



POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
COLLEGE OF COMPUTER AND INFORMATION SCIENCES
DEPARTMENT OF COMPUTER SCIENCE | DEPARTMENT OF INFORMATION TECHNOLOGY

Capstone Project 1

INSTRUCTIONAL MATERIAL FOR STUDENTS

Compiled by:
FLORANTE V. ANDRES
MONINA D. BARRETTO
ILUMINADA VIVIEN R. DOMINGO
RACHEL A. NAYRE
JANELLE KYRA A. SAGUM
RIA A. SAGUM

Table of Contents

CHAPTER 1	- 5 -
INTRODUCTION	- 5 -
1.1 Project Context	- 5 -
1.2 Technical Background	- 5 -
1.3 Problem Analysis	- 7 -
1.4 Purpose And Description	- 13 -
1.5. Specific Objectives	- 14 -
1.6. Scope And Limitations	- 14 -
CHAPTER 2	- 16 -
REVIEW OF RELATED LITERATURE AND STUDIES	- 16 -
CHAPTER 3	- 22 -
METHODOLOGY	- 22 -
3.1 Requirements Analysis	- 22 -
3.1.1 Requirements Feature Matrix	- 24 -
3.1.2 Use Case Diagram	- 27 -
3.1.3 Use Case Report	- 31 -
3.2 Design Specifications	- 33 -
3.3 Development Methodology	- 49 -
3.3.1 Process Model	- 49 -
3.3.2 Development Tools	- 53 -
3.4 Test Methodology/Procedures	- 56 -
3.5 System Requirements	- 63 -
3.6 Quality Plan	- 64 -
3.7 Evaluation Plan	- 71 -

Capstone Project 1

Overview:

This course is the first half of two undergraduate capstone project leading to the Information Technology capstone. In this course, the student is expected to define a capstone project topic, conduct an appropriate literature review, and conceptualize and develop a solution the **research** problem taking note of the underlying methods to be used. The students then write a capstone proposal and presents it before a selected group of panelists.

Course Information:

Course Code: INTE 40163

Course Title: Capstone Project 1

No. of Units: 3 units

No. of Hours: 5 hours

Course Outcomes:

After successful completion of this instructional material, you should be able to:

- Write a capstone proposal.
- Present a capstone proposal for approval before a panel.

Grading System:

Grading in this course rely solely on the grade given by the adviser and the thesis panel after the proposal defense. The final grade of the student is computed as follows:

Final Grade = 40% of Adviser (Project Deliverables) + 60% of Capstone Project 1 Defense

Classroom Policies:

The students must report on progress at least once a week. This can be done via online meetings, email, or Facebook chats.

Consultation:

The student is expected to consult with their respective capstone project adviser regarding the content and direction of the research.

CHAPTER 1

INTRODUCTION

1.1 Project Context

- **Project Definition.** A narrative description of the goals and objectives of the project (excluding being an academic requirement), the project assumptions, and the project constraints
- **Project Overview.** A narrative description of the organizational system/s being studied and analyzed, the operations and basic functions
- **Organizational Background.** A narrative description of the profile of the organization being studied , including the brief history, stakeholders (optional), location, size, organizational structure, mission, vision, organizational objectives/thrusts or services (optional)

1.2 Technical Background

Discuss about the technology that the organization is using and the explanation for all those technical developments.

1.2.1. Equipment/Hardware. Enumerate the current organization's equipment/hardware set-up

Equipment/Hardware	Description

1.2.2. Software. Enumerate the current organization's software being used, if applicable.

Software	Description

1.2.3. Peopleware/Manpower. Enumerate the organization's manpower with their job description.

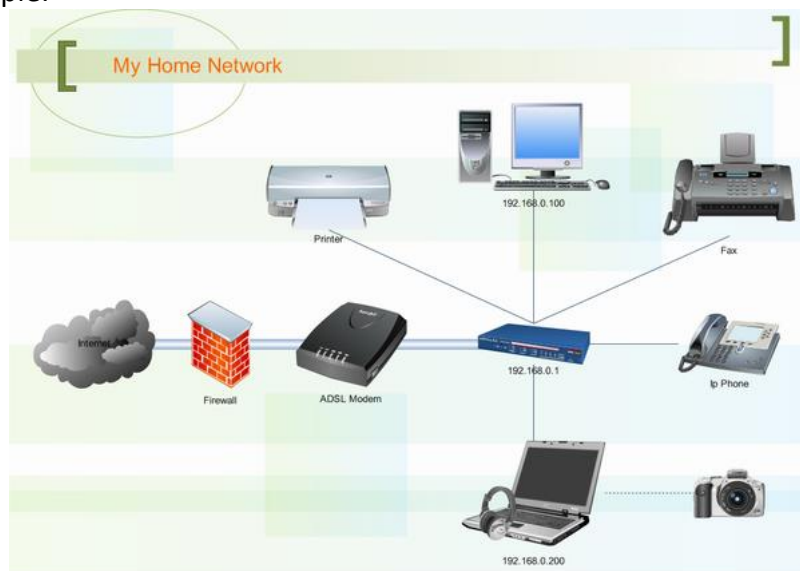
Position	Responsibilities

Example:

Position	Responsibilities
Accountant	In-charge in the accounting of the whole company.
Accounting Staff	Takes the task of monitoring sales and billing and collection. In-charge in inventory of grocery items, and payroll for the company.

1.2.4. Network Infrastructure/Architecture. Discuss briefly the current organization's network infrastructure/Architecture. Include illustration if any.

Example:



1.2.5. Storage, Backup and Recovery Procedure. Discuss the organization's manner of storing data, how they backup and restore these data.

Example:

In storing a file, the company uses an OPEN TEXT application in a way of scanning files, uploading data file or scan file to server/archiver server. In a backup process the company uses a special application to back up their file to local storage once the system is down. In recovering their file, they used special application in data recovery, they manually check the file from other local storage and also retrieve uploading logs in server.

1.2.6. Security Procedures. Explain how the organization execute their security procedures and policies.

Example:

In securing the transactions and data the users make sure that their files will be saved directly to their IP Address and personal hard drive

- 1.2.7. Policies and Procedures. Specify the current business rules and requirements used by the organization in processing their transactions. Business rules are lists of statements that tell you whether you may or may not do something or that give you the criteria and conditions for making a decision. Business requirements are what you need to do to enable the implementation of and compliance with business rules.

Business Requirements

Bus. Req. #	Description

Example:

Bus. Req. #	Description
BRQ#1	Requirements from the employee must meet the requirements of the client.
BRQ#2	The employee must validate all the information before submitting an issue.
BRQ#3	Ability to list, view, and search tickets by its ticket number.

Business Rules

Bus. Rule #	Description

Example:

Bus. Rule #	Description
BR#1	To report an issue, the employee must first inform the helpdesk through email.
BR#2	Employee is not allowed to request without reporting the issue.
BR#3	Helpdesk should send the ticket details to employee for reference.

- 1.2.8. Data and Process. Enumerate the steps on how data are being processed per major process.

1.3 Problem Analysis

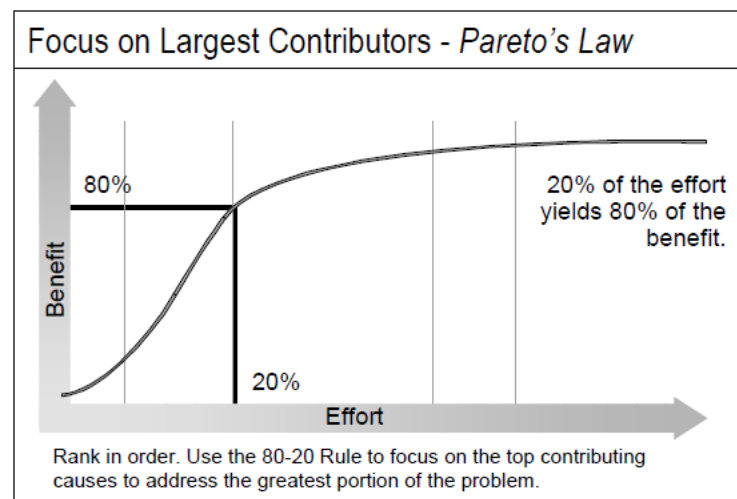
The process of understanding real-world problems and user needs and proposing solutions to meet those needs. A problem can be defined as the difference between things perceived and

things desired. Identify the overriding problem and establishing the causes and effects related to the organization's problem. Ensuring that "root causes," not just the symptoms of the problem, are identified and subsequently addressed in the project design.

Steps in Problem Analysis:

- Gain agreement on the problem definition. Team techniques for understanding the problem that a new software system is intended to solve. One approach is to write it down and see if everyone agrees. Use the Problem Statement format below.
- Understand the root causes – the problem behind the problem. The process by which the team gains an understanding of the causes of the problem under consideration. Techniques include fishbone diagrams and pareto charts.

One way of focusing on the most important root causes is the Pareto Principle, or "80-20 rule." A Pareto chart ranks the contributing causes in the order of their contribution to the problem, so it is easy to see the largest contributors. A Pareto diagram can also be thought of as a "Fishbone diagram with numbers." Assign a weight to each bone and then graph it.



- Identify the stakeholders and users. A stakeholder is anyone who could be materially affected by the implementation of a new system or application. Identifying them involves interviewing decision makers, potential users, and other interested parties.

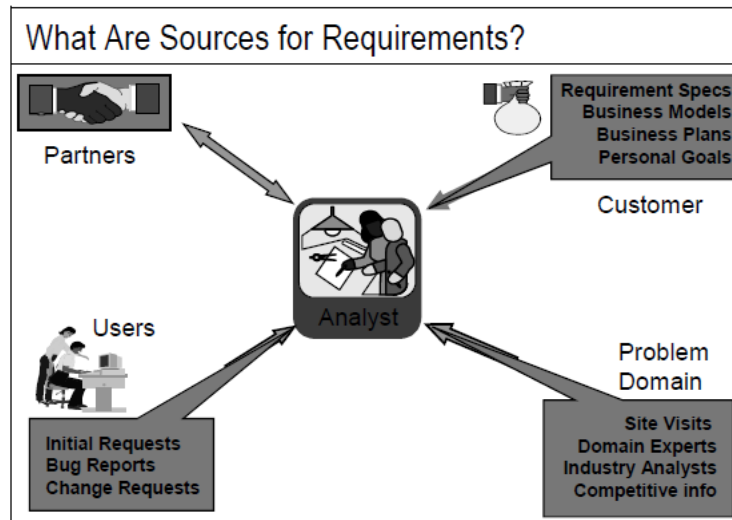
Helpful Questions to Ask in Identifying Stakeholders:

- Who are the users of the system?
- Who is the customer (economic buyer) for the system?
- Who else will be affected by the outputs the system produces?
- Who will evaluate and approve the system when it is delivered and deployed?

- Are there any other internal or external users of the system whose needs must be addressed?
- Who will maintain the new system?
- Is there anyone else who cares?

Understand Stakeholder Needs:

- Identify sources for stakeholder requests.



Other examples of external sources for requirements are:

- Statement of work
- Request for proposal
- Mission statement
- Problem statement
- Business rules
- Laws and regulations
- Legacy systems
- Business models
- Any results from requirements elicitation sessions and workshops
 - Expressing stakeholder requests.

Stakeholder requests come from many sources in many different forms. The most common form is a customer requirements specification. But requests also come via email, meetings, informal discussions over lunch, etc. All of these requests need to be recorded somewhere and negotiated for inclusion into the project.

To understand the need behind a request expressed as a feature, you can perform some problem analysis. For example: "The defect tracking system shall provide a project status trending report." This is a stakeholder request expressed as a feature. To understand the need that drives this feature, ask "why?" The answer could be something like: "I need to be able to understand

if defects being raised faster than they are being resolved.” This is the real need behind the feature, a sub-problem if you like.

- List techniques to elicit stakeholder requests.

Techniques for Eliciting Stakeholder Requests
<ul style="list-style-type: none">♦ Review customer requirement specifications♦ Requirements workshop<ul style="list-style-type: none">▪ Use-case workshops▪ Brainstorming and idea reduction♦ Interviews♦ Questionnaires♦ Role playing♦ Prototypes♦ Storyboards

These elicitation techniques are useful for gathering information about stakeholder needs. The techniques can also be used very effectively for gathering information about feature requirements or detailed software requirements.

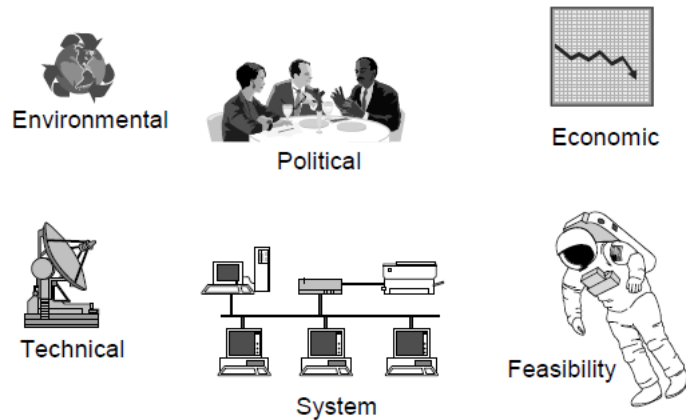
Many of these techniques can be used together. For example, a requirements workshop brings stakeholders together. At the requirements workshop, the participants may brainstorm for new ideas, or they may sketch out the use cases for the system to be built.

- Define the solution system boundary. Determine the boundaries of the solution system. Divide the world in two:
 - Our system Things that interact with our system
 - Someone or something outside the system that interacts with the system is an “actor”

Identifying Actors:

- Who will supply, use, or remove information from the system?
- Who will operate the system?
- Who will perform any system maintenance?
- Where will the system be used?
- Where does the system get its information?
- What other external systems will interact with the system?
- Identify the constraints to be imposed on the solution. A constraint is a restriction on the degree of freedom we have in providing a solution. Some constraints become requirements for the system. Others affect resources, implementation plans, and project plans.

Sources of Constraints:



Potential Causes of Constraints:

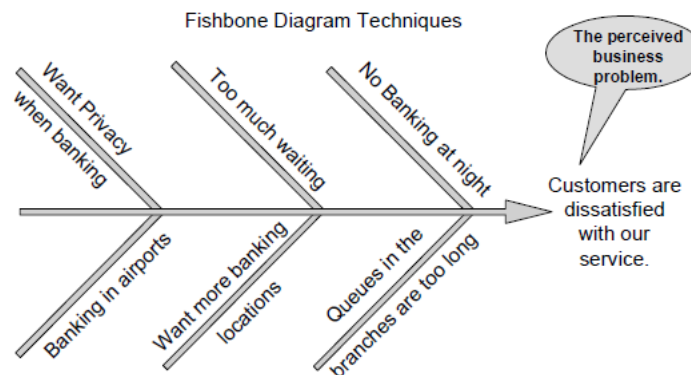
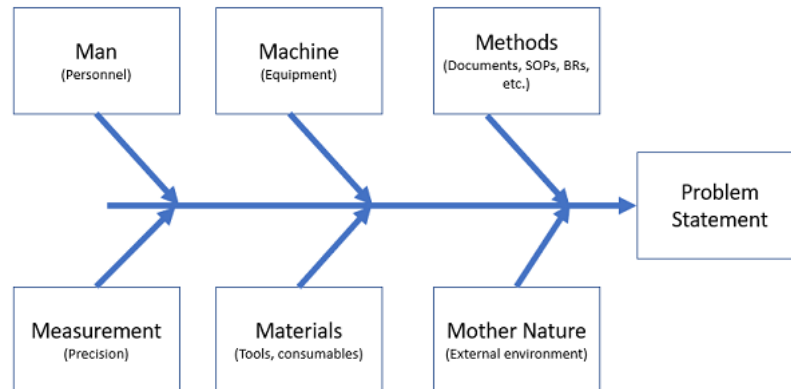
Source	Sample Considerations
Economic	What financial or budgetary constraints apply? Are there costs of goods sold or any product pricing considerations? Are there any licensing issues?
Politics	Do internal or external political issues affect potential solutions? Are there any interdepartmental problems or issues?
Technology	Are we restricted in our choice of technologies? Are we constrained to work within existing platforms or technologies? Are we prohibited from using any new technologies? Are we expected to use any purchased software packages?
Systems	Is the solution to be built on our existing systems? Must we maintain compatibility with existing solutions? What operating systems and environments must be supported?
Environment	Are there environmental or regulatory constraints? Are there legal constraints? What are the security requirements? What other standards might restrict us?
Schedule and resources	Is the schedule defined? Are we restricted to existing resources? Can we use outside labor? Can we expand resources? Temporarily? Permanently?

1.3.1. Fishbone Diagram. The Fishbone diagram is one of the tools used in problem analysis. Use the fishbone diagram to show the root cause of the organization's main problem.

There are four steps in creating a Fishbone Diagram:

1. Identify the main problem and write it on the fish's head.
2. Work out the major factors involved by asking the question "why" plus the main problem.

3. Identify possible causes such as man, machine methods, measure, materials and mother nature.
4. Analyze your diagram and finalize.



- 1.3.2. Problem and Solution Statement. Specify the problem the organization is facing and the possible solution to this problem.

Problem Statement

The problem of	(describe the problem)
affects	(the stakeholders affected by the problem)
the impact of which is	(what is the impact of the problem)
a successful solution would	(list some key business benefits of a successful solution)

The problem of	An inability to make electronic person to person payments
Affects	People that would like to easily send money to another person
The impact of which is	Increased effort required to provide payment using checks or cash
A successful solution would	Allow a person to electronically transfer money to another person in a safe and quick manner.

1.3.3. Problem – Requirements Matrix. [a table derived from problems in the Fishbone Diagram and solutions to the problems]

Problem	Requirements

Example:

Problem	Requirements
Equipments are difficult to trace	The projected system will have an inventory for recording the equipments used for outside viewing. It will monitor the quantity of equipments used for outside viewing and the quantity of the returned equipments.
No back-up files in case of loss	The database of the proposed system will serve as back-up in all transactions made by the company.

1.4 Purpose And Description

A project's purpose explains the reason for its existence, the meaning of what is done, the ambition or dream pursued by the project or the direction it takes and maintains. The definition of this is essential at three levels: for the project and for all stakeholders. A project description is a high-level overview of why you're doing a project. The document explains a project's objectives and its essential qualities. Think of it as the elevator pitch that focuses on what and why without delving into how.

Example:

The purpose of the research project is for the students to learn how to formulate a simple natural language problem/task/application and to experience how to solve it using methods, algorithms and techniques taught in class. The students will conduct experimental evaluation on an interesting dataset and will analyze the obtained results.

1.5. Specific Objectives

Indicating such objectives of the organization that specify the strategic goals and are specific, measurable, achievable & time-bound and are assigned to specific responsible persons in the project plan within the organization.

