Design

CxOne Standard

CxStand_Design.doc

November 3, 2002

Construx
SOFTWARE

cxone

Advancing the Art and
Science of Commercial
Software Engineering

CONTENTS

1 Introduction

This CxOne standard for the design Construx Knowledge Area (CKA) organizes CxOne design support and provides a lightweight basis for developing design processes.

1.1 Overview

Design is the creation of abstracted models and plans for implementing requirements to create a software system. Design is a complex activity that does not lend itself to rigid application of precise rules or templates. Due to the flexible and open-ended nature of design, and the fact that design methodologies are one of the most mature and prolific areas of software engineering, CxOne seeks to leverage existing design knowledge, techniques, and methodologies.

CxOne support is independent of whether you are doing object oriented vs. functional design, and independent of any particular methodology. Instead of redefining readily available support in the public domain, CxOne focuses on making it easier for organizations, projects, and engineers to use existing resources to create custom design processes and do design work.

1.2 Goals

CxOne support for software design focuses on the following goals:

- Defining common design terminology based on the SWEBOK.
- Distillation of common design principles into readily available checklist items.
- Provide a resource for information about existing design best practices, methodologies, and techniques.
- Increase productivity by addressing design issues in a uniform fashion, reducing common errors, duplication of effort, and multiple solutions for the same problem.

1.3 Background

CxOne design materials are synthesized from many sources including the SWEBOK, IEEE software standards, Steve McConnell's works, other industry sources and literature, and Construx's experience with software projects.

1.3.1 When are you doing Design?

Design is sandwiched between requirements and construction activities, and the lines separating design from these upstream and downstream activities are blurred. Requirements, design, and construction work often occurs iteratively or in parallel with each other. For classification, the timing of the activity is not as important as the purpose or intent of the activity.

1.3.2 Design Lifecycle

The general lifecycle for design activities includes a progression from architecture to high-level design to low design. The initial architecture is usually started in parallel with the project planning and requirements activities. The architecture along with high and low level designs are then developed and refined throughout the project. When design activities occur will be highly dependent on the lifecycle followed for the project and the project plan.

Figure 1-1 below shows a view of this lifecycle. Most projects will approach design in an iterative fashion, which this diagram does not attempt to illustrate.

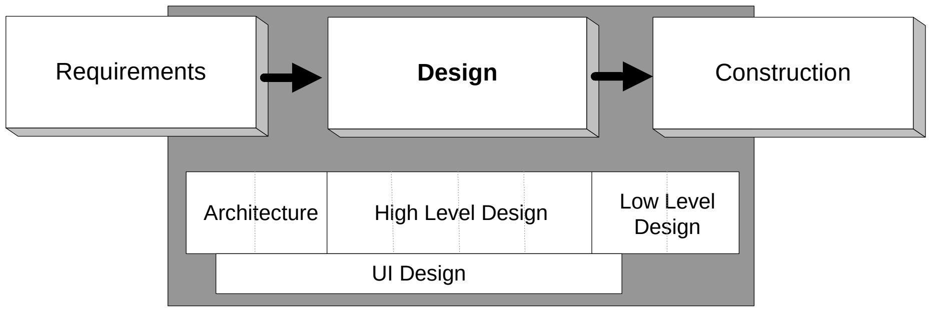

Figure 1-1: Simplified Design Lifecycle

1.4 Relationships with other CKAs

1.4.1 Design and Requirements

The line between when requirements activities stop and design begins can be difficult to pin down, but usually isn't necessary to pin down exactly. Since projects are unique the final decisions on this must be left to the project team, however some general guidelines are:

- Requirements work focuses on determining "what" is the solution.
- Design focuses on determining "how" the solution will be accomplished.
- Requirements and Design may use some similar techniques and may cover similar territory, e.g., domain modeling. Whether specific work is described as requirements or design in these cases should be determined by *intent.*

For example, several UML diagram types may be employed for requirements or design descriptions. CxOne has separate supporting materials for these in the Design and Requirements CKAs, based on whether the diagram is being used to define the "what" or the "how".

