Department of
Computer Science and Engineering

Title: Requirement Specification and SDLC Model Selection for a specific project

Integrated Design Project II

CSE 406

Green University of Bangladesh

1 Objective(s)

- To find the requirement specification (both functional and nonfunctional) of a given Problem.
- Understand system development process as a life cycle.
- To produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

2 Problem analysis

Problem analysis for example of "ATM Management System" which is an electronic telecommunication system. This system involves the customers of financial institutions like banks, to carry out financial transactions, such as cash withdrawal, Balance Enquiry without any requirement of a bank clerk.
For an ATM Management System, a customer is a person who inserts or swipes plastic card with a magnetic stripe or a plastic smart card with a chip into an ATM containing a different card number. Post inserting the card the client is also asked to give his authentication by entering a special identification number (PIN) which must match with the PIN stored in the chip on the card (if the card has it equipped) or in the financial institution's database who has issued the card.
As huge data is to be maintained, so it is necessary to design and develop a system which will be efficient, error-free, automated and easily maintainable. The software AMS is to be developed for Automated Teller Machine (ATM). An Automated Teller Machine is computerized system which will provide a secure platform for customers of banks to perform financial transactions in public places such as roads, malls, offices etc.; without any human intervention. This system will provide a user-friendly interface to access financial facilities provided by banks.

3 Methodology

Requirement Specification Procedure:

3.1 Step 1: Introduction

Identify the product whose software requirements are specified in this document. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single sub system. Describe the different types of user that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

Project Scope

Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. An SRS that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.

3.2 Step 2: Overall Description

Product Perspective

Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

Product Features

Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.

User Classes and Characteristics
Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes.Distinguish the favored user classes from those who are less important to satisfy.

Operating Environment
Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

Design and Implementation Constraints
Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).

3.3 Step 3: System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

System Feature 1
Don't really say "System Feature 1." State the feature name in just a few words.

- Description and Priority

  Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).

- Stimulus/Response Sequences

  List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.

- Functional Requirements

  Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous,verifiable, and necessary.

3.4 Step 4: External Interface Requirements

User Interfaces
Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

Hardware Interfaces
Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

Software Interfaces
Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing

mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

Communications Interfaces

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

Nonfunctional Requirements

Performance Requirements

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

Safety Requirements

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

Security Requirements

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

Software Quality Attributes

Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

Other Requirements

Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.

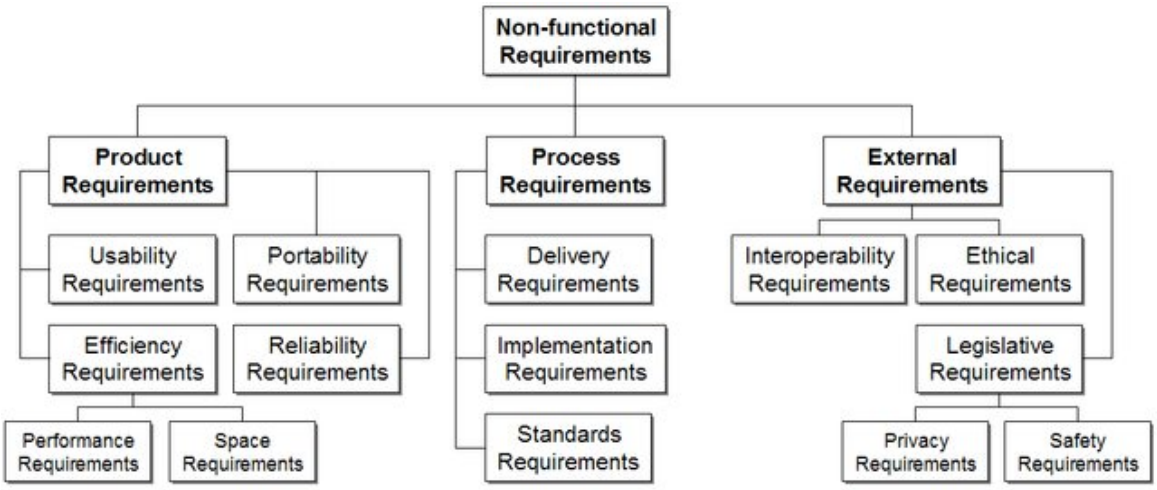

Figure 1: Non-Functional requirements of a System

4 Implementation

4.1 System requirements for ATM Management

- The system will be able to system which can easily store, retrieve and update data.