Example:

1. Making 50 000 euros the first year of product launch
2. In six month 90% of our products will be available in our online shop
3. 100% compliance with HTML 5 standard

1.6. Scope And Limitations

Scope and limitations are two terms that address the details of a research project. The term scope refers to the problem or issue that the researcher wants to study with the project. Limitations is the term used for constraints that impact the researcher's ability to effectively study the scope of the project. The scope and limitations can be presented either in paragraph or bullet format.

Example in paragraph format:

The coverage of the system will generally focus on ticket request. There will be sending of ticket request for the company that allows the employee of each department to input their hardware and software issues through their personal computers. Where the IT/Tech support will be notified regarding the issue of the employee. Part of it also is the approval of tickets by an IT Staff. After the approval of request by an IT staff the system will categorize the issue according to its level of priority. It will also provide reports such as summaries of ticket and IT staff performance. All matters regarding information and management of the staffs are not included in the system.

Example in bullet format:

Scope will include the following:

- Processing of accounts receivable and payable
- Recording of sales and purchases
- All reports pertaining to accounts receivable and accounts payable

Limitations:

- Does not include aging of accounts receivable
- Does not include actual collection from customers

ASSESSMENT:

1. Differentiate business requirements from business rules.
2. How will you know if it is a “real problem?”
3. Explain Pareto Principles in your own words.
4. Briefly explain the different techniques in eliciting stakeholder requests.

5. Why should analysts consider the possible causes of constraints in problem analysis?
6. Enumerate and explain the steps in creating the Fishbone Diagram.

CHAPTER 2

REVIEW OF RELATED LITERATURE AND STUDIES

Overview

This chapter starts with a brief introductory paragraph concerning the researcher's exploration of related literature and studies on the research problem. It states the main coverage of said chapter which are how to review related literatures and studies. It will allow the readers to assess how important a review of literature and studies is in research. It will discuss how review will be done and how to write this portion of the research.

This chapter primarily presents the different research and other literatures from both foreign and local researchers, which have significant bearings on the variables included in the research. It focuses on several aspects that will help in the development of the study. The literatures of any chosen study come from books, journals, articles, electronic materials such as PDF or E-Book, and other existing theses and dissertations, foreign and local that are believed to be useful in the advancement of awareness concerning the study.

*https://www.academia.edu/8222610/chapter_2_review_of_related_literature_and_studies

Objectives

At the end of the chapter, the students will be able to:

1. Discuss the importance and types of RRLS
2. Identify the different process of reviewing a literature in research writing
3. Construct own Review of Related Literature and Studies

Introduction

In research writing the most common problem is how to review different studies and literatures related to your chosen topic. Other researchers just read and write whatever they think is related and summarize everything and that's it, they are done with the portion of documentation in thesis writing, which is called Review of Related Literature and Studies (RRLS). It is imperative to note getting related literature and studies should be past ten years from the time of research writing. Earlier than this will not validate the review of related literature and studies

Types of Literature Review

There are many types of literature review, and the following types of literature review are the most popular in research studies:

Narrative literature review critiques the literature and summarizes the body of a literature. Narrative review also draws conclusions about the topic and identifies gaps or inconsistencies in a body of knowledge. You need to have a sufficiently focused research question to conduct a narrative literature review.

Systematic literature review requires more rigorous and well-defined approach compared to most other types of literature review. Systematic literature review is comprehensive and details the timeframe within which the literature was selected. Systematic literature review can be divided into two categories: meta-analysis and meta-synthesis.

When you conduct meta-analysis you take findings from several studies on the same subject and analyze these using standardized statistical procedures. In meta-analysis patterns and relationships are detected and conclusions are drawn. Meta-analysis is associated with deductive research approach.

Meta-synthesis, on the other hand, is based on non-statistical techniques. This technique integrates, evaluates and interprets findings of multiple qualitative research studies. Meta-synthesis literature review is conducted usually when following inductive research approach.

Argumentative literature review, as the name implies, examines literature selectively in order to support or refute an argument, deeply imbedded assumption, or philosophical problem already established in the literature. It should be noted that a potential for bias is a major shortcoming associated with argumentative literature review.

Integrative literature review reviews, critiques, and synthesizes secondary data about research topic in an integrated way such that new frameworks and perspectives on the topic are generated. If your research does not involve primary data collection and data analysis, then using integrative literature review will be your only option.

Theoretical literature review focuses on a pool of theory that has accumulated in regard to an issue, concept, theory, phenomena. Theoretical literature reviews play an instrumental role in establishing what theories already exist, the relationships between them, to what degree the existing theories have been investigated, and to develop new hypotheses to be tested.

At the earlier parts of the literature review chapter, you need to specify the type of your literature review and provide reasons for your choice. Your choice of a specific type of literature review should be based upon your research area, research problem and research methods. Also, you can briefly discuss other most popular types of literature review mentioned above.

* <https://research-methodology.net/research-methodology/types-literature-review/>

RRLS is very important and it must be written comprehensively for you to state that you have a problem to work on. Different researchers write their RRL in different ways, for this discussion these are the key ideas on how to start reviewing related works (literature and studies) for your research:

1. The review shall be organized thematically to confirm to the specific problems, can be from simple to complex and vice versa;
2. It should synthesize evidence from all studies reviewed to get an overall understanding of the state of the knowledge in the problem area;
3. As much as possible, the reviewed works should be limited within the last ten years, but of course if it is theories better based your work to the original researcher of the theory, and
4. A clinching statement showing how the related materials had assisted the researchers in the present study should be the last part.

Foreign and local studies should be included in this chapter arranged from simple to complex or vice versa.

Conducting a review of literature

- Introduction
- Reorganize your copied and pasted files
- Retrieve all related copied and pasted abstracts
- Evaluate the results

- Author, Title, Date, Source or document type
- Read the abstract
- Benchmark on the suggestions or recommendations taken from sources of literature and studies. Retrieve the full texts of the most useful sources such as Books, Dissertation, and Journals
- Use primary or secondary source depending on the objectives of the study

Why?

As mentioned earlier RRLS is important to write because it gives the ff:

1. Knowledge base upon which your study is built
2. Indebtedness to the past and shows connection between:
 - What was known in the past
 - What was discovered in the present

Writing an effective related literature and studies

Once you have conducted the review of related literature you are now ready to write an effective related literature. This can be done as follows:

I. Update the comprehensive critique of the literature and studies

- Check Journals that publish educational research
- Contact other researchers working in your field of study
- Attend scholarly conferences

II. Reread all resources

The purpose of this is to freshen your understanding of what has been learned

Example:

Topic: Writing a review of the research on teachers' professional development

1. Review each topical file

Example: Stages of teachers' career development

2. Identify each source with an appropriate code number to simplify the way you refer to it in your own work

Example: Career Stages 1

3. Cross-reference sources appropriately

- Make a simple note card reminding you of the source you should check
- Example:
 - Career Stages 1: Stages of teachers' career development
 - Contains Material: Fostering teachers' development
 - See Career Stages 1 + Fostering folder

4. Reread each source and write a brief summary

- Main points
- Perceived value to your review

Example:

- Stages 7. Huberman identifies 5 stages, each with a different theme
- In stages 3 & 4, the teacher may take one of two paths. Very useful review.
 - Clip the summary to the source

Types of sources:

- Opinions about current educational issues
- Reports of practice

- Prescriptions of what should be done

III. Develop the final outline

1. Determine the major components of the chapter

- Includes explanation of the search process that was used in assembling the review
- Discussion of the theoretical literature
- A review of the empirical research

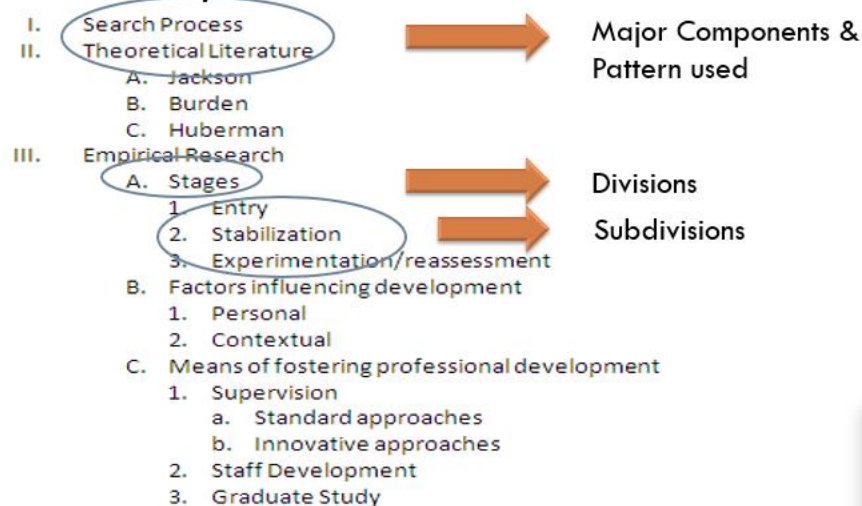
Patterns used:

- The search process
 - The theoretical literature
 - The empirical research
- The search process
 - The empirical research
- The theoretical literature
- The empirical research

2. Analyze the major components into Divisions and Subdivisions

- Question: What are the big pieces of this component?
- Example: How the researcher might identify the divisions of the component
“Methods of fostering teachers’ professional development”
 - Supervision
 - Staff Development
 - Graduate Study
- Subdivisions - For divisions that are both complex and critical to the review
 - Supervision
 - Standard approaches
 - Innovative approaches

3. Develop and Submit the Final Outline



IV. Use levels of Headings that reflect the outline

This is needed to make clear to the reader how the chapter is organized. Consider 4 Heading style: follows the recommendations of the APA Publication Manual (4th ed.)

- Title of the chapter – Level 1

- Main divisions of the chapter – Level 2
- Subdivisions – Level 3
- Sub-subdivisions – Level 4

V. Write an Introductory Paragraph

- Does not have its own heading
- Should be relatively brief
- Providing an overview of the chapter

Write the First Section of the Review

- Provide an Overview
 - It helps the reader understand how that section is organized and what its main divisions are.
- Generalize
 - One or two sentences that generalize what the studies show.
 - Obligation to the reader to make coherent sense of the literature
- Specify
 - Specific evidence, citing and discussing each study relating to that generalization.

Write the Remaining Sections, Including a Summary

- Conclude the chapter with a summary, probably a page or two in length, that reviews the content of the chapter and brings together the key conclusions of all the empirical research

Some Specific Reminders

- Vary the way you refer to studies.
- Paraphrase; do not quote.
- Be sure each section is organized clearly and give the reader verbal signals to indicate organization.
- Use headings that reveal the organizational pattern.

Checklist

- Is the review
 1. 1. Comprehensive, including all major works relating to your topic?
 1. 2. In-depth, providing the reader dept of knowledge about the prior research?
 2. 3. Current, including works published recently?
 3. 4. Selective, discriminating between major and less important studies?
 4. 5. Unbiased, without the writer skewing the prior research to suit his or her point of view?
 5. 6. Clearly organized, so that the reader can easily follow the plan and flow of the chapter?
 6. 7. Coherent, making sense of the studies, not simply describing them?
 7. 8. Effectively written, with a scholarly style?

Evaluate and Revise

- Once done with the writing the chapter, put it aside for a while
- Then revise it, using such technological aids as a spell-check, a thesaurus, and a style-checker.

Synthesis comes at the end of the chapter. This summarizes and combines all related literature and studies gathered to the research paper being conducted. This is usually one to two paragraphs only.

- References:
 - Glatthorn, Allan A. (1998). Writing the winning Dissertation: A Step-by-step guide. Corwin Press, Inc. Thousand Oaks, California.
 - Katz, Steven B. and Penrose, Ann M. (1998). Writing in the Science, Exploring conventions of scientific discourse. St. Martin's Press: New York, NY.
- Online sources:

<http://www.jpsimbulan.net/thesis-writing-guide/how-to-write-a-thesis/how-to-write-chapter-2-of-a-thesis-basic-format>

https://www.academia.edu/8222610/chapter_2_review_of_related_literature_and_studies

<https://research-methodology.net/research-methodology/types-literature-review/>

Chapter Activities:

1. Fill in the matrix of RRLS:

Title of Research/Literature (8 RRLS per student member in the group)	Author/s	Related Keywords	Publisher, complete information of Publication	Summary and conclusion	Recommendation/s

2. Given the matrix of RRLS, produce Chapter 2 given the following outline:
 1. Related Literature (Foreign /local, simple to complex or vice versa)
 2. Related Studies (Foreign /local, simple to complex or vice versa)
 3. Synthesis of the Study

CHAPTER 3

METHODOLOGY

OVERVIEW

Requirement's analysis involves extensive interaction with users to elicit information about the required output. This information is documented primarily through models which serves as powerful techniques to show what processes the user must perform which the new system must support. This module covers introduction to requirements, its types, and how they can be modeled using use cases. It also discusses the use case report which provides more detailed analysis of the use case diagrams.

LEARNING OUTCOMES

1. Analyze real world problems.
2. Distinguish functional and Non-Functional requirements.
3. Apply modeling tool to analyze project requirements
4. Identify appropriate computing solution.

COURSE MATERIALS

3.1 Requirements Analysis

This module discusses how to identify the functionalities that is needed by a system in order to satisfy the customer's requirements. System requirements are the most effective way of meeting the user needs and reducing the cost of implementation. System requirements could cause a company to save a lot of money and time, and also can cause a company to waste money and time. They are the first and foremost important part of any project, because if the requirements are not fulfilled, than the project is not complete.

What Are System Requirements

System requirements are a broad and also narrow detailed statement that the customer makes in order to achieve their requirements. The statement should clearly explain what the customer exactly wants and how they want it. A customer's need might be to satisfy a contract, solve a problem, achieve an objective, meet a standard, or to meet any other guidelines of the project. System requirements will always vary depending on the project and no two systems would have identical requirements. For instance, if two companies were wanting to build a wireless phone. Company A could have requirements for the phone to be different colors, light weight, and

transparent. Company B could have touch screen, all black, and also to be in-between 1 lbs. – 1.20 lbs. for their requirements. These examples are just the simple constraints for a product. There could be thousands of other requirements for the wireless phone. Including, chip maker, what products to use, and cost of course. There is no such thing as a correct amount of system requirements. This can vary from product to product or solution to solution. A flying plane could have millions of requirements, compared to a bicycle that could have a hundred requirements. As well, the requirements for solving the issue of hand washing would be a lot less than the requirements for sending a person to Mars. Yet, having too many system requirements for a project could cause the cost and time to dramatically increase. On the other side of that, having too few requirements might cause your system to not work correctly, or last as long as you like. Thus making the requirements the most important aspect of a system.

Where They Come From

System requirements can come from the customer, the company, or even other larger groups that set the requirements for specific and whole categories. From a macro view to a micro view or system requirements. Think of the highest of groups, the government, that could issue requirements for systems. These requirements are set on the highest stage and have to be implemented down to a single person. For example, the government has hundreds of divisions that set the general bench mark to certain system requirements. These requirements could be for vehicles, food, medical, and anything else. Not only do companies and customers have to meet these requirements, but also the requirements set forth by the customer/company. With system requirements being significant to the process of the project/product or what the customer needs, it is necessary for the person who is submitting the requirements to know exactly what they are. With system requirements being set by many factions. The ending customer usually has the most requirements to be set. For instance, the federal government has requirements on the environmental footprint a manufacture can have making requirements. This is a system requirement that is based on the federal level. The next level could be state level requirements of stricter rules upon the first requirement and also a system requirement of how that product is tested and designed. Then the customer has their system requirements which are the most important and usually the most thorough and extensive list. The company or person who is trying to meet the system requirements should already be aware of the top two tiers of the requirements. The customer's requirements are the ones that are most important and matter the most.

How To Improve Requirements

For system requirements, there are many methods for a company/person to improve their system requirements. Companies that want to increase their percentage of the system requirements met, can do many task. For example, if a company were to do significant system requirements on their product and find an issue in the development stage. It might cost the company a few extra dollars to have that fixed at once. If they didn't have the system requirement for that design, than it could cost the company twice or more to fix the same issue.

Having the technology and skills to master the system requirements and to implement them can save a company a lot of money. Having the right processes and people to test out the requirements can help the customer get exactly what they want. The more improve requirement testing, the better satisfaction for the customer.

Well Designed

Having a well-designed requirements for a project can save the customer a lot of time and money. Listing out the details of specific requirements before the project proceeds and advances will do the company world of good. Having test and making products, and then learning that a requirement was skipped to save money will cost even more money down the road.

Poorly Designed

System requirements could be thought of being well designed to a customer, yet some requirements could have been misplaced simple do to cost or time cutting. In 2010 Toyota had an issue with some of their cars driving out of control. The issue was that the floor mats were becoming lodged under the accelerator. If the company had a requirement to test out all moving parts inside of the car, then that missed this requirement. This incident caused a five billion dollar recall cost to the company. This would be an example of not having enough system requirements for the vehicle. The customer has to decide if the cost would be great to implement a well thorough system requirement, or pay the cost to fix the original design. When system requirements are poorly designed the project itself will more likely be a failure. Companies pay a premium of as much as 60% on time and budget when they use poor requirements practices on their projects. Companies that do not have a well-documented and thorough requirements and other business analysis will have three times more project failures as successes. Meaning system requirements is a necessity to the overall success of projects. These requirements could make or break a company some times. For a lot of companies, they think they will have success because there would be no failures/mistakes in the beginning. Yet, long term wise, the company could see failures in their products years down the road. The most recent example of system requirement failure would be Volkswagen. The requirement they set fourth was to have certain emissions from their diesel engines. These system requirement was a big selling point for their vehicles. Come to find out, the company had a software that could detect when it was being tested. The software than would change the performance of the test to show better than expected numbers. This was a total scandal and will probably ruin the company. There was a requirement that was presented, yet the company was doing illegal methods to reach it. This will cost them about \$9.5 Billion in relation to the emissions scandal. This does not count the losses the stock has seen on the stock market.

3.1.1 Requirements Feature Matrix

Requirements are the capabilities that the new system must have or what characteristics it needs to have.

Two Categories

1. **Functional** – activities that a system must perform

Examples : Compute gross pay, Calculate taxes, etc

2. **Non-functional** – characteristics the system must have (see examples below per type of non-functional requirement)

Functional requirements

1. The system will enable salespersons to create a customer offer.
2. The system will allow salespeople to know whether an offer is pending on a specific vehicle.
3. The system will enable managers to record approval of a customer offer.
4. The system will prepare a sales contract.
5. The system will prepare a shop work order based on customer requested dealer options.
6. The system will record a customer deposit.
7. The system will record a customer payment.
8. The system will create a record of the customer's vehicle purchase.

Kinds of Non-functional Requirements

1. Operational - the physical and technical environments in which the system will operate.
2. Performance – the speed, reliability, and capacity of the system
3. Security – who has authorized access to the system under what circumstances
4. Cultural and political – Cultural and political factors and legal requirements that affect the system.

