SOFTWARE REQUIREMENT ENGINEERING

1. Suppose you are making a withdrawal or transferring money to other bank accounts from the ATM using a debit/credit card. Develop a complete UML diagram for the ATM machine.
2. What are stakeholders? Identify all the actors for the "Result processing system of SUST".
3. How will you validate the client's every requirement?
4. Among the eight core principles that guide processes which is the most important to you? Justify your answer.
5. What is an agile process?
6. What key traits must exist among the people on an effective software team?
7. An analysis rule of thumb is that the model "should focus on requirements that are visible within the problem or business domain.'. What types of requirements are not visible in these domains? Provide few examples.
8. You have been asked to build "A Web-based order processing system for a departmental store". Develop an entity-relationship diagram that describes data objects, relationships and attributes.
9. Now develop a LEVEL 0, LEVEL 1 DFD model for the same system stated in question 8.

Chapter: 3

3.1. Reread "The Manifesto for Agile Software Development" at the beginning of this chapter. Can you think of a situation in which one or more of the four "values" could get a software team into trouble?

Ans:

Manifesto for agile software development:

Better ways of developing software development:

1) Individuals and interactions over processes and tools,
2) Working software over comprehensive documentation,
3) Customer collaboration over contract negotiation,
4) Responding to change over following a plan.

I thought that, all of that value could get a software team into trouble. For example, "Individual and interaction", the key is about communication and organizing the team. So, we have to make good communication in a team, because we will work together all of the project time. And we know that,
communicationis the first step of basic framework activities. So that's why, if we have bad situation, we will get in trouble.

Then, "customer collaboration", we need customer to get in the project because he will decide the work product. If the agile team agrees that the process works, and the team produces deliverable software increments that satisfy the customer, we've done it. But, if the costumer doesn't agree, that will be a trouble, we won't ever finish the project.

But I thought that, **the most important** and could get a software team into trouble is the last one. That's about the importance of responding to change over following plan. We know that, agile development is a software proses that forced us to response quickly to our project. Because on agile development, we won't be able to define the requirement of the project completely at the beginning. That’s why, we have to responsive to any changes whenever. Then, we will not stick at one problem. So, if we don’t respond the changes quickly and still following the plan, we won’t finish our project as soon as possible, and that’s a trouble for software team.

3.2. Describe agility (for software projects) in your own words.

Ans:

The word agility means “To do something very quickly”. Thus, agility in terms of software development is to develop a software which meets all the requirement specified by the customer, and always ready to accept the changes required by the customer even in later phases.

The goals of agility are explained below:

- The development process should always be ready to accept new changes needed in the development.
- Agility means to deliver the working software as soon as possible.
- Agile means to keep the software simple, the part only that is required.
- The agile software development is always growth oriented. That means, it should adapt to new things, but incrementally. Adaptation to new changes should always move the progress in forward direction.
- Agility in the software development keeps the change cost low as it is an incremental process. thus, changes can be easily made in controlled environment.

Agility can be applied to any software process. However, to accomplish this, it is essential that the process be designed in a way that allows the project team to adapt tasks and to streamline them, conduct planning in a way that understands the fluidity of an agile development approach, eliminate all but the most essential work products and keep them lean, and emphasize an incremental delivery strategy that gets working software to the customer as rapidly as feasible for the product type and operational environment.

3.3. Why does an iterative process make it easier to manage change? Is every agile process discussed in this chapter iterative? Is it possible to complete a project in just one iteration and still be agile? Explain your answers.

Ans:

The software team manages change by focusing on a defined increment and postponing any changes until the next increment. All agile process models are iterative/incremental.

An iterative process makes it easier to manage changes. Since each iteration is a mini ‐ project, the project team addresses, to some extent, all the risks associated with the project as a whole each time it builds an increment of the system. As risks become greater, as delays occur, and as the environment become more unstable. The team is able to make necessary adjustments on a relatively small scale and propagate those adjustments across the entire project. An iterative process has greater flexibility to change the plan. Hence it is easier to manage the changes. Almost all the agile process models are iterative.

Yes, it is possible to complete a project in just one iterative and still be agile. Because if the time is the major constraints and all the requirements are well known at the beginning of the project. If there are well suited design patterns are available and the principles of agile development are satisfied then it is possible to complete a project in one iteration that too with agility.

Agile process models deliver software increments followed by customer feedback and revision. Because the increments are small and the customer feedback is prompt, it is relatively easy to incorporate the changes in the next software increment. For example, ASD uses an iterative process that incorporate adaptive cycle planning, relatively rigorous requirement gathering methods, and an iterative development cycle that incorporates customer focus groups and formal technical reviews as real--time feedback mechanisms. The Dynamic Systems Development Method (DSDM) defines three different iterative cycles—functional model iteration, design and build iteration, and implementation—preceded by two additional life cycle activities—feasibility study and business study.

3.4. Could each of the agile processes be described using the generic framework activities noted in Chapter 2? Build a table that maps the generic activities into the activities defined for each agile process.

Ans:

The generic process framework can be applied to each agile process described in this chapter. The table should list all agile process models across the first row and all generic framework activities

1) Communication
2) Planning
3) Modeling
4) Construction
5) Deployment.

Mapping from activates in agile process to generic activities.

1) Extreme programming (XP)
    a) Planning ‐ communication, planning
    b) Design ‐ modeling
    c) Coding ‐ modeling, construction
    d) Test ‐ deployment.
2) Adaptive software development (ASD)
    a) Speculation ‐ communication, planning
    b) Collaboration ‐ modeling, construction
    c) Leaning ‐ Deployment
3) Dynamic systems development method (DSDM)
    a) Feasibility, business study ‐ communication, planning
    b) Functional model, design ‐ Modeling, constructs
    c) Implementation ‐ construction, deployment.

3.5. Try to come up with one more “agility principle” that would help a software engineering team become even more maneuverable.

Ans:

One more additional agility principle ‐

Less is more:

Always model for a purpose, either to communicate, or to understand. Don’t create models for other people unless you know how and by whom they will be used. Travel light and update only when it hurts. Everything that is created, model, document, code, test etc. has a cost. If you decide to keep it, each will need to be maintained over time increasing cost of change and what people must know and understand. The less you keep and maintain, the more agile you are.

Minimize the complexity. Simplicity reduces amount of work to do. The simplest solution is usually the best solution. More complex and detailed artifacts will be more difficult and expensive to change. Plan for change realistically, but don’t overload your models or system with features that you don’t need.

3.6. Select one agility principle noted in Section 3.3.1 and try to determine whether each of the process models presented in this chapter exhibits the principle. [Note: I have presented an overview of these process models only, so it may not be possible to determine whether a principle has been addressed by one or more of the models, unless you do additional research (which is not required for this problem).

Ans:

Agile processes embrace change as essential to the customer's competitive advantage. Each of the process models presented in this chapter exhibits this principle.

Selected one agility principle is: “Working software is the primary measure of progress”.

- Extreme programming (XP):
  Planning activity begins with the creation of a set of stories. A value is assigned to each story and XP team will assess each story & assign a cost. Finally, the XP story will evolve as working software.
- Adaptive software development (ASD):
  Effective collaboration with the customer will only occur if we throw out any “us and them” attributes. ASD teams with intent of learning & then improving its approach.
- Dynamic system development method (DSDM):
  It is an incremental method. The increment may not be 100% complete but there should be some progress compared to last increments
- Scrum:
  Scrums allow us to build softer software. Scrum patterns enable a software development team to work successfully in a world where elimination of uncertainty is impossible.
- Agile modeling (AM):
  The problem can be partitioned effectively among the people who must solve it and quality can be assessed at every step as system is being progressed, engineered and built.

3.7. Why do requirements change so much? After all, don’t people know what they want?

Ans:

Requirements may change for various reasons even if people know their wants. The reasons for the same can be:

· Change in environment: Changes in legislation, organizational dynamics or competition strategy may also lead to change in requirements.

· Missed requirement: A stakeholder working with a system might realize that it's missing a feature.

· Identify a defect: A bug or defect in the system, creates new requirement.

· Actual requirement was not clearly understood: It is common for user to realize that what they asked for is different from what is required.

It is a fact that requirements change a lot in software development life cycle. Changes in requirement are hard to predict and so are the change in customer priorities.

It is often difficult for stakeholders to verbalize their software needs till they see a working prototype, and it is then only that they might realize that they have forgotten to consider some important points. This might happen because people may not know what they actually want until they see something running. This explains importance of active stakeholder participation and short iterations to a system’ s success.

Very rarely, does it happen that stakeholders do not know what they want from the system. A decent requirement specification can be designed by effective collaboration between all the stakeholders, also their experience in the respective fields does make the task a little easier.

Requirements change so much, that it is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as the project proceeds. It is often difficult for people to verbalize their software needs until they see a working prototype and realize that they had forgotten to consider something important.

3.8. Most agile process models recommend face-to-face communication. Yet today, members of a software team and their customers may be geographically separated from one another. Do you think this implies that geographical separation is something to avoid? Can you think of ways to overcome this problem?

Ans:

Most agile process models recommend face--to--face communication. Yet today, members of a software team and their customers may be geographically separated from one another in that case Communication is the key, the consumer and developer should constantly be communicating even if they are geographically separated, they can web-talk or talk over the phone every now and then or write emails, use chatting as the, means or a medium of conference call communication where 2 and more people can talk to each other at the same time.

3.9. Write an XP user story that describes the “favorite places” or “bookmarks” feature available on most Web browsers.

Ans:

Internet bookmarks are stored webpage locations that can be retrieved. The main purpose is to easily catalog and access web pages that a user has visited and choose to save. Saved links are called “favorites”, and by virtue of the browser’s large market share, the term favorite has been synonymous with bookmark since the early days of widely﹣distributed browsers. Bookmark are normally visible in a browser menu and stored on the user’s computer and commonly a metaphor is be used for organization. Book marks are a fundamental feature of web browsers, but some users have expressed frustration with bookmark collections that become disorganized and have looked for other tools to help manage their links. There tools include browser synchronizers and desktop applications.

3.10. What is a spike solution in XP?

Ans:

Extreme Programming (XP):

- Extreme programming (XP) is a popular agile software development methodology used to implement the software project.
- XP uses an object-oriented approach as its preferred development paradigm.
- XP encompasses a set of rules and practices that occur within the context of framework activities.
- Spike solution:

- It is a small technique to investigate the solution to a problem.
- To figure out tough problems or any other design problems, create a spike solution to overcome it.
- When we encounter a tough problem that lack’s an immediately apparent solution, create a spike solution.
- The goal of a spike solution is to keep the solution simple and away from the risk factors.
- That is where you try out different approaches in the code to make the correct solution more apparent.
- This happens all the time in independent development, and it may not require the level of formally those agile programming calls for. Spike solution in XP:

A spike solution in XP is a prototype that is intended to throw away.

- The prototype is implemented and evaluated. The intent is to lower risk when true implementation.
- Starts and to validate the original estimates for the story containing the design pattern.
- To figure out the tough problems or any other design problems, create a spike solution to overcome it.
- Also, the spike solution reduces the potential risk of the problem.
- The spike approach deals to an “end to end” approach to the problem.
- In XP, the spike solution implemented to keep away the risk and to prepare the quality solutions to the tough problems when those agile programming calls for.
-

3.11. Describe the XP concepts of refactoring and pair programming in your own words.

Ans:

<u>Refactoring:</u>

It is a construction technique. Extreme programming (XP) encourages refactoring. Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves the internal structure. It is a disciplined way to clean up the code that minimizes the chance of introducing bugs. The intent of refactoring is to control the modifications by suggesting small design changes that can radically improve the design. The design occurs both before and after coding commences. Refactoring means that design occurs continuously as system is constructed.

<u>Pair programming:</u>

Pair programming recommends that two people or teams work together at one computer work station to create code for a story. This provides a mechanism for real‐time problem solving and real‐time quality assurance. In practice, each person takes on slightly different role. For example, one might think about coding details of particular portion of design while the other ensures that coding standards are being followed and code that is generated will fit to requirement.

3.12. Do a bit more reading and describe what a time-box is. How does this assist an ASD team in delivering software increments in a short time period?

Ans:

A time-box is a project management term that indicates a period of time that has been allocated to accomplish some tasks.

The Time-boxing is for manage time technique to software development, where the schedule is divided into a number of separate time periods with each part having its own deadline, budget and deliverables.

Dynamic Systems Development Method DSDM defines a Timebox as a fixed period of time, at the end of which an objective has been met. The Timebox objective is usually completion of one or more deliverables.

The optimum length for a Timebox is typically between two and four weeks – long enough to achieve something useful, short enough to keep the Solution Development Team focused. On a very short and rapidly moving project, it is possible to have a Timebox as short as a day. In exceptional circumstances, a Timebox might be as long as six weeks. The disadvantage of longer Timeboxes is that the team may lose focus. By evolving the solution through a series of short Timeboxes, the team is able to more frequently assess their true progress – "What have we delivered?" If their progress is not meeting their own expectations, this provides early warning of problems, and an early opportunity to address the problems.

Timeboxing is more than just setting short time periods and partitioning the development work. It is a well-defined process to support the creation of low-level products in an iterative but controlled fashion. Timeboxing incorporates frequent review points to ensure the quality of those products and the efficiency of the Iterative Development process.

