Fall | 16

Software Development Life Cycle

Yan Ting Wong Tiky

The Hong Kong University of Science and Technology

Table of Contents

1. Introduction

Software Development Life Cycle (in short *SDLC*) is a workflow process which defines the core stages and activities of development cycles. It can be used by system analysts, designers and developers to plan and implement the applications and deliver the systems or products on time and within budget. With numerous of development methodologies, it is never an easy task to choose an appropriate strategy that sometimes even it is inevitable to mix-and-match multiple methodologies to fit in a single project.

2. Background

Per *Elliott & Strachan & Radford (2004)*, The initial concepts of SDLC were originated in the 1960s to develop large scale functional business systems in an age of large scale business conglomerates. In the earliest days of computer programming, the only models that were used to develop complex things like that were in construction and manufacturing industries. Thus, it made a lot of sense that the structured approaches used in those industries should be applied on developing computer systems as well. For instance, in the construction fields, the business analysts would first understand the client's requirement. The steps follow by architects designing solutions and engineers to develop and build the buildings, bridges or roads. Coming to the last step, test, refine and sign the certificate for the products. Consequently, in the 1970s, a large groups of business analysts of construction and manufacturing industries had got into the field of computing to analyze the business requirement for the new systems. A significant number of engineers had also entered the field of computing as programmers.

It is a very traditional development process which goes in a sequential manner from start to finish. Some overlapping procedures are inevitable such as testing and refining. Nevertheless, crossing between major phases are not common. In the old days, the programming techniques were very complex and programming languages were not easy to learn and manipulate. Thus, computer system development followed the structured and sequential approaches made a lot of sense.

Over the past 50 years, computer systems have been taking important roles in the corporations. From sending mails with postmen to sending email via internet, from filling paper applications to electronic applications, from auditing financial logbooks to the spreadsheets stored in the enterprise systems, every aspect is closely related with information technology. Hence, numerous companies take it very serious and spend considerable money, resources and effort on information technology security and governance.

Within 5 decades, software development concepts evolved and new perceptions and designs have emerged on customer-oriented applications and solutions. Each approach has its pros and cons, strengths and weaknesses. It is a realistic fact that one single solution can no longer fit in millions of organizations due to different backgrounds, structures, responsibilities, desires and goals. Yet, shared aims on each software development stage can be found. There should not have much variations on how the works are described, organized and managed with different organization backgrounds and requirements. Therefore, the modern software development life cycles are adequately flexible to be used across different types of business, products and services.

Not limited to the listed models below, there are various models used in the software development life cycle process.

- Waterfall
- Iterative
- Agile
- Rapid Application Development (RAD)

Before going into details of the software development life cycle models, firstly should understand what software development life cycle is.

3. What is Software Development Life Cycle (SDLC)?

Software Development Life Cycle consists of details steps and activities which describes how to design, develop, maintain, replace, alter, enhance, test or even launch a software.

The activities can be broken down into a very detail level but at the same time they can be grouped into five (5) core categories: Plan, Design, Develop, Test and Deploy.

Below is a graphic representation which displays a typical Software Development Life Cycle.

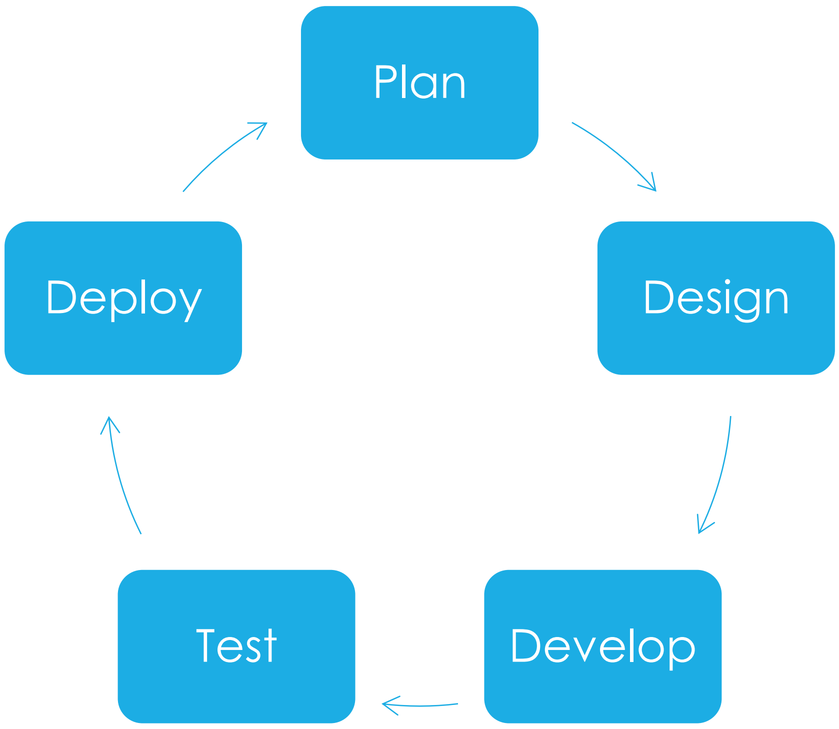

Figure 1 - Software Development Cycle

Stage 1: Plan

Planning usually happens after there is an innovation or initiation that come up from a group of business end-users or a sponsor whom identify a need or an opportunity. Within the planning stage, scope or boundary of concepts are defined. Product feasibility study in financial, operational and technical areas will be conducted by the senior members of the team with the input from the business users. Quality Assurance and Risk management plan are also prepared at planning stage to minimize any unpredictable risks. Business Case Documentation (BSD) should be ready at this stage to summarize all the ideas and have holistic view of the full plan.

Stage 2: Design

Product design is started with a clear definition of requirements. Software Requirement Specification (SRS) document which consists of all the details of the product requirements should be approved by the clients or the customers before product design begins.

With SRS in hands, more than one design of the product architecture will be proposed based on the requirements in SRS. They will be documented in a Design Document Specification (DDS) by the junior members of the team and passed to the senior members, project stakeholders for review. DDS will be evaluated based on various criteria but not limited to budget, time, user-friendliness, risk, integration, etc.

Stage 3: Develop

After the best or the most appropriate design has been selected, implementation starts immediately. Programmers should develop the software according to the DDS and at the same time follow the coding standards defined by the company's closely. Programming tools should be limited to those provided by the company as well to ensure all programmers can align their works. Functional Specification (FS) should be written by programmers to record all the functions that are provided at a technical level.

Stage 4: Test

Software testing should be conducted at all stages as a sub-stage. Nonetheless, two (2) major ones should be done by programmers, end-users and quality assurance experts. The reason is that programmers know the best of how the program works and therefore they can identify the most vulnerable areas of the software. End-users would pay more attentions to their routine tasks which can help to ensure the software can fulfill the requirements. Last but not least, quality control experts examine the software as whole from various perspectives such as architecture, security, integration with other systems, etc. As a result, a few different types of test plans should be prepared for the three groups of testers to conduct at the test stage.

Stage 5: Deploy

First thing to do at deployment stage is to verify all the test cases were run to ensure successful software execution, comprehensiveness and correctness.

Final decision should be made if the software should be deployed to the production environment and therefore approval should be seeking from management in this stage. Deployment Plan (DP) should be well-defined and approved to carry out any changes that is going to make. Guideline documentations should also be prepared such as Installation guide, administration guide and user guide. Support team members should be ready to answer all sort of questions regarding to the software. Finally, Contingency Plans (CP) should be created according to the finalized software. For newly implemented software, a common solution is postposing the software launch day to redo and retest. For a running software, very likely rolling back to the previous version or postposing the launch day to fix the defects would be the choices.

4. Why do we need Software Development Life Cycle?

There is always a huge temptation to implement a software without planning or designing especially for a small or medium size project. Programmers tend to argue that the time that is spent on planning is already good enough for them to do the programming work and deliver the product. Managements also tend to focus on efficiency and making use of the least amount of resources to get the "same" result.

However, there are certain reasons to explain why we need Software Development Life Cycle.

