LEGACY-SYSTEM CYCLE MODEL

The Process of Repeated Interaction Design

Omie R. Walls

University of Wisconsin-Milwaukee

Interaction design should be practical enough to create and maintain productivity – as this is the ultimate intent of any design model. After observing different kinds of interactive designs and comparing the advantages and disadvantages of each process, the Legacy-System Cycle Model was created in order to provide a context-specific layout for businesses who have reason to be concerned with maintaining and upgrading their aging legacy-systems to support longevity and secure a thriving future.

INFOST 310 Human Factors in Information Seeking and Use: Section 202_31573

Fall 2015

LEGACY-SYSTEM CYCLE MODEL

The Process of Repeated Interaction Design

Introduction

In many small businesses, legacy-system replacements and renovations are often rebuffed. Most assume there is not enough support or financing to upgrade their information systems. As result, mounting maintenance costs and system rigidness compile – further complicating the situation and costing their businesses more in the long-run. At this point, the company is ultimately forced to surrender to an upgrade or a complete system migration. Commonly, this decision takes place after the risk of impending loss in market-share or security threats are exposed in the legacy-system.

The Legacy-System Cycle Model, or LSC Model, is the proposed solution to businesses that are apprehensive about upgrading their information systems. This scalable model divides the upgrade into manageable segments that the company can undertake with specified budget and schedule constraints. The LSC Model can be used to gather an overall case for the upgrade, target and prioritize critical areas and weaknesses within the system – while delivering consistent progress, high-level documentation, and regular evaluation. The model's continual design will aid companies in driving down long-run maintenance costs in order to develop and preserve a cost-effective strategy towards improving system performance.

Apart from having a revolving design, the LSC Model is intended to be the most similar to the Waterfall method, however it carries additional principles in its design that were found useful in the Spiral, Dynamic Systems Development, and Contextual Inquiry Models - picking up in areas where each design model leaves off due to the fact that any of these models cannot be realistically proposed as an solitary solution given the disadvantages that lie within each.

In regards to the linear design of the waterfall model which flows steadily downwards through the phases of requirements, design, implementation, verification, and maintenance, it is hard for a business to maintain repairs within a reasonable budget due to the fact that unexpected changes are rather exorbitant. In the spiral lifecycle model, the design is modeled for enormous, complex, and costly projects due to its conjunctional design that takes features from both prototyping models as well as the waterfall model. Though it is an improved fit for a business situated with legacy software, it requires too much risk analysis that the business might not have time or money for. This is because preparing and testing for risks is fundamental in a prototyping design, but not necessarily as much in a design that would be suitable for legacy systems.

Dynamic systems development models generally are considered to carry the core agile project framework. This is one of the first models designed for the RAD (rapid application development) method with fixed costs, prioritization, incremental delivery, and sequential phases. The DSDM project life-cycle spans over three parent phases; the pre-project, project life-cycle, and post-project. The project life-cycle is a sequence of five phases within itself. This model is the closest to that of the LSC Model. The key difference is in the prototype development theme that DSDM is geared towards. A different

approach should be taken and different risks need to be analyzed when considering legacy systems. This is the same situation with contextual inquiry models. Legacy-systems need thorough user analysis that dynamic systems development models aren't created for. There's a certain level of functionality and design that goes above user-centered design in legacy-systems however, so the contextual design methodology should not be sole solution to upgrading any legacy systems.

The LSC Model simplifies using each of the four above-mentioned designs in an easy-to-follow format that at-large mirrors the alternate design of the Agile/SCRUM method which is one of the more popular alternative interaction design models for software development as it borrows principles from lean manufacturing.

The initial effectiveness of the LSC Model depends on the age of the existing system and the constraints that are placed on the process by the business during the analysis phase. This ability to scale down the process on a micro level helps the business to control expenditures with rigid budget. The model's cyclical form ensures that documentation and reviews are performed regularly. The simplistic design of the flow chart can be observed at the end of the document. At-large, it carries the same structure as the Waterfall method, however the sub-task cycle in the production process gives an extra added benefit of flexibility. Breaking down large tasks into smaller components lessens the demand of a highly-skilled team. With adequate research and moderate skill, these sub-tasks can be delivered at a desirable speed and will require less specification or financing by way of rolling update developments.