3.13. Do the 80 percent rule in DSDM and the time-boxing approach defined for ASD achieve the same result?

Ans:

The DSDM approach for each iteration follows the 80 percent rule. That is, only enough work is required for each increment to facilitate movement to the next increment. The ASD time-boxing approach establishes distinct schedule boundaries for the delivery of components of an increment and suggest that work continues only until the time-box boundary is reached. The remaining detail can be completed later when more business requirements are known or changes have been requested and accommodated.

3.14. Using the process pattern template presented in Chapter 2, develop a process pattern for any one of the Scrum patterns presented in Section 3.5.2.

Ans:

Process pattern template for scrum pattern "communicate Early"

Problem: What is the goal of a project and who are members of a team?

Context: Use of scrum in a distributed project (faster, cheaper and quality projects)

Solution: Arrange kick-off meeting for all relevant members present briefly scrum methodology.

Describe the goal and contents of the project.

Describe a release plan with some sprints

Introduce briefly use of an ALM tool everybody presents his / her responsibilities & hobbies.

Consequences: common goal is known by every relevant member a common process and tools for every site can be established by scrum and ALM tools efficient communication channels can be created between team members when they learn to know each other better.

3.15. Why is Crystal called a family of agile methods?

Ans:

The Crystal methodology is one of the most lightweight, adaptable approaches to software development. Several of the key tenets of crystal include teamwork, communication, and simplicity, as well as reflection to frequently adjust and improve the process. Like other agile process methodologies, crystal promotes early, frequent delivery of working software, high user involvement, adaptability, and the removal of bureaucracy or distractions.

Crystal is actually comprised of a family of agile methodologies such as crystal clear, crystal yellow, crystal orange and others, whose unique characteristics are driven by several factors such as team size, system criticality, and project priorities. This crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project 's unique characteristics.

3.17. Visit the Official Agile Modeling Site and make a complete list of all core and supplementary AM principles.

Ans:

Agile modeling defines set of practices which can show the way towards becoming successful Agile modelers. These practices are divided into two sections. One the "Core principles" and the other "Supplementary Principles".

Core Principles:

- Simplicity- Do not make complex model keep it simple. When you can explain your team with a pen and paper do not complex it by using modeling tool like rational rose. Do not add complexity to show of something. If the developer understands only flow chart, then explain him with a flow chart, if he understands pseudo-code then use pseudo-code and so on So look at your team what they understand and prepare document in a similar fashion.
- Welcome change- Requirements grow with time. Users can change the requirements as the project moves ahead. In traditional development cycle you will always hear the word "Freeze the requirement", this has changed with Agile coming in. In Agile we welcome change and the same is reflected in the project.
- Incrementally change- Nothing can be right at the first place itself. You can categorize your development with "The most required", "Needed features" and "Luxury features". In the first phase try to deliver the "The most required" and the incrementally deliver the other features.
- Model exists with a purpose- Model should exist for a purpose and not for the sake of just existing. We should know who our target audience for whom the model is made. For instance, if you are making a technical document, it's for the developers, a power point presentation it's for the top management and so on. If the model does not have target audience, then it should not exist probably. In short "just deliver enough not more".
- It should be light- Any document or artifact you create should be also updated over a period of time. So, if you make 10 documents then you should note that as the source code changes you also need to update those documents. So, make it light as possible.
- Keep multiple models- Project issues vary from project to project and the same project behavior can vary from organization to organization. So do not think one model can solve all issues keep yourself flexible and think about multiple models. For instance, if you are using UML for technical documentation then every diagram in UML can reflect the same aspects in different way. For instance, a class diagram shows the static view of project while a flow charts a dynamic view. So, keep yourself flexible by using different diagrams and see which best fits your project or the scenario.
- Software is the most important thing- The main goal of a software project is to produce high quality software which can be utilized by your end customer in a effective manner. Many projects end up with bulky documents and management artifacts. Documentation is for software and not software for the documentation. So, any document or activity which does not add value to the project should be questioned and validated.
- Get Rapid and regular feedbacks- Software is finally made for the user. So, try to get feedback on regular basis from the end user. Do not work in isolation involve the end user. Work closely with the end customer, get feedback, analyze requirements and try to meet their need.

Supplementary principles:

- Content is important than presentation - The look and feel is not important rather the content or the message to be delivered by the content is important. For instance, you can represent project architecture using complex UML diagrams, simple flow chart or by using simple text. It will look fancy that you can draw complex UML diagrams, but if the end developer does not understand UML, then it ends nowhere. A simple textual explanation could have met the requirement for communicating your architecture to the end developer or programmer.

- Honest and open communication - Take suggestion, be honest and keep your mind open to new model. Be frank with the top management if your project is behind schedule. An open and free environment in project keeps resources motivated and the project healthy.

Chapter: 4

4.1. Since a focus on quality demands resources and time, is it possible to be agile and still maintain a quality focus?

Ans:

Yes, it can be done. Time management and schedule of the team for them to meet the deadline is crucial, Simplicity and accuracy in making the product is sufficient to meet the customers need.

Yes, it is possible to maintain a quality focus while being agile. Agile methodologies are based on the iterative development. They involve a set of engineering best practices intended to allow for rapid delivery of high-quality software. Agile methodologies suggest to, keep the technical approach as simple as possible, keep the work products as concise as possible, and make decisions locally whenever possible. It means that unnecessary usage and wastage of resources must be reduced. So, it does not affect the quality demands resources. Also, by implementing agile methodologies, time can be saved because it suggests making most of the decisions locally.

It is possible to be agile and maintain a quality focus, if the team focus is to deliver the best possible increment to customer and to react to the customer feedback in a responsible manner.

4.2. Of the eight core principles that guide process (discussed in Section 4.2.1), which do you believe is most important?

Ans:

Eight core principles that guide process:

1. Be agile

2. Focus on quality at every step

3. Be ready to adapt

4. Build an effective team

5. Establish mechanisms for communications and coordination

6. Manage change

7. Assess risk

8. Create work products that provide value for others

Out of the eight core principles that guide process, we can consider that the fourth principle – “**building an effective team**” is most important, because the bottom line of a software process is people. Building a self-organizing team that has mutual trust and respect is needed. An effective team can only produce a quality product.

4.3. Describe the concept of separation of concerns in your own words.

Ans:

A large problem is easier to solve if it is subdivided into a collection of elements each of which delivers a distinct functionality that can be developed, and in some cases validated, independently of other concerns.

Separation of concerns:

・A Separation of concerns (SOC) is a large problem is easier to solve if it is subdivided into a collection of elements (or concerns).

・The Separation of concerns is states that a particular given problem involves different kinds of concerns, which should be identified and separated to deal with complexity and to achieve the required engineering quality factors such as robustness, flexibility, maintainability, reusability and different types of concerns may be relevant to different software engineers to different roles, and to achieve to different goals at different stages of the software lifecycle.

・Separation of concerns is a design concept that suggests that any complex problem can be more easily handled if it is subdivided into pieces that can each be solved and optimized independently.

・A concern is a behavior that is specified as part of the requirements model for the software. By separating concerns into smaller and therefore more manageable pieces, a problem takes less effort and time to solve.

・Separation of concerns is manifested in other related design concepts: modularity, aspects, functional independence, and refinement.

4.4. An important communication principle states “prepare before you communicate.” How should this preparation manifest itself in the early work that you do? What work products might result as a consequence of early preparation?

Ans:

An important communication principle states “prepare before you communicate.” Customer communication principles and concepts focus on the need to reduce noise and improve bandwidth as the conversation between developer and customer progresses. Both parties must collaborate for the best communication to occur.

Before I communicate to others regarding my product, just I would understand the total concept considering the do’s & don’ts. And, I identify myself as how that intends to ensure quality.

Planning is also an important role in the communication.

Planning – encompasses a set of management and technical practices that enable the product to define a road map as it travels toward its strategic goal and tactical objectives.

Planning Principles

1. Understand the scope of the product.
    a. Scope provides the product team with a destination.
2. Involve the customer in the planning activity.

   a. The customer defines priorities and established product constraints.

3. Recognize that planning is iterative.
   a. The plan must be adjusted to accommodate changes.
   b. In addition, iterative, incremental process models dictate replanting based on feedback received from users.
4. Estimate based on what you know.
   a. The intent of estimation is to provide an indication of effort, cost, and task duration, based on the team's current understanding of the work to be done.
5. Be realistic.
   a. Realities should be considered as a product plan is established.
   b. People don't work 100 percent of every day.
   c. Noise always enters into any human communication.
   d. Omissions and ambiguity are facts of life.
   e. Change will occur.
   f. Even the best software engineers make mistakes.
6. Adjust granularity as you define the plan.
   Granularity refers to the level of detail that is introduced as a product plan is developed.
   A fine granularity plan provides significant work task detail that is planned over relatively short time increments.
   A coarse granularity plan provides broader work tasks that are planned over longer time periods.

7. Define how your intent to ensure quality.
   If formal technical reviews are to be conducted, they should be scheduled.
   If pair programming is to be used during construction, it should be explicitly defined within the plan.

8. Describe how you intend to accommodate change.
   Example situations:
   Can the customer request a change at any time?
   If a change is requested, is the team obliged to implement it immediately?
   How is the impact and cost of the change assessed?

9. Track the plan frequently and make adjustments as required.
   It makes sense to track progress on a daily basis, looking for problem areas and situations in which scheduled work dons not conform to actual work conducted.
   When slippage is encountered, the plan is adjusted accordingly.

4.5. Do some research on "facilitation" for the communication activity (use the references provided or others) and prepare a set of guidelines that focus solely on facilitation.

Ans:

An important communication principle states "prepare before you communicate." Customer communication principles and concepts focus on the need to reduce noise and improve bandwidth as the conversation between developer and customer progresses. Both parties must collaborate for the best communication to occur.

"Communication" is the exchange of thoughts, messages, or information, as by speech, signals, writing, or behavior.

The process of communication is one that is deeply affected by following factors:

· culture (corporate and societal),

· personal character traits,

· and the environmental setting.

· Good communication is essential for productive design and sound facilitation is essential for encouraging attendees to communicate during the work.

Someone should facilitate the activity. Every communication meeting should have a leader (facilitator) to keep

(1) Conversation moving in a productive direction

(2) To mediate any conflict that does occur

(3) To ensure than other principles are followed.

(4) Provide training to staff for effective communication

(5) Ask everyone to be a good listener.

(6) Provide guidance about maintaining a healthy relationship among team members.

(7) Provide feedback at the end of the communication.

4.6. How does agile communication differ from traditional software engineering communication? How is it similar?

Ans:

The agile view of customer communication and collaboration emphasizes the need for all stakeholders to work (continuously) as a team, be co-located, and recognize the change is part of the process. However, the basic agile philosophy of communication is applicable to all software engineering practice and like all software engineering practices, communication and collaboration are iterative. Each iteration divulges important new information and enhances the overall understanding of the software to be built.

Communication is one of the fundamental activities of software development. Project initiation and requirements gathering takes place in communication activity, no matter whether it is an agile communication or ordinary software communication.

Agility is dynamic, content specific, aggressively change embracing and growth oriented. It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as project proceeds.

The planning activity in agile process begins with creation of set of stories that describe required features of software to be built. Each story is written by the customer and is placed on an index card. The customer assigns a value (priority to it). The software team will look after these stories and make changes.

Whereas traditional software engineering communication is simpler job than agile communication. It has more time to analyze and implement the system.

Both agile communication & traditional software engineering communication are having the same background idea. They both are externally similar but coming to internal point of view i.e. implementation, they differ from each other.

4.7. Why is it necessary to “move on”?

Ans:

It is necessary to “move on” because communication, like any software engineering activity, takes time. Rather than iterating endlessly, the people who participate should recognize that many topics require discussion and that “moving on” is sometimes the best way to achieve communication agility.

<u>Move on</u>

Communication is one of the major software engineering activities. In this “moving on”

Is the one of the concepts of communication practices and sometimes it is the best way to achieve communication agility.

It is necessary to implement in software engineering practice. It works from essence towards the implementation.

Move on is consists of following principle.

1. After agree to something, move on for that task.

2. Can’t agree to something, move on to another task.

3. If a feature or function is unclear and cannot be clarified at the moment.

Example:

The analysis is a task that should “move on” from essential information toward to implement as detail.

If it is needed to spend the time to analyze the problem before meeting with other, then, if necessary, perform some research to analyze business logic. So, it takes long time to understand a complex problem from different kinds of users.

Hence in this scenario, “move on” is the one of the major principles in communication activity.

4.8. Do some research on “negotiation” for the communication activity and prepare a set of guidelines that focus solely on negotiation.

Ans:

“Negotiation” for the communication activity works best when both parties win. Sets of guidelines that focus solely on negotiation are many instances in which the software engineer and the customer must negotiate functions and features, priorities, and delivery dates. If the team has collaborated well, all parties have a common goal. Therefore, negotiation will demand compromise from all parties.

Negotiation is the key decision-making approach used to reach consensus whenever a person, organization or another entity cannot achieve its goals unilaterally. Negotiations appear in a multitude of forms, take place in very different situations and are influenced by ethical, cultural and social circumstances.

The negotiation process as the process of interpersonal communication for the purpose of forming and modifying perceptions and attitudes. Negotiation is every process of social interaction and communication involving distribution and redistribution of power, resources, and commitments. A significant contribution of behavioral research is in numerous heuristics and qualitative models that have been shown to be useful in negotiation practice.

