SE561 Software System Requirements

Formal Methods

Software Engineering and Formal Methods

- Every software engineering methodology is based on a recommended development process
  - proceeding through several phases:
    - Requirements, Specification, Design
    - Coding, Unit Testing
    - Integration and System Testing, Maintenance

- Formal methods can
  - Be a foundation for designing safety critical systems
  - Be a foundation for describing complex systems
  - Provide support for program development

What are Formal Methods?

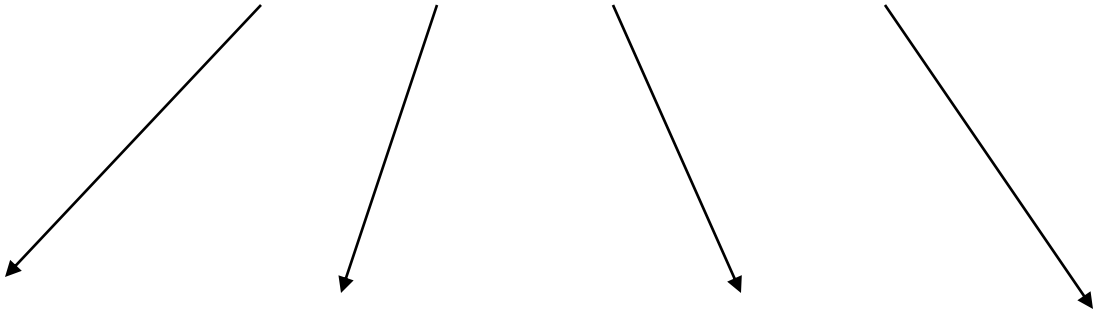
- Techniques and tools based on mathematics and formal logic
- Can assume various forms and levels of rigor
  - Informal
  - Low
  - Medium
  - High

Why Consider Formal Methods?

- The development of a formal specification provides insights and an understanding of the software requirements and software design
  - Clarify customers' requirements
  - Reveal and remove ambiguity, inconsistency and incompleteness
  - Facilitate communication of requirement or design
  - Provides a basis for an elegant software design
  - Traceability
    - System-level requirements should be traceable to subsystems or components

Formal Methods Concepts

Formal Specification Methods



Formal specification

Formal Proofs

Model checking

Abstraction

Formal Specification

- The translation of non-mathematical description (diagrams, table, natural language) into a formal specification language
- It represents a concise description of high-level behavior and properties of a system
- Well-defined language semantics support formal deduction about the specification

Type of Formal Specifications

- Model Oriented: Construct a model of the system behavior using mathematical objects like sets, sequences etc.
  - Statecharts, SCR, VDM, Z
  - Petri Nets, CCS, CSP, Automata theoretic models
- Property Oriented: Use a set of necessary properties to describe system behavior, such as axioms, rules etc.
  - Algebraic semantics
  - Temporal logic models.

Formal Proofs

- Proof is an essential part of specification
- Proofs are constructed as a series of small steps, each of which is justified using a small set of rules
- Proofs can be done manually, but usually constructed with some automated assistance

Model Checking

- A technique relies on building a finite model of a system and checking that a desired property holds in that model
- Two general approaches
  - temporal model checking
  - automaton model checking
- Use model checkers
  - SMV

Abstraction

- Representation of the program using a smaller model
- Allows you to focus on the most important central properties and characteristics
- Getting the right level of abstraction is very important in a specification.

Mathematical Models

- Abstract representations of a system using mathematical entities and concepts
- Model should captures the essential characteristics of the system while ignoring irrelevant details
- Model can be analyzed using mathematical reasoning to prove system properties or derive new behaviors.
- Two types
  - Continuous models
  - Discrete models

Formal Specification Process Model

- Clarify requirements and high level design
- Articulate implicit assumptions
- Identify undocumented or unexpected assumptions
- Expose defects
- Identify exceptions
- Evaluate test coverage