The Legacy-System Cycle is divided into five basic phases- analysis, development, production, deployment, and evaluation. Each phase is divided into sequential sub-tasks that are essential to meeting system requirements in order to ensure quality of the delivered product.

Analysis. The analysis phase breaks down several factors that dictate daily business interactions with the core system. End-user observation is best performed through a variety of means in order to get an encompassing view of the situation. The first step, data-gathering should carry the primary objective of witnessing the setbacks of the current system by measuring its efficiency. Businesses can help assess the pitfalls of their legacy-system by timing processes and comparing the results among the performance of the competitors in the market, conducting interviews, and observing the process-handling sequence of the end-user.

Conducting proper analysis of stakeholders is the part of the process with the widest-range of results. The level at which the stakeholders are assessed in the analysis phase, depends on the scale of the upgrade to the system. Upgrades that impact the business at a higher level should involve a more thorough interaction with stakeholders than it would with a simple, routine maintenance. If stakeholders determine that an immediate legacy-system overhaul is to be completed over a short-term period, the business will have difficulty following the LSC Model in the production phase as sectionalism is a key feature of it. The combined results of the LSC Model are optimal for small businesses that take a conservative to moderate approach towards upgrading and maintaining their legacy-systems. A more basal approach would likely require the Contextual Inquiry Model or Waterfall Model where the analysis phase typically draws the most focus. The purpose of the LSC Model is to give small businesses an immediate start on fixing and optimizing their legacy-systems so that critical performance upgrades do not fall to the wayside and impede their latter progress.

With operational feasibility addressed, the business will have a better understanding of what areas to test their system performance with the list of system requirements gathered from the stakeholder analysis. The goal of analyzing the system's technical feasibility is to address the current system performance, analyze possible design limitations which can negatively impact the upgrade, and assess whether the current system can integrate future applications and processes that the business will need to perform in order to grow or stay in operation.

In this assessment, it is critical to take note of what can be changed in the system without jeopardizing a breach in security or causing sufficient system downtime. Some legacy systems are designed in such a way that a tactical approach will need to be done to secure sensitive information during the upgrade. It is also important to take heed that some updates might expose security weaknesses in other areas. After the critical components have been identified and analyzed, lesser-important system requirements can be addressed immediately or in subsequent cycles of the model.

At the point which system requirements have been made and adjusted to realistically fit the system's technical abilities, financial constrains are necessary. Monetary feasibility is a top-deciding factor in whether to maintain or migrate legacy-systems. A successful budget is also dependent on a multitude of factors, such as determining the different IT skillsets necessary to start and finish the job, maintaining environmental control to keep systems running at peak levels, training and updating end-users on the changes made during the system upgrade, finding any potential return of investment to offset maintenance costs, conducting market research, and making any hardware and software changes to keep the existing system running optimally while installing the new system. In the later cycles of the LSC Model, small businesses might not have the time or reason to perform a cumulative financial review before performing maintenance, and might opt to just set aside a safe amount of funds periodically for updating the system. The amount of funds offered to the development process determines the speed of the upgrade and the amount of functions the business has going on simultaneously. Funneling too much money into the process at once or without a sufficient budget plan might cause for unplanned hindrances in the deployment of the product. Incidentally, delays in the deployment can be costly if a business aims to see a return of investment upon deployment. Due to reasons like this, it is important to come up with a sufficient formula that allocates enough flexibility to prepare for unforeseen consequences of upgrading the legacy system.

Another great benefit of the LSC model is that with repeated cycles of maintenance, time estimation analysis will improve greatly in accuracy. After budgets have been made, it's important to define the schedule of spending, understand the factors that will create limitations with scheduling, and know which areas of the process to allocate extra time for flexibility when handling difficult situations.

