Jia Min

1) Definition of Software Development Life Cycle (SDLC)

It is a conceptual model used in project management that describes the stages involved in an information system development project. There is various methodology that guides processes throughout. Documentation plays a crucial key regardless of the type of model chosen. Furthermore, the SDLC is important because it aids in Software Development, which gives us developers a guide to plan out according to the customer needs. In conclusion, to sum up SDLC methodology, it follows these general steps.

Firstly, the existing system is evaluated and shortcomings are identified. Which we don't have any details from the project specification. However, after some research on the various hotel management system, I would like to proposal the Customer relationship system (CRS). Reason why I choose this is because I think that it will benefit the business in many ways. For example, one of the way is to increases efficiency with the tools provided like its could produce many reports to analyze business to find out what went right or wrong.

Next, the new system requirements are defined. In addition, with the shortcomings of the existing system being addressed with specific proposals for improvement. Which my proposal has to be approved before I can move to the next step. Then, the proposed system can be designs with plans concerning physical construction, hardware, operating systems, programming, communications, and security issues. The new system is then developed. Company will then have to install the new components and programs to train the users of the system to make adjustments if needed. In my case the Customer relationship system (CRS).

After that, the business can either gradually replaced the old system or shut down the old system and implement the new one at once. Last but not least, after the new system is running for a while it should be evaluated and have regular maintenance so that users can be informed of up-to-date modifications and procedures.

The following fig 1 shows a typical Software Development Life Cycle (SDLC):

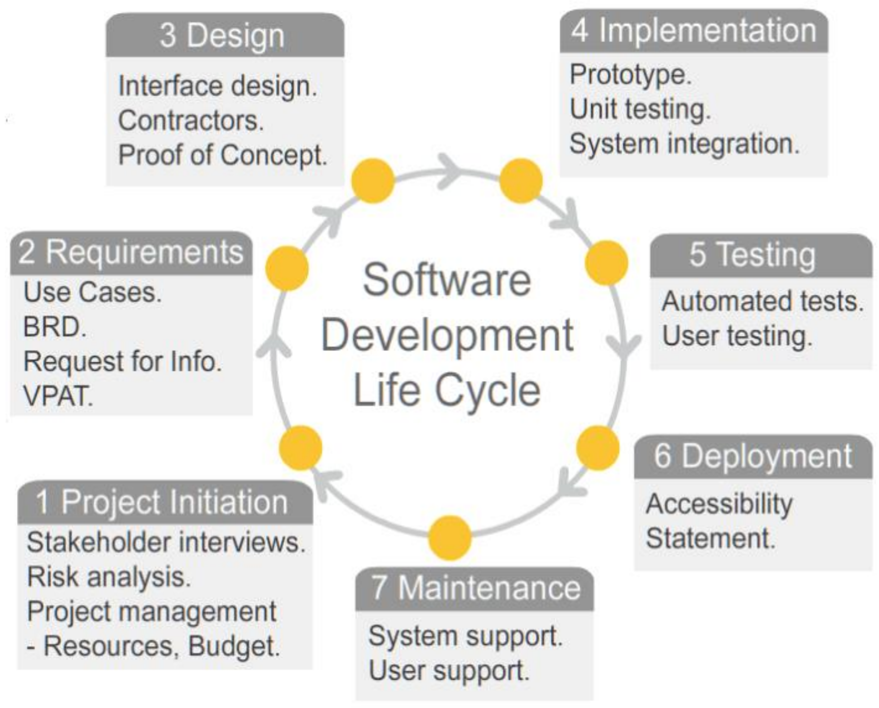


fig 1

1. **Project Initiation**

   This phrase is needed to plan out a rough idea, like a mini proposal to show the business what our strategy can offer them. Also, its shows all the break down of all of the resources, budget and risk analysis so that company can understand all the restraint/risk of the strategy that we are offering to them.

2. **Requirement gathering and analysis**

   The main focus here, is for the managers'/ business owners to determine their requirements. Thus, to work it out what is possible and not very possible to implement in the system. This phrase aids developer in the next phrase which is designing while keep all the focus in mind.