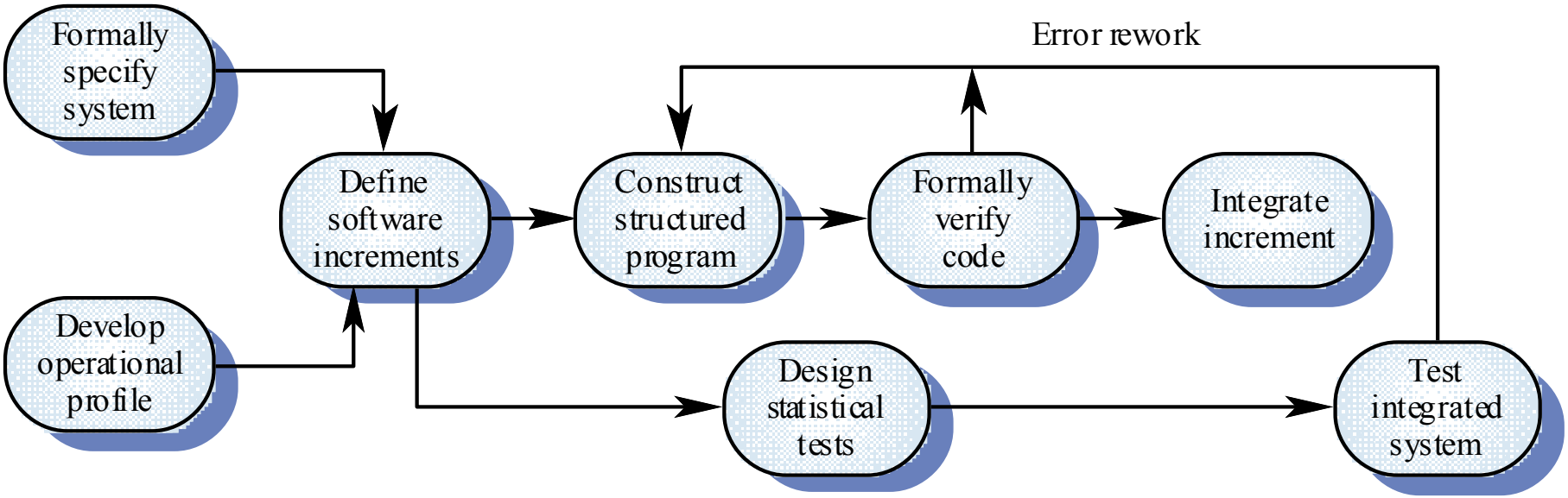
Cleanroom software development

- Spend a lot of effort "up-front" to prevent defects
- Formal specification
- Incremental development
- Statistical methods to ensure reliability

Cleanroom Process

- Formal specification using a state transition model
- Structured programming - limited control and abstraction constructs are used
  - Program resembles state machine
- Static verification using rigorous inspections
  - Mathematical arguments
- Statistical testing of the system reliability

Cleanroom Process



Cleanroom Process

- Incremental development
  - Allows freezing of requirements, so formal work can proceed
  - Work on critical functionality in early revisions, so it receives the most testing

Cleanroom Process

- Specification team.
  - Develop and maintain system specification
- Development team.
  - Develop and verify (mathematically) the software.
  - The software is not executed or even compiled during this process
- Certification team.
  - Develop set of statistical tests to exercise the software after development.
  - Reliability growth models used to determine when reliability is acceptable

Test Results

- Successful in the field
  - Few errors
  - Not more expensive than other processes
- Generally workable
  - Higher quality code resulted

Benefits of Formal Specifications

- Higher level of rigor leads to better problem understanding
- Defects are uncovered that would be missed using traditional specification methods
- Allows earlier defect identification
- Formal specification language semantics allow checks for self-consistency
- Enables the use of formal proofs to establish fundamental system properties and invariants

Limitations to Formal Methods

- Requires a sound mathematical knowledge of the developer
- Different aspects of a design may be represented by different formal specification methods
- Useful for consistency checks, but formal methods cannot guarantee the completeness of a specifications
- For the majority of systems Does not offer significant cost or quality advantages over others