Reason 1: Quality Assurance and Quality Control

The definition of Quality Assurance is a set of activities for ensuring quality in the process of the product development. Meanwhile, the definition of Quality Control is a set of activities for ensuring quality of the developed product. While QA aims on preventing defects by focusing the process of the product development, QC aims of identifying the defects by examining the finished product. The goal of QA is to eliminate as much defect as it can to improve the QC processes. The goal of QC is to identify any defect that is missed in QS processes. Thus, with QA as a proactive quality process and QC as reactive quality process, these two procedures help to ensure the product that is delivered is up to high standard without coming with some unreasonable issues.

Reason 2: Easier implementation control

Within the five (5) core stages in SDLC, multiple documentations should be prepared to give guidelines and instructions for the programmers and testers to follow and for the managements and approvers to be acknowledged and approve on the activities and action taken. Business Case, Software Requirement Specification (SRS), Design Document Specification (DDS), Functional Specification, Test Plan, Deployment Plan, etc. are all well-defined at each stage. With all the documentations, not much surprises or free-style works can be found in the unexpected areas which implies requirements can be fulfilled and project schedule can be met as planned.

Reason 3: Fulfill user requirements or even exceeding their expectations

As mentioned earlier, Quality Assurance (QA) and Quality Control (QC) help to ensure the product delivering as user required with zero to limited number of defects in good quality. Nevertheless, very high chances that users would like to further enhance the delivered product due to business change and technical upgrade is necessary due to technology improvements. Hence, in the design stage, designers not only give resolutions for the requirements but also take considerations of integration with peer systems, flexibility and availability of enhancements, maximum system load due to increasing of users, etc. These “hidden” requirements are usually not stated in the user requirements but they are expected to be well-thought-out in every product.

5. Roles

To effectively implement a software with the above mentioned five core stages, three (3) key members should always involve in the projects whom take up more than 95% of the activities in SDLC. The take up different roles in each stage to cross check and monitor each other's work to ensure each decision made in the SDLC is valid and necessary.

Project Manager

- Define project scope and goals
- Budget control
- Resource allocation
- Business documentations
- Coordinate high-level management aspects of project
- Rollout approval

Business Analyst / System Analyst

- Interact with end-user during implementation
- Business & System Documentations
- Evaluate business requirements
- Design system architecture, business flow and user interfaces
- Ensure business needs are properly analyzed and correctly implemented in the solution
- Facilitate relationship between business and technical roles
- Quality Assurance and Control

Programmer / Solution Developer

- Interpret business requirements and translate them into a deployable solution
- Technical study
- Resolve Product defects
- Prepare functional specifications
- Perform testing in accordance with agreed strategy

6. Software Development Life Cycle Models

There are various software development life cycle models defined which are designed for different types of project. Each model follows a series of unique steps that best fit to its project type to ensure success process of software development. Waterfall, Iterative, Agile & Scum and Rapid Application Development (RAD) are identified as the most popular models being used in the industry and they will be introduced one by one in details as follows:

Model 1: Waterfall

Waterfall model is the earliest, best-known and most commonly used methodology. It is a sequential life cycle that is simple to understand and use. Each phase has to be completely finished before another start which means no overlapping is allowed. The output of each phase serves as the input for the next stage.

A pictorial illustration of Waterfall Model can be found below:

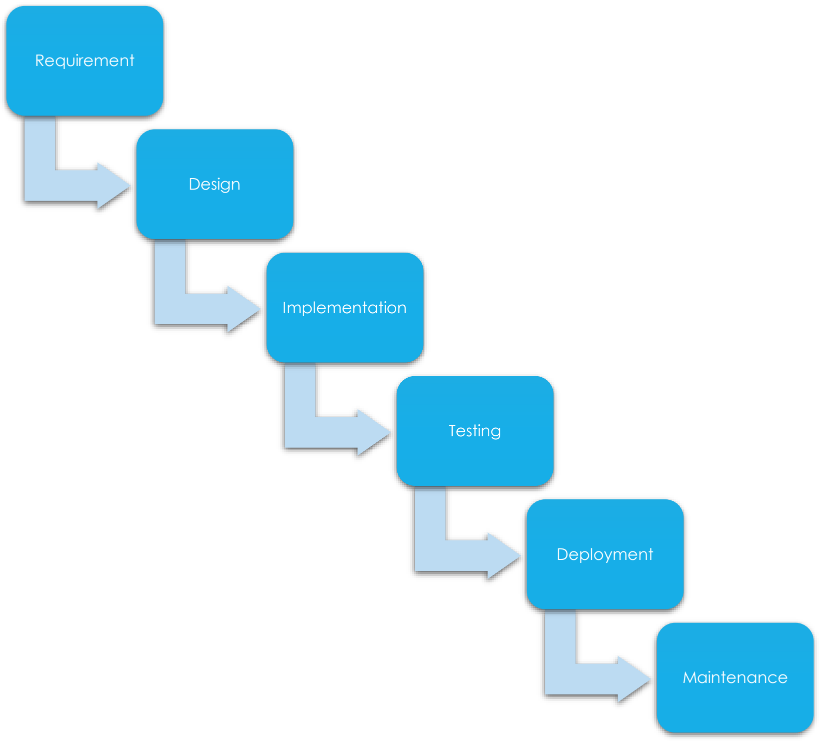

Figure 2 - Waterfall Approach

6 Phases of Waterfall Model

1. **Requirement**
Requirement Phase mainly focuses on communicating with business users to gather and analyze requirements. Project managers try their best to understand and analyze the business, capture all the details of user's needs, define the scope and arrange resources in the Business Case Documentation.
2. **Design**
With the Business Case Documentation in hand prepared in Requirement Phase, Business Analysts evaluate and start on the logical design the software by making use of the information and requirements that are collected by the Project Managers. Based on the high-level design which has fulfilled all the user requirements, System Analysts transform the high-level design to the physical design which put hardware and software technology into consideration. System architecture is defined at Design Phase as well.
3. **Implementation**
Implementation Phase is where the actual code is written. Programmers develop the software according to the instructions recorded on the documents prepared in Requirement and Design phases. Their output is the Functional Specification which files all the details of the functions that are implemented.
4. **Testing**
With the inputs from the Implementation Phase, testers in Testing Phase will draft the Test Plans based of the Functional Specification. Programmers prepare the Test Plan in a check list to examine if every function are executable as expected. Business Analysts prepare the Test Plan for the users which focuses on meeting the user requirements. Finally, Quality Control (QC) experts gather all the documentations from all previous phases and do an overall test on every aspect on a deeper level that are documented including system architecture, technology used, etc.

5. Deployment

After receiving a "PASS" from the Testing Phase, the product is said to be ready to release. Software or Application will either be deployed to production servers or release for users to install on their own machine.

6. Maintenance

In reality, it is inevitable there are some defects or issues come up. In addition, the world keeps changing every day and as a result enhancements are necessary from time to time. Maintenance Phase is for catering such situation and deliver changes to the users again. Within Maintenance Phase, a subset of SDLC Waterfall Model is involved.

