Software Development Lifecycle

Verizon uses a standard software development life cycle to ensure that software requirements are properly analyzed, designed, developed, tested, and implemented. The major phases of the life cycle are summarized below and are illustrated on the attached timeline. This diagram illustrates the typical duration of software development life cycle. The software development process for the Verizon OSS is a complicated undertaking. This schedule has been developed to optimize the amount of business functionality that can be implemented within targeted time frames and quality levels. Experience has demonstrated that attempting to shorten or abbreviate this timeline jeopardizes software quality and causes unproductive re-work for Verizon and the CLECs.

Below is a summary of key activities that occur in each of the phases:

1) Requirements Analysis/ Package and Close. Requirements analysis begins with identification of initiatives to be implemented in the software release. Each initiative describes the desired business functionality to be implemented. During this stage, Verizon develops system requirements that explain to developers at a detailed level the required inputs, business logic and outputs that must be implemented to achieve the desired business functionality. Verizon begins a General Design process, which is an iterative, cooperative effort across the multiple systems within the OSS and concludes with a preliminary technical specification and an estimate of effort and time to implement.

2) Design. Starting with the General Design that outlines the inputs, outputs, business logic and system implementation approach, the development staffs for each of the applications translate those requirements into Detailed Designs by identifying the specific changes to specific modules that need to change, as well as the detailed content and format specifications for data exchange between the applications. Changes such as the introduction of new transactions are complex. New transactions create new specifications for the CLEC interfaces. In addition, new transactions require changes within Verizon’s systems, between the interfaces and the middleware, and between the middleware and the backend systems.

3) Coding and Unit Testing. This phase requires many concurrent activities. The development staff of each impacted application opens the code base and makes changes to the code as specified in the detailed designs. Inconsistencies or errors in design will become apparent during this phase, which results in Verizon’s adjusting the design. After coding, each application change is unit tested to ensure the design was implemented correctly. Next, each application undergoes system testing to make sure that all changes made in the cooperating modules work to produce the designed change. Then the applications perform near-neighbor testing to test all communications between the applications. While the development teams are coding and testing, the analysis teams are preparing changes to the documentation to provide to the CLECs so they can make the appropriate code changes and prepare for testing during the final phase.

4) Integration Testing. During this phase, the development staff of all impacted applications move their new code and other interdependent modules out of the development environment and into an integration test environment for verification by the Integration Test team. The team exercises the new code in two ways. First, the code is *Regression Tested* to ensure that the functionality that was supposed to remain the same did so and that no new defects were introduced. Second, the code is *Progression Tested* to ensure that the changes that were designed, coded and tested were done correctly and have the correct result.

5) CLEC Test Environment Testing. In this phase, Verizon migrates the new code and interdependent modules into a CLEC Test Environment ("CTE") to enable application-to-application testing of pre-order and ordering. During this phase, the CLECs test against the new Verizon software release to ensure the quality of their own internal development, to verify compatibility with the new Verizon code and to facilitate the identification and correction of defects on either side of the interface before new code is migrated to production. Releases in the CTE environment are controlled and announced to CLECs to keep the code base as stable as possible.

Provided that there are no changes to the requirements, the software development life cycle described above takes approximately eight months to complete. Any change in requirements will necessarily lead to delays, as the entire process must be carefully calibrated to take into account all of the ramifications associated with any such change.

The eight month timeline for developing the software in a major software release cannot be shortened by doing the separate steps simultaneously. Instead, each of the five steps must proceed in a linear fashion. The Detailed Designs cannot be developed until all of the requirements are identified and documented. The software code cannot be written until the Detailed Designs are written and the affected systems are identified. And the software code cannot be tested until it is written. Moreover, additional resources are not helpful when multiple initiatives require coding changes to the same modules or application, or when a particular task cannot be accomplished within the allocated development period. In fact, management experts have long recognized that blindly adding bodies to a project only creates problems.[1]

The implementation of major software developments can have major effects on the systems of Verizon and CLECs. It must be done under very precise and controlled conditions – subject to careful testing throughout the implementation. It is for these reasons that Verizon schedules the implementation of major software developments to occur only three times each year – in February, June and October.

[1] Lawrence H. Putnam, Ware Myers, *Executive Briefing: Controlling Software Development* (IEEE Computer Society Press 1996).