1.4.2 Design and Construction

The line between design and construction is a little clearer than between design and requirements, but not by much. Some organizations or projects define low-level design as specifying internal details for implementing a module, and consider any work more detailed than that to be part of construction. Others differentiate based on whether text describing design issues is contained in a separate document or in source code comments.

CxOne treats **design as an abstract activity** and **construction as a concrete activity**, but does not dwell on exactly where the line separating them is. CxOne does define this line implicitly through the materials organization of the design and construction CKAs.

Where that line is drawn and how designs are packaged is less critical than ensuring that an appropriate amount design occurs before coding begins. What the appropriate amount is depends on many variables including complexity, technology and other implementation risks, staff issues, lifecycle, project goals, etc.

The purpose of separating design activities from construction, and specifying software design in as much detail as possible before construction, is to leverage the power of abstraction. By solving design problems at a high level before moving into more detail, rework is reduced, quality is increased, and early visibility is provided. The types and amounts of activities performed and artifacts produced will vary depending on the needs of the project, but in general it is better to err on the side of doing too much vs. too little design.

Documenting Design after Coding

Unfortunately it is not uncommon for designs to be documented after the code implementing the design has been written. This is NOT a design activity; it is a maintenance activity (support for this activity is provided in the maintenance CKA). Once code has been written, the design for that code has already been done; documenting it after the fact is recovery of the implicit design. CxOne's focuses on explicitly defining a design before setting it in code.

1.4.3 Design and Quality

Peer reviews are the primary way to identify defects in the designs prior to construction. The particular type of review and its length varies on the complexity and purpose of the design. Design assumptions and concepts may be tested through the use of functional prototyping. Mathematical modeling can be used to prove correctness of some types of highly constrained designs (network, scaling).

1.5 Relationship to External Standards

There is considerable industry literature available that organizes the software design body of knowledge, but there is no universally accepted taxonomy. CxOne bases its organization and taxonomy on the SWEBOK, but also incorporates elements from other sources. The goal is not to adhere to any particular standard, but rather to provide support that will be natural for experienced design practitioners while still being accessible to novices.

1.5.1 SWEBOK

The CxOne design knowledge area follows the breakdown described in the SWEBOK, but differs from the SWEBOK in the following respects:

1. CxOne overrides add to many areas of the detailed SWEBOK design description.
2. CxOne does not utilize the "Software Design Quality Analysis and Evaluation" topic in the SWEBOK breakdown. These topics are covered in CxOne's Quality CKA.
3. CxOne has divided the "StructureAndArchitecture" topic in the SWEBOK into separate "Structure" and "Architecture" topics. This offers a cleaner organization of architecture artifacts from other high-level and low-level design artifacts.

4. CxOne has not created artifacts for the “Basic Concepts” topic in the SWEBOK breakdown. This area covers underlying concepts and context of software design, general design principles, design processes, and enabling design techniques such as abstraction or decomposition. *This topic will be addressed in future revisions of CxOne.*

5. CxOne has not created artifacts for the “Design Notations” topic in the SWEBOK breakdown. There are many languages, diagram notations and style, and techniques that have been developed for capturing, analyzing, communicating and storing design information. CxOne does not prescribe use of particular notations, instead support is provided for the use of proven notations that are useful in many situations. As described below, the UML plays a large role in CxOne design notations.

1.5.2 UML

The Unified Modeling Language (UML) has a heavy influence on CxOne design materials. The UML encompasses both a design notation and design methodology that is the de facto industry standard for object-oriented design (and requirements modeling as well).

Due to its general utility and widespread adoption, CxOne focuses on the UML when describing design notations and processes. CxOne treats the UML as a set of discrete practices, as opposed to a monolithic whole; notation and concepts are separated from processes and techniques. CxOne does not limit design activities to the UML, as there are approaches and notations outside of the UML that may work well for certain situations.

When there are no inherent advantages in technique or notation, UML has the advantage of standardization so is normally a good choice. Conversely, if you work in an organization or industry where there are particular notations or techniques that work well and are well understood, it may make sense to choose them over UML approaches.