Inputs and Outputs of each phase are summarized and illustrated as below:

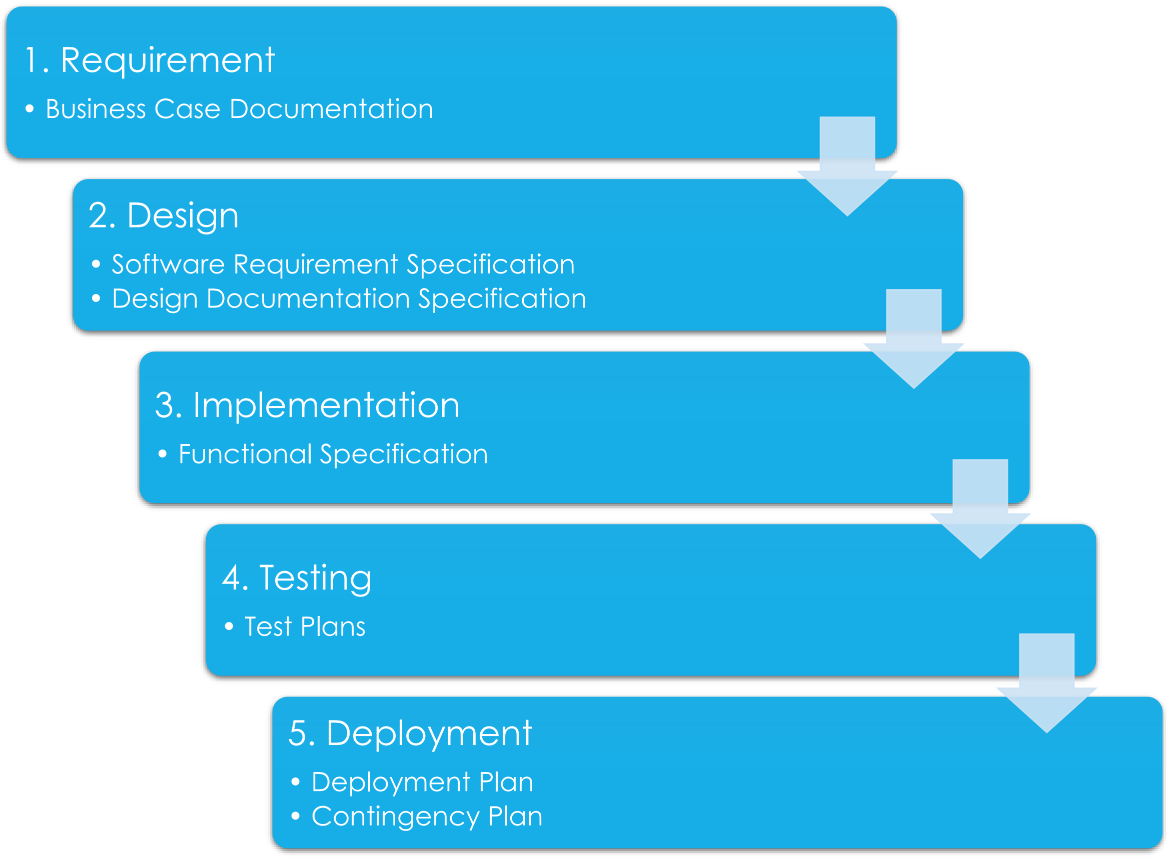
1. Requirement
• Business Case Documentation
2. Design
• Software Requirement Specification
• Design Documentation Specification
3. Implementation
• Functional Specification
4. Testing
• Test Plans
5. Deployment
• Deployment Plan
• Contingency Plan

Figure 3 – Documentations for Waterfall Approach

Applications of model

Appropriate situations for using Waterfall Model in the SDLC:

- Limited amount of ambiguous or unconfirmed requirements
- A software that needs well-documented documentations
- Use of mature technologies and not dynamic
- Management can provide enough resources and experts to pick up the role at each phase
- Simple and small project

Advantages and Disadvantages

Advantages

- Easier to manage as there is clear schedule for each stage that gives clear milestones
- Easier to control with limited external factors as no overlapping development phase
- Provide extensive documentations
- More disciplined and provide distinct actions for the project to move forward as the development move from concept, design, implementation, testing, troubleshooting, running and then maintenance.

Disadvantages

- Cannot have scope change or requirement change
- Cannot preview the product until the deployment phase
- Not flexible to handle unexpected risks
- Limited communication with users as bounded to be done at the beginning and the end of the project
- More resources are needed and some of the teammates might be idle for a long duration
- Poor model for long or ongoing projects as the projects will probably never come to the end and reach the last phase

Model 2: Iterative

Iterative Model works on the simplified requirements which are the subset of the software or application requirements. The "product" is iteratively enhanced and evolved to the final product for deployment. It is called as a "build" for each iteration. So at each build, design amendments and new functionalities are added to the product. With iterative model, the software is implemented by small portions at a time.

A pictorial illustration of Iterative Model is demonstrated below:

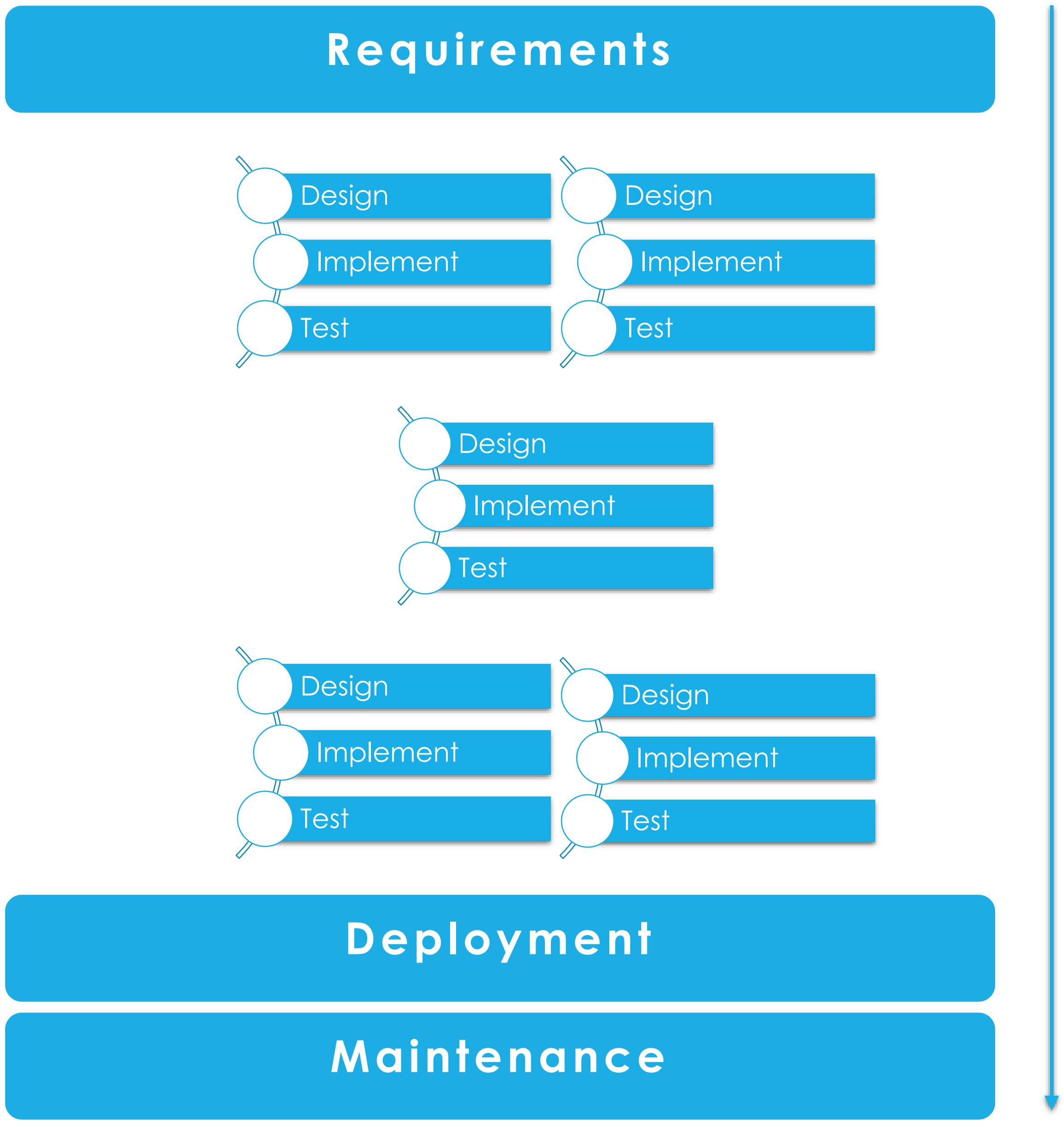

Figure 4 - Iterative Approach

6 Phases of Iterative Model

1. **Requirement**
   Same as Waterfall Model, Requirement Phase focuses on communicating with business users and prepare the Business Case Documentation.
2. **Design**
   Similar to Water Model, Business Analysts and System Analysts work on the logical and physical designs respectively to prepare the Software Requirement Specification and Design Specification Document. However, there is design which holistically recorded how the software is going to be implemented and there are several subset of designs for programmers to go through the implementation and testing which is isolated from other subset of designs. In addition, the subset of designs can be modified after every round of build. Therefore, the subset of designs is not finalized until reaching the Deployment Phase.
3. **Implementation**
   Programmers develop the software according to the subset of design passed from Design Phase. Functional Specification will be prepared for each subset of implementation.
4. **Testing**
   Programmers, business users and QC experts will all be involved for each subset of testing. However, business users will only focus on the limited scope that is covered in the currently build but programmer and QC experts have to cover all the implemented functions every time. In addition, for the last build before going to the Deployment Phase, the three parties not only have to do the subset of testing, they have to conduct the testing as a full system test as well.
5. **Deployment**

   With no difference from Waterfall Model, everything should be ready by this phase and a Deployment Plan for release.

