MINIMUM NUMBER OF PAGES FOR CHAPTER 2 IS 20 PAGES

Chapter 2

METHODOLOGY

This chapter presents software development methodology that was used in this study. The data gathering techniques, and other sources of data are also presented in this chapter.

Software Development Methodology

Software development methodology refers to defined processes involved when developing a software. It is a combination of design philosophies and pragmatic realism that stretches back to the early days of computing. The aim is to provide a systematic approach in developing software.

There are many methodologies that are available but not all methodology is suitable for all projects. Each methodologies have pros and cons so using the most appropriate methodology based on the situation will increase the success rate of developing the software and its efficacy.

The researchers used Rapid Application Development(RAD) which prioritizes rapid prototyping and quick feedback rather than long drawn out development and testing for this study that has limited time.

// INCLUDE DISCUSSION REGARDING RAD METHODOLOGY

Rapid Application Development (RAD) is a development model that favors rapid prototyping and feedback over lengthy development and testing cycles. Developers may use fast application development to make several iterations and upgrades to software without having to start a development schedule from scratch each time.

Rapid Application Development is a development model that arose after developers discovered that the traditional waterfall style of development was ineffective. A fundamental issue in the waterfall paradigm is that once the program is in the testing phase, changing the essential functions and features of the software becomes challenging. This effectively leaves you with software that may or may not meet your changing needs.

Rapid Application Development was invented in the 1980s, thus it is not a new concept. However, unlike the waterfall model, it is not unique. It is a constant growth of development ideas in response to the needs of the moment.

Initially, Barry Boehm, James Martin, and others recognized that software was not bound to traditional engineering processes. It wasn't a one-time resource that necessitated a rigid framework. It might be shaped to meet the demands of the user.

Originally, Rapid Application Development took the form of the Spiral model, in which one or more development models were utilized to work on a specific project.

With these procedures, it may appear that application development is a good choice for all tasks, but that's a bit of a stretch. Rapid Application Development software is ideal for small teams and short-term projects. However, it is not a panacea for all problems. Here are some of the advantages and disadvantages of employing rapid application development.

The advantages of Rapid Application Development are;

Requirements can be changed at any time. You do not need to create a complete and thorough vision for your software. You provide your software development team a basic overview of your ideas but are free to tweak them at any moment.

Encourages and prioritizes customer feedback. If you value your involvement in the development process, Rapid Application Development is an excellent option. Your feedback is a valuable source of information that can affect your software.

Reviews are quick. Because your software is constantly being developed, there is no need to devote too much time to in-depth evaluations, which are carried out on a frequent basis.

Development time is drastically reduced. Developers were unlikely to take a vacation after delivering the product in the traditional waterfall approach. After the initial delivery, clients would usually seek adjustments ranging from the interface to the functionality. With Rapid Application Development, projects are more likely to be completed on time and to the client's satisfaction. The time spent delivering is significantly decreased since the entire project is separated into modules and each module is considered as a distinct prototype.

Integration isn't a problem, since it integrates from project inception. All of the application's modules are integrated from the start. As a result, there are fewer complications in the final integration process. Because all modules are correctly synced from the start, the likelihood of bugs is greatly minimized.

More productivity with fewer people. Rapid Application Development favours a single team with a small number of people. This enables for quick communication as well as quick meetings for information transfer.

Lower Development Cost. In Rapid Application Development, developers create just the systems that the customer demands. In waterfall, IT runs the risk of developing and fleshing out complicated feature sets that the customer may decide to remove from the final product. The time spent developing removed features can never be regained, hence the budget invested on them is wasted. Rapid Application Development programming decreases this risk and, as a result, the cost. Rapid Application Development may necessitate greater investment in competent developers. However, by decreasing the development time, these expenses may be equalized. The key economic advantage of Rapid Application Development is that you never have to restart the project from the beginning if the client requests large modifications, resulting in reduced cost overrun.

Lower Maintenance Cost. Maintenance is often quick and straightforward when a software is released using Rapid Application Development. Traditional approaches need extensive planning, testing, and personnel.