One of the most worrisome situations for small businesses is handling potential security threats with outdated systems as well as new system updates. Determining where the liability falls can affect how the entire process will flow as well as how frequently evaluations are performed. It is critical for businesses with trade secrets and sensitive consumer data to handle proprietary information with the utmost care and caution.

Development. After analysis has been completed, the cycle moves into the development stage. The conceptual design development is aided by user stories and completed user observation from the

analysis phase. In this phase, Entity-Relationship, Web-Flow, Network Architecture, and UML diagrams are helpful in covering multiple aspects of the design and making sure all inputs of data have appropriate and secure outputs. Having a detailed mock-up is an exceptional way for a business to capture the bigger picture which makes it easier for the next part of the development cycle to occur.

Essentially task completion is what aids the process in moving towards deployment. It's difficult to move from analysis to task management without using a diagram model to follow along. It is also hard to define what to do by examining a diagram alone. Translating the diagram into major tasks ensures that there is a realistic definition of done agreed upon before production begins.

If the business is satisfied with the criteria for deployment, then they must follow up with determining which tasks have a higher priority. Prioritizing tasks ensures that appropriate time and other resources are spent in key areas and not to frivolously in lesser ones. After the amount of time it takes to perform each task is assumed, it is necessary to put them in order to define the scope for each progress benchmark and ensure that tasks are being performed in the correct sequential order to avoid wasting efforts or putting tasks on the scope that cannot be performed without a prerequisite task that has not yet been completed.

The scope is equivalently as critical as the timeline as it guarantees consistent reviews can be performed throughout the upgrade in a timely manner, the correct amount of tasks are dispersed evenly in each period of production, and also that project managers are given regular accountability. The scope consists of the specific tasks that will be done in a given period. After it has been set, tasks are ready to be dispersed among members of the team in order to proceed into the production phase.

Production. The production cycle is where the business is responsible for dividing tasks into sub-tasks for manageability and better measurement of daily progress. Production at the sub-task level is a scaled-down iteration of the LSC Model itself. Each individual should be on the same accord with the same interaction design as the business to lessen the amount of conflicting ideas over the manner in which tasks should be completed. With uniformity in the sub-level design process, more productivity can be achieved. In continuing the theme of consistency, documenting the existing code is a major factor in the LSC Model.

Documentation is often a fallacy for many legacy-systems. Many IT individuals imply their code is "self-documenting", however without a clear explanation of the existing functionality of the legacy system, it is possible that time would be wasted re-doing or missing steps in the process when upgrading or performing scheduled tests and maintenance due to the system not being easily legible. This can lead to a breakdown in communication and consequently more errors.

After the code has been documented and understood, the research should continue under the focus of the sub-task at hand. Allocating time for research on sub-tasks is a property of the LSC Model that more so benefits teams who aren't highly skilled, though even the highest skilled IT professionals will find it useful to allocate regular time for research.

Through gathering the solution with research, it is possible for a professional to document the changes to be made. This is a beneficial step that serves as a foundation for performing the sub-task, keeps the sub-task organized, reduces merge conflicts and the documentation process later-on. In the larger model, this section is equivalent to the development stage.

When the sub-task is completed, the task should be reviewed. The LSC Model works best with tasks being self-checked, peer-reviewed, and pushed ahead for a final review. This filtering process is necessary to maintain the deployment deadline and reduce negative feedback about functionality errors.

Deployment & Evaluation. The fourth phase, deployment, compiles all the sub-tasks from the production phase in order to deliver a final product. The final product depends on the scope, version, and cycle that the LSC Model is in. In this phase, the output ranges from beta tests to polished version-controlled releases. It is important to develop a means to run tests, monitor user interactions, and record performance results in the deployment stage in order to have content for the evaluation stage.

In the LSC Model, the evaluation stage is where reviewing feedback, system performance, budgeting, and scheduling takes place in two different ways. Cumulative evaluation is the last phase of the model, however provisional evaluations should happen within every step of the process.