4.2 User requirements for ATM Management

- The system will be able to store, retrieve and update data automatically.
- The user would be able to call money request.
- The user would be able to withdraw money.
- The user would be able to make account inquire.
- The user would be able to transfer money with other accounts.

4.3 Non-Functional Requirements for ATM Management

- Efficiency

  System will be able to perform in real time.

  System will achieve the recognition rate (performance 90

- Validity Checking

  It will show the valid amount of money after any operation.

- Quality

  System must maintain an improved quality in every aspects.

- Authenticity

  System will serve the actual owner of the account.

- Adaptability

  Adaptability will be very high performing.

- Maintainability

  Addition, Deletion and updation any requirement if possible.

- Usability

  User flexibility (easy to configure the system and can train as per its convenience).

- Security:

  Both for the physical installation and from a cyber perspective.

- Reliability:

  This system will work 24/30/365.

5 Methodology Cont.

SDLC selection for a specific project: SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. A typical Software Development Life Cycle consists of the following stages −

5.1 Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas. Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

5.2 Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

5.3 Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product. A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

5.4 Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle. Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

5.5 Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

5.6 Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing). Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

6 SDLC Models

Following are the most important and popular SDLC models followed in the industry −

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model

6.1 Waterfall Model

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

This model is used only when the requirements are very well known, clear and fixed.

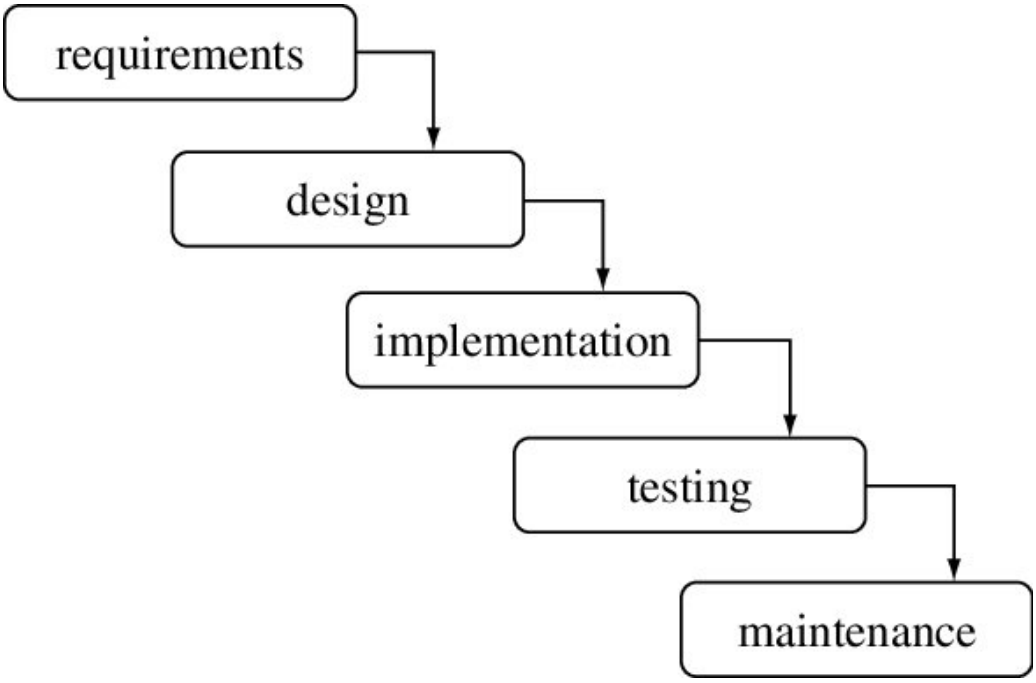

Figure 2: Waterfall model

6.2 Iterative Model

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

When requirements are defined clearly and easy to understand, When the software application is large and When there is a requirement of changes in future, this model is used.