Developer satisfaction. In the typical waterfall model, developers operate in silos, with little feedback or positive affirmation for a well-made product. When they eventually have the chance to exhibit their work to the client, the client may not give them the red carpet treatment. Regardless of how happy developers are with their work, if the client is dissatisfied, they will not obtain the recognition they so much need. In a fast development environment, the client is present at every stage of the route, and the developer has regular opportunities to demonstrate their work. This provides them confidence that when the final product is given, their efforts will be recognized.

The disadvantages of Rapid Application Development are;

Large scale projects. While Rapid Application Development is great for quick turnaround, the continually changing nature of requirements, which leads to design changes, might derail certain larger-scale projects. Because they have immediate access to one another, a close-knit team of engineers, designers, and product managers may quickly implement Rapid Application Development methods. When a project grows beyond a single team or necessitates inter-team collaboration, the development cycle usually slows and muddles the project's direction. Simply said, when your tale is continually shifting, it's tough to keep a huge group of people on the same page.

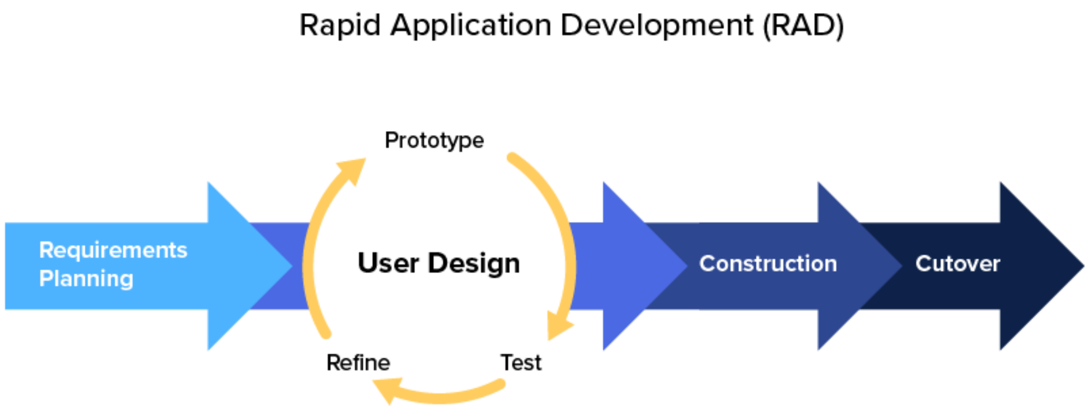
Requires feedback. After finalizing requirements, the client spent the majority of their time away from the development team using the waterfall method. This allowed clients to concentrate on their core activities while developers concentrated on development. In Rapid Application Development, the frequent cycle of prototypes necessitates developers and clients committing to frequent meetings, which may appear to occupy unneeded cycles at first. User feedback is essential for quick application development, but if those users are unavailable or unwilling to work on the project on a constant basis, the quality of the solution produced may suffer.

Requires modularity. One of the model's key limitations is that it can only be applied to software systems that can be modularized into small components. This usually indicates that the Rapid Application Development model is only applicable to modest to medium-sized projects. It has a tendency to have problems with larger, more sophisticated systems.

Requires a high skill level across the team. The capacity of designers and developers to deliver precise work at a rapid speed is critical to the success of a Rapid Application Development prototype. This means that Rapid Application Development may not be the appropriate model for teams without the necessary degree of experience.

Interface-Focus. The rapid application development process encourages developers to create the best possible solution for the customer. The client evaluates the quality of the solution based on what they can engage with—and frequently, all they contact with is a design. As a result, some developers ignore recommended practices on the backend in order to speed up development of the front-end-focused prototype. When it comes time to provide a viable product, they patch up the jerry-rigged server code in order to avoid a refactor.

// INCLUDE A FIGURE WHICH ILLUSTRATE THE PHASES OF RAD



// EXPLAIN THE ACTIVITIES THAT YOU DID IN EACH PHASE

Rapid application development is a common agile project management method in the software development industry.