Most of the existing research concentrates on two directions:

(1) traditional, face-to-face negotiations that rely on human expertise but little, if at all, on the information systems, and

(2) formal models of idealized negotiators involved in complex strategic encounters. Electronic negotiations and processes that involve information systems as actively participants, negotiations among both human and artificial agents who use distributed knowledge and information require consideration of approaches from areas that traditionally were "foreign to each other". We need to continue integrating concepts and approaches proposed in law and social science with those proposed in economic science and management.

In order to establish a common terminology and to facilitate multidisciplinary research, we propose to define negotiations in such a way that it is possible to include the various existing negotiation situations and approaches.

Negotiation as an iterative communication and decision-making process between two or more agents (parties or their representatives) who:

1. Cannot achieve their objectives through unilateral actions;

2. Exchange information comprising offers, counter-offers and arguments;

3. Deal with interdependent tasks; and

4. Search for a consensus which is a compromise decision.

The outcome of a negotiation can be a compromise (an allocation) or a disagreement. Negotiation arena is the accepted place where the negotiators communicate. The agenda specifies the negotiation framework, including the specification of the negotiated issues and format in which they are presented (e.g., sequentially or simultaneously). Decision-making rules are used to determine, analyze and select decision alternatives and concessions. Rules of communication determine the way offers and messages (arguments) are exchanged. Most negotiations follow certain rules; in negotiations in which software carries out some tasks, rules need to be explicitly specified allowing, among others, a distinction between tasks undertaken by a system and by people.

4.9. Describe what granularity means in the context of a project schedule.

Ans:

The term granularity refers to detail with which some element of planning is represented or conducted. It is a type of communication activity.

It refers to the level of detail that is introduced in a project plan and development.

1. A "high granularity" plan provides significant work task detail that is planned over relatively short time increments.

2. A "low granularity" plan provides broader work tasks that are planned over longer time periods.

The communication activity helps a software team to define overall goals and objectives. And these goals and objectives are not the same as defining a plan for getting. Planning activity is encompassing a set of management and technical practices that enable the software team to define a road map as it travels toward its strategic goal and tactical objectives.

So, granularity moves from "high" to "low" as the project timeline moves away from the current data. After that the project can be planned in significant detail. Activities won't occur for much time to do not require fine granularity.

4.10. Why are models important in software engineering work? Are they always necessary? Are there qualifiers to your answer about necessity?

Ans:

The software design model is the equivalent of an architect’ s plans for a house. It begins by representing the totality of the thing to be built (e.g., a three-dimensional rendering of the house) and slowly refines the thing to provide guidance for constructing each detail (e.g., the plumbing layout). Similarly, the design model that is created for software provides a variety of different views of the system, hence it is always necessary.

Models are important in software engineering work.

· They give a prototype of the actual product to be designed.

· They give a better understanding of the product to be designed.

· They help in reducing the complexity of the product to be designed.

· They act as a guide on how the software product can be built efficiently.

· They provide a technical guidance to the users who implement the software.

Models are important as well as necessary in software engineering work.

· They assist in developing the final product.

· Models usually help in getting the product to the customer faster.

· If a model slows down the process of getting the product to the customer, then such models can be avoided and are not necessary.

Time taken to construct the model and complexity of the model can be considered as qualifiers for the necessity of the model in software engineering work.

· If a model slows down the process of getting the product to the customer, then such models can be avoided and are not necessary.

· If a model is taking a lot of time to be built, then such models can be avoided and are not necessary.

4.11. What three “domains” are considered during requirements modeling?

Ans:

Analysis modeling represents the customer requirements by depicting the software in three different domains:

1. The information domain,

2. The functional domain, and

3. The behavioral domain.

Design models represent the characteristics of the software that help practitioners to construct it effectively: the architecture, the user interface, and component-level detail.

4.12. Try to add one additional principle to those stated for coding in Section 4.3.4.

Ans:

One additional principle to those stated for coding would be: Code should be written in a manner it explains the logic even to the user or who is not a developer, self-explanatory and meets the logic of it's working.

Additional principal about coding principal is ‘**secure coding principle**’.

Many security vulnerabilities could easily be prevented if security were taken into consideration at the beginning of the development process. Coding weaknesses degrade the security. So by using the security principles, we can avoid some of the most common security problems like

1. Weak file and group permissions

2. Race conditions

3. Buffer overflows

4. Problems with temporary files

5. Overly complex and unnecessary code

6. Insecure system calls and switches

7. Hard-coding passwords

4.13. What is a successful test?

Ans:

A successful test is one that uncovers an as-yet-undiscovered error. A necessary part of a test case is a definition of the expected output or result. A programmer or programming organization should not test its own programs. Thoroughly inspect the results of each test. Examining a program to see if it does not do what it is supposed to do is only half of the battle. The other half is seeing whether the program does what it is not supposed to do. Do not plan a testing effort under the tacit assumption that no errors will be found. Testing is an extremity creative and intellectually challenging task. Testing is the process of executing a program with the intent of finding errors. A good test case is one that has a high probability of detecting an as - yet undiscovered error. Testing is a never - ending process and Testing is not debugging. Test can never find 100% of the included errors. Test has got goal to find errors and not their reasons.

4.14. Do you agree or disagree with the following statement: "Since we deliver multiple increments to the customer, why should we be concerned about quality in the early increments—we can fix problems in later iterations." Explain your answer.

Ans:

It would be preferable to disagree with the statement, "Since we deliver multiple increments to the customer, why should we be concerned about quality in the early increments–we can fix problems in later iterations." It is always advisable to have a quality check throughout, as the customers will forget you delivered a high-quality product a few days late, but they will never forget the problems that a low-quality product caused them.

"Since we deliver multiple increments to the customer, why should we be concerned about quality in the early increments--we can fix problems in later iterations".

Strongly disagree with above statement.

Explanation:

In this statement, each increment builds on the next level and did not want to build a low-quality foundation.

And from the early increments point of view, if the first increment is low in quality, then customers and users may become concerned for total team. Because each increment builds on the next and no one wants to build on a low-quality foundation. If the first increments are low in quality, then unnecessary tensions may develop. At this time, communication suffers follows.

4.15. Why is feedback important to the software team?

Ans:

Feedback is important to the software team because it provides the information that can be used to correct modeling mistakes, change misinterpretations, and add features or functions that were inadvertently omitted.

The delivered software provides benefit for the end ‑ user, but it also provides useful feedback for software team. As the delivered software application is implementing, the end ‑ users should be encouraged to comment on features and functions, ease of use, reliability and characteristics that are appropriate.

Feedback must be collected from end-user and recorded by a software team. These are used to

· make immediate modifications to the delivered increment (if necessary)

· Define changes to be incorporated into the next planned increment.

· Make necessary design modifications to accommodate changes

· Revise the plan (including delivery schedule) for the next increment to reflect the changes.

Chapter: 5

5.1. Why is it that many software developers don't pay enough attention to requirements engineering? Are there ever circumstances where you can skip it?

Ans:

Understanding the requirements of a problem is among the most difficult tasks that a software engineer face since requirements change continuously, hence they tend to pay little attention to it. In some cases, an abbreviated approach may be chosen. In others, every task defined for comprehensive requirements engineering must be performed rigorously. Requirements engineering builds a bridge to design and construction and cannot be skipped.

Designing and building an elegant computer program that solves the wrong problem server no one needs. That's why it is important to understand what the customer wants before beginning to design and build a computer ‑ based system.

But many software developers do not pay enough attention to requirements engineering, because in developer's point of view ‑

· After all, doesn't the customer know what is required?

· Shouldn't the end users have a good understanding of the features and functions that will provide benefit?

The view is not at all correct.

Requirement engineering helps software engineers to better understand the problem they will work to solve. It encompasses the set of tasks that lead to an understanding of what the business impact of the software will be, what the customer wants and how end – users will interact with software we cannot directly skip requirement engineering but we can be given less importance if the software to be built is very familiar.

5.2. You have been given the responsibility to elicit requirements from a customer who tells you he is too busy to meet with you. What should you do?

Ans:

Create a questionnaire and try to get as much information from the customer as possible. But be sure to make the questionnaire more objective rather than subjective.
based on this questionnaire have another set of questions formulated in continuation to answers provided to the first questionnaire, this way you can make your understanding more concrete on the requirements and get what you are looking for in a few iterations of requestioning.

You might try using an approach like QFD that makes use customer interviews and observation, surveys, and examination of historical data (e.g., problem reports) as raw data for the requirements gathering activity. These data are then translated into a table of requirements—called the customer voice table—that is reviewed with the customers later. A variety of diagrams, matrices, and evaluation methods are then used to extract expected requirements.

Just asking a stakeholder what their requirements are rarely works. It is said that the customer is always right. In fact, the customer may be busy and normally far more urgent things to do rather than speaking to someone in a suit about requirements for a system that is not even due to be delivered.

The first thing to be sure of when eliciting requirements is that we have to get the stakeholder into a state in which they want to talk to us.

Prepare a set of questionnaires and ask him to respond in his free time.

Use the internet facility to overcome the problem of collecting requirements from a busy customer.

Be straight forward, look for the main central idea rather than looking for unnecessary defaults.

However, if the customer refuses to work with you, it's time to get both your management and the customer's management involved. If they don't have the time to help define the software, they probably won't have the inclination to use it.

5.3. Discuss some of the problems that occur when requirements must be elicited from three or four different customers.

Ans:

In reality, the customer and the developer enter into a process of negotiation, where the customer may be asked to balance functionality, performance, and other product or system characteristics against cost and time to market. The intent of this negotiation is to develop a project plan that meets the needs of the customer while at the same time reflecting the real-world constraints (e.g., time, people, budget) that have been placed on the software team, Unfortunately, this rarely happens, each customer has his own views. These views do not match each customer, time is another constraint that matters, each customer may not have time to meet the developer and give the requirements, this further increases the problem.

The following are the problems that occur when requirements must be elicited from three or four different customers.

· The requirements of the customer cannot be understood easily. The customer’ s requirements will change over time such that a customer with a set of requirements at one time can include another set of requirements afterward.

· It is very difficult to understand the requirements of the customers.

· The customers will have a wide range of expectations such that it may lead to disappointments at most of the time.

· The customers will change their requirements rapidly.

· Each and every customer will have some set of requirements such that their expectations can be at different level. Therefore, their expectations may not get fulfilled at most of the time.

· The developmental team of a product will work according to the requirements. Hence, the customers should be available at all time to answer the questions asked by the developmental team of the product.

· The customers should have the knowledge to answer the questions to be asked by the developmental team members. The customers should specify the required input and output clearly.

· The customers may get satisfied or may not get satisfied. They are the decision makers to review a product whether it matches their requirements or not.

To avoid these problems, the customer with some knowledge about the requirements should be available to the developmental team of the product at any time and answerable to the questions asked by them for the good performance and functionality of the product. Also, the customer should be very clear about the input and output data.

5.4. Why do we say that the requirements model represents a snapshot of a system in time?

Ans:

The intent of the analysis model in to provide a description of the required informational, functional and behavioral domains for a computer-based system. The model changes dynamically as software engineers learn more about the system to be built, and stakeholders understand more about what they really require. For that reason, analysis model is a snapshot of requirements at any given time. We expect it to change.

As the analysis model evolves, certain elements will become relatively stable, providing a solid foundation for design. But some elements of model may be more volatile. Indicating the customer doesn’t yet fully understand requirements for the system.

5.5. Let’s assume that you’ve convinced the customer (you’re a very good salesperson) to agree to every demand that you have as a developer. Does that make you a master negotiator? Why?

Ans:

The best negotiations strive for a “win-win” result, hence that does make you a master negotiator. Successful completion of these initial steps achieves a win-win result, which becomes the key criterion for proceeding to subsequent software engineering activities.

There should be no winner and no loser in an effective negotiation. Both sides win because a “deal “that both can live with is solidified.

There is no question of master negotiator. A developer may try to convince the customer but he should remember the “Win -Win” result. The customer wins by getting the system that satisfies the majority of customer’s needs and the software tam wins by working to realistic and achievable budgets and deadlines. Then is negotiation becoming master negotiation but not by convincing the customer as a salesperson.

5.6. Develop at least three additional “context-free questions” that you might ask a stakeholder during inception.

Ans:

The first set of context-free questions focuses on the customer and other stakeholders, the overall goals, and the benefits. For example, the requirement engineer might ask:

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there another source for the solution that you need?

Context-free questions are said to be the questions that are asked during the development of a project or the project which is under construction. These questions will be used to identify the positive and negative sides of a project. These questions will give the clarity for the development of the project.

The following are the context-free questions that would be used to ask during the inception of a project.

· What are the barriers to the development of this project?

· What kind of benefits does the project will provide and who are going to get these benefits?

· Who is going to provide the payment for the work?

These questions will be asked at the inception phase of a project for the development of the project.

5.7. Develop a requirement gathering “kit.” The kit should include a set of guidelines for conducting a requirement gathering meeting and materials that can be used to facilitate the creation of lists and any other items that might help in defining requirements.