Non-Functional requirements - Examples per Type

1. Operational

- 1.1 The system should run on tablet PCs to be used by salespeople.
- 1.2 The system should interface with the shop management system.
- 1.3 The system should connect to printers wirelessly.

2. Performance

- 2.1 The system should support a sales staff of 15 salespeople.
- 2.2 The system should be updated with pending offers on vehicles every 15 minutes.

3. Security

- 3.1 No salesperson can access any other salesperson's customer contacts.
- 3.2 Only the owner and sales manager may approve customer offers.
- 3.3 Use of each tablet PC should be restricted to the salesperson to whom it is assigned.

4. Cultural and Political

- 4.1 Company policy says that all computer equipment is purchased from Dell.
- 4.2 Customer personal information is protected in compliance with the Data Protection Act.

Requirements Feature Matrix

Requirements feature matrix a mapping of the system requirements VS features of the proposed software. It lists down functional / non-functional requirements of the user and maps it against the features available in the product. The list of system requirements is based on (1) requirements driven by the problems identified in Chapter 1.3 and (2) requirements requested by users and stakeholders of the system and (3) requirements integral in the system as identified.

It summarizes what aspects of the user requirements are covered in the product or project which will help to facilitate identification of gaps. The matrix identifies which product feature caters to a specific customer requirement.

A sample template of a requirement / feature matrix is shown below.

REQUIREMENTS /FEATURES	STRQ1:	STRQ2:	STRQ3:	STRQ4:	STQR5:	STQR<n>:
FEAT1:						
FEAT2:						
FEAT<n>:						

Example:

REQUIREMENTS /FEATURES	STRQ1: Provide scheduling of equipments	STRQ2: Make back-up file in case of loss	STRQ3: Produce reports accurately	STRQ4: Produce promissory note/ written agreement for clients with balance	STQR5: Monitor the deployment of the equipments
FEAT1: The system will schedule the deployment of each assigned equipments	√				
FEAT2: The system will provide back-up copy of files in case of lost records		√			

3.1.2 Use Case Diagram

Use Case Modeling

- Used to explain and document the interactions between the users and the system to be able to accomplish the user's task.
- Graphically depicts the system as a collection of **use cases**, **actors(users)**, and their **relationships**

Main Reasons for Writing Use Cases

- Effectively communicate system requirements because the diagrams are kept simple
- Allow people to tell stories
- Make sense to non-technical people
- Do not depend on a special language
- Can describe most functional requirements

Examples

- Company policy says that all computer equipment is purchased from Dell.
- Customer personal information is protected in compliance with the Data Protection Act.

Use Case

Use Case describes the system functions from the perspective of external users in a manner they understand.

It is the result of decomposing the scope of system functionality into many smaller statements of system functionality.

Actor

Actors initiates the use cases

- Refers to a particular role of a user of the system
- Can be human, another system, or a device

Finding Actors

The actors of a system can be identified by answering a number of questions:

- Who will use the functionality of the system?
- Who will maintain the system?
- What devices does the system need to handle?
- What other system does this system need to interact with?
- Who or what has interest in the results of this system?

Use Case Relationships (Behavioral Relationships)

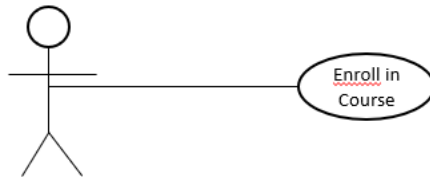
- Communicates

- Includes(uses)
- Extends
- Generalizes

Use Case Relationships – Definitions and Symbols

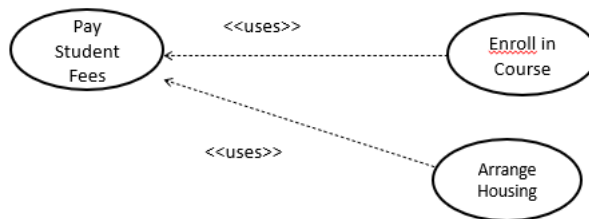
- **Communicates** - Shows an association of an actor with a use case

Symbol – line with no arrowheads



- **Includes (uses)**– describes a situation in which one use case contains a behavior that is common to more than one use case

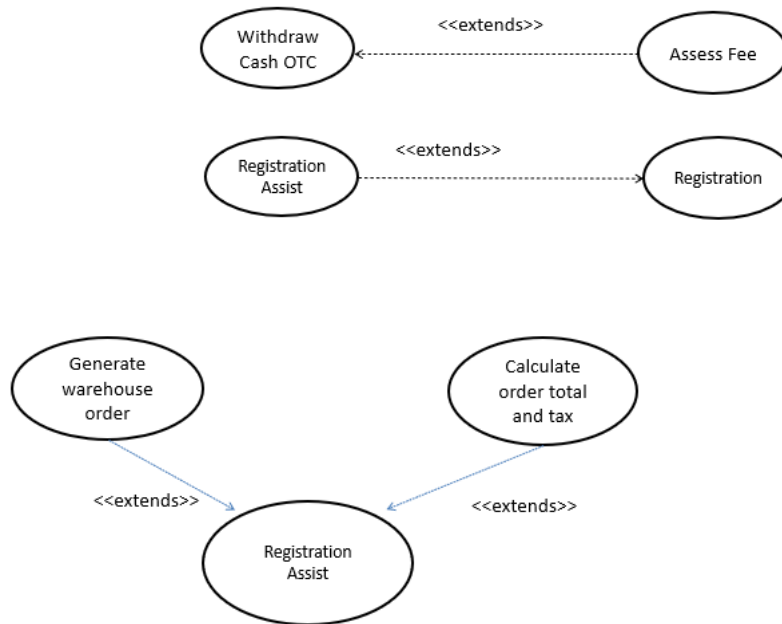
Symbol – Broken line with arrowhead pointing to the common use case



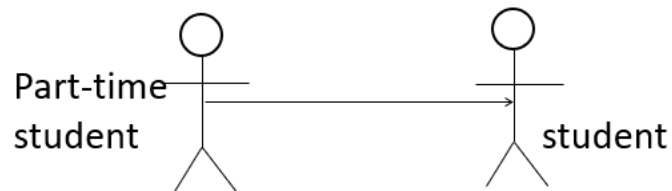
- **Extends** – A use case may contain complex functionality; to simplify, we extract the more complex steps into their own use case

The resulting use case is called an extension use case. It extends the functionality of the base case or the original use case

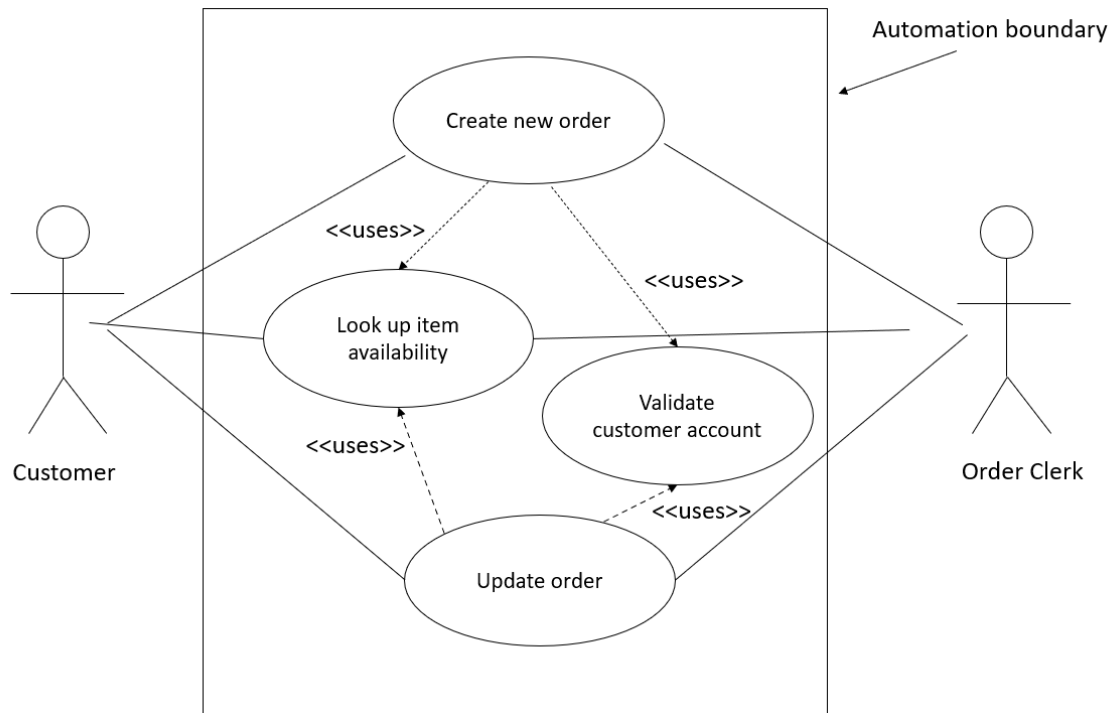
Symbol – Broken line with arrowhead pointing from the extending use case to the extended (base) use case



- **Generalizes** – Inheritance relationship by which a use case/actor inherits the properties of another use case/actor
Symbol – line with arrow pointing to the general thing



Example: The automation boundary is signified by a rectangle which houses the use cases identified as part of the automation process.



The Process of Requirements Use-Case Modeling

- Identify business actors
- Identify business requirements use cases
- Show the relationships between actors and uses cases (construct the use case diagram)
- Document use-case narratives

Techniques for identifying use cases

- List all users and think through what they need the system to do for their jobs
- List all system functions for the existing system, adding new functionalities
- Talk to all users to get them to describe their goals in using the system

Use Cases must not be too low level and must not be too broad

Examples of Use Cases

- Typing customer name on a form - too low level
- Working with customers to add new customers, update data, following up payments, etc. - too broad
- Adding a new customer - defines a complete user goal and is the right level of analysis for a use case

3.1.3 Use Case Report

The use case report provides a narrative of the use case model organized in different compartments.

- The first and second compartments identify the use cases and scenarios within the use cases.
- The third compartment (Triggering Event) identifies the trigger that initiates the use case.
- The fourth compartment is a brief description of the use case or scenario.
- The stakeholders compartment identifies interested parties other than specific actors. They may be users who do not actually invoke the use case but who have an interest in results produced from the use case.
- The next two compartments (preconditions/postconditions) provide critical information about the state of the system before and after the use case executes.

Preconditions state what condition must be true before a use case begins, e.g. what objects must already exist, what specific relationships exist between objects, what information must be available, what values are important etc. Postconditions identify what must be true upon completion of the use case.

- The final two compartments describe the detailed flow of activities with the use case showing the steps performed by the actor and the responses required by the system. The item numbering helps identify the sequence.

Below is a sample of a use case report

Use Case Name:	Create new order
Scenario:	Create new telephone order
Triggering Event:	Customer telephones RMO to purchase items from the catalog
Brief Description:	When customer calls to order, the order clerk and system verify customer information, create a new order, and items to the order, verify payment, create the order transaction, and finalize the order.
Actors:	Telephone sales clerk Customer
Include Use Case:	Includes : Check item availability Includes : Validate customer account
Extend Use Case:	
Preconditions:	Customer must exist. Catalog, Products, and Inventory items must exist for requested items
Postconditions:	Order and order line items must be created Order transaction must be created for the order payment Inventory items must have the quantity on hand updated The order must be related (associated) to a customer

Flow of Events:	Actor	System
	1. Sales clerk answers telephone and connects to a customer 2. Clerk verifies customer information 3. Clerk initiates the creation of a new order 4. Customer requests an item to be added to the order 5. Clerk verifies the item (check availability use case) 6. Clerk adds item to the order 7. Repeat steps 4, 5, and 6 until all items are added to the order 8. Customer indicates end of order; clerk enters end of order 9. Customer submits payment; clerk enters amount	3.1 Create a new order 5.1 Display item information 6.1 Add an order item 8.1 Complete order 8.2 Compute totals 9.1 Verify payment 9.2 Create order transaction 9.3 Finalize order
Exception Conditions:	2.1 If customer does not exist, then the clerk pauses this use case and invokes <i>Maintain customer information</i> use case. 2.2 If customer has a credit hold, then clerk transfers the customer to a customer service representative 4.1 If an item is not in stock, the customer can <ol style="list-style-type: none"> choose not to purchase item, or request item be added as a back-ordered item, 9.1 If customer payment is rejected due to bad-credit verification, then <ol style="list-style-type: none"> order is canceled, or order is put on hold until check is received. 	

Activities: Apply all activities to your capstone project

1. Identify functional and non-functional requirements
2. Create a requirements / feature matrix
3. Create a use case diagram
4. Create a use case report.

References:

- Systems Analysis and Design Methods 6th Ed, Whitten, Bentley, Dittman

- Object Oriented Analysis and Design with the Unified Process, Satzinger, Jackson, Burd
- Systems Analysis and Design 5th Ed, Dennis, Wixom, Roth
- <http://www.uml-diagrams.org/use-case-extend.html>
- <https://www.uml.edu/~sauterv/analysis/F2015/System%20Requirements.html>

3.2 Design Specifications

OVERVIEW

The design specification is a document that presents the complete design for the new information system. The system design specification is the baseline against which the operational system will be measured. Unlike the system requirements document, which is written for users to understand, the system design specification is oriented toward the programmers who will use it to create the necessary programs.


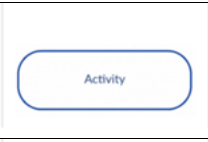


Activity Diagram











Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. It describes the flow of control of the target system, such as the exploring complex business rules and operations, describing the use case also the business process. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows).

Activity diagrams can be used in all stages of software development and for various purposes. And because they are a lot similar to flowcharts, they are generally more popular than other UML diagram types.

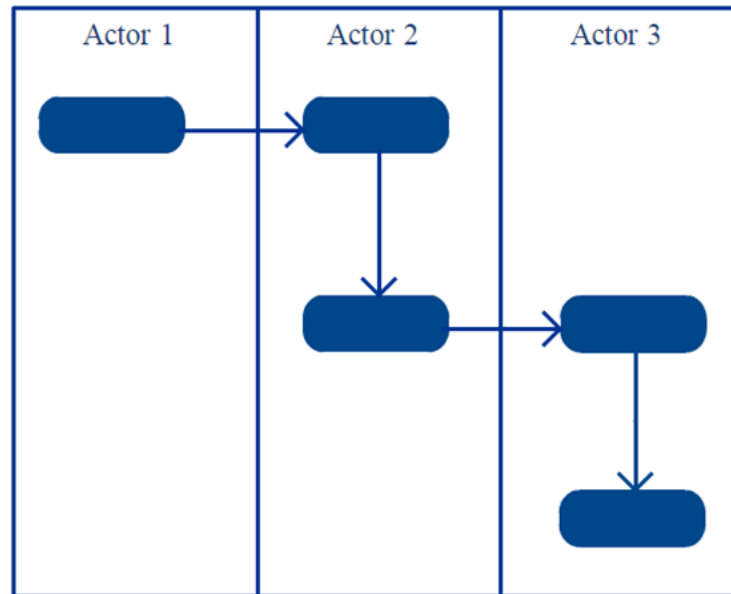
Activity Diagram Symbols

UML has specified a set of symbols and rules for drawing activity diagrams. Following are the commonly used activity diagram symbols with explanations.

Symbol	Name	Use
	Start/ Initial Node	Used to represent the starting point or the initial state of an activity
	Activity / Action State	Used to represent the activities of the process
	Action	Used to represent the executable sub-areas of an activity
	Control Flow Edge	Used to represent the flow of control from one action to the other

		Object Flow Control Edge	Used to represent the path of objects moving through the activity
		Activity Final Node	Used to mark the end of all control flows within the activity
		Flow Final Node	Used to mark the end of a single control flow
		Decision Node	Used to represent a conditional branch point with one input and multiple outputs
		Merge Node	Used to represent the merging of flows. It has several inputs, but one output.
		Fork	Used to represent a flow that may branch into two or more parallel flows
		Merge	Used to represent two inputs that merge into one output
		Signal Sending	Used to represent the action of sending a signal to an accepting activity
		Signal Receipt	Used to represent that the signal is received
		Note/ Comment	Used to add relevant comments to elements

Activity Diagrams with Swimlanes



In activity diagrams swimlanes – also known as partitions – are used to represent or group actions carried out by different actors in a single thread. Here are a few tips you can follow when using swimlanes.

- Add swimlanes to linear processes. It makes it easy to read.
- Don't add more than 5 swimlanes.
- Arrange swimlanes in a logical manner.

How to Draw an Activity Diagram

Activity diagrams can be used to model business requirements, create a high-level view of a system's functionalities, analyze use cases and for various other purposes. In each of these cases, here's how to draw an activity diagram from the beginning.

Step 1: Figure out the action steps from the use case

Here you need to identify the various activities and actions your business process or system is made up of.

Step 2: Identify the actors who are involved

If you already have figured out who the actors are, then it's easier to discern each action they are responsible for.

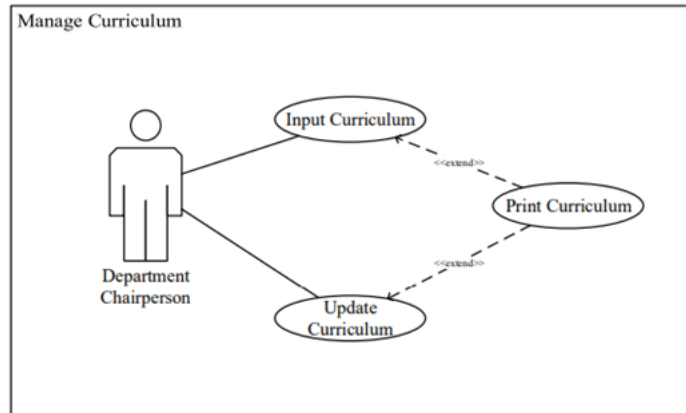
Step 3: Find a flow among the activities

Figure out in which order the actions are processed. Mark down the conditions that have to be met in order to carry out certain processes, which actions occur at the same time and whether you need to add any branches in the diagram. And do you have to complete some actions before you can proceed to others?

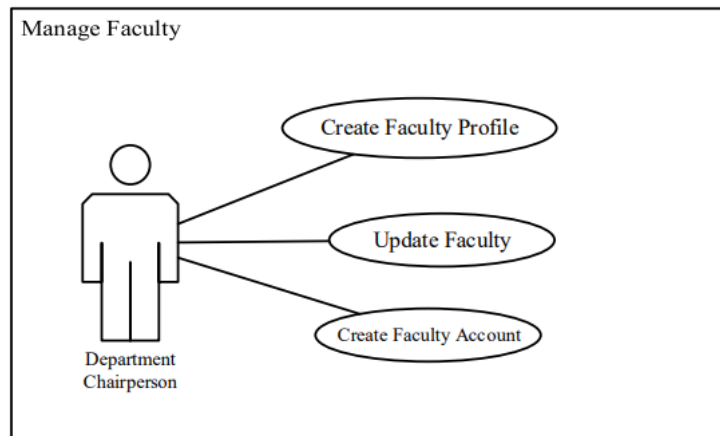
Step 4: Add swimlanes

You have already figured out who is responsible for each action. Now it's time to assign them a swimlane and group each action they are responsible for under them.