3. **Design**

   System Design helps in specifying hardware and system requirements which gives us a good overall system architecture/feel about our product. Also, testers can start coming out with test strategy to implement when the product is done.
4. **Implementation/ Coding**

This is where the coding starts, which is the hardest and longest phase of the software development life cycle.

5. **Testing**

   After the coding is done, the system is being tested against the requirements to make sure that the product actually solve the problem for the business. All types of testing are being done at this phrase. This includes functional, unit, integration, system, acceptance, as well as non-functional testing.

6. **Deployment**

   After testing is passed, the final product will be sent to business for their use. Beta testing is done first by business to catch any bugs which is turn send for fixing before the final deployment happen.

7. **Maintenance**

   Maintenance is one of the most important part, as from time to time we will never know if any real problem will arise when business start to use system. Thus, by keep up to date system will be able to last for a longest time as possible.

2) **Three Models**

   1. **Prototyping**

      This help client gets a glimpse of the system to understand the requirements of the desired system. Prototyping is a very good idea for those business that have no existing system, it is to provide a simple system to get an overall look of it. It can also be easily changed as it is in a smaller scale form.

      Advantages:

- Client and developers have more communicate, leads to lesser problem arise as developers can understand more about their requirement.
- Problem can be detected at a much earlier stage
- Client have a better understanding of the system
- Can be done incrementally, allowing client to make changes
- Missing & confusing functions can be identified

Disadvantages:

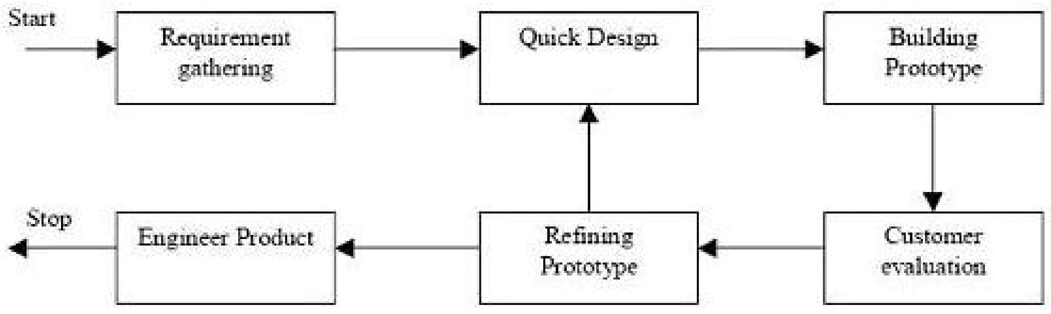
- Too many changes can corrupt software structure and may even lose it functionality

- There is no fix budget in the amount of prototype produces, many lead to over use of funds as the prototype may not be what the client wants
- Planning a prototype might be difficult as there is no regular deliverance
- Incomplete application may cause system not to be used as designed

Usefulness:

Helpful for Mr & Mrs wang business as they we are not very sure yet the specific requirement they want us to design. Prototype will be a very good model for us to show them. Also, our system overall needs to have a lot of interaction with the end users. Thus, prototype is the more recommended model to use.

Diagram of Prototype model:

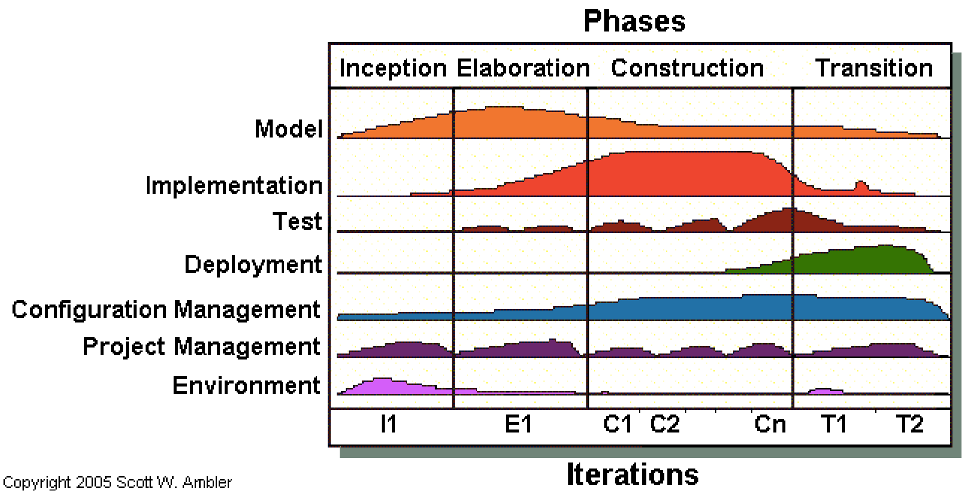


Prototyping Model

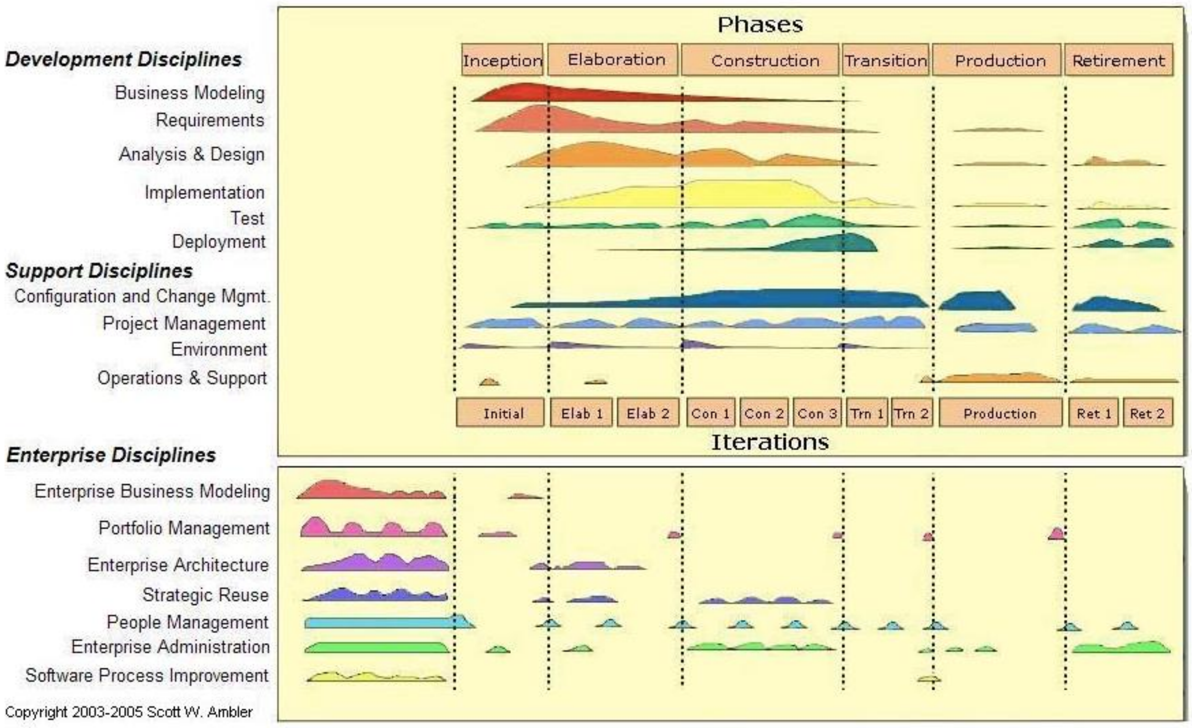
2. **Unified process (UP)**

   This method has a certain lifecycle to follow and its iterative and incremental, driven by use cases, and focus on addressing the risks early. The four project phases are Inception, Elaboration, Construction and Transition. A lot on documentation and using use cases to depict the system. Which sometimes may not be useful as some are unnecessary and waste of efforts as mostly documentation have to be in a formal writing. There are many types of UP like the Agile Unified process (AUP)figure1 and Enterprise Unified Process (EUP)figure2. Basically, UP is just a framework that developers used to customize a one that suits their business needs.