6. Maintenance

Again, like Waterfall Model, it is inevitable that every software needs to be maintenance. Therefore, a subset of SDLC Iterative Model is going to take part in Maintenance Phase.

Inputs and Outputs of each phase are summarized and illustrated as below:

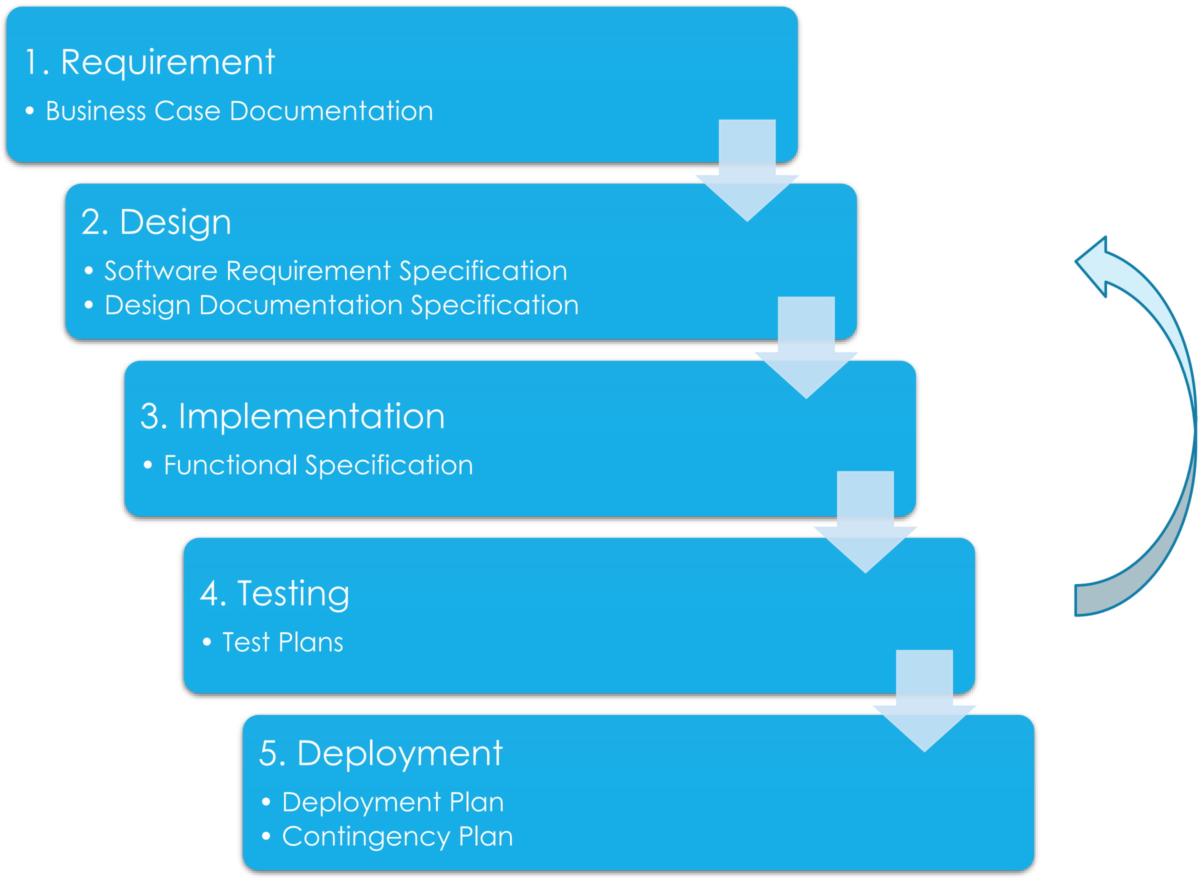
1. Requirement
• Business Case Documentation
2. Design
• Software Requirement Specification
• Design Documentation Specification
3. Implementation
• Functional Specification
4. Testing
• Test Plans
5. Deployment
• Deployment Plan
• Contingency Plan

Figure 5 - Documentations for Iterative Approach

After every subset of testing, it loops back to the Design Phase and starts on the next design until it comes to the very last one.

Applications of model

Appropriate situations for using Iterative Model in the SDLC:

- Major requirements are defined but the minor details might evolve when time goes
- New technologies are being used and there is a learning curve for the programmers to learn
- Resources are limited to do a huge project as if a small project or teammates are in contract rather than permeant
- Very high risk as the goal of the project might change from time to time

Advantages and Disadvantages

Advantages

- Easier to start on a complex project
- Preview the project periodically
- Parallel implementation is allowed
- Project can still be managed like waterfall Model with clear schedule and milestones
- Easier testing and troubleshooting at each build
- Support and less costly for scope or requirement change
- Suitable for huge and core projects
- Better communication with business users as feedbacks can be gather at each build

Disadvantages

- High risk due to system architecture and designs keep changing
- Issues might occur for integration of each build
- More management work to do to ensure each build can meet the standard
- Overlapping implementation could be chaotic
- Need more involvement of business users
- Easier but more time is needed for each testing as each testing have to cover all the previous works

Model 3: Agile

Agile Model extends the advantages of Iterative Model and aims on user satisfaction and product adaptability by rapid delivering of product. From Requirement phase to Deployment Phase, Agile Model breaks the product into small builds. Instead of going back to Design Phase like Iterative Model after each subset of testing, Agile Model goes to Deployment Phase and release the product. Thus, each build contains of some new features and for the very last build it contains all the required features of the software.

A pictorial illustration of Agile Model is demonstrated below:

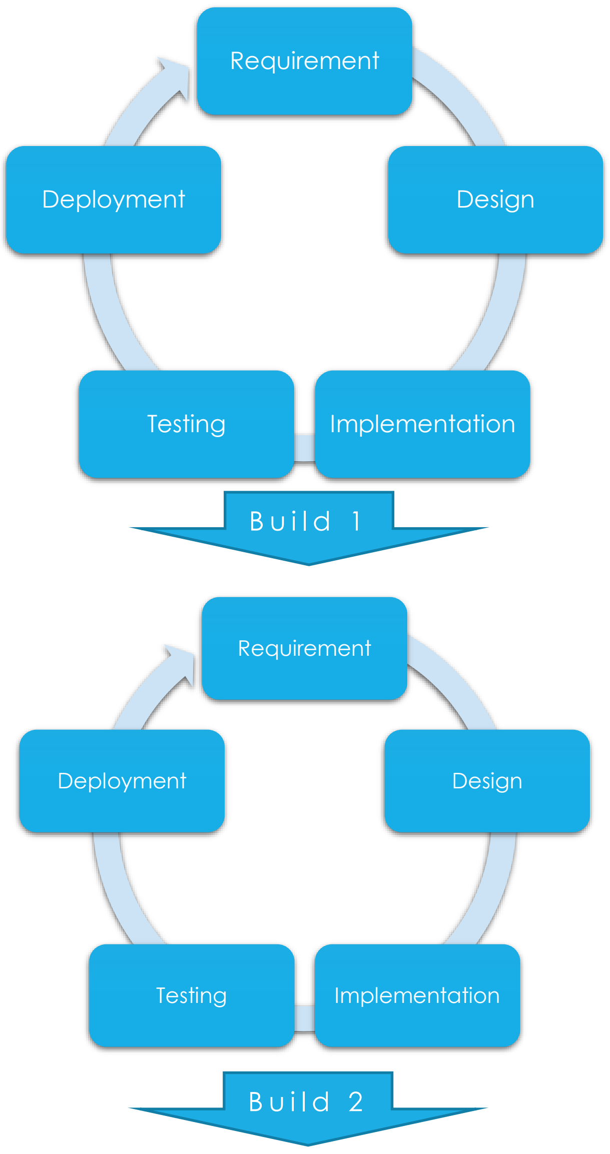

Figure 6 - Agile Approach

5 Phases of Agile Model

1. **Requirement**
   As requirements cannot be gathered completely at the beginning, close relation with business users is necessary to gather feedbacks after every release. However, a Business Case Documentation is still needed at the startup of the project to briefly describe the scope and goal of the project. Resources might have to evaluate and rearrange at each build.
2. **Design**
   Very limited amount of time will be put on designing the software as a whole due to the uncertainty. Designers mainly focus on the build that is working on but the goal of all the builds will still follow the scope that is defined in the Business Case Documentation. Software Requirement Specification and Design Specification Documentation are expected to be short and simple listing out what is covered in the current build.
3. **Implementation**
   Programmers tend to have more "freedom" in Agile Model implementation due to the brief documentations provided. However, they are still required to follow strictly on the coding standard. Functional Specification usually covers the core functions and skipping the details.
4. **Testing**
   A very high responsibility falls on the testers due to limited information found in the documentations especially for the QC experts. Business users tend to test on a very high level or not even includes business users in the Testing Phase.
5. **Deployment**
   Usually product release to users two – three (2-3) weeks after the requirements have been placed. Deployment Plan tends to focus on how to deliver the product but with limited information of the contingency plan because another build will be coming up tightly and the issue can be fixed there.

6. **Maintenance (Not necessary)**
   Not necessary for Agile Model as the next build is coming up and can be done in the next build.

Inputs and Outputs of each phase are summarized and illustrated as below:

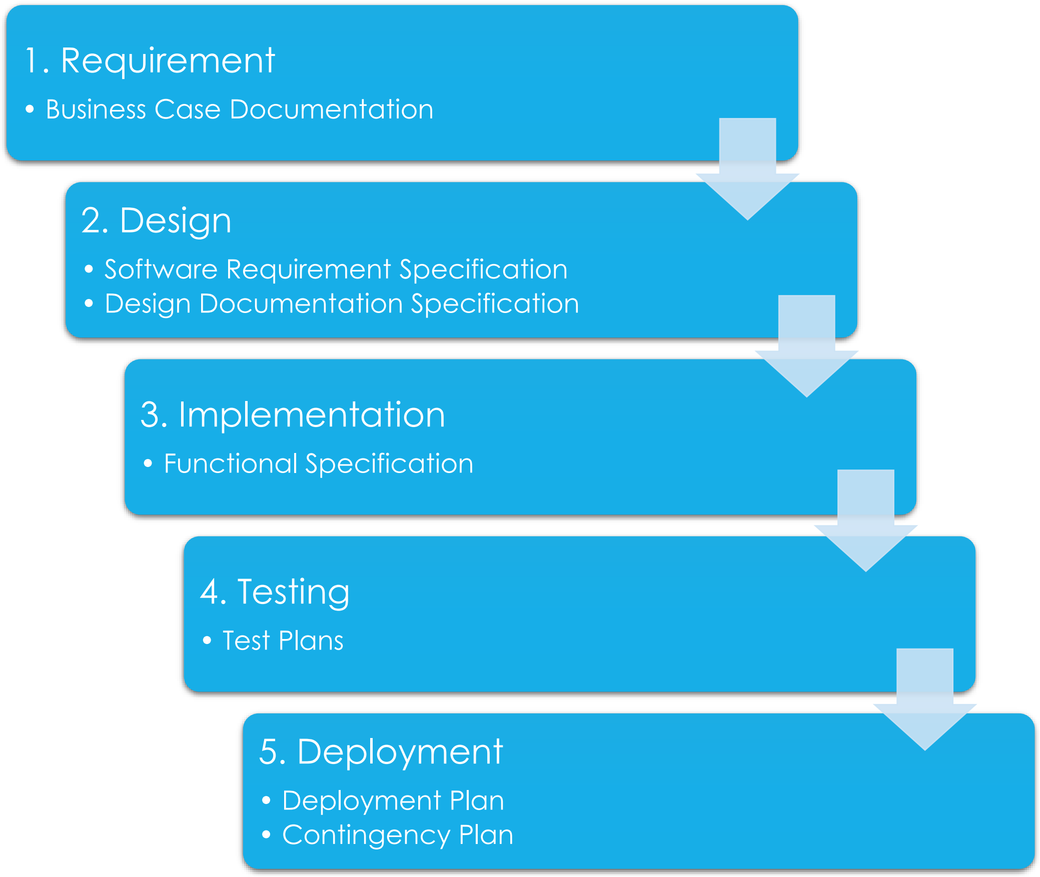
1. Requirement
• Business Case Documentation
2. Design
• Software Requirement Specification
• Design Documentation Specification
3. Implementation
• Functional Specification
4. Testing
• Test Plans
5. Deployment
• Deployment Plan
• Contingency Plan

Figure 7 - Documentations for Agile Approach

Agile Model doesn't pay too much attention on the documentation as Waterfall and Iterative Models. Although the same set of documentations are expected to be ready at each phase, the information that can be found in each documentation is very limited.

Applications of model

- No detail information is provided from business users
- Features driven project
- Product requirements change dynamically
- Have resources on testing
- Close collaboration within the team
- Close relationship with business users

Advantages and Disadvantages

Advantages

- Realistic approach – what you need the most, implement first; what is less important, implement last
- Functionalities can be developed and delivered promptly
- Less resources are required due to the by build approach
- Deliver the project periodically with new functions
- More freedom and flexibility at each phase especially for programmers
- Less documentations and rules to follow

Disadvantages

- Very high risks for maintenance and extendibility
- Not suitability for complex and core projects
- Project Managers have to follow closely at all time to check if the builds still following the scope that is defined
- Depends heavily on business users' feedback which could delay the projects and deliver wrong product if business users are not sure what they actually want
- Too much individual dependency as not much documentation to follow
- Transfer of knowledge to new joiners could be hard due to lack of documentations

Model 4: Rapid Application Development (RAD)

Rapid Application Development (RAD) focuses on gathering user requirements through workshops, test on the pre-released prototypes conducted by users and then reuse the prototypes to further develop the product. A prototype is a working model that is functionally equivalent to part of the releasing product. Minimal or no specific planning is involved at all which can make the team to cope with the changes in the development process and favors faster product delivery. RAD Model is basically assembling the working parts together to generate the product in limited of time for business users to quickly provide feedback regarding the requirements.