Review

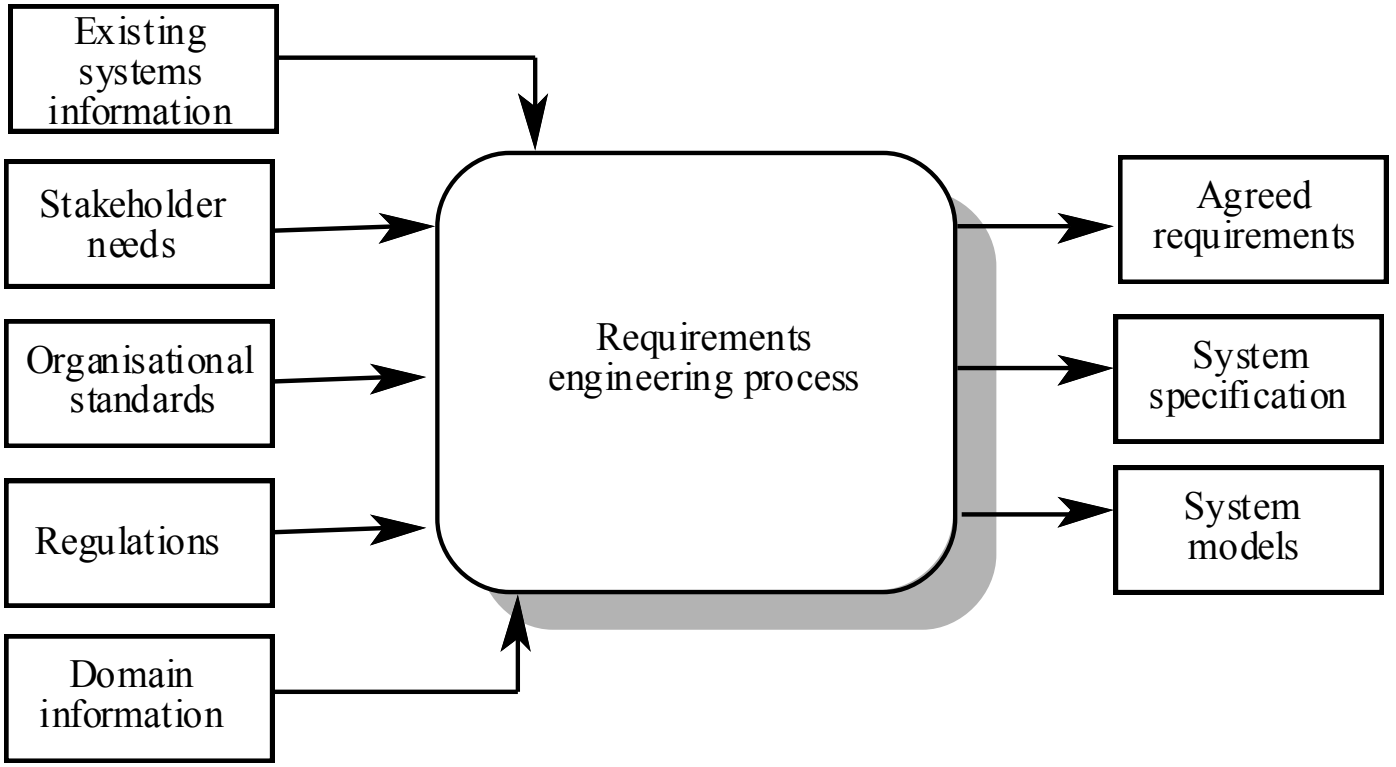
What We learned …

- Fundamental requirements engineering concepts
- Requirements engineering processes
- Requirements engineering techniques

