

# Formal Methods —Contracts

George Huling  
Disciplined Software Consulting  
[g.huling@ieee.org](mailto:g.huling@ieee.org)  
<http://www.laacn.org/firms/dsc/>

# Introduction

- Based on Chapter 14 of Larman: System Behavior—  
Contracts
- Purpose of contracts
- What do contracts depend on
- What are the key parts of a contract

# Purpose

- To ensure the system has all the information (knowledge) necessary for correct future behavior
  - ◆ system information is contained in class instances (objects) and association instances (links)
  - ◆ the attributes of the set of objects
  - ◆ the identities and attributes of the specific objects associated by each link
  - ◆ if there is an association class, the attributes of the link objects

# Contracts Depend On

- Conceptual model (Ch. 9–11)
- System sequence diagrams (Ch. 13)
  - ◆ only enough detail to identify the system events and operations that contribute to the value added of the use case being diagramed
- System events and operations (Ch. 13)
  - ◆ contributing to the **overall** value added of the use case
- System behavior as a “black box”
  - ◆ what (as seen by the user), not how (as seen by the developer)

# Contracts

- What each operation promises to achieve
  - ◆ operation may be the overall effect of a use case
- An operation may have more than one result
  - ◆ there may be more than one way to succeed
  - ◆ there may be one or more ways to fail
- Contracts need to cover all the different results of importance to the users (risk driven)
  - ◆ reveal overlooked operations or results
  - ◆ reveal overlooked attributes

# Key Parts

- Responsibilities — an informal description of the purpose of this operation as seen by the user
- Post-condition — what the system knows after the operation is complete
- Pre-condition — what the system must know or assume before the operation starts
- Inputs from outside system boundary
  - ◆ not mentioned by Larman
- Outputs sent outside the system

# Post-condition

- How did the system knowledge change?
- How is the post operation system knowledge different for each of the possible operation results?
- How do the operation results depend on the pre-condition and the inputs?

# System Knowledge

- objects created or destroyed
  - ◆ to be destroyed something must already exist
  - ◆ to be created something must not already exist
- links created or destroyed
  - ◆ if created, exactly what objects are at the ends
  - ◆ if destroyed, what happens to the objects at the ends
- attributes changed
  - ◆ some attributes cannot change
- What happens when information has been incorrectly entered?

# Operation—Another View

- Post-condition and output: what is the value added for the user — “what’s in it for me”
- Pre-condition: what the system must know to begin the operation
- Input: what additional information must the user give the system to obtain a successful result
- The 4 are related:  
pre-condition + input  $\Rightarrow$  post-condition + output
- Describe the relationship — not how it is obtained

# Larman's *Stage and Curtain View*

- The system and its objects are presented on a stage. The users are the audience.
- Take a picture of the stage before the operation.
- Close the curtain and apply the operation (scene change).
- Open the curtain and take a second picture of the stage.
- Describe the post-condition as the changes in the state of the stage
  - ◆ not how the changes were (or might be) made

# In Later Phases

- The idea of contracts can be applied to subsystems, packages, classes, down to methods
- The context model is always external to the concept for which the contract is being written
- The *stage and curtain* idea always applies
  - ◆ “what”, not “how”
- Contracts trace back to client expectations
  - ◆ system contracts trace to user expectations
  - ◆ method contracts trace to method clients and the class contract

# Pitfalls

- Forgetting about association instances (links)
  - ◆ in relational tables, associations are formed using “foreign keys”
- Using the wrong model for the context
  - ◆ the model is always external to the (sub)system being described
  - ◆ in Ch. 14 the conceptual model is used — the model of the problem domain not the solution domain
- Describing operations in terms of how something is done rather than what the result is