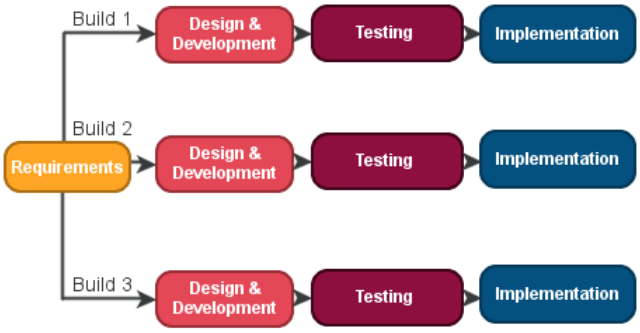

Figure 3: Iterative Model

6.3 V-Model

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

V-Model is used for small projects where project requirements are clear, Simple and easy to understand and use. This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product

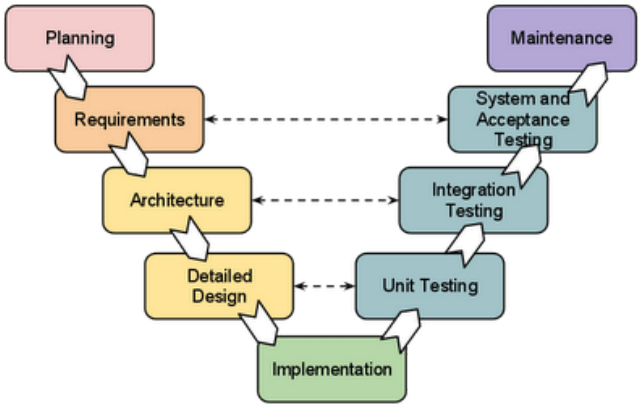

Figure 4: V-shape Model

6.4 Spiral Model

The spiral model has four phases – Identification, design, build, evaluation and risk analysis. A software project repeatedly passes through these phases in iterations called Spirals.

A Spiral model in software engineering is used when project is large, When releases are required to be frequent, When creation of a prototype is applicable and When risk and costs evaluation is important.

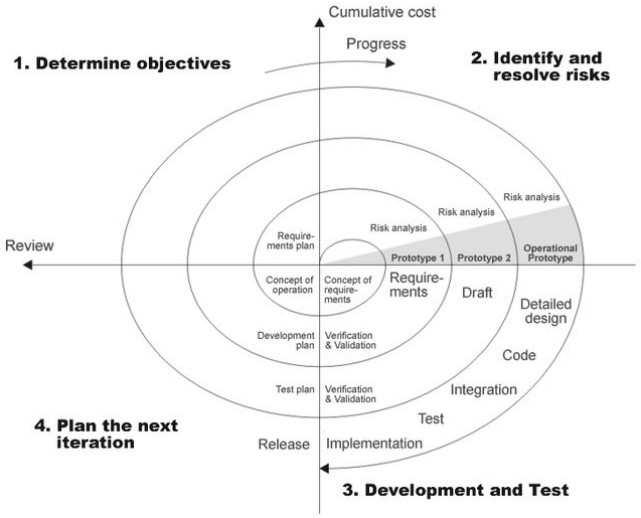

Figure 5: Spiral Model

6.5 Agile Model

The meaning of Agile is swift or versatile."Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.

The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

6.6 Prototyping Model

Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small scale facsimile of the end product and is used for obtaining customer feedback.