Ans:

Facilitated Requirements gathering “kit”

A project manager’s tool kit includes different collaborative sessions. In that requirement gathering is a major session. It consists of

1. Team building

2. Team communication

3. Change management

Requirements gathering meeting consists of

1. Gathering requirements using workshops or focus groups.

2. Prototyping and early, reiterative user testing of designs.

3. The re-use of software components.

4. A good schedule is necessary to design improvements to the next product version.

<u>Requirements gathering session kit</u>

This session is designed to aid in the gathering of requirements to support a project. This session is of great value when there is a need to gain alignment & consensus between stakeholders as to the requirements of a project.

<u>Outcomes & deliverables of session kit</u>

1. Accelerated requirement gathering

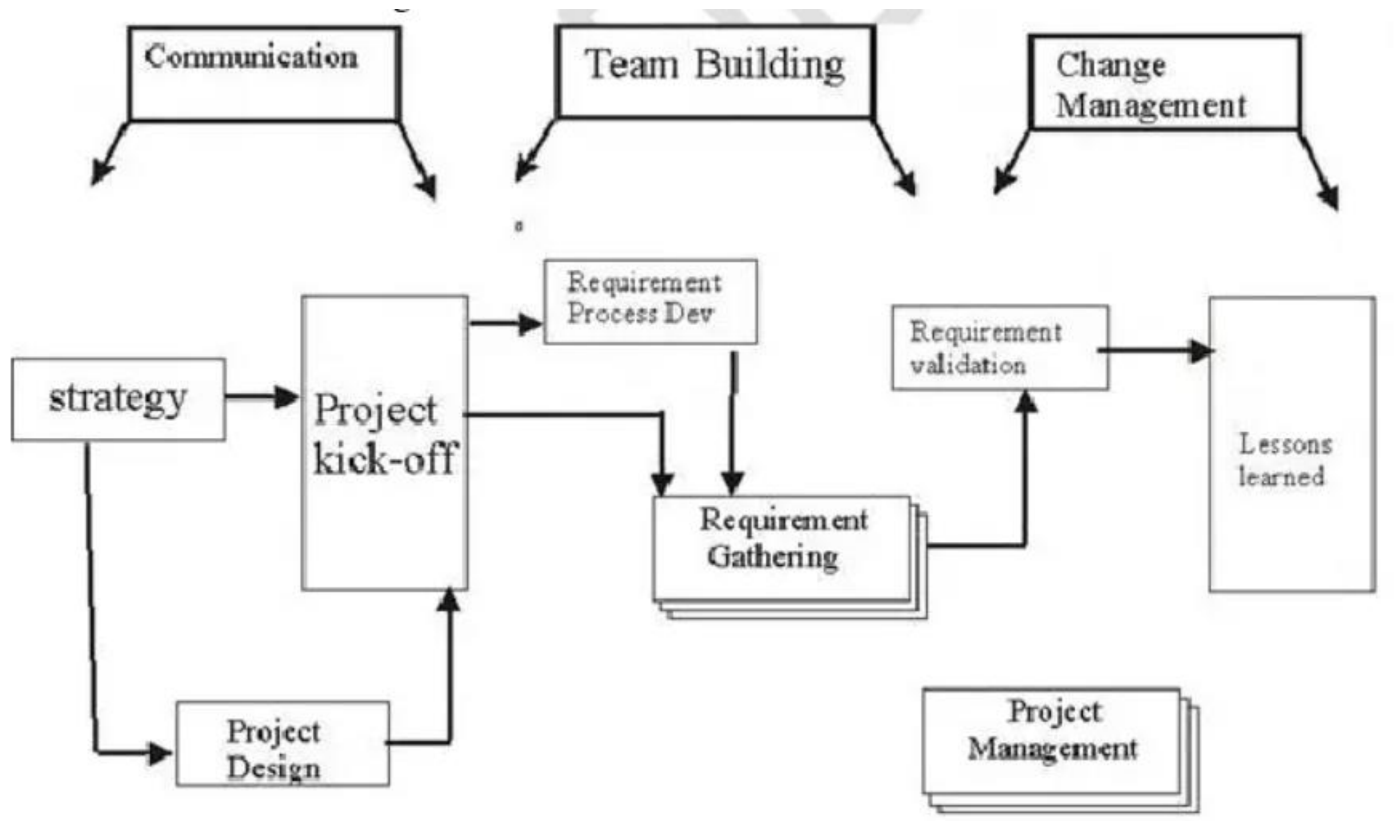
2. Consensus among user groups around project requirements.

3. Reduced rework.

4. Establishment of requirement priorities

5. Increased ownership for the project

6. Join contribution & alignment between business and IT stakeholders



5.8. Your instructor will divide the class into groups of four to six students. Half of the group will play the role of the marketing department and half will take on the role of software engineering. Your job is to define requirements for the Safe-Home security function described in this chapter. Conduct a requirements gathering meeting using the guidelines presented in this chapter.

Ans:

Requirements gathering essentials:

· Fours and clarity

· Format for specifying requirements

· The author of requirements document

· The language of requirements

· Accuracy is critical

· Minimizing risk of errant interpretation

· Conclusion

Safe home security function.

· Access camera surveillance via the internet

· Security & secret cameras

· Configure sate home system parameters

· Set alarms

· Activate sensors

5.9. Develop a complete use case for one of the following activities:

a. Making a withdrawal at an ATM

b. Using your charge card for a meal at a restaurant

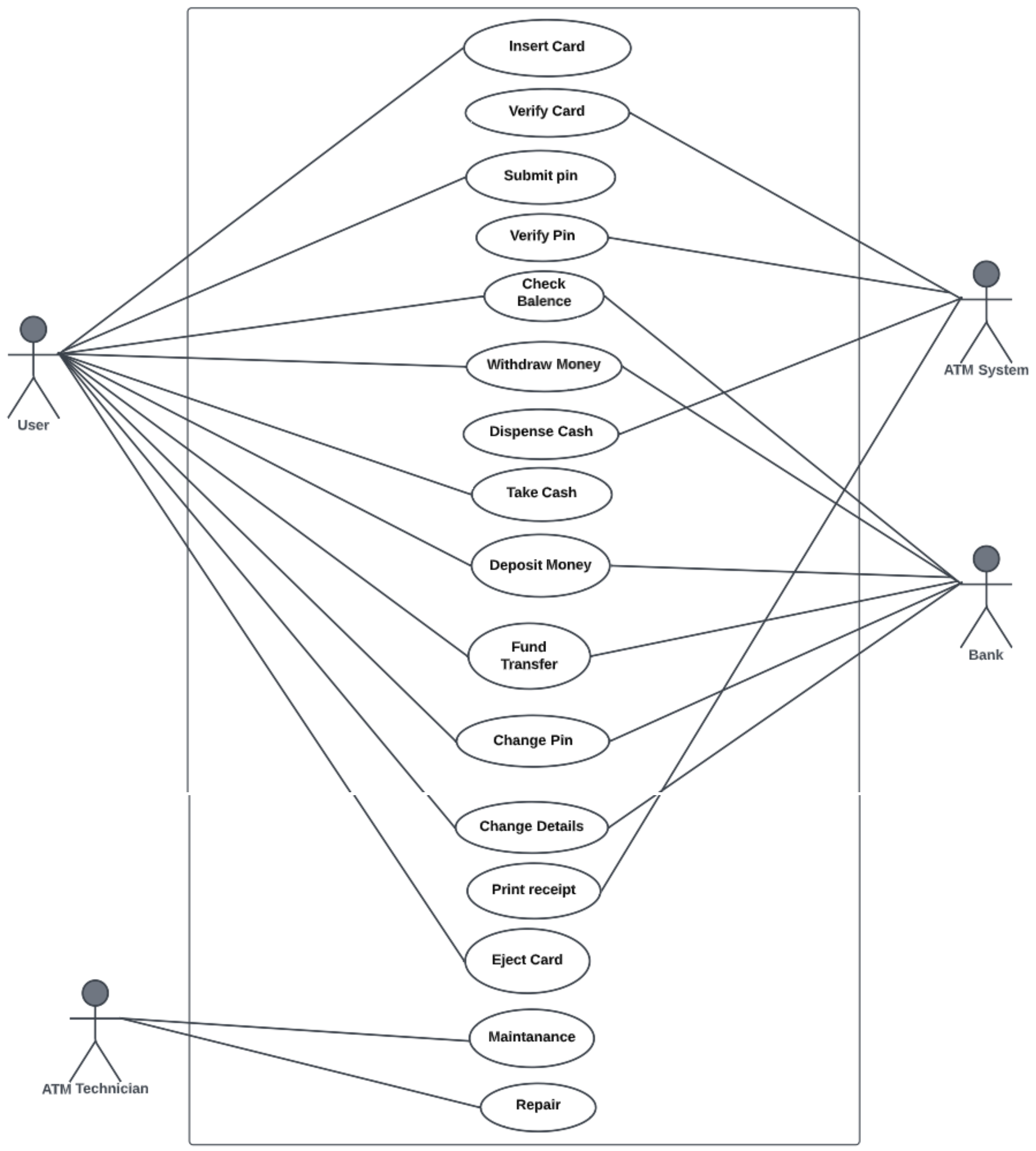
c. Buying a stock using an on-line brokerage account

d. Searching for books (on a specific topic) using an on-line bookstore

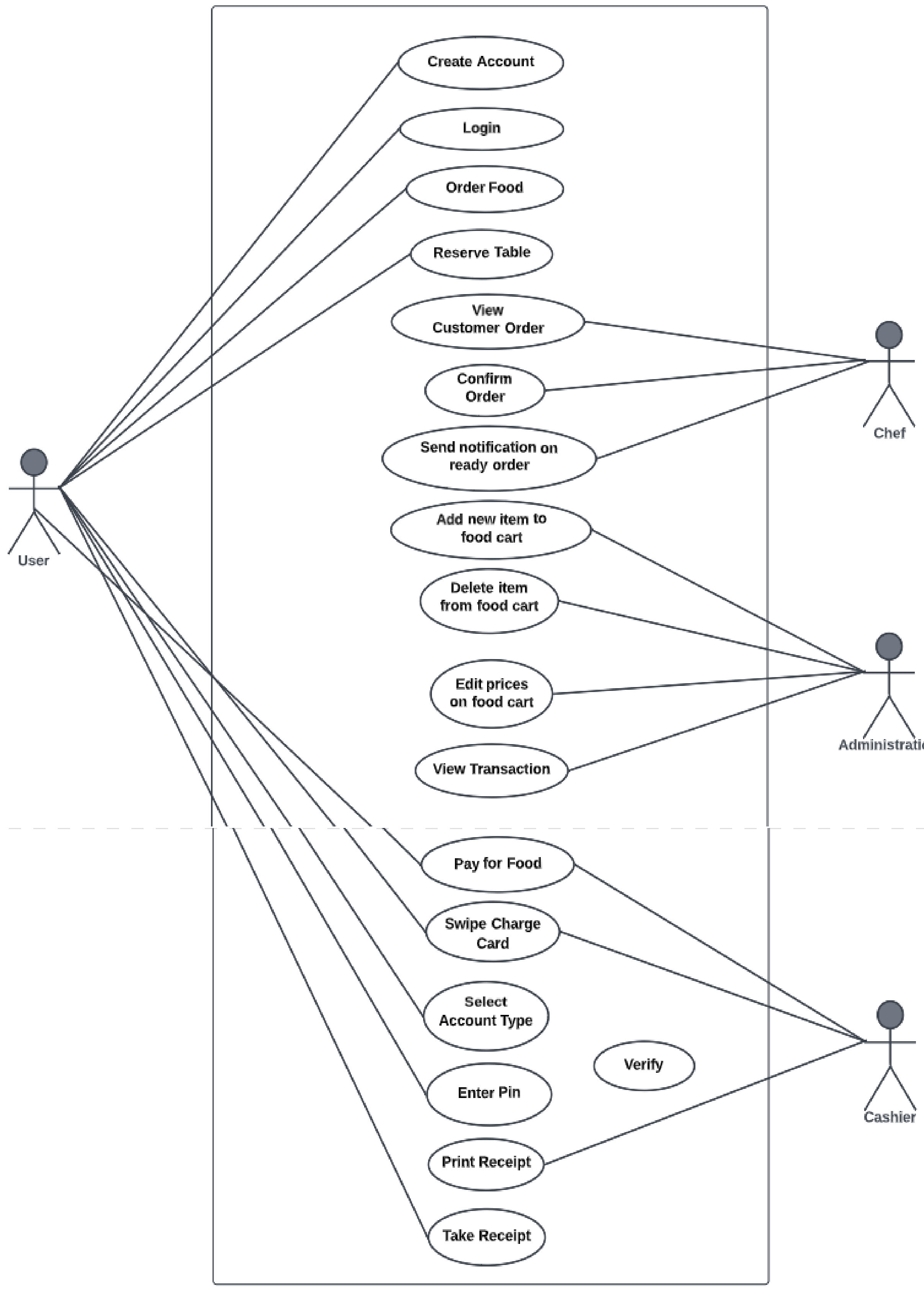
e. An activity specified by your instructor.