A pictorial illustration of RAD Model can be found below:

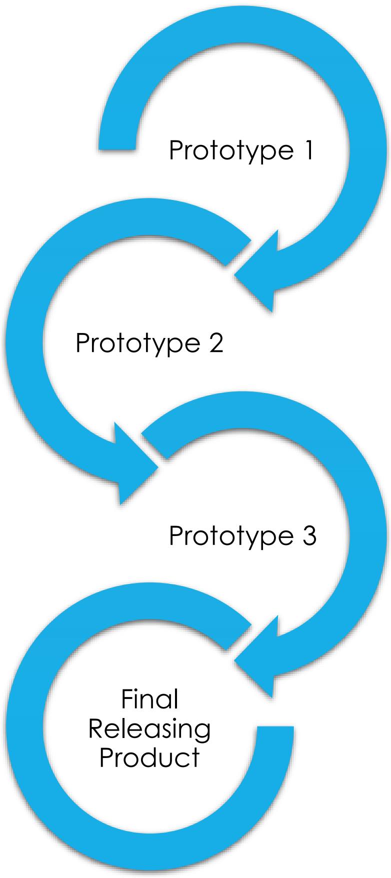

Figure 8 - Rapid Application Development Approach

5 Phases of RAD Model

1. **Business Modeling**
   The information flow and the information distribution are identified between different business channels. A Business Analysis Report is prepared to find out the essential information for the business such as how it can be acquired, how it can be processed and what are the elements driving the information flow and distribution.
2. **Data Modeling**
   With the inputs of Business Modeling Phase, all the necessary information should have been identified. At Data Modeling Phase, the identified information is transformed to certain data sets or data objects which will be further evaluated and defined their relationships in relevance to the Business Model.
3. **Process Modeling**
   The defined data set passed from Data Modeling Phase will be further processed by adding business information flow to achieve the business objectives that are identified at Business Modeling Phase. Any changes or enhancements on the data sets will be done at Process Modeling Phase. Operation of create, retrieve, update or delete (CRUD) of a data object should be defined at this phase as well.
4. **Application Generation**
   Automated tools will be used to convert all the Process Models into program code and pull them together as a prototype.
5. **Testing & Turnover**
   The newly generated prototype will be independently tested at this phase without taking consideration of the functions that are implemented in other prototypes. However, the integration between prototypes should be tested thoroughly.

A pictorial illustration of the inputs and outputs for each prototype is shown:

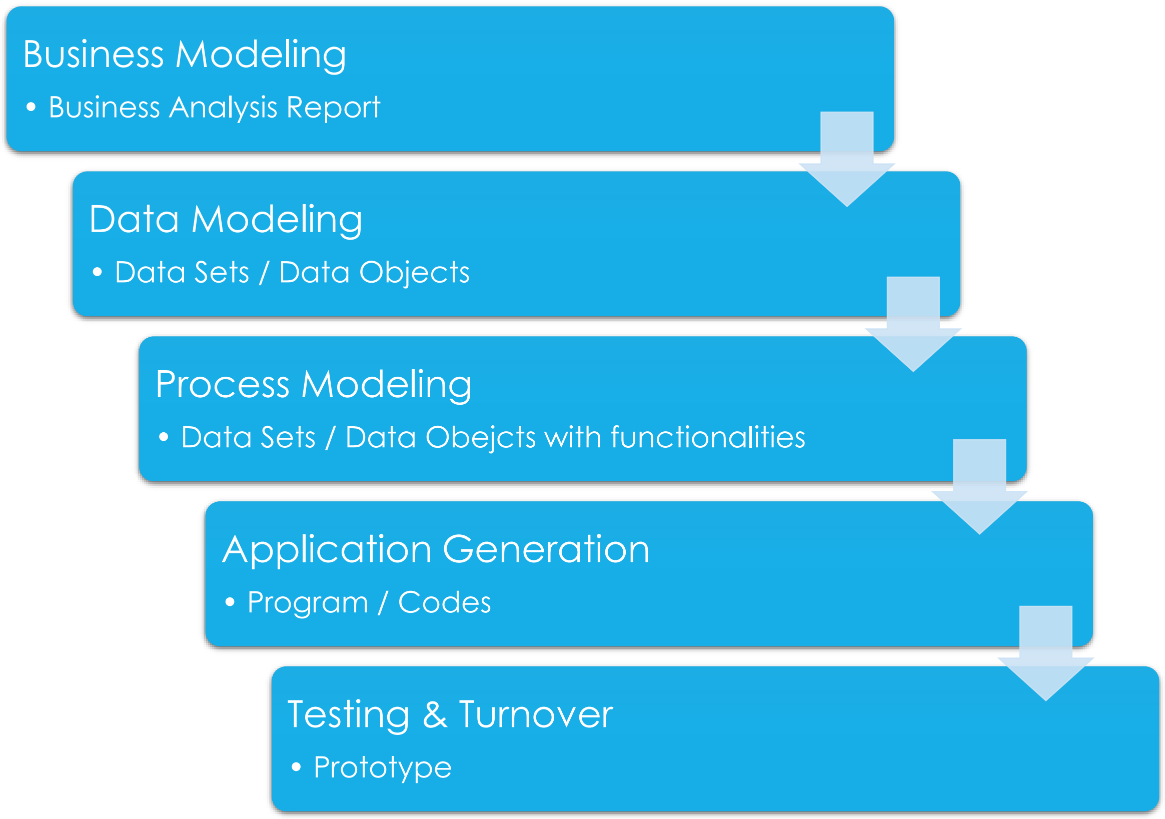

Figure 9 - Documentations for Rapid Application Development Approach

Application of model

- Business users tend to change their requirements from time to time due to the dynamic business environment
- Software that is feasible to be modularized
- Software that is acceptable to be delivered part by part
- Designers who have the business knowledge and know the relationship between prototypes
- For companies that have enough budget to own automated code generator

Advantages and Disadvantages

Advantages

- Can tolerate frequent requirement change
- Measurable progress
- High integration as it is designed to integrate with other prototypes at all time
- More communication with business users which can be done after each prototype release
- Preview and Review the product periodically
- Less programmer dependent with automated code generator
- Shorter SDLC RAD time

Disadvantages

- Depending too much on the automated code generating tools
- Could have performance or technically issues with automation
- Costly automation tools
- Required skillful designers who have the business knowledge and the technical skills
- Limited to modularization available software
- Too much dependency on business users' feedback and they have to be involved at the whole SDLC RAD process
- Requirements could be too dynamic
- Not much documentations
- Highly depends on the Business Analysis Report

7. Comparison Studies

A total of two (2) cases which demonstrate the characteristics of the four (4) different models that have introduced above.

Case 1: Quantity Surveying Application (Waterfall) vs Mobile Application (Agile)

In this case study, two applications will be focused on – Quantity Surveying Application and Mobile Approval Application. Quantity Surveying Application will demonstrate with the waterfall approach while Mobile Approval Application will make use of Agile approach.

Background of Quantity Surveying Application (QS system)

Quantity Surveyors are mainly responsible for preparing tender and contract documents, undertaking cost analysis, performing risk, value management and cost control. Thus, a helpful QS system should at least provide full features of tendering and cost control to assist the Quantity Surveyors on their daily tasks. In addition, the features that are provided have to be precise. Otherwise, even the system is equipped with lots of features are treated as useless.

Background of Mobile Approval Application (Approval app)

Approvers are authorities with power to approve or reject on a task. It could be an application for Human Resource such as Annual Leave Approval App. It could also be an application for project team to digitally signing the documents. Thus, a mobile app which can offer functionalities for approving and rejecting tasks have already served the purpose. Topping up the mobile app with security, user-friendly user interface, instant notifications will be some pluses.

Project Variables

Before going into details, first looking at the project variables of the two applications. The diagram shown below gives us a brief understanding of what is negotiable for the two applications. Those triangles in blue are the fixed deliverables while the grey areas are the negotiable variables.

