Executable Specifications

Jim Fawcett

CSE687 – Object Oriented Design

Software Development Process

- Decomposition in application space
  - Understand customer requirements, needs
  - Partition into application objects
  - Develop user views
  - Think about usability, esthetics, extensions, critical issues
- Recomposition in solution space
  - Define and develop solution objects
  - Aggregate and package
  - Test behavior and packaging
  - Think about simplicity, maintainability, performance

Architecture and Detailed Design

- System architecture can be loosely associated with the application space
  - User interactions with the system
  - Partitioning into subsystems and application objects
  - Possible and planned extensions to delivered product
- Detailed design focuses on the solution space
  - Building reusable objects
  - Salvaging existing code
  - Integration and testing

Specifications

- Specifications describe:
  - What will be built
  - How the system and built parts behave
  - <u>Not</u> how it is designed and <u>not</u> how it is implemented
  - Specifications are the only basis for testing
- Problem:
  - We write specifications for the application space
  - We don’t traditionally write specifications for the solution space
  - So what is our basis for testing solutions?

Executable Specifications

- An **executable specification** is code for a test driver
  - Prologue that describes, in comments:
    - Designer, platform, and project
    - Tested code, e.g., versioned file names
    - Test description
      - Name, build process
      - Expected behavior, and performance
    - Test procedure
      - Steps the driver will execute and expected results – essentially pseudocode
  - A main function and possibly other functions that implement the test procedure

Implementation

- Interface defines contracts for:
  - Logging specification
  - Test execution
- Executable Specification Implements interface
  - Holds specification text as an embedded resource
  - Uses logger and test vector generator to implement test
- Each test is packaged as a DLL for test execution

Incremental System Development

- Architecture
  - Define application subsystems and objects
  - Write executable test descriptions as a semi-formal specification (only spec we will use)
- Detailed Design
  - Define solution side objects
  - Write executable test descriptions, e.g., specification of solution
  - Write solution-side production code and complete executable specifications concurrently
  - Test solutions and iterate
- System Test
  - Write application-side production code and complete executable specifications concurrently
  - Test applications and iterate

Tools

- Test harness for running automated tests
  - Test vector generator delivers inputs that drive tests
  - Executor loads and executes test libraries (DLLs)
  - Logger records results and supports queries
- Test specification parser
  - Extracts readable test description for specification document
  - Extracts names of classes (with help of static analyzer), objects, and descriptions of behavior for design document
  - Extracts test description, procedure and logger results for test document

Conclusions

- Executable specifications
  - Support specification-driven development
    - Support for both application and solution domains
  - Are compatible with Test-Driven development
  - Are compatible with continuous integration
    - automated daily tests
    - longer cycle regression testing