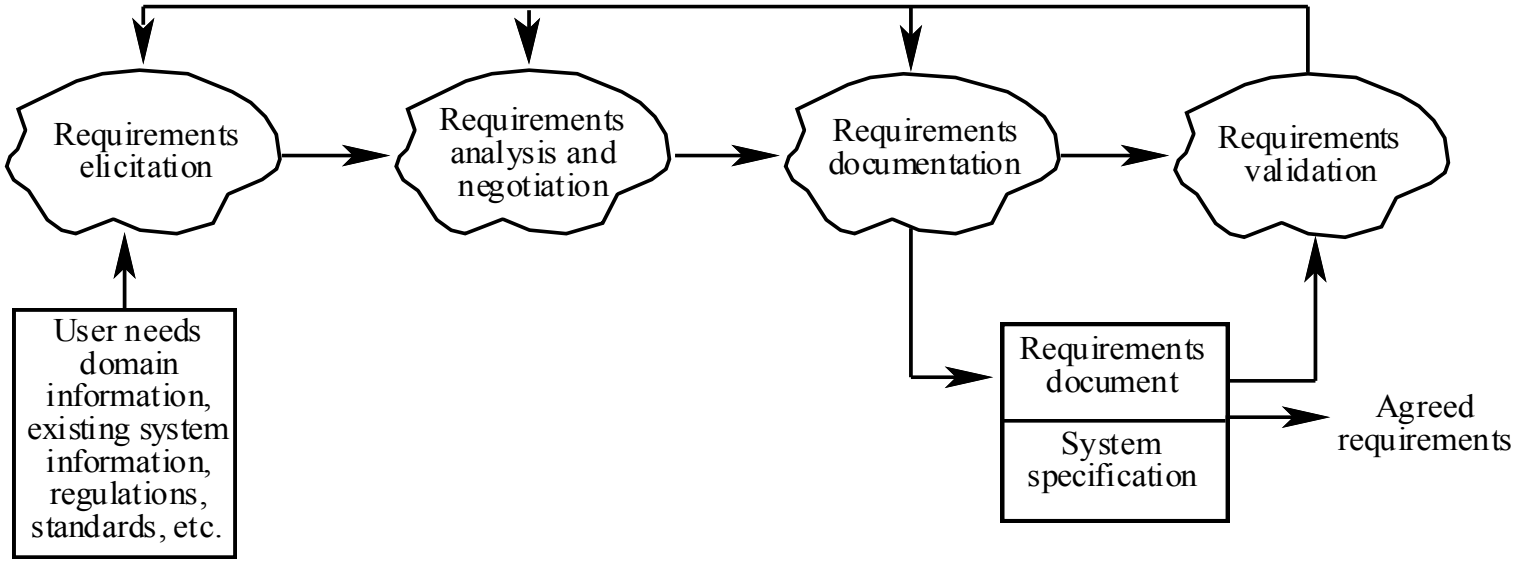
Requirements Engineering Concepts

- Requirements – define what a system is required to do and the constraints under which it is required to operate
- Requirements engineering – all activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system
- The term engineering implies that systematic and repeatable techniques (based on Best Practices) should be used
- The first step in system development
- Include
  - Functional requirements
  - Non-functional requirements
- Stakeholders
  - Software engineers, system end-users, managers of system end-users, external regulators, domain experts

Requirements Engineering Processes



Requirements Engineering Processes



- IBM Rational RequisitPro for requirements documentation and management
- SRS template for final specification

Requirements Engineering Techniques

- Process of requirements engineering (RE) is usually guided by a requirements method
- Requirement methods are systematic ways of producing system models
- System models are important bridges between the analysis and the design process
- Types
  - Structured analysis
  - Object-oriented analysis

Requirements Engineering Techniques

- Data flow modeling
  - One of the most popular structured methods
  - DFD provides a description of a system based on modeling
    - the transformational processes of a system,
    - the collections (stores) of data that the system manipulates, and
    - the flows of data between the processes, stores and the outside world.
  - The DFD describes the functional viewpoint of the system e.g. it describes the system in terms of its operation (tasks).
  - Conducted hierarchically.

Requirements Engineering Techniques

- Object-oriented approach
  - integrate data and functions
  - Use case diagrams
  - Activity diagrams
  - Class diagrams
  - Sequence diagrams
  - Collaboration diagrams
  - State diagrams

Requirements Engineering Techniques

- Non-functional requirements
  - Define the overall qualities or attributes of the resulting system
  - Examples of NFR include safety, security, usability, reliability and performance requirements.
  - Classification
    - Product requirements
    - Process requirements
    - External requirements
  - Derive NFRs
    - Concern decomposition
    - Goal-based
- Formal methods in requirements engineering

Final Exam

- Dec. 18, 1:00 – 3:00, BH 223
- Open book, open notes, no laptop
- One problem on drawing DFD, context level and level 1
- One problem on drawing class diagram and sequence diagrams
- One problem on non-functional requirements
- One problem on formal methods