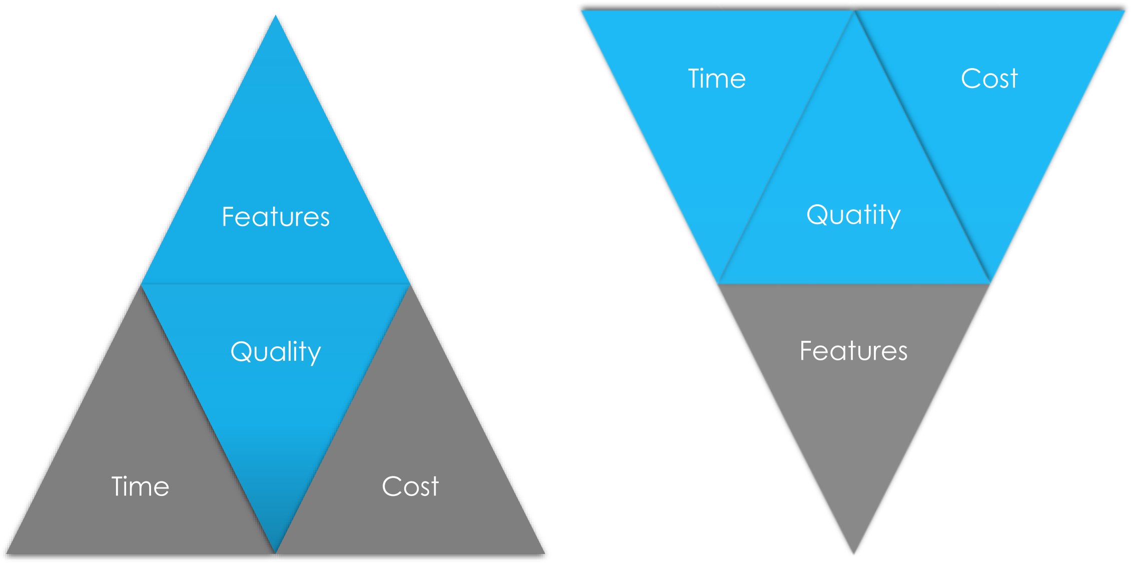

Figure 10 - Project Variables for Waterfall Approach (left) & Agile Approach (right)

Waterfall Approach

Features and *Quality* are some fixed deliverables for the waterfall approach. In QS System, the full set of tendering functions should be provided as a workable system. Even only 5% of the functions are not implemented, the other 95% of the implemented functions are treated as useless work. Consequently, *Time* and *Cost* have a much big chance to be varied in order to fulfill the requirements of *Features* and *Quality*. It would be more appropriate to follow the six (6) phases of waterfall approach to ensure all features and quality to be delivered because the team cannot move forward to the next phase until the current phase is done. For example, the programmers cannot swap or skip to the next phase until they have finished all the implementations. Likewise, testers cannot proceed to deployment unless all the tests are passed. With waterfall approach, it can help ensuring QS System to provide full features at launch. Nevertheless, the application might not be able to be delivered at the schedule time or within budget if any of the phases postponed.

Agile Approach

In reverse, *Time* and *Cost* are the fixed deliverables for agile approach. The team is given a fixed time to implement the application with quality. The tradeoff will be the only left variable – *Features*. The most important feature of the Approval App is the approval function – be able to approve or reject the tasks. Other features are necessary but can be missed out for the application to work. Hence, in order to deliver on time with limited cost together with certain quality, giving up the number of features is the only choice.

Can we swap the approaches for the two applications?

Assume we swap the approaches for QS System to use Agile approach and Approval App to use Waterfall approach. To recap, Agile Approach targets to deliver the application in small builds and each build implements within a 2 – 4 weeks timebox. However, it is quite impossible to deliver a huge application like QS System by small builds as the subcontract tendering should be a completed workflow to be treated as useable. Even though programmers can rush for the coding, there is not enough time for the testers to fully examining the application which could lead to money loss. Besides, lacking detail documentations of the application implementation might not be able to fulfill the auditory standards and requirements especially for a monetary related application.

How about Approval App using Waterfall approach? To recap, Waterfall Approach requires to implement the application step-by-step without any parallel work or partial work. That means, “must-do”, “should-do” and “could-do” tasks are implemented all together before the delivery. Not only it would prolong the application rollout time but increase the risk for a dynamic application. By the time of delivery, some of the features might no longer necessary or the design might not be the best fit due to the user requirements have changed after previewing the product. Thus, the time that are used for planning, implementing and testing could end up to be wasted and the “high-quality” features are treated as useless.

Case 2: Website Development (RAD) vs Enterprise Financial Application (Iterative)

This case study will drill into the SDLC of another two applications – Ecommerce Website and Enterprise Financial Application. RAD approach will be chosen to implement the Ecommerce Website and Enterprise Financial Application will make use of Iterative approach.

Background of Ecommerce Website

The ecommerce website provides a channel for the buyers to get the basic product information such available colors, sizes, price, etc. Besides, it is an excellent portal for the sellers to put as much product details as they can and to advertise to the buyers who are interested to their product. On the other hand, buyers can spend as much time as they can to go through all the specifications of the product and they can make up their decision to get the product anytime 24/7.

Background of Enterprise Financial Application

An Enterprise Financial Application is a combination of Account Payable, Account Receivable, General Ledger, etc. which helps finance department to record all the monetary transactions of a company and lower the risks associated with human errors. With all the transactions, not only it assists the management team to analyze the productive gains but also monitors the performance of the company. Hence, the fundamental principles of an enterprise financial application have to be precise and reliable.

Project Variables

Let's check out the project variables of the two applications before going further. The diagram shown below gives us a brief idea of what is negotiable for the two applications. Those triangles in blue are the fixed deliverables while the grey areas are the negotiable variables.

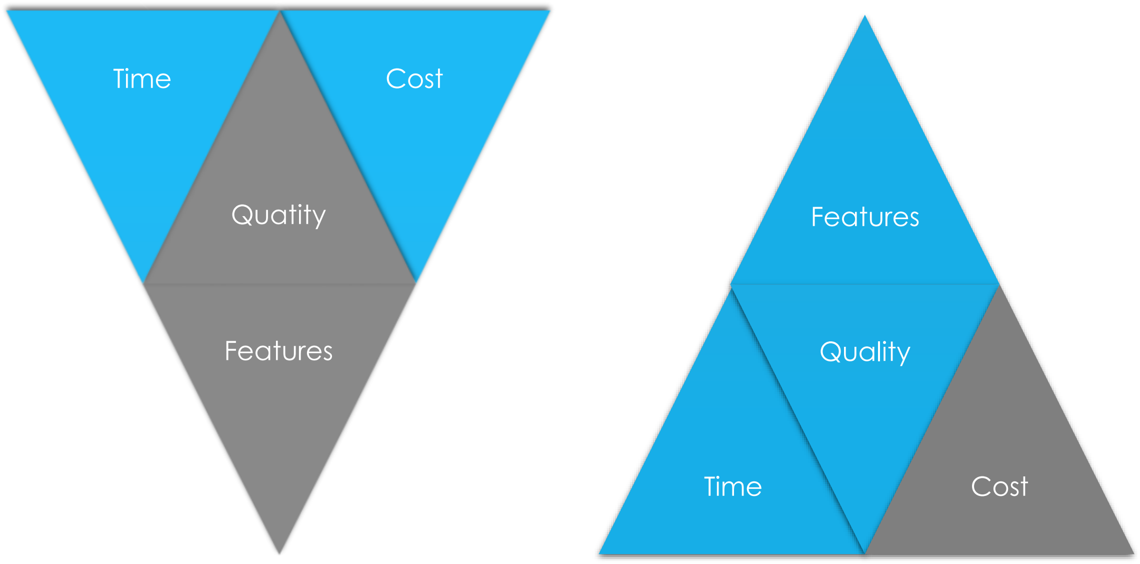

Figure 11 - Project Variables for RAD Approach (left) & Iterative Approach (right)

Rapid Application Development (RAD)

Time and *Cost* are the fixed deliverables in RAD approach. It best fits for the applications or software that have to be released while the need for it is hot and be there when a market is for it. Users can tolerate if there are not many choices in the market and they are willing to wait for the fixings and enhancements of the application. As a result, *Quality* and *Features* are negotiable.

Hence, the first prototype of the ecommerce website can simply just provide a platform that can present all the product information and the second prototype can offer a channel for the buyers to make order. The design of the user interface of website doesn't have to look extraordinary beautiful in the start-up prototypes. After being tested by the pilot buyers for a certain period of time and gathered most of the functionalities and enhancements from multiple prototypes, user interface and user friendliness can be enhanced in the last phase and deliver as a stable and mature application when there are more competitors in the market.