The main advantage of a Rapid Application Development method is quick project turnaround, which makes it an appealing alternative for engineers working in a fast-paced setting such as software development. This quick turnaround is made feasible by Rapid Application Development's emphasis on eliminating the planning stage and emphasizing prototype development. Rapid Application Development enables project managers and stakeholders to precisely monitor progress and communicate in real time on emerging issues or changes by decreasing planning time and prioritizing prototype iterations. This leads to increased efficiency, quicker development, and more effective communication.

There are other ways to break down the process, but in general, Rapid Application Development follows four major phases.

<u>Requirements Planning.</u> This stage is analogous to a project scoping meeting. Although the planning phase is shorter than in other project management approaches, it is crucial to the project's overall success.

During this stage, developers, customers (software users), and team members communicate to define the project's goals and expectations, as well as present and potential problems that must be solved during the development. Rapid Application Development distinguishes itself from typical software development techniques from the start. It does not necessitate meeting with end customers to obtain a precise list of requirements; instead, it requests a general demand. The wide nature of the criteria allows you to provide particular requirements at various stages of the development cycle.

It is critical that everyone gets the chance to examine and remark on the project's goals and expectations. Teams may prevent miscommunications and costly change orders by obtaining approval from each important stakeholder and developer.

User Design. Once the project has been scoped out, it's time to go right into development, building out the user design through multiple prototype iterations.

This is the foundation of the Rapid Application Development methodology, and it is what distinguishes it from other project management approaches. During this phase, clients collaborate with developers to ensure that their demands are addressed at every stage of the design process. It's similar to customizable software development in that consumers may evaluate each prototype of the product at each step to make sure it satisfies their expectations. This is where the main development occurs. Instead of adhering to a specific list of criteria, developers produce prototypes with various features and functions as quickly as possible. These prototypes are subsequently exhibited to clients, who select what they like and dislike. Iteratively, all of the problems and kinks are sorted out. The developer creates a prototype, the client tests it, and then they meet to discuss what worked and what didn't.

This strategy allows developers to modify the model as they go until they achieve a satisfying design.

Both the software developers and the clients learn from the experience to ensure that nothing falls through the cracks.

Most of the time, these prototypes are rushed to completion in order to demonstrate specific features without adequate polish. This is typical, and the final product is only developed during the finalization stage, when both the customer and the developer agree on the final result.

Rapid Construction. Phase 3 turns the prototypes and beta systems from the design phase into the functioning model. Because most of the problems and adjustments were handled during the extensive iterative design process, developers can build the final functioning model faster than they could using a traditional project management method.

During this stage, the software development team of programmers, coders, testers, and developers collaborate to ensure that everything is running well and that the final output meets the client's expectations and objectives.

This third phase is critical since the customer may still provide feedback throughout the process. They can provide modifications, revisions, or even new ideas to tackle problems as they develop.

Feedback on what's excellent, what's not, what works and what doesn't is provided at this point. Feedback is given not only on functionality, but also on appearances and interfaces.

Construction will continue with this feedback in mind. These two processes are continued until a final product that meets the needs of both the developers and the client is created.

Cutover. This is the phase of implementation in which the final product is released. It comprises data conversion, testing, and system transition, as well as user training.

Here, the client and the software's features, functionalities, aesthetics, and interface are completed. Before providing to the customer, stability, usability, and maintainability must be prioritized. While the developers and customers continue to seek for flaws in the system, all final improvements are done.

It is important to guarantee that all aspects of RAD are adequate for the high-speed development process. The four most important features of Rapid Application Development are as follows:

Methodology. The Rapid Application Development methodology covers the following key elements that are utilized to assure timely delivery of a high-quality product:

Using the best available methodologies for the development process and determining the job sequence

Making use of prototypes

Rather than conducting interviews, requirements are gathered through workshops.