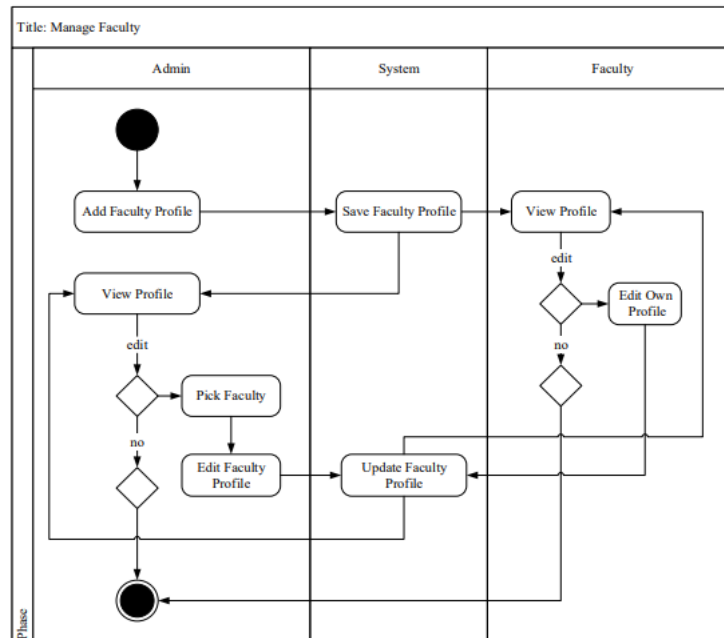
Activity Diagram Example based on Use-Case Diagram



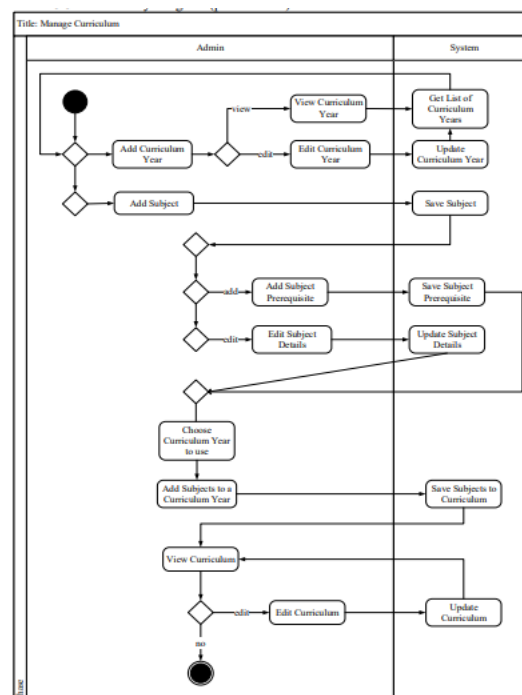
USE-CASE DIAGRAM: MANAGE CURRICULUM



USE-CASE DIAGRAM: MANAGE FACULTY



ACTIVITY DIAGRAM: MANAGE FACULTY



ACTIVITY DIAGRAM: MANAGE CURRICULUM

Class Diagram

The class diagram is a central modeling technique that runs through nearly all object-oriented methods. This diagram describes the types of objects in the system and various kinds of static relationships which exist between them.

Purpose of Class Diagrams

1. Shows static structure of classifiers in a system
2. Diagram provides a basic notation for other structure diagrams prescribed by UML
3. Helpful for developers and other team members too
4. Business Analysts can use class diagrams to model systems from a business perspective

A UML class diagram is made up of:

- A set of classes and
- A set of relationships between classes

What is a Class

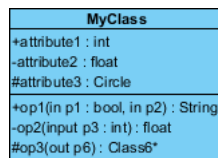
A description of a group of objects all with similar roles in the system, which consists of:

- **Structural features** (attributes) define what objects of the class "know"
 - Represent the state of an object of the class
 - Are descriptions of the structural or static features of a class
- **Behavioral features** (operations) define what objects of the class "can do"
 - Define the way in which objects may interact
 - Operations are descriptions of behavioral or dynamic features of a class

Class Notation

A class notation consists of three parts:

1. **Class Name**
 - The name of the class appears in the first partition.
2. **Class Attributes**
 - Attributes are shown in the second partition.
 - The attribute type is shown after the colon.
 - Attributes map onto member variables (data members) in code.
3. **Class Operations (Methods)**
 - Operations are shown in the third partition. They are services the class provides.
 - The return type of a method is shown after the colon at the end of the method signature.
 - The return type of method parameters is shown after the colon following the parameter name.
 - Operations map onto class methods in code



The graphical representation of the class - MyClass as shown above:

- MyClass has 3 attributes and 3 operations
- Parameter p3 of op2 is of type int
- op2 returns a float

- op3 returns a pointer (denoted by a *) to Class6

Class Relationships

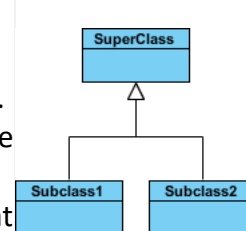
A class may be involved in one or more relationships with other classes. A relationship can be one of the following types: (Refer to the figure on the right for the graphical representation of relationships).

Relationship Type

Graphical Representation

Inheritance (or Generalization):

- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class



Simple Association:

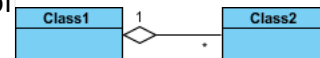
- A structural link between two peer classes.
- There is an association between Class1 and Class2
- A solid line connecting two classes



Aggregation:

A special type of association. It represents a "part of" relationship.

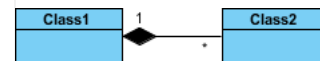
- Class2 is part of Class1.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.
- A solid line with an unfilled diamond at the association end connected to the class of composite



Composition:

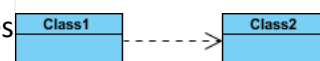
A special type of aggregation where parts are destroyed when the whole is destroyed.

- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite



Dependency:

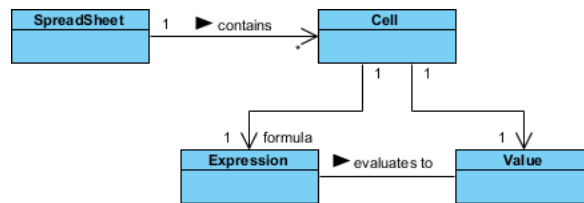
- Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).



- Class1 depends on Class2
- A dashed line with an open arrow

Relationship Names

- Names of relationships are written in the middle of the association line.
- Good relation names make sense when you read them out loud:
 - "Every spreadsheet **contains** some number of cells",
 - "an expression **evaluates to** a value"
- They often have a **small arrowhead to show the direction** in which direction to read the relationship, e.g., expressions evaluate to values, but values do not evaluate to expressions.



Relationship - Roles

- A role is a directional purpose of an association.
- Roles are written at the ends of an association line and describe the purpose played by that class in the relationship.
 - E.g., A cell is related to an expression. The nature of the relationship is that the expression is the **formula** of the cell.

Navigability

The arrows indicate whether, given one instance participating in a relationship, it is possible to determine the instances of the other class that are related to it.

The diagram above suggests that,

- Given a spreadsheet, we can locate all of the cells that it contains, but that
 - we cannot determine from a cell in what spreadsheet it is contained.
- Given a cell, we can obtain the related expression and value, but
 - given a value (or expression) we cannot find the cell of which those are attributes.

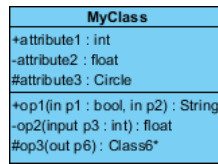
Visibility of Class attributes and Operations

In object-oriented design, there is a notation of visibility for attributes and operations. UML identifies four types of visibility: public, protected, private, and package.

The +, -, # and ~ symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

- + denotes public attributes or operations
- - denotes private attributes or operations
- # denotes protected attributes or operations
- ~ denotes package attributes or operations

Class Visibility Example



In the example above:

- attribute1 and op1 of MyClassName are public
- attribute3 and op3 are protected.
- attribute2 and op2 are private.

Access for each of these visibility types is shown below for members of different classes.

Access Right	public (+)	private (-protected)) (#)	Package (~)
Members of the same class	yes	yes	yes
Members of derived classes	yes	no	yes
Members of any other class	yes	no	in same package

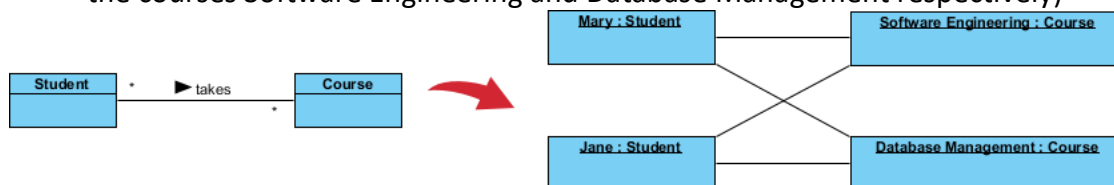
Multiplicity

How many objects of each class take part in the relationships and multiplicity can be expressed as:

- Exactly one - 1
- Zero or one - 0..1
- Many - 0..* or *
- One or more - 1..*
- Exact Number - e.g. 3..4 or 6
- Or a complex relationship - e.g. 0..1, 3..4, 6.* would mean any number of objects other than 2 or 5

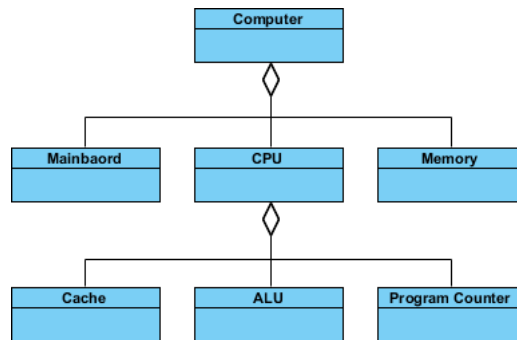
Multiplicity Example

- Requirement: A Student can take many Courses and many Students can be enrolled in one Course.
- In the example below, the class diagram (on the left), describes the statement of the requirement above for the static model while the object diagram (on the right) shows the snapshot (an instance of the class diagram) of the course enrollment for the courses Software Engineering and Database Management respectively)



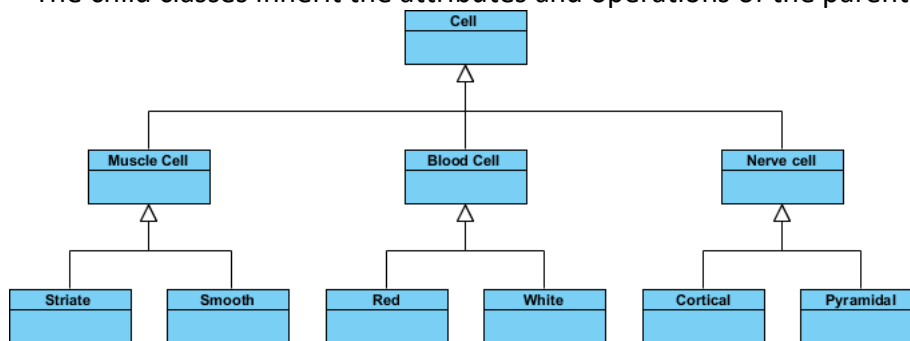
Aggregation Example - Computer and parts

- An aggregation is a special case of association denoting a "consists-of" hierarchy
- The aggregate is the parent class, the components are the children classes



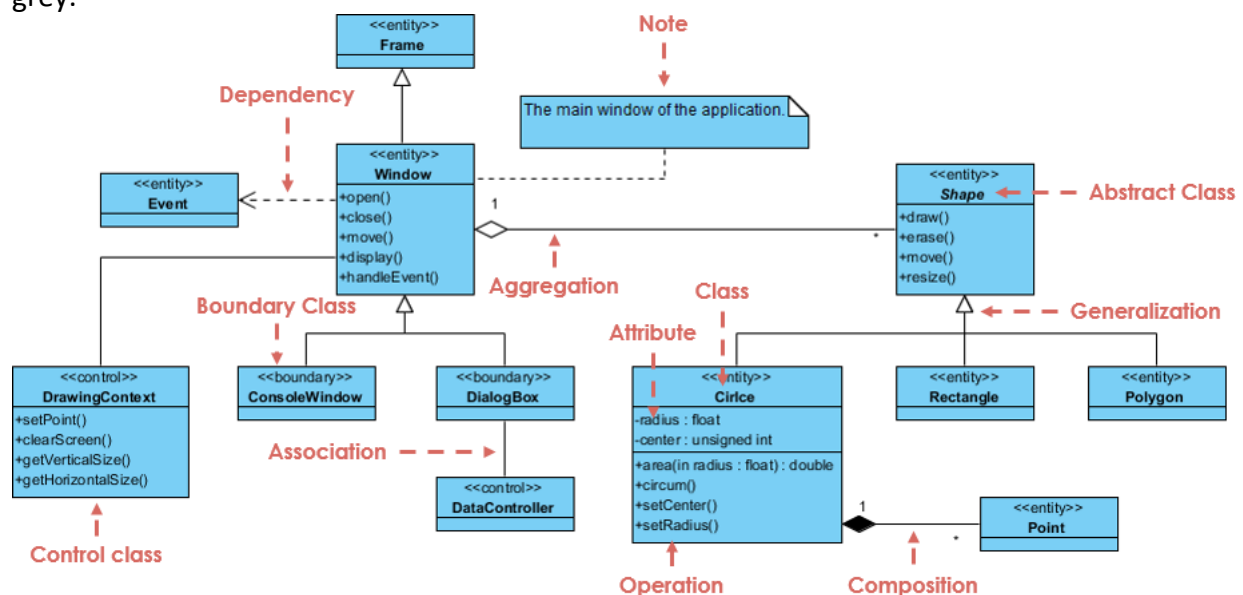
Inheritance Example - Cell Taxonomy

- Inheritance is another special case of an association denoting a "kind-of" hierarchy
- Inheritance simplifies the analysis model by introducing a taxonomy
- The child classes inherit the attributes and operations of the parent class.



Class Diagram - Diagram Tool Example

A class diagram may also have notes attached to classes or relationships. Notes are shown in grey.

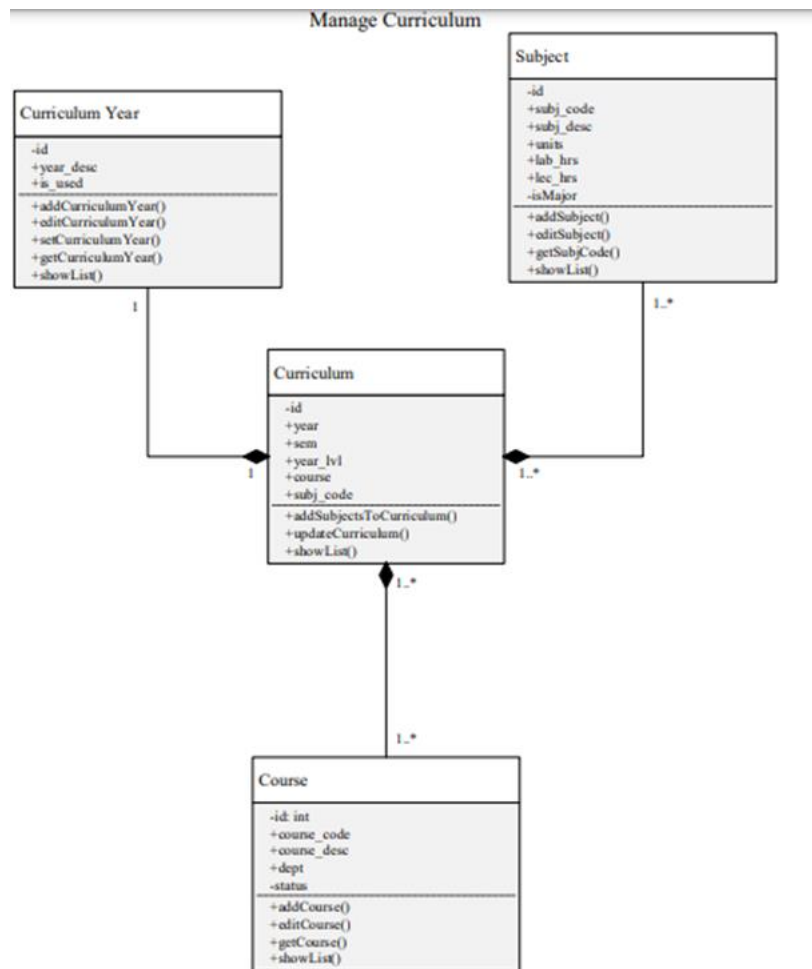


In the example above:

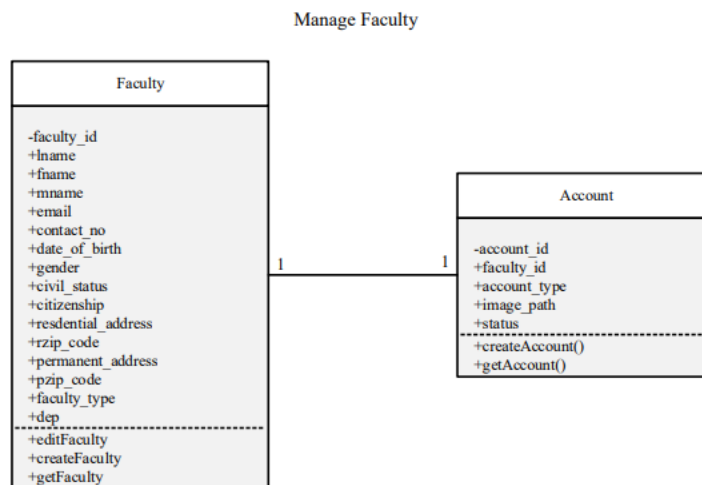
We can interpret the meaning of the above class diagram by reading through the points as following.

1. Shape is an abstract class. It is shown in *Italics*.
2. Shape is a superclass. Circle, Rectangle and Polygon are derived from Shape. In other words, a Circle is-a Shape. This is a generalization / inheritance relationship.
3. There is an association between DialogBox and DataController.
4. Shape is part-of Window. This is an aggregation relationship. Shape can exist without Window.
5. Point is part-of Circle. This is a composition relationship. Point cannot exist without a Circle.
6. Window is dependent on Event. However, Event is not dependent on Window.
7. The attributes of Circle are radius and center. This is an entity class.
8. The method names of Circle are area(), circum(), setCenter() and setRadius().
9. The parameter radius in Circle is an in parameter of type float.
10. The method area() of class Circle returns a value of type double.
11. The attributes and method names of Rectangle are hidden. Some other classes in the diagram also have their attributes and method names hidden.