Figure 1. The lifecycle of the Agile Unified Process (AUP).



igure 2. The lifecycle for the Enterprise Unified Process (EUP).



Advantages:

- Emphasis on accurate documentation which give a very detailed analysis
- Less time for integration, as it goes throughout the SDLC
- Development time is less due to re-use of components

Disadvantages:

- Development process too complex and disorganized
- Integration throughout the process od software development, in theory sounds like a good thing. However, on big project with multiple development is will be chaos and cause more issues at the testing stage

Usefulness:

Our project is small scale, don’t really need this as prototype is already the better strategy for the simple system as we also do not have a lot of time frame for creating our system.

3. Rapid Application Development (RAD)

The RAD model are developed in parallel. Which time boxing helps to keep project on task. As the name already suggest that its emphasizes on the fast and iterative release of prototypes and applications. Thus, it can quickly give client something to see for them to provide feedback. There are five components for this model.

Business modeling – Analyze the current process of business to improve system flow.

Data modeling – Collect Information from business modeling to define data objects needed for business

Process modeling - After defining the object from data modeling, it is then converted to achieve some business objective. Its track the process of the system development. CRUD is created after identifying from the information collected.

Application generation – Automated tools used to convert process models into code and implemented in the actual system.

Testing and turnover – Test with various users of the system to test out the new components and all the interfaces.

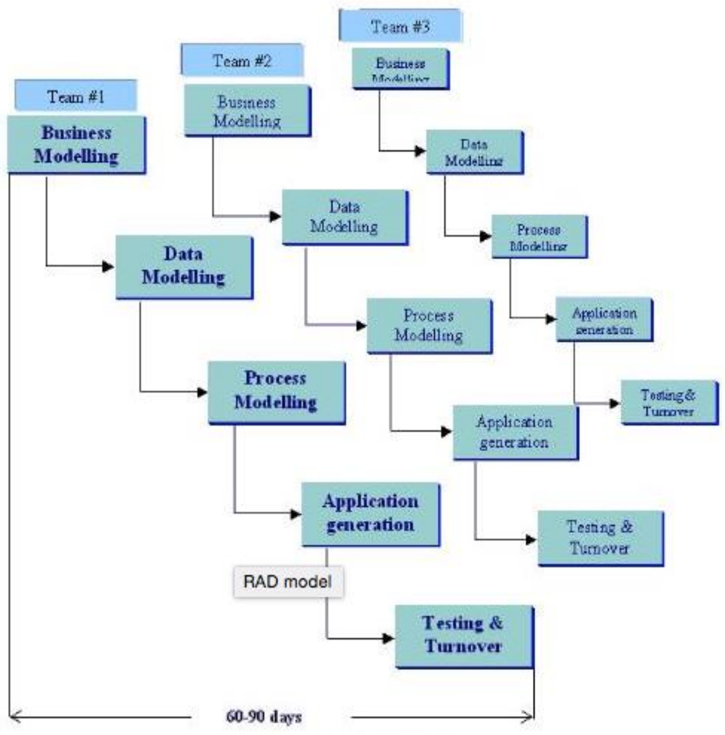


Figure 1.5 – RAD Model

Advantages:

- Good for business that want a quick result
- More Users feedback to improve on system
- Re-use software to save time, hence, reusability increase
- Save cost as the shorten the project, lesser changes

Disadvantages:

- Need a strong team/ individual performance for identifying business requirements, which its doesn't work if your team is not that experience
- Only modularized system can be built using RAD
- High requirements of both developers/ designers
- High cost for project as modeling not only need skill but money, however, overall it might still be cheaper depending on the project requirement

Usefulness:

We don't really have the recourses and skills to carry this model. Hence, not recommended for us to use.

References:

http://www.unimelb.edu.au/accessibility/users/development
http://searchsoftwarequality.techtarget.com/definition/systems-development-life-cycle
http://www.customer-relationship-system.com
http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/
http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/
http://www.agilemodeling.com/essays/agileModelingRUP.htm
http://istqbexamcertification.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/
http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-rup-software-development.html