Choosing a set of CASE tools

Putting in place a time-bound development method

Defining risks and providing rules for a successful product

<u>People.</u> Rapid Application Development promotes fast, high quality, and cheap cost. To do this, not only high-quality tools but also highly competent individuals are necessary; everybody engaged in the RAD process should be highly skilled. People should be highly motivated and talented in order to reduce any delays or solve any difficulties that may arise throughout the development process. Each significant end-user should be accessible to participate in workshops during the requirements planning and user design stages. Furthermore, the Construction team, which is in charge of the CASE toolset (used for design and

code production), and the cutover team (used for training and cutover), should be able to move swiftly.

Management. Management must be highly motivated in order to drive both the IT team and the users. Management should exercise caution while choosing and managing the SWAT team, as well as providing training for the tools and procedures that will be employed in the development process. The success of the Rapid Application Development project is dependent on both people and tools; management should know this and keep each team member motivated on a regular basis in order to boost their productivity, which will eventually aid in producing the product faster and with higher quality.

Tools. The tools are one of the most crucial aspects of the RAD project's success. These tools are utilized throughout the building phase and can assist with design-automation approaches, code creation, and computer-aided planning and analysis. Rapid Application Development's power tools are Computer-Aided Systems Engineering (CASE) tools. CASE tools are used to automate the operations of the software development life cycle.

Rapid Application Development diagram shows how the research and process is being develop from the start of the process until it was been finished, Rapid Application Development is easy to use as a methodology model because as you can see from the diagram it was been processed step by step so that the problems that might encountered can be polished or be refined. In the Rapid Application Development model the functions are developed as the prototype is being integrated to make the complete process faster for product delivery, it makes it easier to incorporate the changes within the development process and can quickly give the customer something to see and use to provide feedback regarding the delivery and their requirements. The advantage of the Rapid Application Development model from the other methodologies is that it reduced the development time, it encouraged the customer to give feedback to the prototype system, and the integration that was done from the beginning solved a lot of integration issues.

Scope and Delimitation

The researcher focused on the Inventory Management with Point of Sale System of Pipay's Coffee Shop improving the process of managing inventory and recording of transactions. The study shall include recorded transactions at the counter, current number of stocks, monitor stocks that are low on stocks, record of expenses, and generate reports about the sales.

The study did not include hardware support of devices like receipt printer, barcode scanner, credit card reader and POS machine.

Data Gathering Techniques

The researchers used instruments and techniques necessary to complete the development of an Inventory Management with Point of Sale System for Pipay's Coffee Shop which includes observation, interview, survey and document analysis for determining the study's worth and significance. The data gathering was done in a time that is most convenient for both the respondents and the researcher, which led to attaining important information that was crucial in achieving the objectives of this study.

Observation. The researchers observed the process of managing inventory and recording transactions and studied how respondents have problems with the current method of managing inventory and recording transactions.

Interview. The researchers interviewed the staff and owner to learn more about the processes involved in managing inventory and recording transactions and identify the problems that need to be addressed. The interview was conducted to effectively gather all the needed information.

Survey Questionnaires. The researchers conducted a survey that was used for the assessment of the current method of managing inventory and recording transactions. The respondents were asked to assess the current method and give opinions about having an Inventory and Point of Sale System.

Document Analysis. The researchers gathered information about study from literature review, which gives the researchers some insights based on past studies that will be used in developing the Inventory Management with Point of Sale system.

Sources of Data

Primary sources of data were gathered through observation, interview and survey. The researcher observed Pipay's Coffee Shop's inventory management and transaction recording operations, which assisted the researcher in identifying the problems in the current method. The responses obtained through the interviews with Pipay's Coffee Shop's workers and owner. The information gathered during the interviews was used to create Pipay's Coffee Shop's proposed Inventory Management System with Point of Sale system. The data gathered through respondent's response in survey have been considered for the features of the Inventory Management System with Point of Sale system.

Secondary sources of data were gathered through document analysis. The data gathered through document analysis provides background on things needed to consider in developing Inventory Management System with Point of Sale system.