Class Diagram Example based on Use-Case Diagram



CLASS DIAGRAM: MANAGE CURRICULUM



CLASS DIAGRAM: MANAGE FACULTY

GUI Design (screenshots)

Graphical User Interface (GUI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.

Best Practices for Designing an Interface

Everything stems from knowing your users, including understanding their goals, skills, preferences, and tendencies. Once you know about your user, make sure to consider the following when designing your interface:

- **Keep the interface simple.** The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.
- **Create consistency and use common UI elements.** By using common elements in your UI, users feel more comfortable and are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the site.
- **Be purposeful in page layout.** Consider the spatial relationships between items on the page and structure the page based on importance. Careful placement of items can help draw attention to the most important pieces of information and can aid scanning and readability.
- **Strategically use color and texture.** You can direct attention toward or redirect attention away from items using color, light, contrast, and texture to your advantage.
- **Use typography to create hierarchy and clarity.** Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scan ability, legibility and readability.
- **Make sure that the system communicates what's happening.** Always inform your users of location, actions, changes in state, or errors. The use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for your user.
- **Think about the defaults.** By carefully thinking about and anticipating the goals people bring to your site, you can create defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where you might have an opportunity to have some fields pre-chosen or filled out.

Database Schema

What is Database Schema Design?

Database schema design refers to the practices and strategies for constructing a database schema.

You can think of database schema design as a “blueprint” for how to store massive amounts of information in a database. The schema is an abstract structure or outline that represents the logical view of the database as a whole. By defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

Database schema design organizes the data into separate entities, determines how to create relationships between organized entities, and how to apply the constraints on the data. Designers create database schemas to give other database users, such as programmers and analysts, a logical understanding of the data.

Why is Database Schema Design Important?

Databases that are inefficiently organized suck up tons of energy and resources, tend to be confusing, and are hard to maintain and administer. That’s where database schema design comes into play.

Without a clean, efficient, consistent database schema, you’ll struggle to make the best use of your enterprise data. For example, the same data might be duplicated in multiple locations—or even worse, might be inconsistent between these locations.

Relational database systems heavily depend on having a solid database schema in place. The goals of good database schema design include:

- Reducing or eliminating data redundancy.
- Preventing data inconsistencies and inaccuracies.
- Ensuring the correctness and integrity of your data.
- Facilitating rapid data lookup, retrieval, and analysis.
- Keeping sensitive and confidential data secure, yet accessible to those who need it.

How to Design a Database Schema

Database schemas outline the architecture of a database, and helping to ensure database fundamentals such as the following:

- Data has consistent formatting
- All record entries have a unique primary key
- Important data is not omitted

A database schema design can exist both as a visual representation and as a set of formulas, or use constraints, that govern a database. Developers then express these formulas in different data definition languages, depending on the database system you’re using. For example, even though the leading database systems have slightly different definitions of what schemas are, MySQL, Oracle Database, and Microsoft SQL Server each support the CREATE SCHEMA statement.

For example, suppose that you want to create a database to hold information for your organization’s accounting department. A specific schema for this database might outline the structure of two simple tables:

Table1

Title: Users

Fields: ID, Full Name, Email, Date of Birth, Department

Table2

Title: Overtime Pay

Fields: ID, Full Name, Time Period, Hours Billed

This single schema contains valuable information such as: The title of each table

- The fields that each table contains
- The relations between tables (e.g. linking an employee's overtime pay to their identity via their ID number)
- Any additional relevant information

Developers and database administrators can then convert these schema tables into SQL code.

Best Practices for Database Schema Design

In order to make the most of database schema design, it's important to follow these best practices to ensure that developers have a clear point of reference about what tables and fields a project contains, etc.

- **Naming conventions:** Define and use appropriate naming conventions to make your database schema designs most effective. While you may decide on a particular style or else adhere to an ISO standard, the most important thing is to be consistent in your name fields.
- Try not to use reserved words in table names, column names, fields, etc., which will likely deliver a syntax error.
- Don't use hyphens, quotes, spaces, special characters, etc. because they will not be valid or will require an additional step.
- Use singular nouns, not plural nouns, for table names (i.e. use StudentName instead of StudentNames). The table represents a collection, so there's no need to make the title plural.
- Omit unnecessary verbiage for table names (i.e. use Department instead of DepartmentList, TableDepartments, etc.)
- **Security:** Data security starts with a good database schema design. Use encryption for sensitive data such as personally identifiable information (PII) and passwords. Don't give administrator roles to each user; instead, request user authentication for database access.
- **Documentation:** Database schemas are useful long after they've been created, and will be viewed by many other people, which makes good documentation essential. Document your database schema design with explicit instructions, and write comment lines for scripts, triggers, etc.
- **Normalization:** Briefly, normalization ensures that independent entities and relationships are not grouped together in the same table, reducing redundancy and improving integrity. Use normalization as necessary to optimize the database's performance. Both over-normalization and under-normalization can result in worse performance.
- **Expertise:** Understanding your data and the attributes of each element helps you build out the most effective database schema design. A well-designed schema can enable your data to grow exponentially. As you keep expanding your data, you can analyze each field in relation to the others you are collecting in your schema.

Data Dictionary

What Is a Data Dictionary?

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. It describes the meanings and purposes of data elements within the context of a project, and provides guidance on interpretation, accepted meanings and representation. A Data Dictionary also provides metadata about data elements. The metadata included in a Data Dictionary can assist in defining the scope and characteristics of data elements, as well the rules for their usage and application.

Why Use a Data Dictionary?

Data Dictionaries are useful for a number of reasons. In short, they:

- Assist in avoiding data inconsistencies across a project
- Help define conventions that are to be used across a project
- Provide consistency in the collection and use of data across multiple members of a research team
- Make data easier to analyze
- Enforce the use of Data Standards

Data Dictionary Example

DATA DICTIONARY – CURRICULUM						
Column	Type	Null	Default	Links to	Description	Sample Data
curriculum_id (Primary)	int(11)	No			Primary key for curriculum table.	1
curriculum_yr	int(11)	No		curriculum_year → curr_year_id	Foreign Key (FK)	2
sem	varchar(10)	No			Semester for curriculum year.	1st
year_lvl	varchar(20)	No			Year level.	1st
course	int(11)	No		course → course_id	Foreign Key (FK)	1
subj_code	int(11)	No		subject → subj_id	Foreign Key (FK)	2

3.3 Development Methodology

3.3.1 Process Model

A model is a representation of reality. Just as a picture is worth a thousand words, most models are pictorial representations of reality. Better understand existing system and Document requirements for proposed system. Logical models show what a system is or does. They are implementation independent; that is, they depict the system independent of any technical implementation. Physical models show not only what a system is or does, but also how the system is (to be) physically and technically implemented. They are implementation dependent because they reflect technology choices.

LOGICAL MODEL

A **nontechnical** pictorial representation that depicts what a system is or does. Synonyms are *essential model*, *conceptual model* and *business model*.

PHYSICAL MODEL

A **technical** pictorial representation that depicts what a system is or does. Synonyms are *implementation model* and *technical model*.

Why Logical System Models?

- Logical Models remove biases that are the result of the way the system is currently implemented, or the way that any one person thinks the system might be implemented.
- Logical Models reduce the risk of missing business requirements because we are too preoccupied with technical results.
- Logical Models allow us to communicate with end-users with nontechnical or less technical languages.

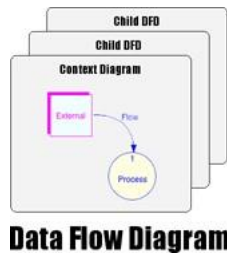
Process Modeling and DFDs

Process modeling is a technique for organizing and documenting the structure and flow of data through a system's processes, and/or the logic, policies, and procedures to be implemented by a system's processes. It is a technique used to organize and document a system's processes.

- Flow of data through processes
- Logic
- Processes

- Procedures

A **data flow diagram (DFD)** is a tool (and type of process model) that depicts the flow of data through a system and the work or processing performed by that system. DFDs have become a popular tool for business process redesign. It shows the flow of information through a system. Each process transforms inputs into outputs. Flow lines represent data flowing between nodes including processes, external entities and data stores.



The model generally starts with a context diagram showing the system as a single process connected to external entities outside of the system boundary. This process explodes to a lower level DFD that divides the system into smaller parts and balances the flow of information between parent and child diagrams. Many diagram levels may be needed to express a complex system.

Process Specifications

Primitive processes don't explode to a child diagram. They are usually described in a connected textual specification. This text is sometimes referred to as a mini-spec or process specification. It textually describes how the outputs are generated from the inputs.

Data Dictionary

When drawing data flow diagrams, the designer adds an entry for each data flow or store into a data dictionary. The data dictionary integrates the stack of diagrams into a cohesive model by defining all the names and data composition.

DIFFERENCE BETWEEN DFDs AND FLOWCHARTS

- Processes on DFDs can operate in parallel (at-the-same-time)
- Processes on flowcharts execute one at a time
- DFDs show the flow of data through a system
- Flowcharts show the flow of control (sequence and transfer of control)
- Processes on one DFD can have dramatically different timing
- Processes on flowcharts are part of a single program with consistent timing

Gane & Sarson DFD

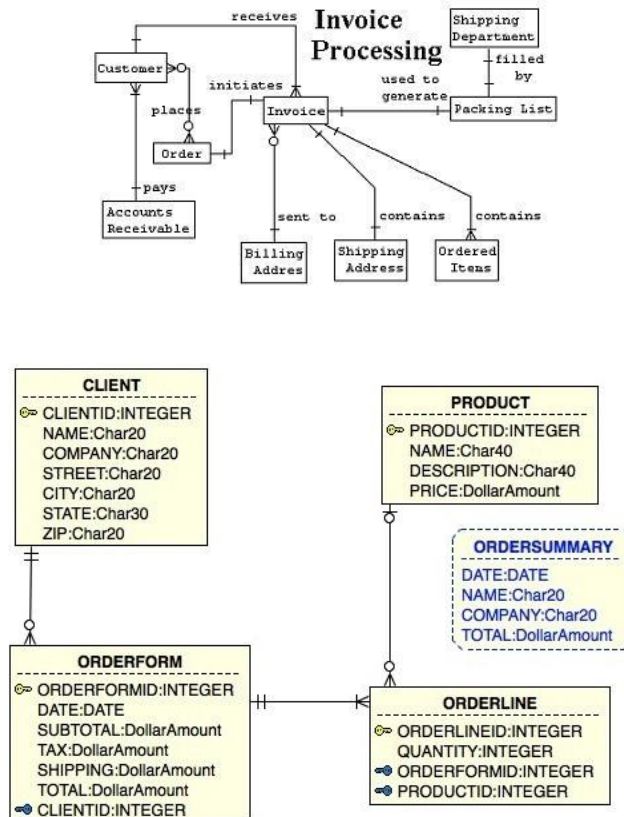
The Gane & Sarson style DFD is typically used for information systems. Process nodes are named round rectangles. Flows names are adjacent to perpendicular lines. Here we show the flow of information in a small software company.

DATA MODEL (Entity-Relation Diagram)

An entity-relation diagram, called an ERD illustrates the data structure of an information system. A database can be designed using logical and physical data models that highlight primary and foreign keys. Martin's Information Engineering notation is typically used for data models.

Logical Data Model

An entity-relation diagram (ERD), also referred to as a data model, typically shows the name of each entity, its list of attributes and relationships with other entities. An information system can be represented as a logical or physical data model. A logical data model typically uses longer more expressive names for entities and attributes but hides other details.



Physical Data Model

A physical data model can identify primary keys for each entity and foreign keys used to express referential relationships between entities. Attribute names in the physical model are usually more concise and sometimes predetermined by the existing database schema.

Data types, trigger definitions and other details can be assigned to each element of the data model. An SQL schema customized for a specific database can be generated from the data.

Often the data model is later implemented by generating SQL for an RDBMS where entities become tables and attributes becomes columns.

A class diagram gives a visual representation of the classes you need. And here is where you get to be really specific about object-oriented principles like inheritance and polymorphism.

Describing the interactions between those objects lets you better understand the responsibilities of the different objects, the behaviors they need to have.

— Other diagrams

There are many other diagrams we can use to model the system from different perspectives; interactions between objects, structure of the system, or the behavior of the system and how it responds to events.

It is always about selecting the right diagram for the right need. You should realize which diagrams will be useful when thinking about or discussing a situation that is not clear.

System Modeling

System modeling is the process of developing models of the system, with each model representing a different perspective of that system.

The most important aspect about a system model is that it leaves out detail; It's an abstract representation of the system.

The models are usually based on graphical notation, which is almost always based on the notations in the Unified Modeling Language (UML). Other models of the system like mathematical model; a detailed system description.

Models are used during the analysis process to help to elicit the requirements, during the design process to describe the system to engineers, and after implementation to document the system structure and operation.

Different Perspectives

We may develop a model to represent the system from different perspectives.

1. **External**, where you model the context or the environment of the system.
2. **Interaction**, where you model the interaction between components of a system, or between a system and other systems.
3. **Structural**, where you model the organization of the system, or the structure of the data being processed by the system.

Behavioral, where you model the dynamic behavior of the system and how it respond to events.

3.3.2 Development Tools

A **programming tool** or **software development tool** is a computer **program** that **software** developers use to create, debug, maintain, or otherwise support other programs and applications.

A **programming tool** or **software development tool** is a computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs, that can be combined together to accomplish a task, much as one might use multiple hands to fix a physical object. The most basic tools are a source code editor and a compiler or interpreter, which are used ubiquitously and continuously. Other tools are used more or less depending on the language, development methodology, and individual engineer, often used for a discrete task, like a debugger or profiler. Tools may be discrete programs, executed separately – often from the command line – or may be parts of a single large program, called an *integrated development environment* (IDE). In many cases, particularly for simpler use, simple ad hoc techniques are used instead of a tool, such as print debugging instead of using a debugger, manual timing (of overall program or section of code) instead of a profiler, or tracking bugs in a text file or spreadsheet instead of a bug tracking system.

The distinction between tools and applications is murky. For example, developers use simple databases (such as a file containing a list of important values) all the time as tools.[†] However a full-blown database is usually thought of as an application or software in its own right. For many years, computer-assisted software engineering (CASE) tools were sought after. Successful tools have proven elusive. In one sense, CASE tools emphasized design and architecture support, such as for UML. But the most successful of these tools are IDEs.

USES OF PROGRAMMING TOOLS

Translating from human to computer language

Modern computers are very complex and in order to productively program them, various abstractions are needed. For example, rather than writing down a program's binary representation a programmer will write a program in a programming language like C, Java or Python. Programming tools like assemblers, compilers and linkers translate a program from a human write-able and readable source language into the bits and bytes that can be executed by a computer. Interpreters interpret the program on the fly to produce the desired behavior.

These programs perform many well defined and repetitive tasks that would nonetheless be time consuming and error-prone when performed by a human, like laying out parts of a program in memory and fixing up the references between parts of a program as a linker does. Optimizing compilers on the other hand can perform complex transformations on the source code in order to improve the execution speed or other characteristics of a program. This allows a programmer to focus more on higher level, conceptual aspects of a program without worrying about the details of the machine it is running on.

Making program information available for humans

Because of the high complexity of software, it is not possible to understand most programs at a single glance even for the most experienced software developer. The abstractions provided by high-level programming languages also make it harder to understand the connection between the source code written by a programmer and the actual program's behavior. In order to find bugs in programs and to prevent creating new bugs when extending a program, a software developer uses some programming tools to visualize all kinds of information about programs.

For example, a debugger allows a programmer to extract information about a running program in terms of the source language used to program it. The debugger can compute the value of a variable in the source program from the state of the concrete machine by using information stored by the compiler. Memory debuggers can directly point out questionable or outright wrong memory accesses of running programs which may otherwise remain undetected and are a common source of program failures.

What is a development framework?

“A framework is a basic conceptual structure used to solve or address complex issues, usually a set of tools, materials or components. Especially in a software context the word is used as a name for different kinds of toolsets, component bases; it has since become a kind of buzzword or fashionable keyword.”

In the context of this document an ETL Framework is therefore a “fashionable” term for a development practice that will help solve the complex problem of developing the ETL solutions for Data Integration projects such as: Data Warehousing, Data Consolidation, Data Migration, and System Integration.

This development practice will comprise an ETL toolset along with: prebuilt reusable components, customizable templates and standard practices for solving certain problems.

Why use a development framework?

In the current data integration climate organizations are looking to reduce overheads and are turning to temporary staff and or third party organizations to implement solutions on a project basis.

In this way a “virtual” team is assembled comprising differing numbers of contractors, third party consultants and internal staff with a range of abilities, experience and knowhow. The members of these teams may be housed locally, remotely or a combination of both – with respect to the purchasing client.

There is often a high turnover of staff within the client organization, the third party organization and both may also make use of contractors to fulfil need.

This means that team members are often fluid between projects and within projects, especially if the project is for a long period of time.

The modern toolsets allow for many ways to solve a specific problem – many are considered good practice and therefore are perfectly valid ways of solving the problem; some are not of such good practice, but are easy to develop and therefore find their way into the solution mix. Many organizations talk of the notion of “Best Practice” however with a number of good and valid ways to approach and solve any given problem this is a difficult idea to pin down. In fact it is often subjective with one expert’s notion of best practice not necessarily being the same as another’s.

Good project organization applies standards to their development initiatives – but even these often apply more to style than to the practical sense of building the same component in the same way every time or solving a problem the same way every time.

With these challenges it is very easy for multiple projects within the same delivery organization, with differing staff of varying competence to build their ETL work quite differently; this increases the downstream maintenance costs to the client organization in that they have to support multiple solutions to a given problem. Without close coordination, substantial effort can be put into each project, re-developing, testing, deploying and then maintaining different code that effectively performs the same function. By identifying these common development areas using an ETL Development planning matrix, the Framework sets out to help an organization define a set of Good Practice service components that can be used consistently within that organization. A framework does not completely solve this problem; however by applying reusable components and templates and enforcing their use with review processes, the risk of inconsistent delivery is reduced.

Every project will have some specific problems to solve and these may well be unique, equally, every project will have a large portion of the requirements that are common (or could be common) to every other project.

The task of the Framework is to provide pre-designed, pre-built code components and templates to solve these areas that could be considered common, along with a planning process to ensure they are used; thus allowing more time to focus on those more difficult and or specific business problems.

Other benefits are also found in: faster and more accurate deliveries, more productive and potentially smaller development teams; reduced training in taking on new or replacement staff; reduced testing, reduced documentation, reduced effort in production support. These benefits collectively transcend into cost savings.

A framework needs to be developed to ensure that many programs can be developed in a consistent, efficient and accurate manner by a reasonably large team of developers, many of whom may well be located remotely or even offshore, physically separated from the design teams. Therefore a high level of structure is required in the organization, design and development processes to ensure timely deliveries and accurate data integration processing.

