Baseline SCIs

Brynn Riebe

Grantham University, CS 406

Professor Atkison

9/25/19

1. BER\_CS406\_SAMS

-Planning

a. Project\_Management

b. Configuration\_Management

-CI\_documentation

c. Risk\_Management

-Risk\_documentation

-Iteration\_1

a. Planning

b. Requirements

c. Design

d. Implementation

e. Integration

f. Validation

-Iteration\_2

a. Planning

b. Requirements

c. Design

d. Implementation

e. Integration

f. Validation

-Iteration\_3

a. Planning

b. Requirements

c. Design

d. Implementation

e. Integration

f. Validation

-Close Project

1. The SCIs needed for each phase within the structure would be the same for each iteration. For the planning baseline: the SCIs required would be domain models, use cases/use case diagrams, a use case delivery structure, the requirements for the project, drafting the architectural design, and making a requirement-use case traceability matrix. For the requirements baseline: the SCIs would be to update any items within the planning baseline if there are any, creating the domain model or refining it, create a validation test plan, and creating the preliminary user’s manual. For the design baseline: the SCIs would be to create the software design document and integrate the test plan. For the implementation baseline: the SCIs would be unit test cases, creating the source code lists, and create the unit test reports. For the integration baseline: the SCIs would be creating use case-based integration test cases and creating the test report. For the validation baseline: the SCIs would be creating the user’s guide, creating a user’s manual as built, creating the installation manual, and creating the test cases and report for validation.
2. Dm, use\_case/use\_case\_diagram, use\_case\_delivery, requirements, architectural\_design\_draft, traceability\_matrix, refine\_dm, validation\_tp, prelim\_user\_manual, software\_designdoc, integrate\_tp, source\_code\_list, test\_reports, users\_guide, user\_manual, installation\_manual, validation\_test, and validation\_report.
3. Changes that could occur could impact multiple items within the model due to inheritance, aggregation, and association relationships. If one of the diagrams are updated, then the other diagrams will most likely need to be updated. This is the same case if a diagram is deleted. If these things occur the design specifications will also have to be changed. Some events that could occur could be changing operational requirements or improving/enhancing requests from the users or even changes to the budget, project duration, and schedule.
4. An ECP, or engineering change proposal, consists of administrative forms and supporting technical and administrative materials that specifies a few things. These things include a description of the proposed changes, identification of originating organization or developer, rationale for the changes, identification of affected baselines and SCIs, the effort/time/cost required to implement the proposed changes as well as the priority of each of the proposed changes, and the impact to the project schedule.
5. A. The steps needed for a baseline to transition from TBE to a formal baseline include establishing all of the associated configuration items, passing an SQA inspection, reviewing and/or testing, and entering them into the SCM system.

B. Visual inspection will be performed to verify the configuration items.

C. The CIs will be run through a group of people to ensure that the expanded use case matches the user’s expectations.

D. Change control auditing will ensure that there will be quality and consistency when changes are made in the configuration object. This is where the change request is submitted to the software configuration manager. This will cover everything that the ECP goes over and therefore will promote a change request to the project in a timely manner by making the configuration items available during the software lifecycle. The project manager will be in charge of making sure that the processes and policies are followed for changing the project and then testing it.

Kung, D. C. (2014). Object-Oriented Software Engineering. Retrieved September 26, 2019, from https://online.vitalsource.com/#/books/0077508440/cfi/589!/4/4@0.00:29.9.

Software Configuration Management in Software Engineering. (2019). Retrieved September 26, 2019, from https://www.guru99.com/software-configuration-management-tutorial.html#6.