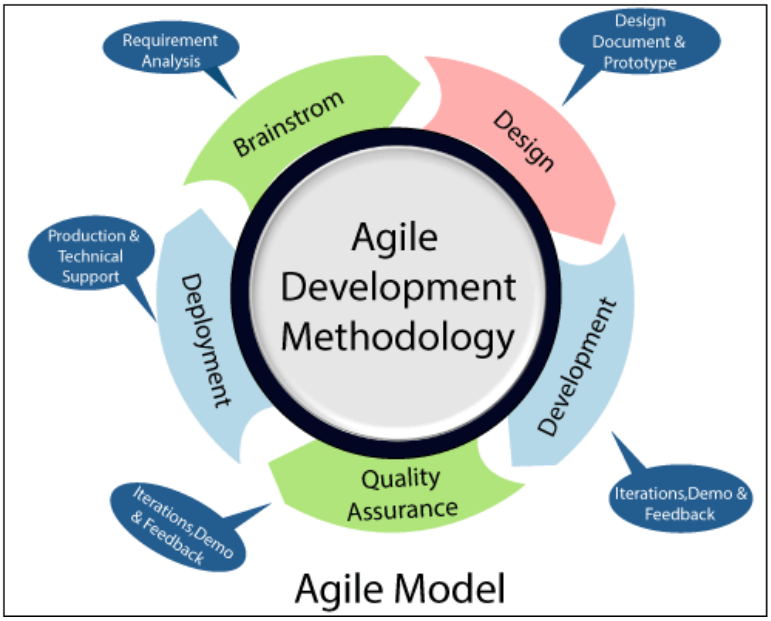

Figure 6: Agile Model

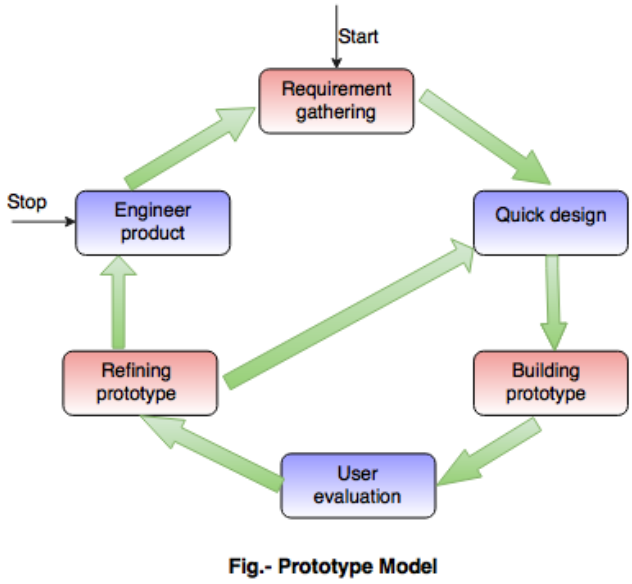

Figure 7: Prototype Model

7 Implementation in ATM Management system

Here we can see a table for SDLC Model for our project. We chose some category which is suitable for our project. Then we select some priority above the requirement for our project. Then see which category is support by which model. Then we chose the best suitable Model above the the priority we set for each category. Then we will set this SDLC Model for our Project.

In this SDLC Models, Waterfall scores 11, V-Shaped scores 23, Iterative scores 11, Spiral scores 22, Agile scores 5 and Prototype scores 11. Here our most important categories are supported by V-Shaped Model. So we will prefer V-Shaped Model as a best option for our project.

Table 1: Comparison matrix with different models

Priority	Criteria	Waterfall	V-shape	Iterative	Spiral	Agile	Prototype
5	Well known requirement	Yes	Yes	No	No	No	No
3	Technological knowledge	Yes	Yes	No	No	No	No
6	Efficiency	No	Yes	Yes	Yes	No	Yes
3	Risk analysis	No	No	No	Yes	No	No
5	User testing ability	No	No	Yes	Yes	Yes	Yes
5	Dependability and Security	No	Yes	No	Yes	No	No
3	Time consuming	Yes	Yes	No	Yes	No	No
Total- 30	Over all	11	23	11	22	5	11

8 Discussion & Conclusion

Based on the focused objective(s), to understand about system architecture and about the right choice of model selection (low level design), the additional lab exercise made us more confident towards the fulfilment of the objectives(s).

9 Lab Task (Please implement yourself and show the output to the instructor)

1. Prepare a functional and non-functional project requirements of a library management system.
2. Select the best model of a library management system by creating a model matrix.

10 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.