SE 422 Advanced Software Engineering

SOFTWARE SPECIFICATIONS

The Role of Sepcifications

- Requirements are seldom communicated accurately
  - Why? Not understood or recorded correctly.
  - Consequences? The development team produces a functioning system that is not correct. There exists a *gap* between the requirements and the implementation.
- Formal Methods provide a foundation for:
  - describing complex systems
  - Reasoning about their behavior

The Role of Sepcifications

- Most application domains are complex
  - Nuclear, avionics, medical, weather, etc.
- Application domains demand *dependable* software
  - Reliable, safe, secure, and correct.
  - Dependability must be understood in the context of environmental interactions –not just conformance to its local system properties.
- Formal Methods intensify the rigor with which requirements are gathered, analyzed and specified.

Complexity

- Complex systems are large (LOC, coupling, people, resources, communication, documentation, processes…)
  - We would like to control complexity
  - We would like to reduce complexity by introducing simplicity in the construction of large systems

Complexity

- Simple System
  - Components can be modeled in a simple way and the interactions are governed by well-defined deterministic rules.
  - The overall behavior becomes predictable to a high degree of accuracy.
- Complex System
  - Components are difficult to model accurately.
  - Interactions amongst components are not governed by well defined rules.
  - Behavior is not predictable.

Complexity Types

- Size
- Structural
- Environmental
- Domain
- Communication

Size Complexity

- The size of a system refers to:
  - The number of components
  - The number of requirements to describe each component
  - The number of interactions between the components
  - The number of quality constraints
  - … many more

Size Complexity

- The behavior of a large system is governed by the behavior of the individual parts as well as the interactions between the parts.
- Large Size → higher complexity

Formal methods help us manage complexities by removing conflicts

Structural Complexity

- Two aspects:
  - Management
    - Each phase in the lifecycle adds new complexity.
    - The organizational structures of teams are reflected in the software.
  - Technical
    - Coupling between modules
    - Packaging

Environmental Complexity

- Environment refers to the physical and logical structure within which the software will operate.
- Software environment refers to the combination of:
  - Operating system
  - Software tools
  - Interfaces
  - Database systems

Environmental Complexity

- The software environment is the sum of:
  - The requirements specification environment
  - The development environment
  - The testing environment
  - The deployment environment
  - Etc.
- To earn the trust of a client, the attributes and constraints of the environment must be taken into consideration when designing the software.

Domain Complexity

- The domain refers to a particular field of Knowledge
  - If the domain is complex (most are) then the software must necessarily also be complex in order to provide reasonable solutions.
  - In this case the software developers cannot remove complexities inherent in the domain, but they can control it.
- Formal methods help minimize complexity

Domain Complexity

- Additionally:
  - Domains can be interrelated
  - Domains can have fuzzy boundaries
  - Knowledge acquisition in a domain will be incomplete. Not all objects are known.

Communication Complexity

- Communication complexities exist at the technology as well as the organizational (people) level.
- Technology
  - Distributed systems
  - Multi-core processors
  - Network protocols
- People
  - Team organizations
  - Different lifecycles between collaborating teams
  - Geographic distribution
  - People may play different roles.

Formal Methods help us model different types of complexities thus reducing the errors made in the requirements analysis of software.

Software Specification

1) A proper specification can control and adequately contain certain types of complexity

2) Without specification software complexity is uncontrollable

What is Specification?

- A statement that describes structural and behavioral details of the software to be developed.
- The software specification must contain:
  - A precise description of the system objects
  - A set of methods to manipulate the objects
  - A statement of their collective behavior for the duration of their existence in the system

Why Specify?

- Because we want to produce software products that "*successfully work in the environment where they are intended to be used*"
- The vast amount of information and data in the target domain is otherwise unmanageable
- Abstraction and decomposition are the most useful tools when specifying.
  - Abstraction : Helps simplify
  - Decomposition : Helps precission

What to Specify?

- For each object:
  - Description
  - Properties
    - Simple or structured. State constraints, invariants.
  - Operations
    - Behaviors
    - Pre and Post conditions.
- For each pair of objects:
  - Interaction rules

Controlling Complexity

- It is not possible to control domain or environmental complexity.
- Size complexities are best approached through:
  - Decomposition. We can partition objects into manageable collections.
  - Reuse. Use existing and well understood software components.
  - Abstraction. Top down functional decomposition produces hierarchies with simpler concepts (abstractions) at the top of the hierarchies.

Controlling Complexity

- Structural complexities are best dealt with:
  - Set theory
  - Predicate logic
  - Relations and function abstraction
- Communication complexity is best dealt by:
  - Understanding organization of teams
  - Precise notations and protocols