Conclusion. After gathering and reviewing the feedback from beta tests, the cycle continues as the business uses these evaluations to analyze and plan their next development cycle in order to improve performance and manageability of the legacy system. Through continued usage, an establishment applying the Legacy-System Cycle Model will enjoy having a business management solution that supports the company's quality standards and enables them to keep growing.

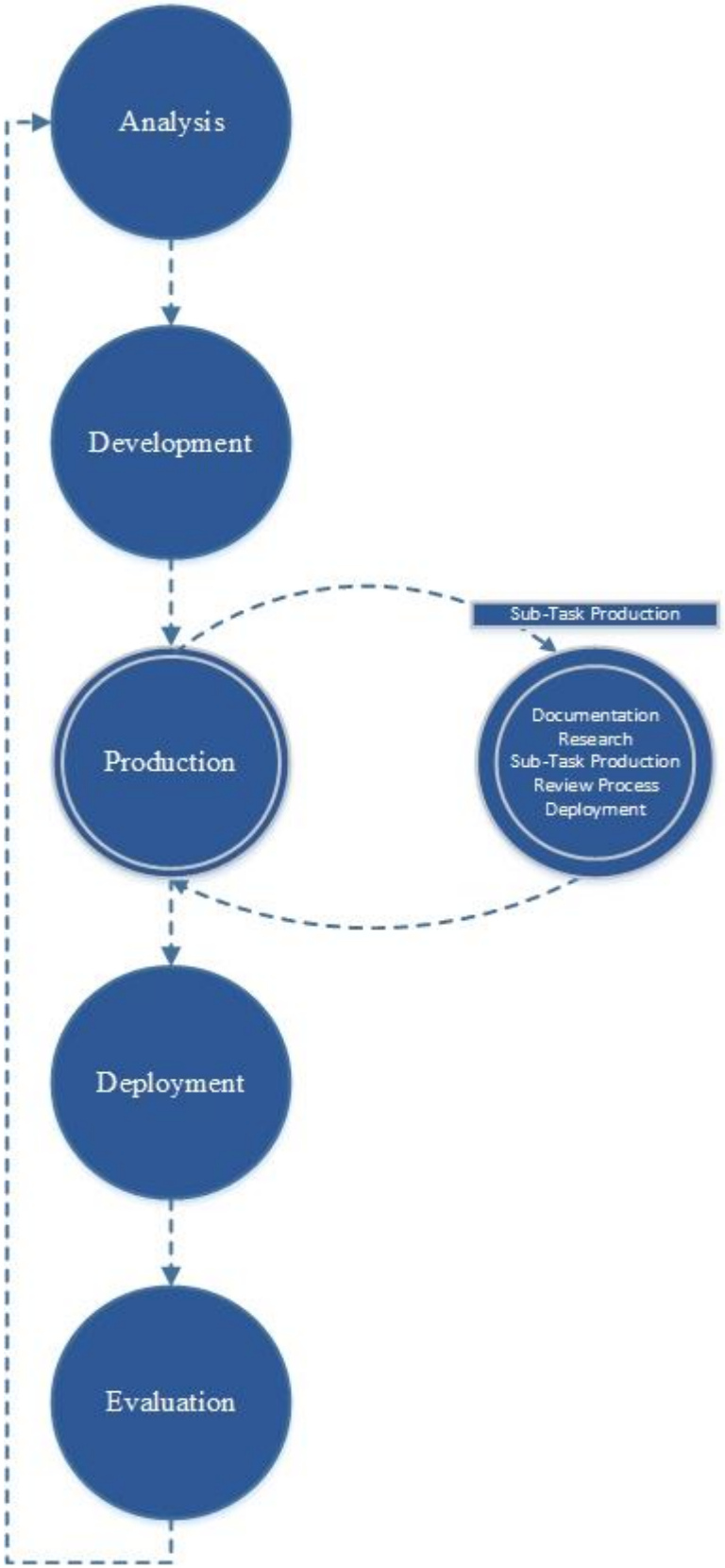

Figure: Flow Chart for Legacy-System Cycle Model