The framework sets out to break down the ETL process into the various activities that may/should/must take place and provides: organization, process, common terminology,

components, templates, design, standards and guidelines for building each piece of the ETL process.

This work breakdown should allow for parameter driven components and templates to be built which will substantially speed up the code delivery and increase the accuracy of the transformed data due to prior testing processes.

API (Application Programming Interface)

API(Application Programming Interface) is a set of instructions, standards or requirements that enables a software or app employ features/services of another app, platform or device for better services. In short, it's something that let apps communicate with each other.

For example, when we hit on the 'Connect Facebook' button on Candy Crush, it does not ask us to enter our Facebook account details. Rather, it accesses the data from the Facebook server and let us enjoy playing – all thanks to API.

An API is the base of all the apps that deal with data or enable communication between two products or services. It empowers a mobile application or platform to share its data with other apps/platforms and ease the user experience without involving the developers. On the top of it, APIs eliminate the need to build a similar program or platform from scratch; you can use the existing one of some other app/platform. Because of these factors, both app developers and business leaders focus on API development.

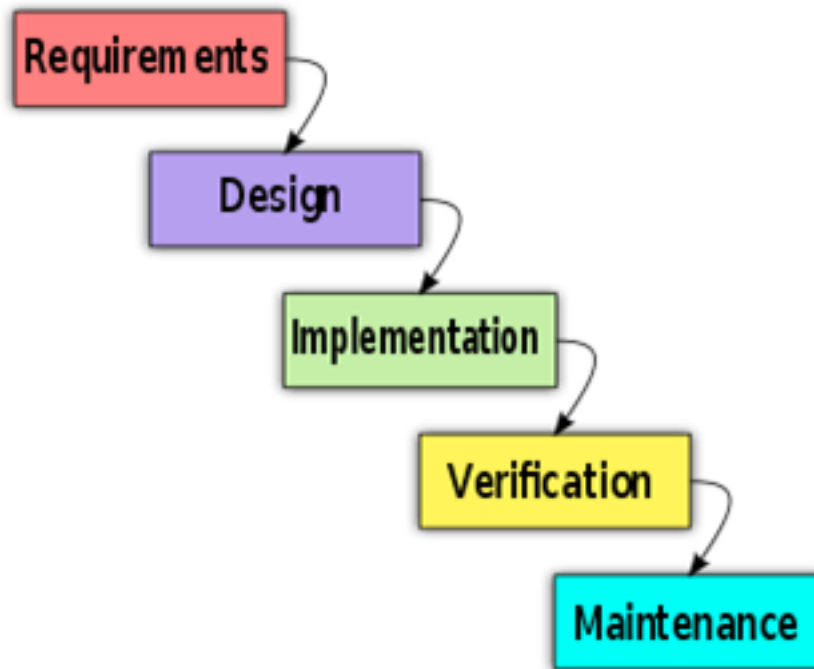
ASSESSMENT/ACTIVITY

You are expected to construct your chosen System's ERD and DFD and present it to class. Document for submission to Teacher-in-Charge.

3.4 Test Methodology/Procedures

Software Testing Methodology is defined as strategies and testing types used to certify that the Application Under Test meets client expectations. Test Methodologies include functional and non-functional testing to validate the AUT. Examples of Testing Methodologies are Unit Testing, Integration Testing, System Testing, Performance Testing etc. Each testing methodology has a defined test objective, test strategy, and deliverables.

Waterfall Model



What is it?

In the waterfall model, software development progress through various phases like Requirements Analysis, Design etc. - **sequentially**.

In this model, the next phase begins only when the earlier phase is completed.

What Is The Testing Approach?

The first phase in the waterfall model is the requirements phase in which all the project requirements are completely defined before starting the testing. During this phase, the test team brainstorms the scope of testing, test strategy and drafts a detailed test plan.

Only once the design of software is complete, the team will move on to execution of the test cases to ensure that the developed software behaves as it expected.

In this methodology, the testing team proceeds to the next phase only when the previous phase is completed.

Advantages

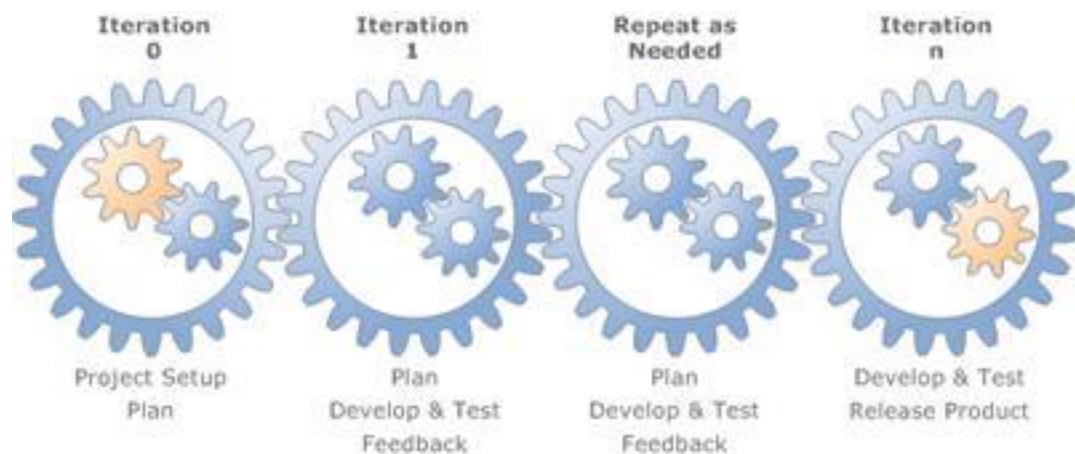
This software Engineering model is very simple to plan and manage. Hence, projects, where requirements are clearly defined and stated beforehand, can be easily tested using a waterfall model.

Disadvantages

In the waterfall model, you can begin with the next phase only once the previous phase is completed. Hence, this model cannot accommodate unplanned events and uncertainty.

This methodology is not suitable for projects where the requirements change frequently.

Iterative development



What is it?

In this model, a big project is divided into small parts, and each part is subjected to multiple iterations of the waterfall model. At the end of an iteration, a new module is developed or an existing module is enhanced. This module is integrated into the software architecture and the entire system is tested all together

What is the testing Approach?

As soon as iteration is completed, the entire system is subjected to testing. Feedback from testing is immediately available and is incorporated in the next cycle. The testing time required in successive iteration can be reduced based on the experience gained from past iterations.

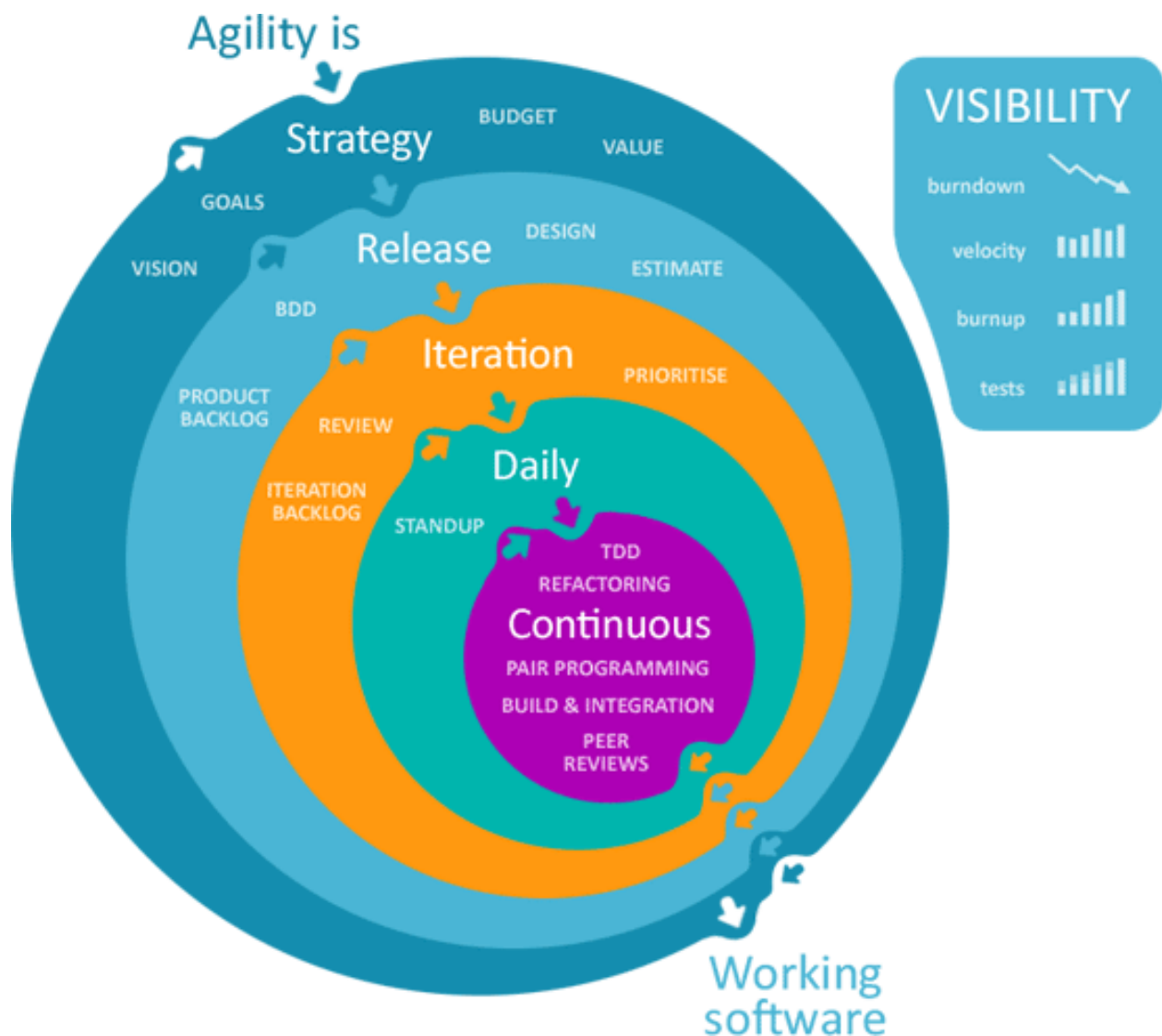
Advantages

The main advantage of iterative development is the test feedback is immediately available at the end of each cycle.

Disadvantages

This model increases communication overheads significantly since, at the end of each cycle, feedback about deliverables, effort etc must be given.

Agile methodology



What is it?

Traditional software development methodologies work on the premise that software requirements remain constant throughout the project. But with an increase in complexity, the requirements undergo numerous changes and continuously evolve. At times, the customer himself is not sure what he wants. Though the iterative model addresses this issue, it's still based on the waterfall model.

In Agile methodology, software is developed in incremental, rapid cycles. Interactions amongst customers, developers and client are emphasized rather than processes and tools. The agile methodology focuses on responding to change rather than extensive planning.

What Is the Testing Approach?

Incremental testing is used in agile development methods and hence, every release of the project is tested thoroughly. This ensures that any bugs in the system are fixed before the next release.

Advantages

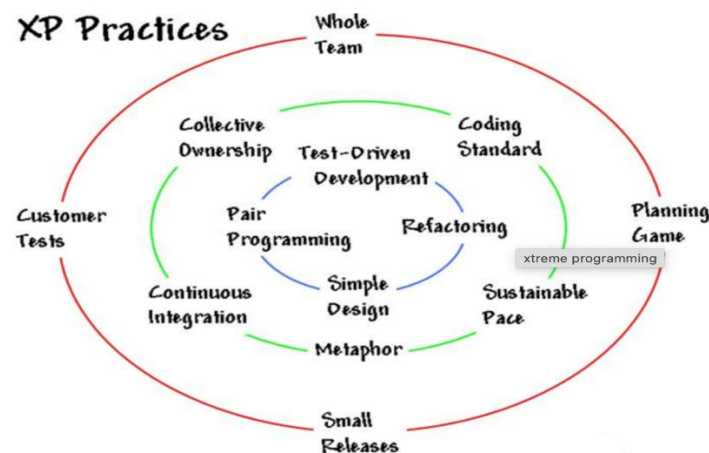
It is possible to make changes in the project at any time to comply with the requirements.

This incremental testing minimizes risks.

Disadvantages

Constant client interaction means added time pressure on all stakeholders including the client themselves, software development and test teams.

Extreme Programming



What is it?

Extreme programming is a type of agile methodology which beliefs in short development cycles. A project is divided into simple engineering tasks. Programmers code a simple piece of software and get back to the customer for feedback. Review points from the customer are incorporated and the developers proceed with the next task.

In extreme programming developers usually, work in pairs.

Extreme Programming is used in places where customer requirements are constantly changing.

What Is The Testing Approach?

Extreme programming follows a Test-driven development which is described as follows -

1. Add a Test Case to the test suite to verify the new functionality which is yet to be developed
2. Run all the tests and obviously the new test case added must fail since the functionality is not coded yet
3. Write some code to implement the feature/functionality
4. Run the test suite again. This time, the new test case should pass since the functionality has been coded

Advantages

Customers having a vague software design in mind could use extreme programming.

Continuous testing and continuous integration of small releases ensure software code is delivered is of high quality

Disadvantages

Meetings amongst the software development team and clients add to time requirements.

Which Software Methodology to choose?

There are tons of methodologies available for software development and its corresponding testing. Each testing technique and methodology is designed for a specific purpose and has its relative merits and demerits.

Selection of a particular methodology depends on many factors such as the nature of a project, client requirement, project schedule, etc.

From a testing perspective, some methodologies push for testing input early in the development life cycle, while others wait until a working model of the system is ready.

How to setup software testing methodologies?

Software testing methodologies should not be set up just for the sake of testing software code. The big picture should be considered and the prime goal of the project should be satisfied with the testing methodology.

Scheduling

Realistic scheduling is the key to the implementation of successful testing methodology and the schedule should meet the needs of every member of the team.

Defined deliverables

In order to keep all the members of the team on the same page, well-defined deliverables should be provided. The deliverables should contain direct content without any ambiguity.

Test approach

Once scheduling is complete and defined deliverables are made available, the testing team should be able to formulate the right test approach. Definition documents and developer meetings should indicate the team about the best test approach that can be used for the project.

Reporting

Transparent reporting is very difficult to achieve, but this step determines the effectiveness of the testing approach used in the project.

Assessment:

Device an Evaluation Testing for your propose software. Based on your Objectives.

References:

<https://www.guru99.com/testing-methodology.html>

3.5 System Requirements

This part of the system requirements define what the proposed system needs in terms of computers/hardware, software that will be needed to run the application.

3.5.1 Equipment/Hardware. Enumerate the organization's equipment/hardware to be used in the proposed system together with their specifications.

Equipment/Hardware	Description

Example:

Workstation	
Category	Specifications
Processor	I5 10 th gen.
Hard Drive & Memory	1tb HDD RAM: 4gb Integrated graphics

3.5.2 Software. Enumerate the organization's software to be used for the proposed system together with their specifications.

Software	Description

Example:

Category	Specifications
Operating System	Windows 10
Productivity Tool	Microsoft Office Pro
Application	Adobe Acrobat Reader 8.0 Adobe Photoshop CS2
Web Browser	Microsoft Edge Google Chrome

3.6 Quality Plan

A **quality plan** sets out the desired product qualities and how these are assessed and defines the most significant quality attributes. The quality plan should **define the quality assessment process**. It should set out which organizational standards should be applied and, where necessary, define new standards to be used. Quality plans should be short, succinct documents; if they are too long, no-one will read them. Quality plan structure:

- Product introduction;
- Product plans;
- Process descriptions;
- Quality goals;
- Risks and risk management.

Quality management is particularly important for large, complex systems. The quality documentation is a record of progress and supports continuity of development as the development team changes. For smaller systems, quality management needs less documentation and should focus on establishing a quality culture. Techniques have to evolve when agile development is used.

Software quality

Quality, simplistically, means that **a product should meet its specification**. This is problematic for software systems because there is a tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.); some quality requirements are difficult to specify in an unambiguous way; software specifications are usually incomplete and often inconsistent. The focus may be 'fitness for purpose' rather than specification conformance.

Software fitness for purpose

- Has the software been properly tested?
- Is the software sufficiently dependable to be put into use?
- Is the performance of the software acceptable for normal use?
- Is the software usable?
- Is the software well-structured and understandable?
- Have programming and documentation standards been followed in the development process?

The subjective quality of a software system is largely based on its non-functional characteristics. This reflects practical user experience - if the software's functionality is not what is expected, then users will often just work around this and find other ways to do what they want to do. However, if the software is unreliable or too slow, then it is practically impossible for them to achieve their goals.

Page Break

Software quality attributes

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency

Robustness	Complexity	Learnability
------------	------------	--------------

It is not possible for any system to be optimized for all of these attributes - for example, improving robustness may lead to loss of performance. The **quality plan should therefore define the most important quality attributes** for the software that is being developed. The plan should also include a definition of the quality assessment process, an agreed way of assessing whether some quality, such as maintainability or robustness, is present in the product.

The quality of a developed product is influenced by the quality of the production process. This is important in software development as some product quality attributes are hard to assess. However, there is a very complex and poorly understood relationship between software processes and product quality. The application of individual skills and experience is particularly important in software development. External factors such as the novelty of an application or the need for an accelerated development schedule may impair product quality.

Quality managers should aim to develop a 'quality culture' where everyone responsible for software development is committed to achieving a high level of product quality. They should encourage teams to take responsibility for the quality of their work and to develop new approaches to quality improvement. They should support people who are interested in the intangible aspects of quality and encourage professional behavior in all team members.

Software standards

Software standards define the required attributes of a product or process. They play an important role in quality management. Standards may be international, national, organizational or project standards. Encapsulation of best practices avoids repetition of past mistakes. They are a framework for defining what quality means in a particular setting i.e. that organization's view of quality. They provide continuity - new staff can understand the organization by understanding the standards that are used.

Product standards apply to the software product being developed. They include document standards, such as the structure of requirements documents, documentation standards, such as a standard comment header for an object class definition, and coding standards, which define how a programming language should be used. Product standards may include:

- Design review form
- Requirements document structure
- Method header format
- Java programming style
- Project plan format
- Change request form

Process standards define the processes that should be followed during software development. Process standards may include definitions of specification, design and validation processes, process support tools and a description of the documents that should be written during these processes. Process standards may include:

- Design review conduct
- Submission of new code for system building
- Version release process
- Project plan approval process
- Change control process

- Test recording process

Problems: Standards may not be seen as relevant and up to date by software engineers. They often involve too much bureaucratic form filling. If they are unsupported by software tools, tedious form filling work is often involved to maintain the documentation associated with the standards.

Practitioners should be involved in development of standards. Engineers should understand the rationale underlying a standard. Review standards and their usage regularly. Standards can quickly become outdated, and this reduces their credibility amongst practitioners. Detailed standards should have specialized tool support. Excessive clerical work is the most significant complaint against standards. Web-based forms are not good enough.

