CHAPTER 28 FORMAL METHODS

Overview

This chapter discusses the role of formal methods in software engineering. Formal methods allow software engineers to create specifications using mathematical notation that is more complete, more consistent, and unambiguous. The mathematics used in formal software engineering methods relies heavily on set theory and logic. In many safety critical or mission critical systems, failures can have a high cost. Many safety critical systems cannot be completely tested without endangering the lives of the people they are designed to protect. Use of formal methods reduces the number of specification errors dramatically, which means that the customer will encounter fewer errors when the product is deployed.

Using Formal Methods

- Define the data invariant, state, and operations for each system function
  - data invariant - condition true throughout execution of function that contains a collection of data
  - state - stored data accessed and altered by function
  - operations - system actions that take place when data are read or written to the state (a precondition and postcondition is associated with each operation)
- Specification is represented in some set theoretic type notation from some formal language (e.g., OCL or Z)
- Specification correctness can be verified using mathematical proofs (set operations, logic operations, sequences, induction)

Formal Specification Properties

- Unambiguous - formal syntax used by formal methods has only one interpretation (unlike natural language statements)
- Consistency - ensuring through mathematical proof that initial facts can be mapped (using inference rules) into later statements within the specification
- Completeness - difficult to achieve in a large system even using formal methods

Weaknesses of Less Formal Approaches

- Contradictions - statements do not agree with one another
- Ambiguities - statements have more than one interpretation
- Vagueness - specifications in large documents are often not written precisely enough
- Incompleteness (e.g., failing to list limitations and error handling required of a function)
- Mixed levels of abstraction - occurs when very abstract statements are intermixed randomly with statements written at lower levels of detail)
Necessary Mathematics

- Constructive set specification (also known as set builder notation)
- Set operations (membership, subset, union, intersection, difference, cross product or Cartesian product, powerset)
- Logic operators (and, or, not, implies, universal quantification)
- Sequence properties (order, domain, range)
- Sequence operators (concatenation, head, tail, front, last)

Writing Formal Specifications

- Begin by defining state in terms of abstract items to be manipulated by the function (similar to variable declaration in a programming language)
- Define the data invariant by writing the data relations that will not change during the execution of the function using mathematical notation
- Write the precondition and postcondition for the function using mathematical notation to show the system state before and after function execution

Formal Specification Language Components

- Syntax that defines the specific notation used to represent a specification
- Semantics that help to define the universe of objects that will be used to describe the system
- Set of relations that define the rules that indicate which objects properly satisfy the specification

Object Constraint Language (OCL)

- Notation developed to allow users to add more precision to UML specifications
- OCL is a modeling language that has all the attributes of a formal language
- Typically one begins with a set of UML diagrams (class, state, activity)
- OCL Boolean expressions (constraints) are created for the diagram elements
- OCL constraint expressions involve operators operating on objects
- Any implementation derived from UML model must ensure that each constraint always remains true
- OCL can be used to specify the preconditions and postconditions for operations

Z Specification Language

- Applies typed sets, relations, and functions within the context of first-order predicate logic to build schemas
- Schemas are box-like structures that introduce variables and specify the relationships between these variables
- Schemas are the formal specification analog of a programming language component
• Schemas describe the stored data used to define the state of a system and
describes what data the operations alter to define a new state

Ten Commandments of Formal Methods

1. Choose the appropriate notation
2. Do not over-formalize
3. Estimate costs
4. Have a formal methods guru on call
5. Do not abandon traditional development methods
6. Document sufficiently
7. Do not compromise quality standards
8. Do not be dogmatic in assuming formal specifications are flawless
9. Use of formal methods does not eliminate the need to test products
10. Reuse is still important