Lecture 12: Introduction to Specifications

Kenneth M. Anderson

Foundations of Software Engineering

CSCI 5828 - Spring Semester, 1999

Today's Lecture

- Introduction to Specifications
  - – Present an extended example

Specification

- Records Results of a Creative Activity
  - – Requirements, design, code
  - – Project plan, test plan, configuration
- Agreement Among Parties to a Service
  - – Clients and vendors; vendors and engineers
  - – Note: relative and subjective

Specification Qualities

- Clear, Unambiguous, and Understandable

  Are these self-contradictory?
- Consistent

  How can we check this?
- Internally and Externally Complete

  Does completeness reduce understandability?

  What about normal vs. exceptional behavior?

Specifications are Software

- Have a *Lifecycle*
- Should be *Modular*
- Come in *Versions*
- Exhibit *Dependencies*

Specifications Can Be Wrong

- Need to Validate and Verify (V&V)
- V&V is a “W.R.T.” Activity
  - Implies existence of another specification
  - But how do we V&V that other specification?
- Human Holds the Ultimate Specification

Getting Specifications Right

- (Reusable) Layers and/or Modules Help

  A confidence game
- Formality Helps

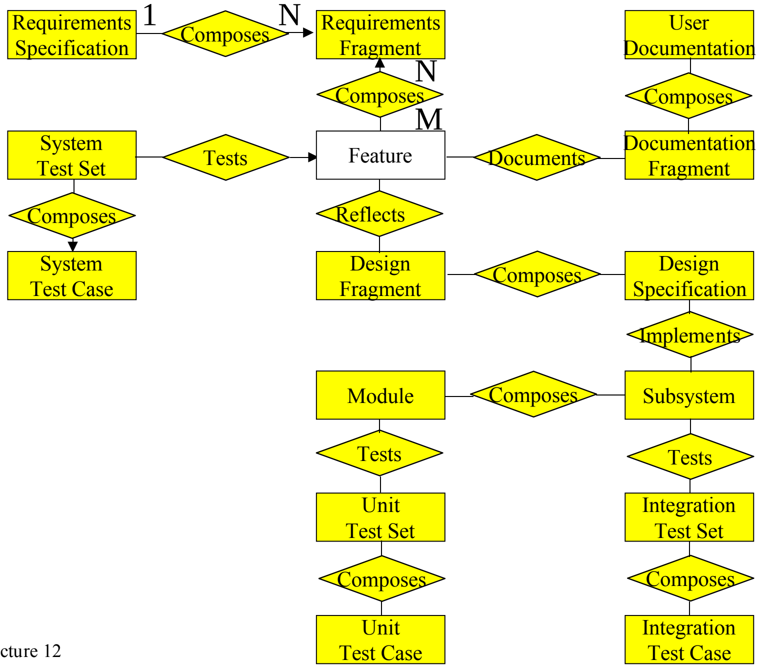
  For example, declarations vs. uses

Specification Notations

- Key to Qualities
- Affect V&V Options
- Most are Equivalent in *Expressive Power*
- Differ in *Expressive Convenience*

Specification/Modeling Styles

- Operational (or Imperative)
  - Described according to desired *actions*
  - Usually given in terms of an *execution model*
- Descriptive (or Declarative)
  - Described according to desired *properties*
  - Usually given in terms of *axioms* or *algebras*
- Structural (or Relational)
  - Described according to desired *relationships*
  - Usually given in terms of *multi/hyper graphs*



An Informal Specification

- A system consists of a set of object files. Each object file is derived from one or more source files. Object and source files have a timestamp indicating when they were last modified. If an object file is older than any source file, then the object file must be rederived.

Make Specification Language

- Hybrid Declarative/Imperative/Relational
- Dependencies are Relational
- Rules are Declarative
- Actions are Imperative