Ans:

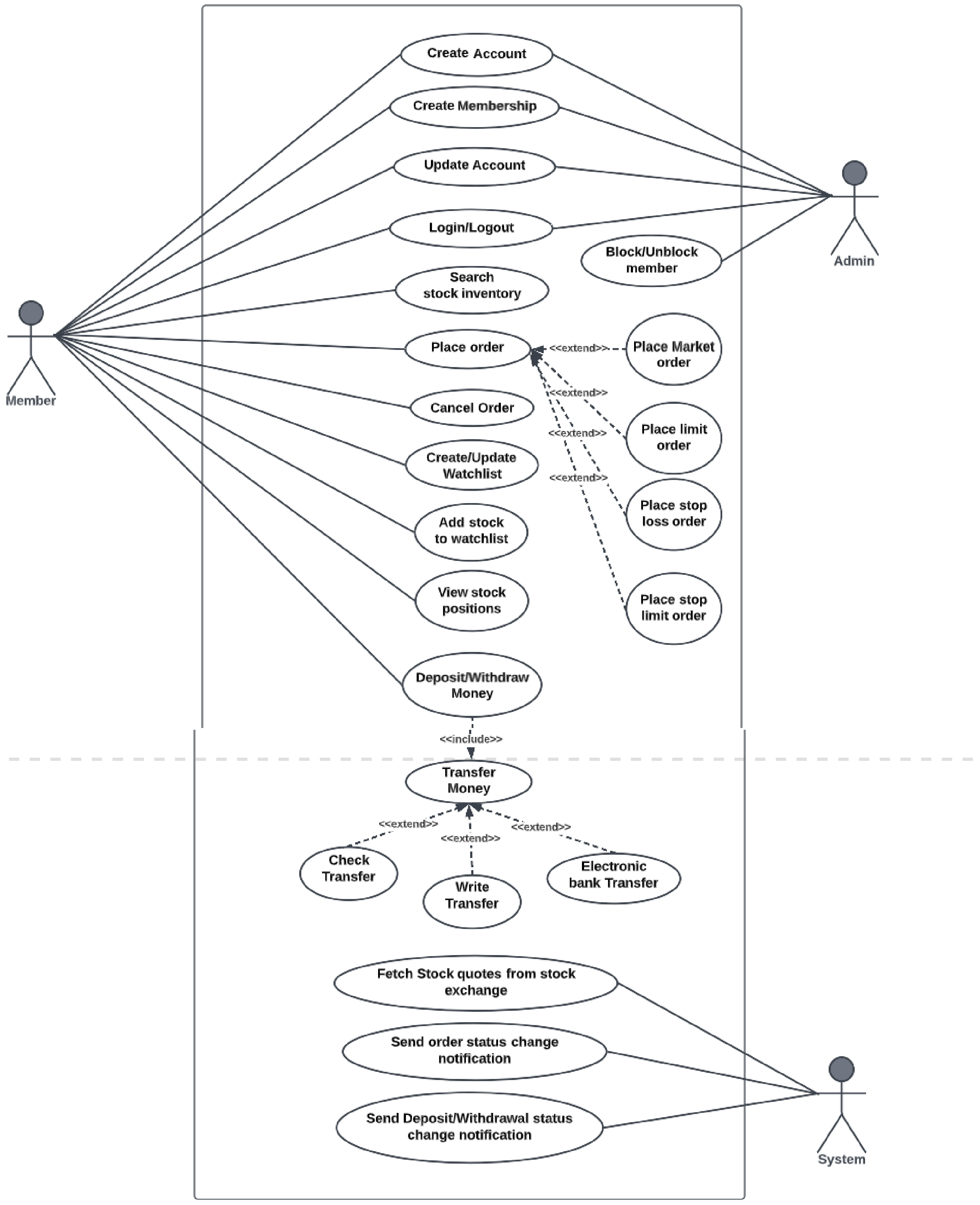
(a) Use Case Diagram for Making a withdrawal at an ATM



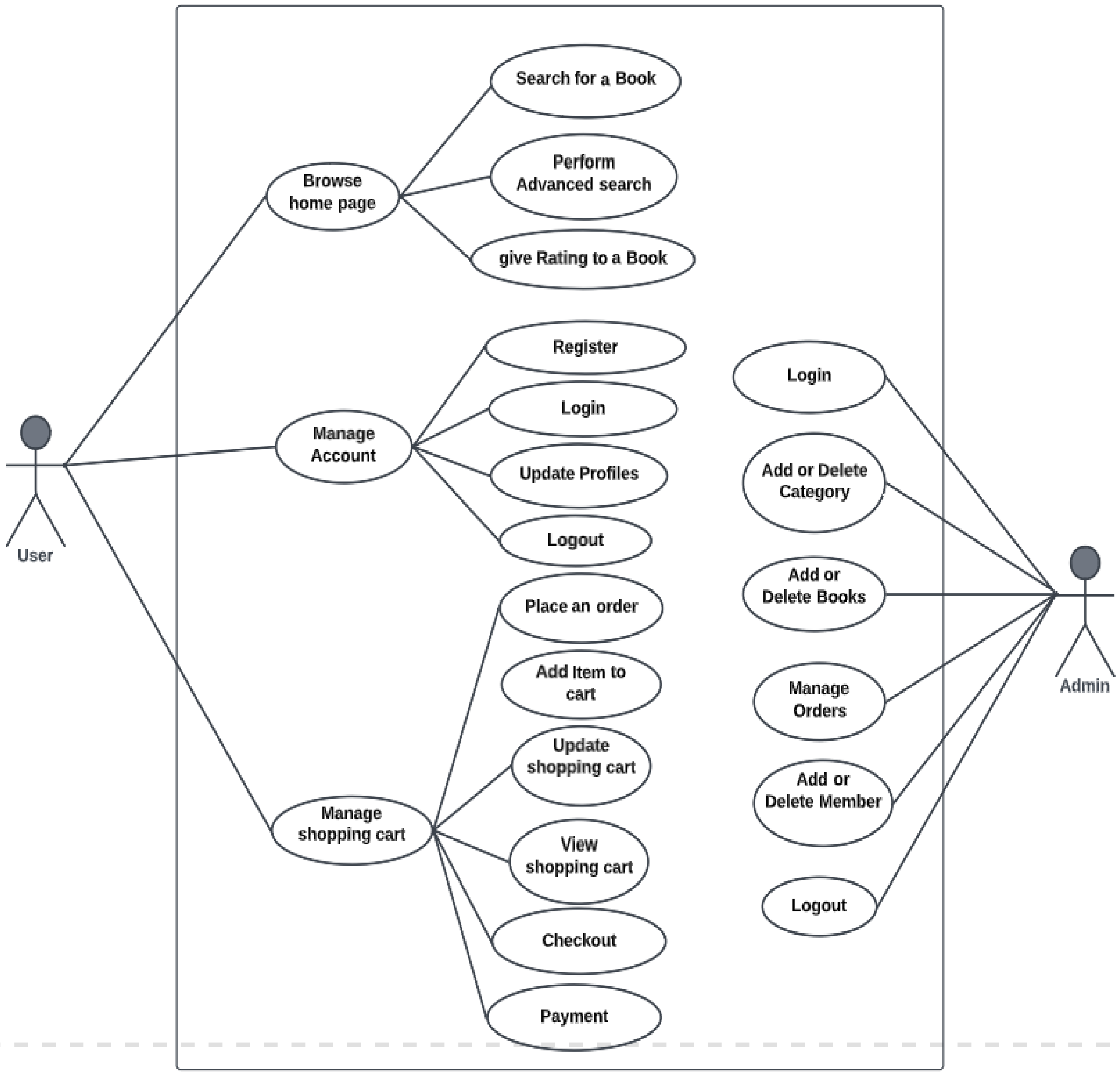
(b)Use Case Diagram for Using your charge card for a meal at a restaurant



(c) Use Case Diagram for Buying a stock using an on-line brokerage account



(d) Use Case Diagram for Searching for books (on a specific topic) using an on-line bookstore.



5.10. What do use case “exceptions” represent?

Ans:

It represents some error conditions the actor might encounter while using some function the system provided.

The purpose of exceptions in use-case is to identify some of the situations that are not covered in the preliminary use case.

· The situations that are identified in exceptions are covered while refining the preliminary use case.

· The use case must be complete and must deliver proper meaning to the user.

In the process of redefining the preliminary use case, at each step, various questions should be asked to get a clear picture of the purpose of that step.

Some of the questions to be asked are:

· Is there any alternative action that can be taken by the actor?

· Will there be any errors conditions to be handled that the user may encounter?

5.11. Describe what an analysis pattern is in your own words.

Ans:

Analysis pattern:

We can notice that certain things reoccur across all projects with in a specific application domain. These can be called as analysis patterns and represent something within the application domain that can be reused when modeling many applications. Information about an analysis pattern is presented in a standard template:

Pattern mane: A descriptor that captures essence of pattern.

Intent: Describes what the pattern accomplishes & represents.

Motivation: A scenario that illustrates how pattern can be used to address the problem.

Forces and context: A descriptor of external issues that can affect how the pattern is used and also the external issues that will be resolved when pattern is applied.

Solution: A description of how the pattern is applied to solve the problem with an emphasis on structural and behavioral issues

Consequences: Addresses what happens when the pattern is applied.

Design: Discusses how the analysis pattern can be achieved through the use of known design patterns.

Known uses: Examples of uses within actual system.

Related patterns: one or more patterns that are related to the named pattern.

5.13. What does win-win mean in the context of negotiation during the requirements engineering activity?

Ans:

A “Win-Win” situation is where the customer wins by getting the system or product that satisfies the majority of the customer’s needs and the software team wins by working to realistic and achievable budgets and deadlines.

The customer and the developer enter into a process of negotiation, where the customer may be asked to balance functionality performance and other product or system characteristic against cost and time to market. The intent of negotiation is to develop a project plan that meets the needs of the customer while at the same time reflecting the real – world constraints that have been placed on software team.

The best negotiations strive for a “Win – Win” result. The customer wins by getting the system that satisfies the majority of the customer’s needs and the software team wins by working to realistic and achievable budgets and deadlines.

Win – win result can be achieved by successful completion of

1. Identification of system or subsystem’s key stakeholders

2. Determination of the stokeholds “Win conditions”.

3. Negotiate of stakeholders win conditions to reconcile them into a set of win-win condition for all concerned.

5.14. What do you think happens when requirement validation uncovers an error? Who is involved in correcting the error?

Ans:

When the requirement validation uncovers an error, it has each requirement against a set of checklist questions. It then has a review team looking into it. The review team includes software engineers, customers, users, and other stakeholders who examine the specification looking for errors in content or interpretation, areas where clarification may be required, missing information, inconsistencies (a major problem when large products or systems are engineered), conflicting requirements, or unrealistic (unachievable) requirements.

It is extremely desirable to detect errors in the requirements before the design and development of the software begin. The basic objective of the requirements validation is to ensure that the SRS reflects the actual requirements accurately and correctly.

Omission, inconsistency, incorrect fact and ambiguity are the common errors in requirements. As requirements are generally textual documents that cannot be executed, inspections are eminently suitable for requirement validation. Requirement validation reviews are conducted to uncover the errors that commonly occur during requirements gathering.

Because requirement specification formally specifies something that is originally existed informally in people’s minds, requirements validation must involve the clients and the users.

Chapter: 6

6.1. Is it possible to begin coding immediately after an analysis model has been created? Explain your answer and then argue the counterpoint.

Ans:

The analysis model will serve as a basis for the design and coding. It is possible to begin coding after objects, attributes; relationships are analyzed in analysis phase however the design will suffer as a result of explicit architecture design will not have been considered. Interfaces will have been developed in haphazard manner and global data structure will not have been explicitly designed.

6.2. An analysis rule of thumb is that the model "should focus on requirements that are visible within the problem or business domain." What types of requirements are not visible in these domains? Provide a few examples.

Ans:

<u>Thumb rule of Analysis</u>

· The model should focus on requirements which are visible with in problem or business domain.

· Abstraction levels are high in thumb rule

· New elements added in the analysis model should help in better understanding of the software requirements of the system.

· These elements provide better understanding of the functionality, behavior and information domain of the system.

· Coupling should be minimized throughout the system

· The model should be simple

<u>The following are the requirements not visible in these domains:</u>

· Infrastructure requirements

o Functions required to manage global data

o Functions required to implement network communications

o Functions required to implement user interface

6.3. What is the purpose of domain analysis? How is it related to the concept of requirements patterns?

Ans:

Domain analysis is an on-going software engineering activity that is not connected to anyone software project

<u>Purpose of domain analysis:</u>

The key to reusable software is captured in domain analysis in that it stresses the reusability of analysis and design.

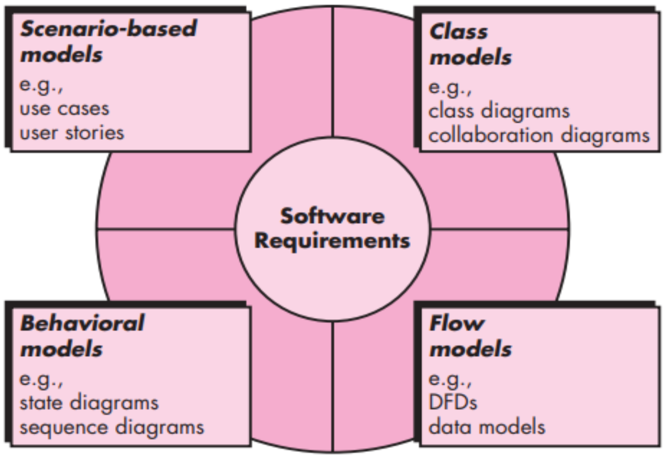
<u>Domain analysis for requirement patterns:</u>

The requirement is to find analysis pattern that are widely used with in specific application domain and to identify a set of objects and classes that characterize the domain.