Iterative Approach

To implement a huge application like that, it is very important to have detailed requirement specification to state clearly what features have to be included before implementation starts. Furthermore, enterprise financial application has to be exanimated completely before delivering to ensure the preciseness and reliability. Thus, *Feature* and *Quality* are the fixed deliverables which has no flexibility at all. To fulfill *Feature* and *Quality*, logically *Time* and *Cost* should be adjusted accordingly to satisfy such requirement. The six (6) phases of Iterative approach should be followed closely to safeguard the correctness of the implementation. Since the application can be modularized (i.e. account payable is a module and account receivable is another module), it means that more than one team can go through the six phases for each module and seal the module while another team is working on some other modules. Not only it can help simplify the implementation complexity of a huge application, parallel implementation can be done in different teams to fulfill the fixed *Time* deliverable. It can also give a clear milestone for the team to go back and forth when debugging the application or across the modules so that no need to drill down to the ground whenever there is an issue found.

Can we swap the approaches for the two applications?

Let's say, the enterprise financial application is being developed in a short period without much testing and release the prototype for use. First, it is not feasible at all for an enterprise financial application to adopt RAD approach simply because it is not acceptable for anyone to give up *Quality* by risking the monetary transactions without testing the application out thoroughly. Furthermore, it is quite impossible for the development team to implement such a huge application or even just one of the module within the limited *Time*. In addition, *Features* are not sacrificial that an enterprise finance application is not useable if there is only account payable but no account receivable. Likewise, lacking documentation for a huge monetary application cannot fulfill the auditory standards as well.

What about adopting Iterative approach in e-commerce website? *Features* should be set at the early stage which might still be an unknown to the team as there is not much information in the market. All the implemented functions are fully tested and in high *Quality* which is always considered as a good gesture if the resource is sufficient. However, very likely the implemented functions are considered as not necessary after being put into the market which would have wasted much of the time and resources on something that is uncertain. From the shareholders' point of view, uncertainty tends to be risky and non-profitable which leads to low budget application implementation. Situation can only get better when the market share in the industry gets higher. Shareholders can evaluate the performance and revise or adjust the *Cost* from time to time. Unfortunately, iterative approach can only provide flexibility to adjust Cost but do not have the luxury to negotiate for *Features*, *Quality* or *Time*. Thus, for e-commerce website implementation, the project can still come to the end as a usable product. However, it cannot help the business to generate the most value with the lowest *Cost* when the project is achievable to do so. In short, using iterative approach in e-commerce website implementation is wasting the business resources and cannot maximize the value that the project can generate.

Can we swap Iterative with Waterfall, RAD with Agile?

If it is not workable for QS System to go with Agile approach, what about a similar approach like Iterative? Like Waterfall approach, Features and Quality are fixed deliverables in Iterative approach. They can help to ensure all the necessary parts to be included in the application and are in good shape without many buggy problems. Iterative approach also spends reasonable amount of time on documenting the application which helps to fulfill the auditory standards for monetary systems. The only difference is that Waterfall approach does not modularize the project from design stage to testing stage but limited to the implementation stage. Nevertheless, it wouldn't be a complicated task to expand the modularization further and there shouldn't be any critical side effects caused by the modularization. In reverse, it would even benefit the project by spotting out the issues at the early stage and prevent bringing the problems along with the project to run into another iteration. Additionally, multiple teams can take part in the different iterations to speed up the development cycle.

What about Enterprise Finance Application go with Waterfall approach instead of Iterative? Since Iterative approach is an enhanced Waterfall approach, it would not be surprised that it is applicable to the Enterprise Finance Application. The main different would be as stated above, the modularization will shrink from spanning from design stage to testing state back to limited to implementation stage. The means a full design should be finalized before getting into the implementation stage. The development of account payable, account receivable should all be done before starting the testing stage. Such approach should have no side effect to the ultimate product as *Features* and *Quality* are reminded fixed but it would use up more *Time* and *Cost* than adopting Iterative approach due to extensive design and testing will be involved.

Can Approval app get along with RAD then? Likewise, RAD shares the similarities of Agile which both have *Time* and *Cost* as fixed deliverables. Yet, RAD has a more flexible perspective on *Quality* as RAD deliver the product at the beginning stage as prototype rather than an officially tested product. Thus, it might cause some hiccups at the launch such as not able to do a simple task like approving a payment which might disappoint the users. Therefore, it is suggested that the beginning stage should get only a small group of pilot users who are in good relationship with the development team. Not only the users can report the problems promptly and directly, the level of tolerance would be higher whenever they hit on any issues.

Can E-commerce Website Implementation go with Agile approach? Supposing Agile approach can deliver a higher standard product than RAD given that *Quality* is one of the fixed deliverables. Thus, the website is implemented with more testing than RAD. As a tradeoff, less features will be provided given that the fixed Time cannot be extended and part of it has been used for extensive testing. Although Agile is one of the speedy approach that can deliver the website to come with some nice and functional pages with detail product information, it does not provide any channel for the buyers to place order. Thus, the market share cannot be as wide as expected due to lack of some core e-commerce features and the shareholders might be still not willing to invest more for the website or application.

8. Summary

Deliverables of SDLC

	Waterfall	Iterative	Agile	RAD
Features	Fixed	Fixed	Negotiable	Negotiable
Quality	Fixed	Fixed	Fixed	Negotiable
Cost	Negotiable	Negotiable	Fixed	Fixed
Time	Negotiable	Fixed	Fixed	Fixed

Phases of SDLC

	Waterfall	Iterative	Agile	RAD
Plan	Important	Important	Less Important	Less Important
Design	Important	Important	Less Important	Less Important
Develop	Less Important	Less Important	Important	Important
Test	Important	Important	Less Important	Less Important
Deploy	Important	Important	Important	Important
Maintain	Important	Important	Less Important	Less Important

Guideline to choose SDLC for Projects

	Waterfall	Iterative	Agile	RAD
Monetary Project	Suitable	Suitable	Not Suitable	Not Suitable
Incremental Project	Not Suitable	Suitable	Suitable	Suitable
High User Involvement Project	Not Suitable	Suitable	Suitable	Suitable
Trial Project	Not Suitable	Not Suitable	Suitable	Suitable

9. Meeting Minutes

Meeting Minutes of 23rd September, 2016

Date	23rd September, 2016
Time	13:45 – 14:05
Place	Room 3512
Present	Dr. David Rossiter Yan Ting Tiky Wong
Discussion	• Confirm the scope and objectives of the project • Set schedule for each section of the project
Follow up	Drafted report will be attached in the email and sent to Dr. David Rossiter to update the progress before the next meeting Proposed date: 16th October, 2016
Next Meeting	29th October, 2016 16:45 Room 3512 Cancelled 21st November, 2016 08:15 Coffee Shop (Lift 25-26, Ground Floor)

Meeting Minutes of 21st November, 2016

Date	21st November, 2016
Time	08:15 – 08:45
Place	Coffee Shop (Lift 25-26, Ground Floor)
Present	Dr. David Rossiter Yan Ting Tiky Wong
Discussion	• To check on the progress of the project • To confirm if more roles should be added and explain further • To discuss if the case studies are on track • To come up with what should be included in the summary • To schedule for the next meeting
Follow up	Finish up the case studies, summary and roles sections
Next Meeting	1st December, 2016 08:15 Coffee Shop (Lift 25-26, Ground Floor)

Meeting Minutes of 1st December, 2016

Date	1st December, 2016
Time	08:15 – 08:45
Place	Coffee Shop (Lift 25-26, Ground Floor)
Present	Dr. David Rossiter Yan Ting Tiky Wong
Discussion	• To check on the progress of the project • To review the summary • To schedule for the next meeting
Follow up	Refine the summary, figures and roles sections
Next Meeting	10th December, 2016 08:15 Coffee Shop (Lift 25-26, Ground Floor)

Meeting Minutes of 10th December, 2016

Date	10th December, 2016
Time	08:15 – 08:45
Place	Coffee Shop (Lift 25-26, Ground Floor)
Present	Dr. David Rossiter Yan Ting Tiky Wong
Discussion	• To review the whole report
Follow up	Final review and send out the final report
Next Meeting	N/A