An international set of standards that can be used as a basis for developing quality management systems. **ISO 9001, the most general of these standards, applies to organizations that design, develop, and maintain products, including software.** The ISO 9001 standard is a framework for developing software standards. It sets out general quality principles, describes quality processes in general and lays out the organizational standards and procedures that should be defined. These should be documented in an organizational quality manual.

Basics of ISO 9001 certification:

- Quality standards and procedures should be documented in an organizational quality manual.
- An external body may certify that an organization's quality manual conforms to ISO 9000 standards.
- Some customers require suppliers to be ISO 9000 certified although the need for flexibility here is increasingly recognized.

The ISO 9001 certification is inadequate because it defines quality to be the conformance to standards. It takes no account of quality as experienced by users of the software. For example, a company could define test coverage standards specifying that all methods in objects must be called at least once. Unfortunately, this standard can be met by incomplete software testing that does not include tests with different method parameters. So long as the defined testing procedures are followed and test records maintained, the company could be ISO 9001 certified.

Reviews and inspections

Reviews and inspections involve a group who examines part or all of a process or system and its documentation to find potential problems. Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management. There are different types of review with different objectives:

- Inspections for defect removal (product);
- Reviews for progress assessment (product and process);
- Quality reviews (product and standards).

Phases in the review process:

- **Pre-review activities** are concerned with review planning and review preparation.
- During the **review meeting**, an author of the document or program being reviewed should 'walk through' the document with the review team.
- **Post-review activities** address the problems and issues that have been raised during the review meeting.

The processes suggested for reviews assume that the review team has a face-to-face meeting to discuss the software or documents that they are reviewing. However, project teams are now often distributed, sometimes across countries or continents, so it is impractical for team members to meet face to face. **Remote/distributed reviewing** can be supported using shared documents where each review team member can annotate the document with their comments.

Program inspections are peer reviews where engineers examine the source of a system with the aim of discovering anomalies and defects. Inspections do not require execution of a system so may be used before implementation. They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.). They have been shown to be an effective technique for discovering program errors.

Checklist of common errors should be used to drive the inspection. Error checklists are programming language dependent and reflect the characteristic errors that are likely to arise in the language. In general, the 'weaker' the type checking, the larger the checklist. Examples: Initialization, Constant naming, loop termination, array bounds, etc.

Quality management and agile development

Quality management in agile development is informal rather than document-based. It relies on establishing a quality culture, where all team members feel responsible for software quality and take actions to ensure that quality is maintained. The agile community is fundamentally opposed to what it sees as the bureaucratic overheads of standards-based approaches and quality processes as embodied in ISO 9001.

Good agile practices:

- **Check before check-in:** Programmers are responsible for organizing their own code reviews with other team members before the code is checked in to the build system.
- **Never break the build:** Team members should not check in code that causes the system to fail. Developers have to test their code changes against the whole system and be confident that these work as expected.
- **Fix problems when you see them:** If a programmer discovers problems or obscurities in code developed by someone else, they can fix these directly rather than referring them back to the original developer.

The review process in agile software development is usually informal. In Scrum, there is a review meeting after each iteration of the software has been completed (a sprint review), where quality issues and problems may be discussed. In Extreme Programming, pair programming ensures that code is constantly being examined and reviewed by another team member.

Pair programming is an approach where 2 people are responsible for code development and work together to achieve this. Code developed by an individual is therefore constantly being examined and reviewed by another team member. Pair programming leads to a deep knowledge of a program, as both programmers have to understand the program in detail to continue development. This depth of knowledge is difficult to achieve in inspection processes and pair programming can find bugs that would not be discovered in formal inspections. Pair programming weaknesses include:

- **Mutual misunderstandings:** Both members of a pair may make the same mistake in understanding the system requirements. Discussions may reinforce these errors.

- **Pair reputation:** Pairs may be reluctant to look for errors because they do not want to slow down the progress of the project.
- **Working relationships:** The pair's ability to discover defects is likely to be compromised by their close working relationship that often leads to reluctance to criticize work partners.

When a **large system** is being developed for an external customer, **agile approaches to quality management** with minimal documentation **may be impractical**. If the customer is a large company, it may have its own quality management processes and may expect the software development company to report on progress in a way that is compatible with them. Where there are several geographically distributed teams involved in development, perhaps from different companies, then informal communications may be impractical. For long-lifetime systems, the team involved in development will change. Without documentation, new team members may find it impossible to understand development.

Software measurement

Software measurement is concerned with deriving a numeric value for an attribute of a software product or process. This allows for objective comparisons between techniques and processes. Although some companies have introduced measurement programs, most organizations still don't make systematic use of software measurement. There are few established standards in this area.

Software metric is any type of measurement which relates to a software system, process or related documentation: lines of code in a program, the fog index (a code readability test), number of person-days required to develop a component. Allow the software and the software process to be quantified. May be used to predict product attributes or to control the software process. Product metrics can be used for general predictions or to identify anomalous components. Process metrics include:

- **The time taken for a particular process to be completed** This can be the total time devoted to the process, calendar time, the time spent on the process by particular engineers, and so on.
- **The resources required for a particular process** Resources might include total effort in person-days, travel costs or computer resources.
- **The number of occurrences of a particular event** Examples of events that might be monitored include the number of defects discovered during code inspection, the number of requirements changes requested, the number of bug reports in a delivered system and the average number of lines of code modified in response to a requirements change.

Software measurements can be used to:

- **To assign a value to system quality attributes** by measuring the characteristics of system components, such as their cyclomatic complexity, and then aggregating these measurements, you can assess system quality attributes, such as maintainability.
- **To identify the system components whose quality is sub-standard.** For example, you can measure components to discover those with the highest complexity. These are most likely to contain bugs because the complexity makes them harder to understand.

Software metrics assumptions:

- A software property can be measured accurately.
- The relationship exists between what we can measure and what we want to know. We can only measure internal attributes but are often more interested in external software attributes.
- The relationship has been formalised and validated.
- It may be difficult to relate what can be measured to desirable external quality attributes.

Problems with measurement in industry:

- It is impossible to quantify the return on investment of introducing an organizational metrics program.
- There are no standards for software metrics or standardized processes for measurement and analysis.
- In many companies, software processes are not standardized and are poorly defined and controlled.
- Most work on software measurement has focused on code-based metrics and plan-driven development processes. However, more and more software is now developed by configuring ERP systems or COTS.
- Introducing measurement adds additional overhead to processes.

Software measurement and metrics are the basis of **empirical software engineering**. This is a research area in which experiments on software systems and the collection of data about real projects has been used to form and validate hypotheses about software engineering methods and techniques. Research on empirical software engineering, this has not had a significant impact on software engineering practice. It is difficult to relate generic research to a project that is different from the research study.

A quality **product metric** should be a predictor of product quality. Classes of product metrics:

- Dynamic metrics which are collected by measurements made of a program in execution;
- Static metrics which are collected by measurements made of the system representations;
- Dynamic metrics help assess efficiency and reliability;
- Static metrics help assess complexity, understandability, and maintainability.

Dynamic metrics are closely related to software quality attributes. It is relatively easy to measure the response time of a system (performance attribute) or the number of failures (reliability attribute). Static metrics have an indirect relationship with quality attributes. You need to try and derive a relationship between these metrics and properties such as complexity, understandability, and maintainability.

System **components can be analyzed separately** using a range of metrics. The values of these metrics may then compare for different components and, perhaps, with historical measurement data collected on previous projects. Anomalous measurements, which deviate significantly from the norm, may imply that there are problems with the quality of these components.

When you collect quantitative data about software and software processes, you have to analyze that data to understand its meaning. It is easy to misinterpret data and to make inferences that

are incorrect. You cannot simply look at the data on its own. You must also consider the context where the data is collected.

Processes and products that are being measured are not insulated from their environment. The business environment is constantly changing, and it is impossible to avoid changes to work practice just because they may make comparisons of data invalid. Data about human activities cannot always be taken at face value. The reasons why a measured value changes are often ambiguous. These reasons must be investigated in detail before drawing conclusions from any measurements that have been made.

Software analytics is analytics on software data for managers and software engineers with the aim of empowering software development individuals and teams to gain and share insight from their data to make better decisions. The automated collection of user data by software product companies when their product is used. If the software fails, information about the failure and the state of the system can be sent over the Internet from the user's computer to servers run by the product developer. The use of open-source software available on platforms such as Sourceforge and GitHub and open-source repositories of software engineering data. The source code of open-source software is available for automated analysis, and this can sometimes be linked with data in the open-source repository.

Software analytics is still immature, and it is too early to say what effect it will have. Not only are there general problems of 'big data' processing, our knowledge depends on collected data from large companies. This is primarily from software products, and it is unclear if the tools and techniques that are appropriate for products can also be used with custom software. Small companies are unlikely to invest in the data collection systems that are required for automated analysis so may not be able to use software analytics.

Reading links

1. Defining and Assessing Software Quality by Quality Models, <http://mediatum.ub.tum.de/doc/1169637/955300.pdf>
2. Applying the ISO 9126 Quality Model to Test Specifications, <https://subs.emis.de/LNI/Proceedings/Proceedings105/gi-proc-105-024.pdf>

Assessment:

1. Create a Software Quality Questionnaire for your capstone proposal project. Make sure that it will help you answer your questions or attain your objectives.

References:

CS 530 - Software Engineering Quality Management. (n.d.). Retrieved from cs.ccsu:

[https://cs.ccsu.edu/~stan/classes/CS530/Notes18/24-](https://cs.ccsu.edu/~stan/classes/CS530/Notes18/24-QualityManagement.html#:~:text=A%20quality%20plan%20sets%20out,new%20standards%20to%20be%20used.)

[QualityManagement.html#:~:text=A%20quality%20plan%20sets%20out,new%20standards%20to%20be%20used.](https://cs.ccsu.edu/~stan/classes/CS530/Notes18/24-QualityManagement.html#:~:text=A%20quality%20plan%20sets%20out,new%20standards%20to%20be%20used.)

Sommerville, I. (n.d.). Software Engineering. Retrieved from My Courses:

https://mycourses.aalto.fi/pluginfile.php/1177979/mod_resource/content/1/Sommerville-Software-Engineering-10ed.pdf

3.7 Evaluation Plan

What is Evaluation?

The systematic collection of information about a program in order to enable stakeholders to better understand the program, to improve program effectiveness, and/or to make decisions about future programming.

What's in it for you?

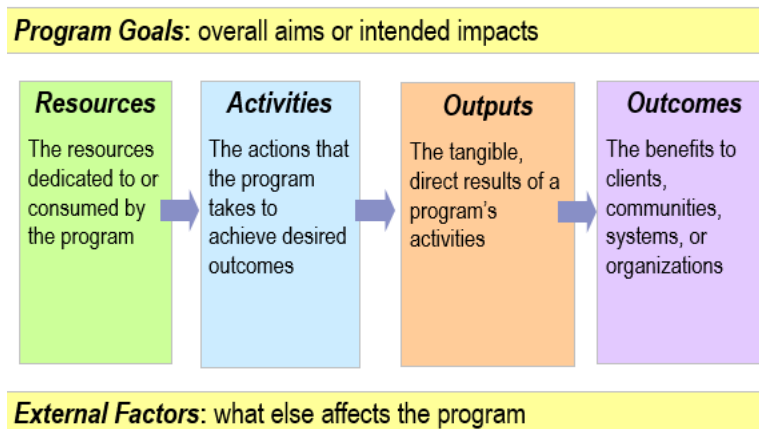
- Understand and improve your program
- Test the theory underlying your program
- Tell your program's story
- Be accountable
- Inform the field
- Support fundraising efforts

Evaluation Principles

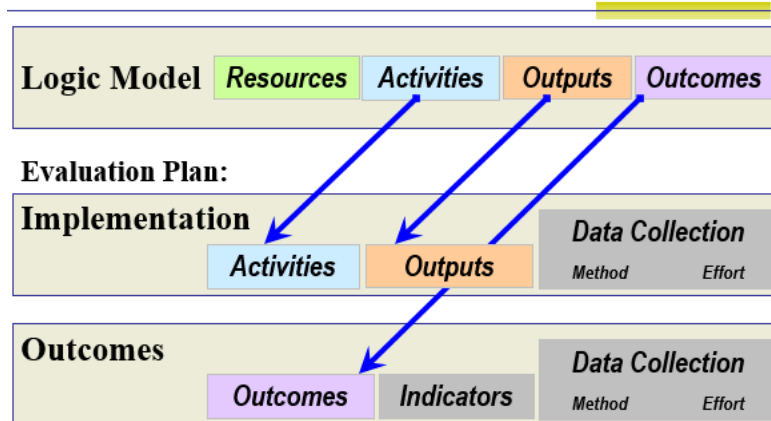
Evaluation is most effective when it:

- Is connected to program planning and delivery
- Involves the participation of stakeholders
- Supports an organization's capacity to learn and reflect
- Respects the community served by the program
- Enables the collection of the most information with the least effort

Logic Model



Putting Your Plans Together



Implementation and Outcomes

- Evaluating Outcomes: What changes occurred as a result of your work?
- Evaluating Implementation: What did you do? How well did you do it?

Evaluating Outcomes

Outcomes: the changes you expect to see as a result of your work

Indicators: the specific, measurable characteristics or changes that represent achievement of an outcome. They answer the question: How will I know it?

Indicators = What to Measure

- Meaningful
- Direct
- Useful
- Practical

Direct v. Indirect Indicators

Outcome	Indicator
Participating new mothers have their children immunized.	Indirect: #/% of new mothers who are aware of importance of immunization. Direct: #/% of children of participating mothers who are up-to-date in immunizations.
Increase referrals to your services from targeted doctors.	Indirect: #/% of increase in your client base. Direct: #/% of increase in your client base from targeted doctors.
Targeted teens learn about certain environmental health hazards.	Indirect: # of students who receive brochure on topic. Direct: #/% of students who ID 3 health hazards.

Example

Outcome	Indicators	Data Collection Method	Data Collection Effort (have, low, med, high)
Participants learn job-seeking skills	#/% of participants who can meet criteria in mock interview	Observation of mock interview conducted at end of training session using observation checklist	Low
	#/% of participants who develop a quality resume	Job counselors review resumes based on quality checklist	Have
Participants obtain and carry out job interviews	#/% of participants who go on at least 2 job interviews	Tracking by job counselors	Have

Evaluating Implementation

- **Activities and Outputs:** The “what” —the work you did, and the tangible results of that work
- **Additional Questions:** The “why” —understanding how well you did, and why

Understanding how well you did

What information will help you understand your program implementation? Think about:

- Participation
- Quality
- Satisfaction
- Context

Example

Activities	Outputs & Implementation Questions	Data Collection Method	Data Collection Effort (have, low, med, high)
<ul style="list-style-type: none"> ■ Develop/revise curriculum for training series ■ Meet with potential program clients ■ Provide training session series to two groups of clients 	Outputs <ul style="list-style-type: none"> ■ Updated curriculum ■ 2 training session series held ■ #/rate of participation by group 	<ul style="list-style-type: none"> ■ Program records ■ Records ■ Records, logs 	<ul style="list-style-type: none"> ■ Have ■ Have
	Questions <ul style="list-style-type: none"> ■ Are we getting the clients we expected to get?(Partic.) ■ Are they satisfied w/ training? What did they like most, least? (Satisfaction) 	<ul style="list-style-type: none"> ■ Review of participant intake data ■ Participant survey 	<ul style="list-style-type: none"> ■ Low ■ Med

Data Collection

Determine what methods will you use to collect the information you need?

- Choose the method
- Decide which people, or records will be the source of the information
- Determine the level of effort involved in using that method with that population.

Data Collection Methods

- Review documents
- Observe
- Talk to people
- Collect written information
- Pictorial/multimedia

Issues to Consider

- Resist pressure to “prove”
- Start with what you already collect
- Consider the level of effort it will take to gather the data.
- Prioritize. What do you need to collect now, and what can wait until later?

Data Collection: Level of Effort

- Instrument development
- Cost/practicality of actually collecting data
- Cost of analyzing and presenting data

Quantitative Data

- Pieces of information that can be expressed in numerical terms, counted, or compared on a scale
- Collected in surveys, attendance logs, etc.

Qualitative Data

- Usually in narrative form—not using numbers
- Collected through focus groups, interviews, open-ended questionnaire items, but also poetry, stories, diaries, and notes from observations

Both Types of Data are Valuable

- Qualitative information can provide depth and insight about quantitative data
- Some information can only be collected and communicated qualitatively
- Both methods require a systematic approach

What do your data tell you?

- Are there patterns that emerge?
 - Patterns for sub-groups of your population?
 - Patterns for different components of your program?
- What questions do the data raise?
 - What is surprising? What stands out?
- What are other ways the data should be analyzed?
- What additional information do you need to collect?

Communicating Findings

- Who is the information for?
- How will you tell them what you know?

“Information that is not effectively shared with others will not be effectively used.”

Source: Building a Successful Evaluation Center for Substance Abuse Prevention

Audience: Who needs the findings, and what do they need?

Who are the audiences for your results? Which results?

- Staff
- Board
- Funders
- Partners
- Other agencies
- Public

Different ways to communicate

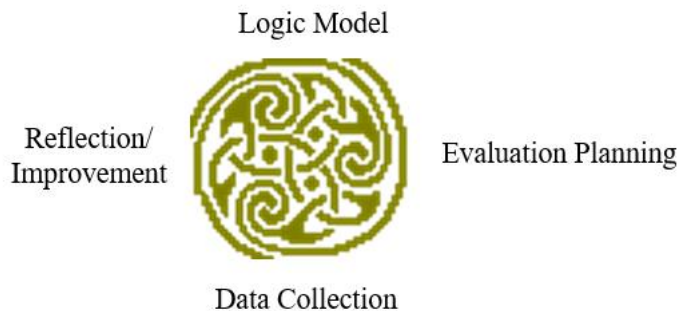
Decide what format is appropriate for different audiences.