6.4. Is it possible to develop an effective analysis model without developing all four elements shown in Figure 6.3? Explain.

Ans:

It is possible to develop an effective analysis model without developing all four elements. The specific content of each element may differ from project to project. Each element of the requirements model presents the problem from a different point of view.



Scenario-based elements depict how the user interacts with the system and the specific sequence of activities that occur as the software is used.

Class-based elements model the objects that the system will manipulate, the operations that will be applied to the objects to affect the manipulation, relationships (some hierarchical) between the objects, and the collaborations that occur between the classes that are defined.

Behavioral elements depict how external events change the state of the system or the classes that reside within it.

Finally, flow-oriented elements represent the system as an information transform, depicting how data objects are transformed as they flow through various system functions.

However, each element presents a different view of the problem to be solved and therefore provides the ability to see potential problems, inconsistence or omissions more clearly. In addition, if all four elements of the analysis model are developed, the transition to design is simplified.

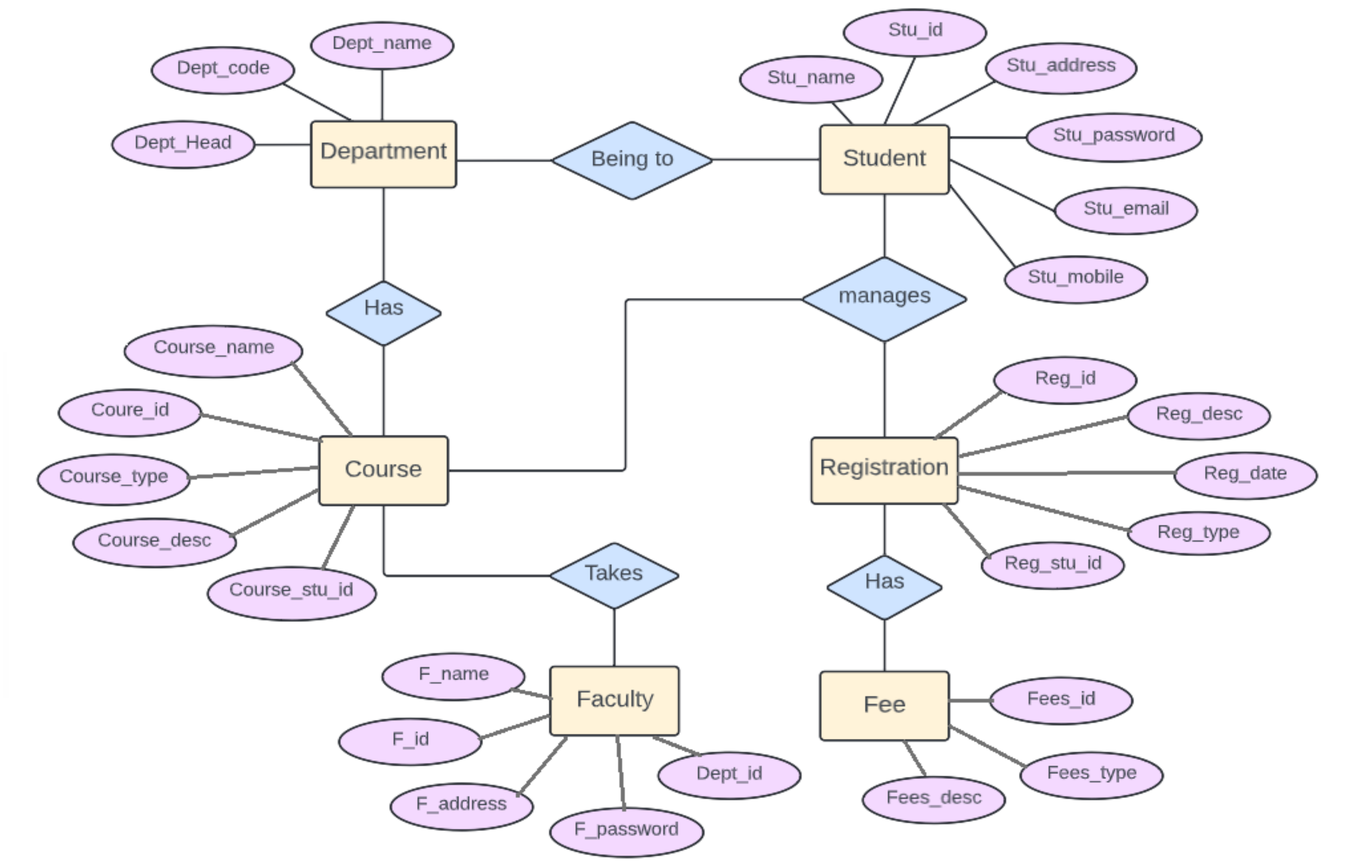
6.5. You have been asked to build one of the following systems:

a) **a network-based course registration system for your university.**
b) **a Web-based order-processing system for a computer store.**
c) **a simple invoicing system for a small business.**
d) **an Internet-based cookbook that is built into an electric range or microwave.**

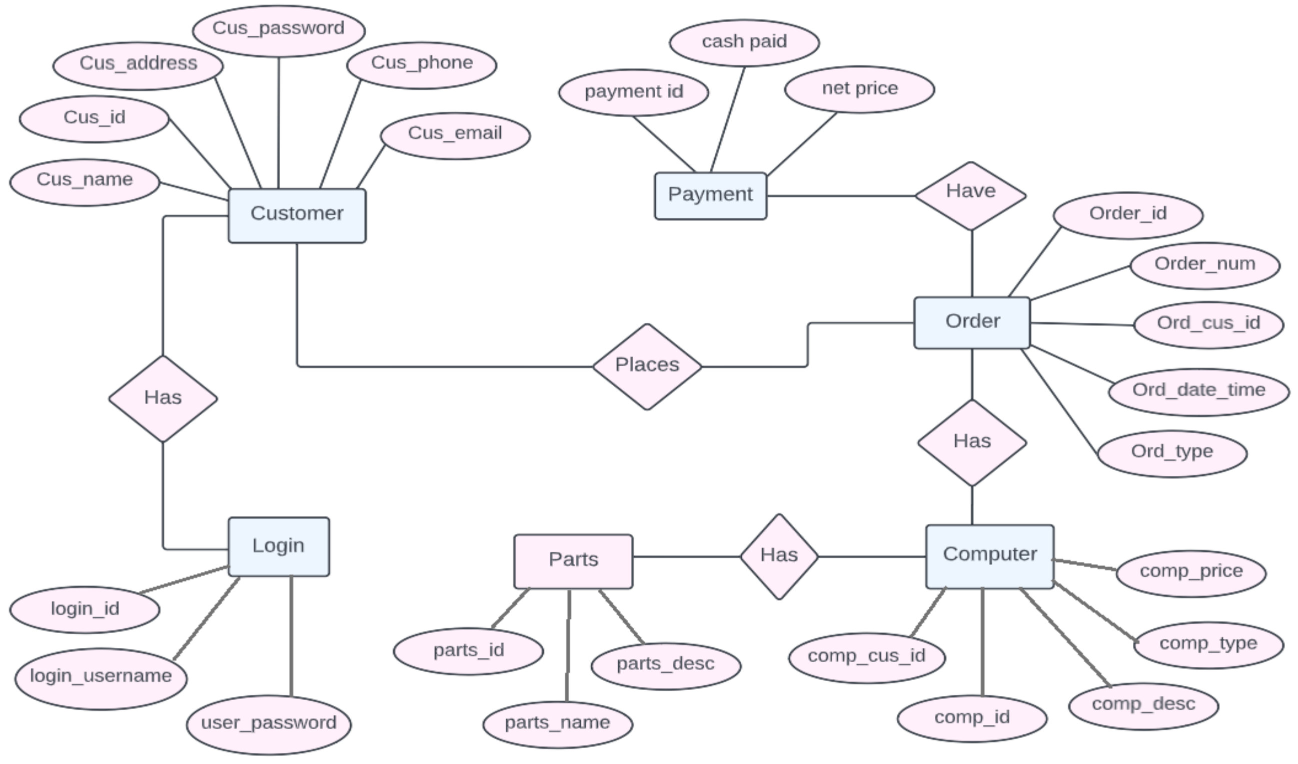
Select the system that is of interest to you and develop an entity-relationship diagram that describes data objects, relationships, and attributes.

Ans:

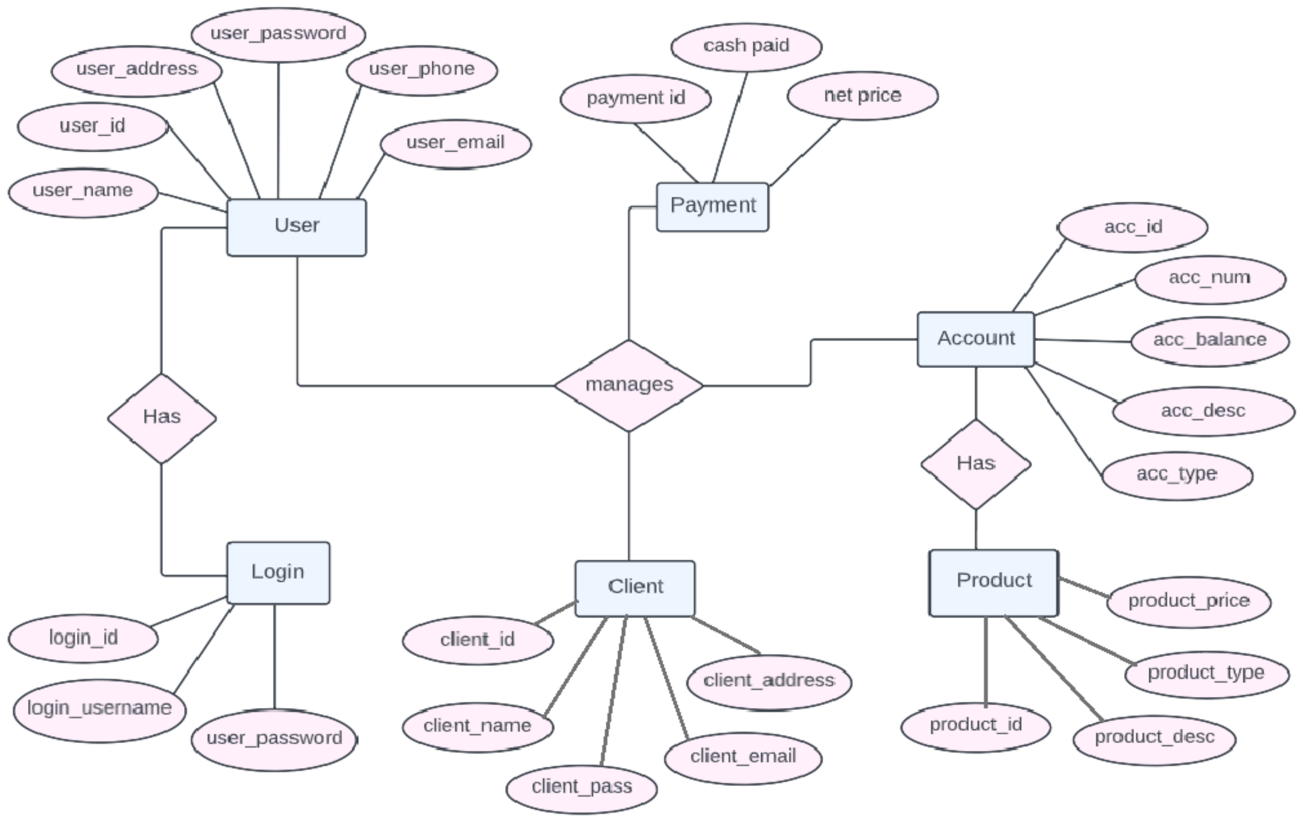
(a) a network-based course registration system for your university.



(b) a Web-based order-processing system for a computer store.



(c) a simple invoicing system for a small business.



6.6. The department of public works for a large city has decided to develop a Web-based pothole tracking and repair system (PHTRS). A description follows: Citizens can log onto a website and report the location and severity of potholes. As potholes are reported they are logged within a "public works department repair system" and are assigned an identifying number, stored by street address, size (on a scale of 1 to 10), location (middle, curb, etc.), district (determined from street address), and repair priority (determined from the size of the pothole). Work order data are associated with each pothole and include pothole location and size, repair crew identifying number, number of people on crew, equipment assigned, hours applied to repair, hole status (work in progress, repaired, temporary repair, not repaired), amount of filler material used, and cost of repair (computed from hours applied, number of people, material and equipment used). Finally, a damage file is created to hold information about reported damage due to the pothole and includes citizen's name, address, phone number, type of damage, and dollar amount of damage. PHTRS is an online system; all queries are to be made interactively.

a. Draw a UML use case diagram for the PHTRS system. You'll have to make a number of assumptions about the manner in which a user interacts with this system.

b. Develop a class model for the PHTRS system.

c. Develop an activity and swim-lane diagram for one aspect of PHTRS.

Ans:

(a)

The department of public works for a city has decided to develop a web-based Pot Hole Tracking and Repair System (PHTRS). A UML use case diagram states the relationship between the actors in the system with the use cases.

