Running Head: SDLC PHASES

SDLC Phases: Factory Systems Development Project

Jered McClure

Walden University

Note: This paper is based on the SDLC as defined in: Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2009). *Systems Analysis and Design in a Changing World* (5th ed.). Boston: Course Technology, Cengage Learning.SDLC Phases: Factory System Development Project

The “Factory System Development Project” is a high risk high reward systems upgrade (Satzinger, Jackson, & Burd, 2009, p. 69). Sally Jones has an opportunity to increase the overall input/output of the organization through automated streamlining efforts. However, this must be done without drastically interfering with worker production on the factory floor. This means an iterated approach such as a spiral model is not going to work. As such, I would recommend the modified waterfall approach (Satzinger, Jackson, & Burd, 2009, p. 42) .

Approaching this system development in a systematic step by step phase is the only way of achieving a working system without interfering with factory operations. However, a pure waterfall model would not leave room for adjustments within the project where changes are required (Satzinger, Jackson, & Burd, 2009, p. 41). The modified waterfall model gives the analysts and developers a way to back track before moving on to the next phase. The down side to all of this is that there will be no prototyping, once the system is live, it moves into the support phase.

In the analysis phase of the waterfall model, management and workers should be actively involved with determining requirements. This includes acceptance forecasting of the changes to be made, proposed training required for the system, and how the new system will interfere with the status quo of factory operations. Management should specifically be part of determining the prioritization of system changes and reviewing alternative recommendations from various vendors. Overall, the key factor in the analysis phase is to gather all the information and collate it into a workable model for the design phase.

The design phase should include both workers and management when developing the user interfaces. This is because workers and management both will be involved with using the system post implementation. If they cannot understand how the system operates, or are unhappy with how to interface with the system, then it will be a detriment to the overall operations of the factory floor. However, management should specifically be involved when developing the tracking software of the new system. After all, the goal is to automate the tracking of work in progress as well as inventory post production (Satzinger, Jackson, & Burd, 2009, p. 69).

In a traditional Systems Development Life Cycle (SDLC), analysis is done in order to inform the designer about the specific specifications required. That is “by deferring decisions about technology, developers can sharply focus their efforts on what is needed, not on how to do it” (Satzinger, Jackson, & Burd, 2009, p. 56). This means that analysts can liaise with all appropriate parties, find out what the problem is that needs to be addressed, and brainstorm possible solutions, all before any design work is underway.

On the other hand, the object-oriented approach of the SDLC analysis is one of “defining all of the types of objects that do the work in the system and showing what use cases are required to complete tasks” (Satzinger, Jackson, & Burd, 2009, p. 60). Similar to the design phase of traditional techniques, the analyst creates a diagram defining each individual object outside the system and how it interacts with the system as an object. For instance, users and what they do with the system, as well as, additional systems which interconnect with the system being developed.

Once the analysis part of the project is complete, designing the system can begin in earnest. In a traditional SDLC, this means laying out the entire programmatic structure in diagrams, which can then be broken down and developed into their component parts. Each module of the diagram should be independent of the other modules, but work together to solve the overarching problem. Also, each module should accomplish a specific task unto itself, thereby alleviating the need to create redundant modules throughout the system.

In the object-oriented SDLC, design is focused on defining all of the interacting objects within the system which communicate with the users as well as other devices which connect to the system (Satzinger, Jackson, & Burd, 2009, p. 60). This encompasses creating a diagram showing object interactions, tasks for each object, as well as further definitions describing each object. The diagram should act as a map on how the differing object in the system will interact and react to one another allowing programmers to better meet the analysis goals.

Now that the system is designed it is time to develop the system. “All the prior activities must come together to culminate in an operational system” (Satzinger, Jackson, & Burd, 2009, p. 47). This is the actual software programming, testing, installation, and training phase of the traditional and object-oriented SDLC. Meaning that everything in the project has been leading up to this point, this is the final output.

Implementation is where both traditional and object-oriented approaches overlap. Both could inherently use object-oriented or traditional structured programming languages. Alternatively, object-oriented design could use traditional design at its base diagram levels, thereby giving rise to a hybrid of the two techniques. Unfortunately, there is no real way to hybridize the traditional technique as each part of the process is designed towards a structural means.

When designing a system for a live workplace where downtime is not an option, a spiral model with active prototyping is not an option. The analysis and design must take a systematic approach whereby the final outcome is the implementation phase moving on to the support phase thereafter. That being said, the technique used for such a model can go either down the traditional path or the object-oriented path, each having its benefits. Overall, however, the object-oriented approach is capable of hybridization within its technique, whereas the traditional structured approach is unable to move from its systematic methods. In essence, a choice must be made at the project planning phase whether to use one or the other, as once the ball is rolling it cannot be changed with the traditional technique (see page 6 for a chart comparing the two techniques).

Reference

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2009). *Systems Analysis and Design in a Changing World* (5th ed.). Boston: Course Technology, Cengage Learning.

Traditional vs. Object-Oriented Techniques

|  |  |  |
| --- | --- | --- |
|  | Traditional Technique | Object-Oriented Technique |
| Analysis | * Define system requirements * Define Data types and uses * Define system events | * Define object types * Define object interactions |
| Design | * Design system structure * Design system modules * Design subsystem modules | * Design object types * Design object interactions |
| Implementation | * Develop software * Test Software * Train Users * Install software | * Develop software * Test Software * Train Users * Install software |