have identified?

* 1. Research it by asking these questions:

- Is it likely to be an attribute about some other thing I have identified?

- Is it something I might need if assumptions change?

Relationships Among Things

* Relationships between things apply in two directions.
* A relationship’s cardinality refers to its number of associations, e, g, one-to-one, or one-to-many relationships between things.
* Multiplicity refers to the number of associations in the object-oriented approach, as defined by UML.
* The range of the cardinality expressed as a max and min cardinality are ‘cardinality constraints.’
* Relationships between two things can be: binary, unary, ternary, and n-ary relationships.
* Store information about relationships as well as the things.

Attributes of Things

* Identifier or key, compound attribute.

The Entity-Relationship Diagram

* data entities

Examples of ERD Notation

* “Crow’s feet” symbols

Refinements made to the ERD

* An “associative entity” represents a many-to-many relationships between two other data entities.
* When designing relational databases, refinement is the process of “normalization.”

The RMO’s ERD

The Domain Model Class Diagram

Domain Model Class Diagram Notation

* UML symbol with its name and attributes
* Relationships between objects (other classes) with multiplicity notation

More Complex Issues about Classes of Objects

* Generalization/ Specialization hierarchies
* Inheritance
* Whole-part hierarchies

- Aggregation: Parts can exists separately.

- Composition: is a stronger association; once associated, parts cannot be separated.

Where You Are Headed: Traditional Approach or Object-Oriented Approach?

Events, use cases, and event table.

Things:

Traditional Approach

Entity Relationship Diagram (ERD)

* Context Diagram
* Data Flow Definitions
* DFD Fragments
* Other Traditional Models

Object-Oriented Approach

Class Diagram

* Use Case Diagrams
* System Sequence Diagrams
* Use Case Descriptions
* Activity Diagrams
* State Machine Diagrams

5TH Class – Tuesday Evening, September 22, 2015

Uploaded to the Webclass – Course Documents is the midterm study guide ‘620 Study guide.docx.’

Covered it and identified topics in each chapter and their page numbers.

Break – then team up with object.

Goal: Build a better marble toy (mouse trap)

1 Facilitator / team

1 Group builds the toy.

1 Group create a flowchart entity diagram

1 Group create a descriptive interview

1 Group take this to market.

Prepare for midterm exam (Celebration of learning!)

Preparation for midterm celebration focused on coverage of topics outlined in handout document “Topics for the Celebration of Knowledge” All material is found in chapters 1–5 (and some in chapter 6).

6TH Class – Thursday Evening, September 24, 2015

Midterm Exam (Celebration of learning!)

Took the Midterm exam (Celebration!) and I attained 18/20 correct for a score of 90%.

Chapter 8 Reading: Evaluating Alternatives for Requirements, Environment, and Implementation

Figure 8-6 on page 297 – Implementation Alternatives. Enterprise resource planning solutions.

- Facilities management/ Service Provider solutions occur when the entire system, including development and operation, is contracted to another company.

- Packaged software is shrink-wrapped, off-the-shelf. Turnkey system is a customized package.

Both require some modification to fit the existing environment with a “build” component.

- Enterprise resource planning (ERP) solutions begin with a standard system, but they require substantial integration with a company’s business processes. ERP solutions are integrated so tightly with the entire organization and all its systems that the implementation frequently requires substantial effort. Custom-built systems require substantial programming by either in-house staff or outsourced consultants and programmers.

7TH Class – Tuesday Evening, September 29, 2015

Watched video “CIS 121 – System Development Life Cycle, by Al Zimmermann (25 minutes)

Submitted our team member’s names/student ID # to Noel (TA), [nstorms@coleman.edu](mailto:nstorms@coleman.edu)

Team Key Assignment – Got into our groups

Team Tasks – Detailed Outline:

- 3-5 Articles

- Overview of project

- Roles and responsibilities (Design, Build, Deploy, Support, Analysis)

Reminder: Key Assignment Paper (IC EX 8B) – Due Thursday 6PM, (24 hours)

Reminder: Key Assignment Presentations – Present next Tuesday

Detailed Outline – Paper: “System Analysis of GlowForge SCM System”

* Michael, 100% – Cover Page
* Michael, 90% – TOC
* Introduction, –
* Methodology
* Michael, 100% - m Tools
* Conclusion
* References

Detailed Outline – Presentation: “System Analysis of GlowForge SCM System”

* Cover Page
* TOC
* Introduction
* Methodology
* Tools
* Conclusion
* References

8TH Class – Thursday Evening, October 1, 2015

Key Assignment Presentations

Present.