To construct a unified modelling language (UML) use case diagram first identify the actors for the system. For PHTRS, actors can include: -

· Citizen

· PHTRS employees

· Contractor

· Repair Crew

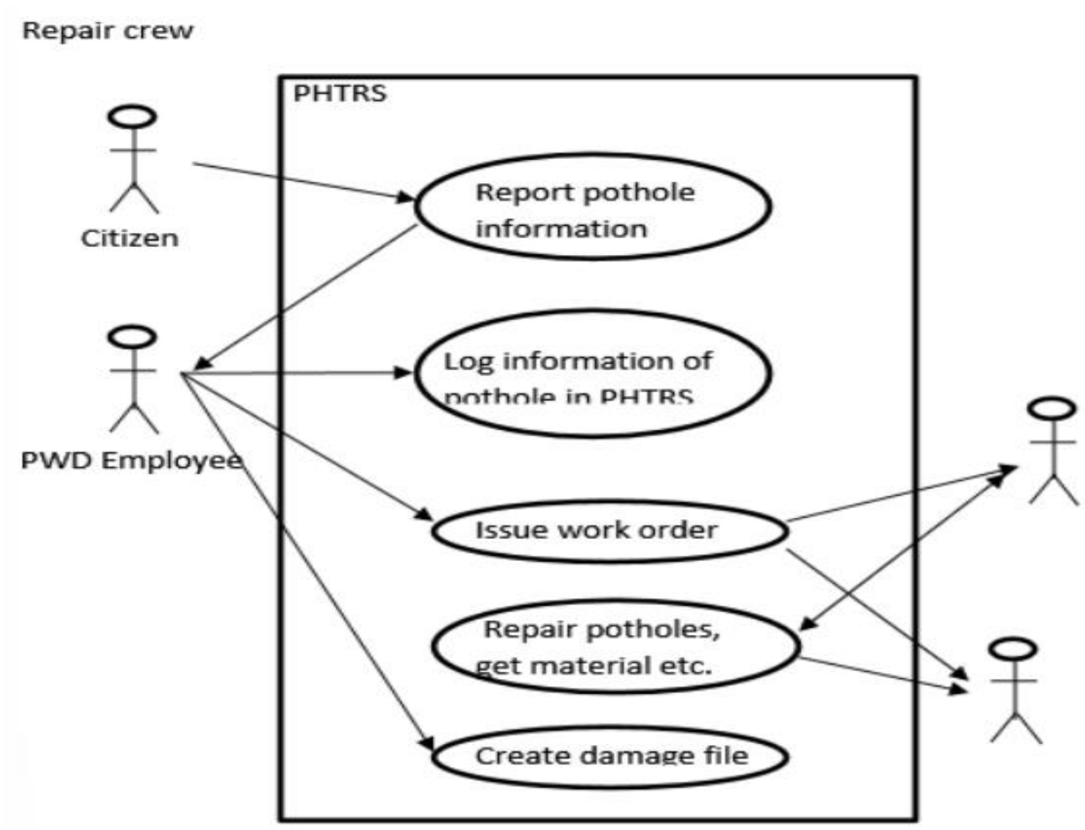
Then, identify the functionalities to be identified as use case for the system. For PHTRS, functionalities can include: -

· Citizen reports pothole information like its severity and location.

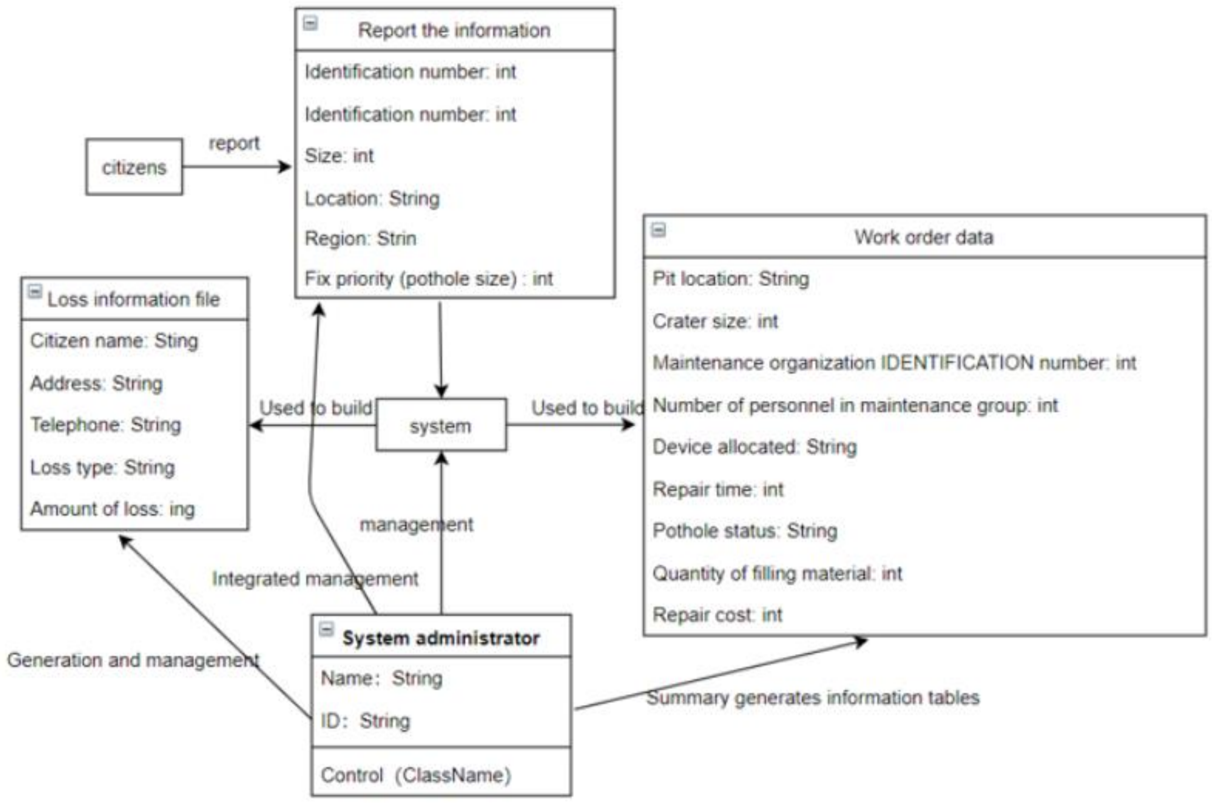
· Record pothole information like identification number, size, location, address, repair priority in PHTRS.

· Issue work order with data like size, location, repair crew id number, crew size, equipment assigned, hours worked, status, cost or repair etc.

· Create a PHTRS damaged file with information like citizen name, phone number, and address, type of damage and cost of damage.

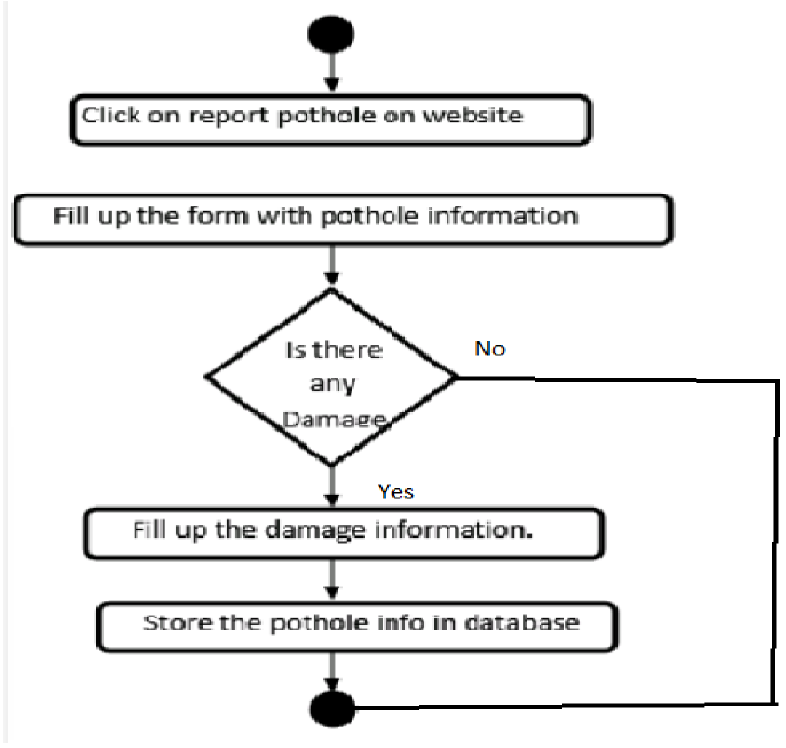

Repair crew
PHTRS
Citizen
PWD Employee
Report pothole information
Log information of pothole in PHTRS
Issue work order
Repair potholes, get material etc.
Create damage file

(b) a class model for the PHTRS system

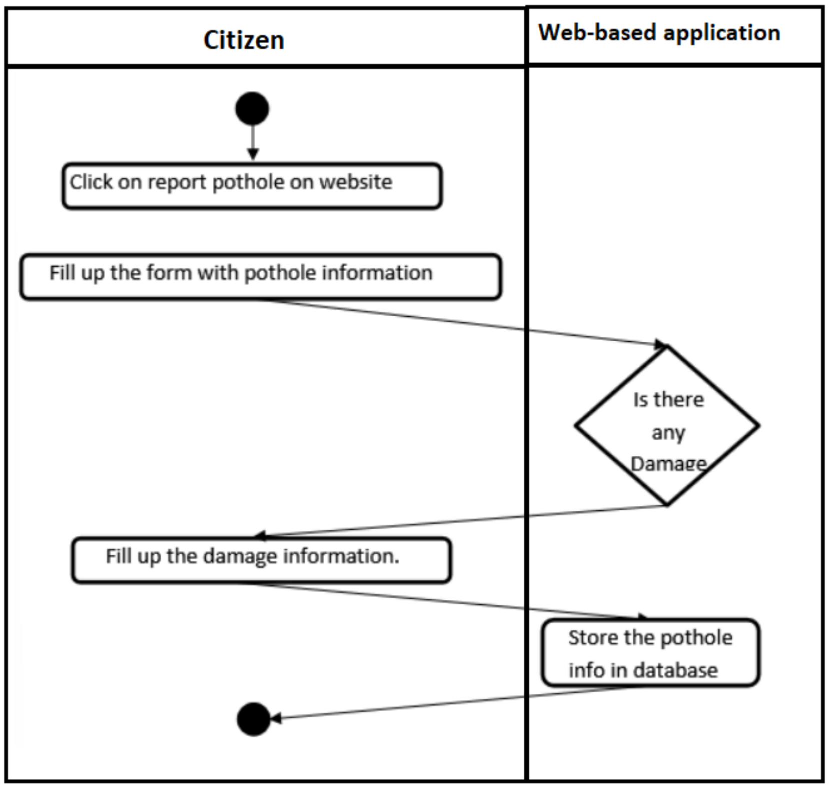


(c)

Activity diagram:


Click on report pothole on website
Fill up the form with pothole information
Is there any Damage
No
Yes
Fill up the damage information.
Store the pothole info in database

Swim-lane diagram


Citizen
Web-based application
Click on report pothole on website
Fill up the form with pothole information
Is there any Damage
Fill up the damage information.
Store the pothole info in database

6.7. Write a template-based use case for the Safe-Home home management system described informally in the sidebar following Section 6.5.4.

Ans:

Use case: Same Home residential management functionality
Key players: homeowner, residential management system
Objective: To control all kinds of electronic devices in the house
Early conditions: The system supports sensor function and has Internet interface
Trigger: The owner decides to control the equipment in the house through an interface or the Internet

Scene:

- Access to the system: The homeowner USES the residential management system through the residential management interface on the PERSONAL computer or the Internet, and enters the account password or USES face recognition to access the system
- Function selection: The system management menu is displayed on the PC screen for the owner to select the corresponding function
- Operate the lighting management system: If the owner chooses to control the lighting in the room, the lighting in the house will be controlled by the wireless interface controller.

- Thumbnails of each room and the location and number of lighting equipment should be displayed on the screen, which is convenient for the owner to choose and control
- Functional application: If the owner decides to change a certain lighting equipment, the system will control to change the state of the corresponding lighting equipment through wireless sensor and control interface
- Return menu: The owner selects the return menu and the computer screen displays each control system module again
- Homeowner chooses thermostat: If the homeowner chooses this feature, the system immediately displays the floor plan of the house and shows the location of each heating and air conditioning unit for the homeowner to choose from
- Temperature regulation by the system: If the owner determines the location and data of temperature regulation, the system will regulate the temperature of the air conditioning or heating equipment at the corresponding location through wireless sensors
- The owner returns and selects the controlled audio-visual equipment: a list of all available equipment is displayed on the screen
- Owner determines the equipment: The system sends data and control information through sensors and applies it to the equipment to complete the control

Exception:

- Failed to connect to the Internet: check the network interface and network status and try to connect again
- Incorrect account password: Re-enter the account or password
- Face recognition failure: recertification within a limited number of times
- Sensor system error: you can choose to reply the default setting of the sensor in the system or use it after checking the sensor system offline

Priority: Must be achieved

When available: The first increment

Frequency of use: multiple times per day

How to use: from a control panel on your PC or the Internet

Secondary actor: sensor system

Sensor: Wired or wireless interface

Unresolved issues:

- Can you easily operate this system without entering the password?
- How many times can face recognition be done at most?
- How many meters can the system be operated in the absence of network
- How to carry out safety inspection and maintenance of the system

6.8. Develop a complete set of CRC model index cards on the product or system you chose as part of Problem 6.5.

Ans:

Class-Responsibility-Collaborator (CRC) Models provide a means to identify and organize the classes that are significant to system requirement.