1.5.3 IEEE Standards

There is implicit and explicit support for IEEE design standards in CxOne material, but the design CKA extends the scope and level of support provided by the IEEE standards.

2 Organization

The organization of CxOne Design CKA materials is described below.

2.1 Structure

The Structure topic covers design techniques, practices, and artifacts used to define the internal structure and operation of a software system. The definition of specific design levels and views along with reusable design elements such as patterns occur in this topic.

2.2 Architecture

The architecture section focuses on architectural techniques, practices, and artifacts. The definition of specific architecture views along with reusable elements such as architectural styles are included in this topic. Architecture was mentioned in the previous section as the first level of design abstraction. In CxOne this topic has diverged from the Structure section to give proper attention to the unique issues discussed at the architectural level. See *CxGuide_Architecture* for details of these issues.

2.3 Key Issues

Key issues are critical software design issues must be dealt with on all software designs to ensure quality. Examples are control and data flow, concurrency, distribution, error handling, etc. CxOne does not provide materials for key issues at this time.

2.4 Strategies and Methods

Materials in this area support specific proven strategies, techniques, methodologies, processes, tools or best practices that are directly related to design activities. This is where issues specific to design approaches such as object oriented vs. functional design are supported.

3 Structure

3.1 Design Levels

There are potentially many levels of design abstraction as software specifications are transformed into a form that may be directly constructed. The terms *architecture* and *detailed design* are commonly used to differentiate higher and lower levels of design abstraction. Architecture has reasonably consistent use as a term describing top-level design. Detailed design is used less consistently. It is often applied to just about anything that is not considered architecture, and is a very general term to some while being a very specific term for others.

3.1.1 CxOne Design Levels

To clarify terminology, CxOne defines four major levels of design abstraction. These levels are a way of breaking design activity and artifacts down by intent, which is separate from breakdowns based on design views, techniques, etc. They are described below and shown in figure 1-1 from the previous section.

Architecture – Top level design, often focused on decomposing a system into subsystems and describing relationships between those subsystems. Should also describe global design issues such as environment, error handling, etc. See *CxCheck_Architecture* under the Architecture topic for items an architecture should cover.

High Level Design (HLD) – High-level design defines the behavior, internal structure, data flow, key algorithms, states, interfaces, etc. of sub-systems or other components called out in the architecture or in other HLDs. Some projects have multiple HLD levels, often organized as a hierarchy under the architecture. See *CxCheck_HighLevelDesign* under the Structure topic for items a high level design should cover.

Low Level Design (LLD) – Low-level design defines details necessary to directly support construction activities. Low-level design is performed on a per unit basis, with a unit being the lowest reasonable level of a design (e.g., for OO designs a unit of design often covers a class, for functional design a unit covers a module of related functions and/or data). See *CxCheck_LowLevelDesign* under the Structure topic for items a low level design should cover.

User Interface Design – User interface design is a specialized level that encompasses the details necessary to translate requirements into a human-machine interface. Unlike other design levels it is not focused on the internals of the system, but rather the interface between the internals and humans. This design level is often done implicitly as part of requirements. Some projects explicitly define user interface design as part of requirements through the use of a detailed functional specification. CxOne identifies this activity with design because the focus is on "how" vs. "what". Many times the "how" and "what" of user interface definition can be successfully combined, so projects should structure this activity to best meet their needs. See *CxCheck_UserInterfaceDesign* under the Structure topic for items that should be addressed during user interface design.

3.1.2 Design Level Vernacular

When used in CxOne the term **detailed design** encompasses HLD and LLD. When used in industry settings the term sometimes refers only LLD. HLD is often lumped together with **architecture** or is simply called **design**. The more precisely defined design levels of CxOne are useful abstractions that allow description of common classes of design activity. CxOne does not prescribe these design levels; they are simply a framework for organizing levels of design abstraction. These levels may be split into many specific flavors or hierarchies depending on a project's nature and size.