- Written report
- Short summaries
- Film or videotape
- Pictures, displays
- PowerPoint presentations
- Graphs and other visuals

Whatever communication strategy you choose:

- Link the findings to the program’s desired outcomes
- Include the “good, the bad, and the ugly”
- Be sure you can support your claims
- Acknowledge knowledge gaps

Continuous Learning Cycle



To effectively complete or implement most projects, an evaluation plan is needed. There are two basic types of evaluation plans:

1. Formative
2. Summative

Page Break

Formative Evaluation Plan

A formative evaluation plan is completed before or during the project. A formative evaluation has the following characteristics:

- Evaluates upcoming or continuing activities of a project
- Covers activities from development to implementation stages
- Contains reviews from principal investigators, evaluators, and governing committees

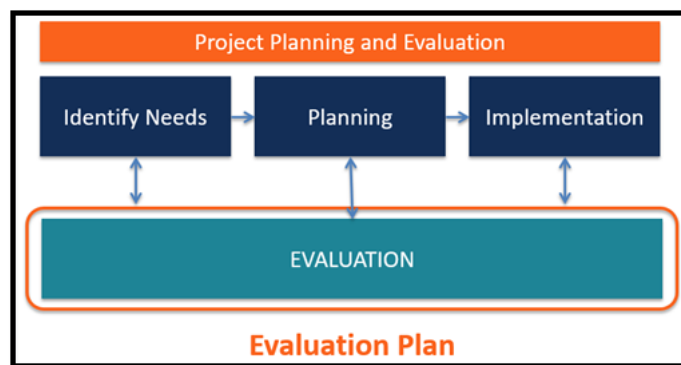
Summative Evaluation Plan

A summative evaluation plan “sums” up the project. As such, it is written at a project’s completion. A summative plan is characterized by having the following features:

- Evaluates whether the goals that were achieved are the goals that were set. If not, the evaluation should state the extent of the variation and the reasons for it.
- Contains the details of the outcomes and information obtained during the project.
- Reports the outcome of the project to the principal investigator of the project, evaluators, and any governing committees.

There are some common content elements that should be included in an evaluation plan regardless of whether it is classified as formative or summative. They are as follows:

- The project to be evaluated
- Purpose of evaluation
- Key evaluation questions
- Notation of methods used, including methods for collecting and analyzing all the necessary data
- The reports and reviews of the stakeholders and investors directly involved in the project
- Resources needed to fund and facilitate the project
- Expected findings and outcomes of the project, as well as the expected time of the final report



How to Write an Evaluation Plan

Before writing an evaluation plan for your business, it is advisable to consult prior plans to see if certain formats are preferred. In general, however, the plans should include methods such as interviews, administration of questionnaires, and consultation that will be carried out during the project. Other items include:

1. **Clear title** – The recommended way of writing the title is that you should write it on a page of its own. The title page should contain a recognizable name of the project, dates of the project, and the general focus of the evaluation plan.
2. **Uses and users of the evaluation plan** – It is essential to describe the use of the evaluation plan clearly. For transparency and accountability, under this section, you should clearly show the users of the plan. Again, you should describe the involvement of stakeholders and the financiers of the project in this same section.
3. **Project description** – Under this section, the developer of the evaluation plan should critically assess and describe what the entire project is all about. Here, it is essential to state what the project focuses on achieving, and the process for evaluating how successfully the project met its goals.
4. **Methodology** – In this section, an evaluation plan should clearly state the methods that will be used to collect data, expected data sources, and the roles and responsibilities of each participant in the project. This is the section that should also describe which methods will be used to ensure that the project is completed successfully.
5. **Analysis** – This section contains a thorough analysis of the project. It will show findings and reasons for any unexpected outcomes. It may also contain data analyses done before the project's completion and how it affected the project's continuation.
6. **Sharing plan** – In most cases, the sharing plan section is often overlooked, despite the fact that it can play a major role. Toward the end of the plan, there should be a proper way of sharing evaluation findings. This section should also state how the findings and outcomes of the project will reach (be reported to) the involved stakeholders.

The Importance of an Evaluation Plan

1. An evaluation plan is a valuable asset that can help ensure that a project runs smoothly. A well-documented plan states the roles of all participants in the project and the sources of all resources. This implies that there should be minimal delays, as everything should have been communicated ahead of time. Furthermore, if the plan clearly states the dates on which specific activities should take place, then the involved participants will be encouraged to be right on schedule.
2. A good evaluation plan should cater to the smooth running of the project from its initial stages to its completion.
2. An effective evaluation plan will also ensure better results in upcoming projects of the same nature.
3. A well-documented evaluation plan enhances transparency and accountability. Involved participants, contractors, and stakeholders share the plan among themselves. The methodology section clearly outlines and describes how they obtained each finding and outcome.
4. The practice of using evaluation plans should improve the success and effectiveness of projects undertaken by an organization. If the plans are well documented and filed, the organization can learn from previous projects and be able

to better gauge the success of certain projects and project practices. The plans can also come in handy in helping the foundation or organization make critical decisions. This is because the information in the plan is not just gathered randomly – it is obtained after thorough research and evaluation of the project.

WHY SHOULD YOU HAVE AN EVALUATION PLAN?

After many late nights of hard work, more planning meetings than you care to remember, and many pots of coffee, your initiative has finally gotten off the ground. Congratulations! You have every reason to be proud of yourself and you should probably take a bit of a breather to avoid burnout. Don't rest on your laurels too long, though--your next step is to monitor the initiative's progress. If your initiative is working perfectly in every way, you deserve the satisfaction of knowing that. If adjustments need to be made to guarantee your success, you want to know about them so you can jump right in there and keep your hard work from going to waste. And, in the worst case scenario, you'll want to know if it's an utter failure so you can figure out the best way to cut your losses. For these reasons, evaluation is extremely important.

There's so much information on evaluation out there that it's easy for community groups to fall into the trap of just buying an evaluation handbook and following it to the letter. This might seem like the best way to go about it at first glance-- evaluation is a huge topic and it can be pretty intimidating. Unfortunately, if you resort to the "cookbook" approach to evaluation, you might find you end up collecting a lot of data that you analyze and then end up just filing it away, never to be seen or used again.

Instead, take a little time to think about what exactly you really want to know about the initiative. Your evaluation system should address simple questions that are important to your community, your staff, and (last but not least!) your funding partners. Try to think about financial and practical considerations when asking yourself what sort of questions you want answered. The best way to insure that you have the most productive evaluation possible is to come up with an evaluation plan.

HERE ARE A FEW REASONS WHY YOU SHOULD DEVELOP AN EVALUATION PLAN:

- It guides you through each step of the process of evaluation
- It helps you decide what sort of information you and your stakeholders really need
- It keeps you from wasting time gathering information that isn't needed
- It helps you identify the best possible methods and strategies for getting the needed information
- It helps you come up with a reasonable and realistic timeline for evaluation
- Most importantly, it will help you improve your initiative!

WHEN SHOULD YOU DEVELOP AN EVALUATION PLAN?

As soon as possible! The best time to do this is before you implement the initiative. After that, you can do it anytime, but the earlier you develop it and begin to implement it, the better off your initiative will be, and the greater the outcomes will be at the end.

Remember, evaluation is more than just finding out if you did your job. It is important to use evaluation data to improve the initiative along the way.

WHAT ARE THE DIFFERENT TYPES OF STAKEHOLDERS AND WHAT ARE THEIR INTERESTS IN YOUR EVALUATION?

We'd all like to think that everyone is as interested in our initiative or project as we are, but unfortunately that isn't the case. For community health groups, there are basically three groups of people who might be identified as stakeholders (those who are interested, involved, and invested in the project or initiative in some way): community groups, grantmakers/funders, and university-based researchers. Take some time to make a list of your project or initiative's stakeholders, as well as which category they fall into.

WHAT ARE THE TYPES OF STAKEHOLDERS?

- **Community groups:** Hey, that's you! Perhaps this is the most obvious category of stakeholders, because it includes the staff and/or volunteers involved in your initiative or project. It also includes the people directly affected by it--your targets and agents of change.
- **Grantmakers and funders:** Don't forget the folks that pay the bills! Most grantmakers and funders want to know how their money's being spent, so you'll find that they often have specific requirements about things they want you to evaluate. Check out all your current funders to see what kind of information they want you to be gathering. Better yet, find out what sort of information you'll need to have for any future grants you're considering applying for. It can't hurt!
- **University-based researchers:** This includes researchers and evaluators that your coalition or initiative may choose to bring in as consultants or full partners. Such researchers might be specialists in public health promotion, epidemiologists, behavioral scientists, specialists in evaluation, or some other academic field. Of course, not all community groups will work with university-based researchers on their projects, but if you choose to do so, they should have their own concerns, ideas, and questions for the evaluation. If you can't quite understand why you'd include these folks in your evaluation process, try thinking of them as auto mechanics--if you want them to help you make your car run better, you will of course include them in the diagnostic process. If you went to a mechanic and started ordering him around about how to fix your car without letting him check it out first, he'd probably get pretty annoyed with you. Same thing with your researchers and evaluators: it's important to include them in the evaluation development process if you really want them to help improve your initiative.

Each type of stakeholder will have a different perspective on your organization as well as what they want to learn from the evaluation. Every group is unique, and you may find that there are other sorts of stakeholders to consider with your own organization. Take some time to brainstorm about who your stakeholders are before you begin making your evaluation plan.

WHAT DO THEY WANT TO KNOW ABOUT THE EVALUATION?

While some information from the evaluation will be of use to all three groups of stakeholders, some will be needed by only one or two of the groups. Grantmakers and funders, for example, will usually want to know how many people were reached and served by the initiative, as well as whether the initiative had the community-level impact it intended to have. Community groups may want to use evaluation results to guide them in decisions about their programs, and where they are putting their efforts. University-based researchers will most likely be interested in proving whether any improvements in community health were definitely caused by your

programs or initiatives; they may also want to study the overall structure of your group or initiative to identify the conditions under which success may be reached.

WHAT DECISIONS DO THEY NEED TO MAKE, AND HOW WOULD THEY USE THE DATA TO INFORM THOSE DECISIONS?

You and your stakeholders will probably be making decisions that affect your program or initiative based on the results of your evaluation, so you need to consider what those decisions will be. Your evaluation should yield honest and accurate information for you and your stakeholders; you'll need to be careful not to structure it in such a way that it exaggerates your success, and you'll need to be really careful not to structure it in such a way that it downplays your success!

Consider what sort of decisions you and your stakeholders will be making. Community groups will probably want to use the evaluation results to help them find ways to modify and improve your program or initiative. Grantmakers and funders will most likely be making decisions about how much funding to give you in the future, or even whether to continue funding your program at all (or any related programs). They may also think about whether to impose any requirements on you to get that program (e.g., a grantmaker tells you that your program may have its funding decreased unless you show an increase of services in a given area). University-based researchers will need to decide how they can best assist with plan development and data reporting.

You'll also want to consider how you and your stakeholders plan to balance costs and benefits. Evaluation should take up about 10--15% of your total budget. That may sound like a lot but remember that evaluation is an essential tool for improving your initiative. When considering how to balance costs and benefits, ask yourself the following questions:

- What do you need to know?
- What is required by the community?
- What is required by funding?

HOW DO YOU DEVELOP AN EVALUATION PLAN?

THERE ARE FOUR MAIN STEPS TO DEVELOPING AN EVALUATION PLAN:

- Clarifying program objectives and goals
- Developing evaluation questions
- Developing evaluation methods
- Setting up a timeline for evaluation activities

CLARIFYING PROGRAM OBJECTIVES AND GOALS

The first step is to clarify the objectives and goals of your initiative. What are the main things you want to accomplish, and how have you set out to accomplish them? Clarifying these will help you identify which major program components should be evaluated. One way to do this is to make a table of program components and elements.

DEVELOPING EVALUATION QUESTIONS

For our purposes, there are four main categories of evaluation questions. Let's look at some examples of possible questions and suggested methods to answer those questions. Later on, we'll tell you a bit more about what these methods are and how they work

- **Planning and implementation issues:** How well was the program or initiative planned out, and how well was that plan put into practice?

- *Possible questions:* Who participates? Is there diversity among participants? Why do participants enter and leave your programs? Are there a variety of services and alternative activities generated? Do those most in need of help receive services? Are community members satisfied that the program meets local needs?
- *Possible methods to answer those questions:* monitoring system that tracks actions and accomplishments related to bringing about the mission of the initiative, member survey of satisfaction with goals, member survey of satisfaction with outcomes.
- **Assessing attainment of objectives:** How well has the program or initiative met its stated objectives?
 - *Possible questions:* How many people participate? How many hours are participants involved?
 - *Possible methods to answer those questions:* monitoring system (see above), member survey of satisfaction with outcomes, goal attainment scaling.
- **Impact on participants:** How much and what kind of a difference has the program or initiative made for its targets of change?
 - *Possible questions:* How has behavior changed as a result of participation in the program? Are participants satisfied with the experience? Were there any negative results from participation in the program?
 - *Possible methods to answer those questions:* member survey of satisfaction with goals, member survey of satisfaction with outcomes, behavioral surveys, interviews with key participants.
- **Impact on the community:** How much and what kind of a difference has the program or initiative made on the community as a whole?
 - *Possible questions:* What resulted from the program? Were there any negative results from the program? Do the benefits of the program outweigh the costs?
 - *Possible methods to answer those questions:* Behavioral surveys, interviews with key informants, community-level indicators.

DEVELOPING EVALUATION METHODS

Once you've come up with the questions you want to answer in your evaluation, the next step is to decide which methods will best address those questions. Here is a brief overview of some common evaluation methods and what they work best for.

Monitoring and feedback system

This method of evaluation has three main elements:

- *Process measures:* these tell you about what you did to implement your initiative;
- *Outcome measures:* these tell you about what the results were; and
- *Observational system:* this is whatever you do to keep track of the initiative while it's happening.

Member surveys about the initiative

When Ed Koch was mayor of New York City, his trademark call of "How am I doing?" was known all over the country. It might seem like an overly simple approach, but sometimes the best thing

you can do to find out if you're doing a good job is to ask your members. This is best done through member surveys. There are three kinds of member surveys you're most likely to need to use at some point:

- Member survey of goals: done before the initiative begins - how do your members think you're going to do?
- Member survey of process: done during the initiative - how are you doing so far?
- Member survey of outcomes: done after the initiative is finished - how did you do?

Goal attainment report

If you want to know whether your proposed community changes were truly accomplished-- and we assume you do--your best bet may be to do a goal attainment report. Have your staff keep track of the date each time a community change mentioned in your action plan takes place. Later on, someone compiles this information (e.g., "Of our five goals, three were accomplished by the end of 1997.")

Behavioral surveys

Behavioral surveys help you find out what sort of risk behaviors people are taking part in and the level to which they're doing so. For example, if your coalition is working on an initiative to reduce car accidents in your area, one risk behavior to do a survey on will be drunk driving.

Interviews with key participants

Key participants - leaders in your community, people on your staff, etc. - have insights that you can really make use of. Interviewing them to get their viewpoints on critical points in the history of your initiative can help you learn more about the quality of your initiative, identify factors that affected the success or failure of certain events, provide you with a history of your initiative, and give you insight which you can use in planning and renewal efforts.

Community-level indicators of impact

These are tested-and-true markers that help you assess the ultimate outcome of your initiative. For substance abuse coalitions, for example, the U.S. Centers for Substance Abuse Prevention (CSAP) and the Regional Drug Initiative in Oregon recommend several proven indicators (e.g., single-nighttime car crashes, emergency transports related to alcohol) which help coalitions figure out the extent of substance abuse in their communities. Studying community-level indicators helps you provide solid evidence of the effectiveness of your initiative and determine how successful key components have been.

SETTING UP A TIMELINE FOR EVALUATION ACTIVITIES

When does evaluation need to begin?

Right now! Or at least at the beginning of the initiative! Evaluation isn't something you should wait to think about until after everything else has been done. To get an accurate, clear picture of what your group has been doing and how well you've been doing it, it's important to start paying attention to evaluation from the very start. If you're already part of the way into your initiative, however, don't scrap the idea of evaluation altogether--even if you start late, you can still gather information that could prove very useful to you in improving your initiative.

Outline questions for each stage of development of the initiative

We suggest completing a table listing:

- **Key evaluation questions** (the five categories listed above, with more specific questions within each category)

- **Type of evaluation measures** to be used to answer them (i.e., what kind of data you will need to answer the question?)
- **Type of data collection** (i.e., what evaluation methods you will use to collect this data)
- **Experimental design** (A way of ruling out threats to the validity - e.g., believability - of your data. This would include comparing the information you collect to a similar group that is not doing things exactly the way you are doing things.)

With this table, you can get a good overview of what sort of things you'll have to do in order to get the information you need.

When do feedback and reports need to be provided?

Whenever you feel it's appropriate. Of course, you will provide feedback and reports at the end of the evaluation, but you should also provide periodic feedback and reports throughout the duration of the project or initiative. In particular, since you should provide feedback and reports at meetings of your steering committee or overall coalition, find out ahead of time how often they'd like updates. Funding partners will want to know how the evaluation is going as well.

When should evaluation end?

Shortly after the end of the project - usually when the final report is due. Don't wait too long after the project has been completed to finish up your evaluation - it's best to do this while everything is still fresh in your mind and you can still get access to any information you might need.

WHAT SORT OF PRODUCTS SHOULD YOU EXPECT TO GET OUT OF THE EVALUATION?

THE MAIN PRODUCT YOU'LL WANT TO COME UP WITH IS A REPORT THAT YOU CAN SHARE WITH EVERYONE INVOLVED. WHAT SHOULD THIS REPORT INCLUDE?

- *Effects expected by shareholders*: Find out what key people want to know. Be sure to address any information that you know they're going to want to hear about!
- *Differences in the behaviors of key individuals*: Find out how your coalition's efforts have changed the behaviors of your targets and agents of change. Have any of your strategies caused people to cut down on risky behaviors, or increase behaviors that protect them from risk? Are key people in the community cooperating with your efforts?
- *Differences in conditions in the community*: Find out what has changed. Is the public aware of your coalition or group's efforts? Do they support you? What steps are they taking to help you achieve your goals? Have your efforts caused any changes in local laws or practices?

You'll probably also include specific tools (i.e., brief reports summarizing data), annual reports, quarterly or monthly reports from the monitoring system, and anything else that is mutually agreed upon between the organization and the evaluation team.

WHAT SORT OF STANDARDS SHOULD YOU FOLLOW?

Now that you've decided you're going to do an evaluation and have begun working on your plan, you've probably also had some questions about how to ensure that the evaluation will be as fair, accurate, and effective as possible. After all, evaluation is a big task, so you want to get it right. What standards should you use to make sure you do the best possible evaluation? In 1994, the Joint Committee on Standards for Educational Evaluation issued a list of program evaluation standards that are widely used to regulate evaluations of educational and public health programs. The standards the committee outlined are for utility, feasibility, propriety, and accuracy. Consider using evaluation standards to make sure you do the best evaluation possible for your initiative.

Assessment:

1. Prepare an evaluation plan for your capstone project proposal. Take note of the information needed to include in an evaluation plan.

References:

Corporate Finance institute. (2015-2021). What is an Evaluation Plan? Retrieved from Corporate Finance institute:

<https://corporatefinanceinstitute.com/resources/knowledge/other/evaluation-plan/>

hampton, C. (1994-2021). Community Tool Box. Retrieved from Ctb:

<https://ctb.ku.edu/en/table-of-contents/evaluate/evaluation/evaluation-plan/main>

Pearson, S. (2015, September 5). Evaluation Plan. Retrieved from Slide player:

<https://slideplayer.com/slide/6965849/>