A CRC model is a set of index cards which embody classes. They are divided into three portions. Top portion is for class name. The body of the card is used to list the class responsibilities on the left and the collaborators on the right.

Responsibilities are basically attributes and operations in the class. Collaborators are related classes that are essential to provide a class with the information required to complete a responsibility.

The set of CRC model index cards for a network based university course registration system, can be:-

Student	
Student id Name Address Phone No Email Average marks Guardians Name Guardians Address Guardians Phone no Provide list of courses taken Calculate average marks Calculate Total fees & Bill	Course Taken Course

Professor	
Professors Id Name Address Phone No Email Salary Qualification Experience Provide list of course taught	Section Course

Course	
Course Id Course Name Fees Professor Credit hours Assign a course to a teacher	Professor

Course Taken	
Student Id Course Id Marks Fees paid Get all courses taken by a student Get all students enrolled in a course Get all courses taught by a professor	Course Student Professor

Section	
Section id Registered Student Waiting list of students Course id Professor id Room Add a student to course section Drop a student from a course section	Student Course Taken

6.9. Conduct a review of the CRC index cards with your colleagues. How many additional classes, responsibilities, and collaborators were added as a consequence of the review?

Ans:

Reviewing the Class-Responsibility-Collaborator (CRC) Model for a network-based course registration system in a university, the following changes can be incorporated.

Send the schedule of the classes that student needs to attend via email. To accomplish this, add:

- Class: StudentSchedule
- Responsibilities:
    - In class StudentSchedule
        - studentID: stores the identification of student for whom the schedule is required.
        - sectionID: stores the identification of section for whom the schedule is required.
        - ProfessorID: stores the identification of professor for whom the schedule is required.
        - Date: date of scheduled lecture
        - Time: Time of scheduled lecture
        - addSectionToSchedule(): Add schedule elements to the table for a student. This is called when a student takes up a new course
        - dropSectionFromSchedule() : Drop Schedule elements from the able for a student. This is called when a student drop a course.
    - In class Student (update)
        - printSchedule(): Print the lecture schedule for a student.
    - In class professor (update):
        - printSchedule(): Print the lecture schedule for a professor

- Collaborations:
    - In class StudentSchedule
        - Student: store information of student
        - Professor: stores information of professor
        - Course: stores information of professor
        - CourseTaken: stores information of course taken by a student
    - In class Student (update)
        - StudentSchedule() : To get the lecture schedule for a student
    - In class Professor (update)
        - StudentSchedule() : To get the lecture schedule for a professor

Hence, the CRC model will contain a new index card namely

Student Schedule	
Student id Section Id Professor Id Date Time Add section to a student's schedule Drop section form a student's schedule	Course Taken Course

Student	
Student id Name Address Phone No Email Average marks Guardians Name Guardians Address Guardians Phone no Provide list of courses taken Calculate average marks Calculate Total fees & Bill Print a schedule for student	Course Taken Course StudentSchedule

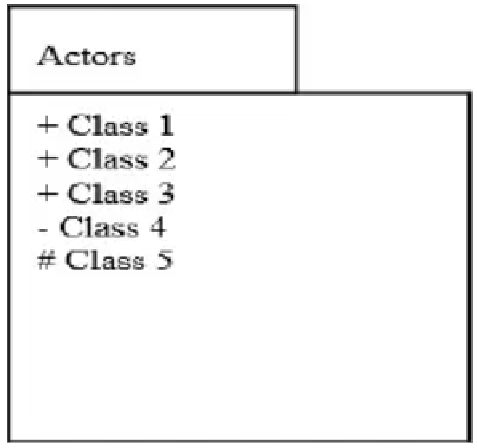
Professor	
Professors Id Name Address Phone No Email Salary Qualification Experience Provide list of course taught Print a schedule for a professor	Section Course StudentSchedule

6.10. What is an analysis package and how might it be used?

Ans:

In a software engineering, various elements of the analysis model such as use cases, analysis classes are categorized in a manner that packages them as a grouping. This is called an analysis package. To build a software, lot of classes are there which fall in one category and other classes fall in other categories. Thus, classes that come under a category are grouped under a name, that is called an analysis package.

An analysis package is represented shown in figure:



Analysis package+, -, and # are access specifiers for public, private and protected respectively.is for

Uses:

The analysis packages are used to derive the requirements.

Chapter: 7

7.1. What is the fundamental difference between the structured analysis and object-oriented strategies for requirements analysis?

Ans:

In structured analysis, data objects are modeled in a way that defines their attributes and relationships. An object-oriented analysis focuses on the definition of classes and manner in which they collaborate with one another to effect customer requirements.

1. Structured Analysis :
Structured analysis is a method of development that allows and gives permission to the analyst to understand and know about the system and all of its activities in a logical way. It is simply a graphic that is used to specify the presentation of the application.

2. Object-Oriented Analysis :
Object-Oriented Analysis (OOA) is a technical approach generally used for analyzing and application designing, system designing, or even business designing just by applying object-oriented programming even with the use of visual modeling throughout the process of development to just simply guide the stakeholder communication and quality of the product. it is actually a process of discovery where a team of development understands and models all the requirements of the system.

7.2. In a data flow diagram, does an arrow represent a flow of control or something else?

Ans:

An arrow represents Data flow in a flowchart

Flowchart Arrow symbol

- Special shapes are used in flowcharts to depict various types of actions or steps in a process. The chronology of the stages, as well as the links between them, is depicted using lines and arrows.
- An arrow line sign in a flow chart is a connector that depicts the relationship between the representing shapes. The arrow line symbol depicts the flow's direction and order.

The arrow represents the in-flow and out flow of data. In a data flow diagram labeled arrows are also used. The labeled arrows used to represent data objects.

7.3. What is "information flow continuity" and how is it applied as a data flow diagram is refined?

Ans:

Information flow continuity is type of dataflow from level to level. The data objects that flow into the system or into any transformation at one level must be the same data objects or their constituent parts. That flow into the transformation at a more refined level.

7.4. How is a grammatical parse used in the creation of a DFD?

Ans:

While DFD is expanded from level 0 to level 1 model, the grammatical parse is used. It is described the context level bubble and identify all nouns (and noun phrases) and verbs in the narrative.

Here nouns and verbs are associated with one another. Nouns represent the data objects, external entities and data stores. Verbs represent the processes.

DFD is constructed by review the data model to isolate data objects and use a grammatical parse to determine operations and external entities (producers and consumers data).

7.5. What is a control specification?

Ans:

The control specification (CSPEC) used to indicate (1) how the software behaves when an event or control signal is sensed and (2) which process are invoked as a consequence of the occurrence of the event. The CSPEC contains a number of important modeling tools.

A control specification (CSPEC) represents the behavior of the system (at the level from which it has been referenced) in two different ways. The CSPEC contains a state diagram that is a sequential specification of behavior. It can also contain a program activation table (PAT)—a combinatorial specification of behavior.

A control specification represents the behavior of the system in two different ways.

Representation through a state diagram that is a sequential specification of behavior through a program activation table, a combinatorial specification of behavior.

7.6. Are a PSPEC and a use case the same thing? If not, explain the differences.

Ans:

No, they are not the same thing. PSPEC is a "mini-spec" that describes as a guide for design of a software component that will implement the bubble. It is used to describe all flow model process that appears at the final level of refinement. Whereas, use case determines the scenario of how system will be used. It is used to identify the primary elements and processes that form the system.

7.7. There are two different types of “states” that behavioral models can represent. What are they?

Ans:

In the context behavioral modeling, two different characterizations of states must be considered:

- The state of each class as the system performs its function
- The state of the system as observed from the outside as the system performs its function

<u>State diagram:</u> This is for analysis classes. One component of behavioral model is a UML state diagram that represents active states for each class and the events that cause changes between these active states.

<u>Sequence Diagram:</u> The second type of behavioral representation, called a sequence diagram in UML, indicates how events cause transitions from object to object.

7.8. How does a sequence diagram differ from a state diagram? How are they similar?

Ans:

State diagrams show the various state that are valid for an object. That could be a particular class or the system as a whole. This type of diagram shows what actions are valid for a given object, depending on what state it is currently in.

A sequence diagram typically shows the execution of a particular use case for the application and the objects that are involved in carrying out that use case. It could either show a single path or all of the various paths through the use case starting with an actor initiating some kind of action.

<u>Difference:</u>

A sequence diagram differs from a state diagram in the way of representing the behavior of the system.

• Unlike a state diagram that represents behavior without noting the classes involved, a sequence diagram represents behavior, by describing how classes move from state to state.

• A state diagram represents how an individual class changes its state, based on external events where as a sequence diagram shows the behavior of the software as a function of time.

<u>Similarity:</u>

Both the sequence diagram and the state diagram are the notation used for behavioral modeling. They are the two different behavioral representations.

7.9. Suggest three requirements' patterns for a modern mobile phone and write a brief description of each. Could these patterns be used for other devices? Provide an example.

Ans:

Modern mobile phones need various applications like camera, Bluetooth, Internet, Radio etc. Some of the requirement's patterns for such modern mobile phones are:

<u>Cameracontroller</u>

This pattern shows how to specify the sensors and actuators in a phone, used to perform camera controlling functions. This pattern uses the sensors for the zoom-in and zoom-out functions of a camera. A pull mechanism is used for passive sensors and a push mechanism for the active sensors. This pattern can be used in various devices that need a camera. For example, a computer that needs a web camera can use this pattern for camera controlling operations.

<u>MobileRadio</u>

This pattern provides functioning of an in-built mobile radio. It uses some technologies that enable stations to communicate over wide areas. This pattern can be used in cars to provide the facility of a radio system

<u>WirelessInternet</u>

This pattern provides the mobile phones the access to internet without any modems or wires. This pattern can also be used in laptops, which are mobile. Due to their mobility, a wireless Internet access is much useful for laptops. So, this pattern can be used

7.11. How much analysis modeling do you think would be required for SafeHomeAssured .com? Would each of the model types described in Section 7.5.3 be required?

Ans:

For a SafeHomeAssured.com application, the degree to which analysis modeling for an application is emphasized depends on the following factors:

• Size and complexity of WebApp increment

• Number of stakeholders

• Size of the WebApp team

• Degree to which members of the WebApp team worked together before

• Degree to which the organization's success is directly depends on the success of the WebAppSmaller the project and lesser the number of stakeholders and if the application is less critical, it is reasonable to apply a more lightweight analysis approach. In the case ofSafeHomeAssured.com, a more lightweight analysis approach can be applied because,

• We need to analyze only that part of the problem that is relevant to the design work for the increment to be delivered.

• It does not need any stakeholders. Only the owner of the house will be concerned about the project.

• Also, the success of a home is not a direct dependent on this SafeHomeAssured.com There are five main classes of models.

They are:

• Content model

• Interaction model

• Functional model

• Navigation model

• Configuration model

Some specific models depend largely upon the nature of the WebApp. ForSafeHomeAssured.com not each of these model types are required. For this system, we could validly design the overall website aesthetics (layouts, color schemes, etc.) without having analyzed the functional requirements for e-commerce capabilities. So, the functional model is not required.

7.12. What is the purpose of the interaction model for a WebApp?

Ans:

The interaction model describes the manner in which users interact with the WebApp. The purpose of the interaction model for a Web Apps is to enable a "Conversation" between an end user and application functionality, content, and behavior. This conversation can be described using an interaction model that can be composed of one or more of the following elements:

• Use cases

• Sequence diagrams

• State diagrams

• User interface prototypes

<u>• Use cases:</u>

It is the main element of WebApp interaction models. A set of use cases insufficient to describe the interaction at analysis level.

<u>• Sequence diagrams:</u>

It provides representation of manner in which user actions collaborate with analysis classes.

• State diagrams:

It indicates information required to move users between states and represents behavioral information. It can also represent potential navigation pathways.

• User interface prototype:

It provides the layout of the user interface, the content it presents, and the interaction mechanisms it implements

7.13. It could be argued that a WebApp functional model should be delayed until design. Present pros and cons for this argument.

Ans:

It could be argued that a WebApp functional model should be delayed until design. The pros and cons for this argument are:

Pros:

• Some Web Apps does not need functional requirements up to some phase. In such cases, it reduces the effort of unnecessary gathering of functional requirements.

• It saves some time.

Cons:

• Some requirements may not be addressed due to this delay till the design phase.

• Some Web Apps deliver a broad array of computational and manipulative functions that can be associated directly with content and the major goal of the WebApp. So, in some cases, even a major goal of user-WebApp interaction may sometimes go unassigned.

7.14. What is the purpose of a configuration model?

Ans:

The purpose of the configuration model is to describe the environment and infrastructure in which the WebApp resides. The configuration model in some cases is nothing more than a list of server-side and client-side attributes for Webapps.

7.15. How does the navigation model differ from the interaction model?

Ans:

A navigation model differs from an interaction model in addressing the requirements.

• A navigation model is concerned about how each user category will navigate from one-Web-App element to another, where as an interaction model presents the conversation between an end user and application functionality.

• The mechanics of navigation are defined as part of design, while the interaction model is basically described at an analysis level and further introduced in detail during design.