Introduction to Formal Methods in SE

Getinet Yilma, Ravindra Babu

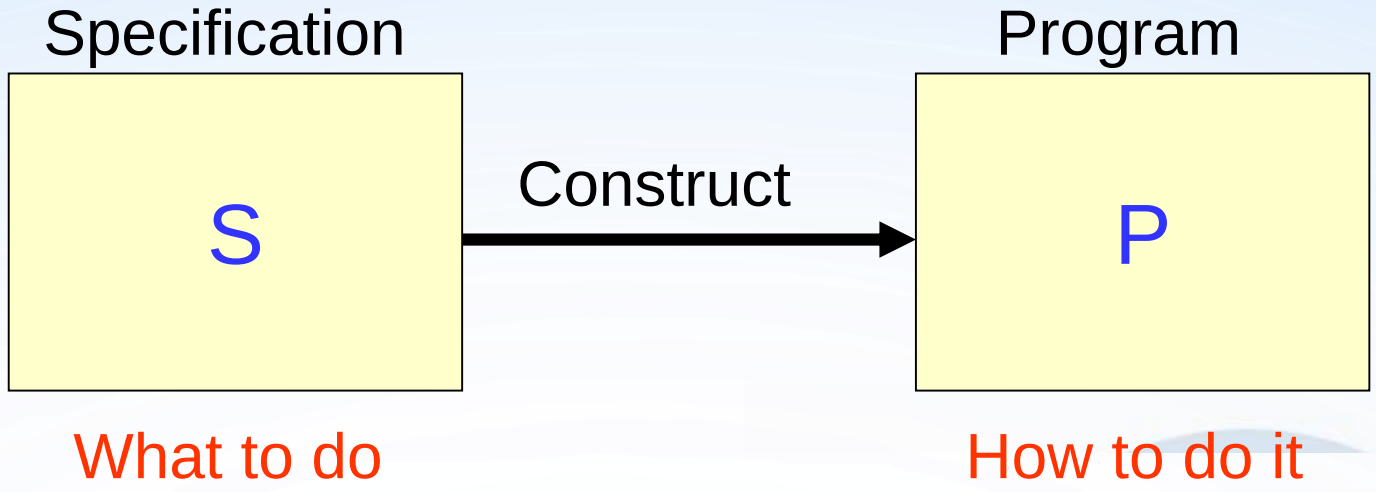
Formal Methods in Software Engineering - SEng3203

Outline

- Introduction to Formal Method
- SDLC
- Formal Method Specification techniques
- Critical Software
- Integrity Level
- Pros and Cons

Introduction to Formal Method

Problems in software development



- How to ensure that S is not ambiguous so that it can be correctly understood by all the people involved?
- How can S be effectively used for inspecting and testing P?
- How can software tools effectively support the analysis of S, transformation from S to P, and verification of P against S?

Introduction to Formal Method

An example of **informal** specification:

"A software system for an Automated Teller Machine (ATM) needs to provide services on various accounts. The services include Transaction on current account, operations on savings account, transferring money between accounts, managing foreign currency account, and change password. The operations on a current or savings account include deposit, withdraw, show balance, and print out transaction records."

Introduction to Formal Method

A better way to write the same specification:

"A software system for an automated teller machine (ATM) needs to provide services on various accounts.

The services include

① Tran on current account
② Tran on savings account
③ transferring money between accounts
④ managing foreign currency account,
⑤ change password.

The Tran on a current or savings account include

① deposit
② withdraw
③ show balance
④ print out transaction records."

Introduction to Formal Method

The major problems with informal specifications:

- Informal specifications are likely to be ambiguous, which is likely to cause misinterpretations.
- Informal specifications are difficult to be used for inspection and testing of programs because of the big gap between the functional descriptions in the specifications and the program structures.
- Informal specifications are difficult to be analyzed for their consistency and validity.
- Information specifications are difficult to be supported by software tools in their analysis, transformation, and management (e.g., search, change, reuse).

What are Formal Methods?

Formal methods for the problems

What is formal methods?

Formal methods = Formal Specification

+

Refinement

+

Formal Verification

Set theory, logics, algebra, etc.

What are Formal Methods?

- ✓ Formal method is branch of software engineering:
  - Ensure a computer system meets its requirements.
  - Help to make system descriptions precise and support system analysis.
  - Used in **safety-critical, business critical, mission critical** application areas.
  - Bring different perspectives to any given variables and functions that could have been hidden behind the natural language.
- ✓ The Encyclopedia of Software Engineering defines formal methods :
  - Employ mathematical techniques for describing system properties within which people can specify, develop, and verify systems in a systematic, rather than ad hoc manner.
- ✓ Formal methods use formal languages such as **Z notation**, **Vienna Development Method** (VDM), **Algebra**, **Functional Programming**, etc.

What are Formal Methods?

- **Definition**

  Mathematical based techniques for the specification, development and verification of software and hardware systems.

  Can determine if:

  – Specifications are satisfied.

  – Implementation of a system is correct.

  – Proving properties of a system.

What are Formal Methods?

- Formal Language Specification Formal Methods centred around a notation known as a formal specification language.

  Formal Semantics

  – Mathematical base allows precise notions.

  – Unambiguous.

  – Allows consistency, correctness, specification and implementation to be expressed.

  Increase Human Understanding of Specified System.

  &

  Allow the possibility of formal reasoning and development.

What is Formal methods ?

- ✓A method (FM) is typically given by a formal specification language. This basis provides a means of precisely defining notions like consistency, completeness, and more relevantly specification, implementation and correctness.
- ✓**Correctness**, the property that an abstract model fulfils a set of well defined requirements.
- ✓**Consistency**, to be consistent, facts stated in one place in a specification should not be contradicted in another place
- ✓FM Used to specify programs, what the system is suppose to do.
- ✓FM is used for constructing programs.
- ✓FM is used to verify the program.

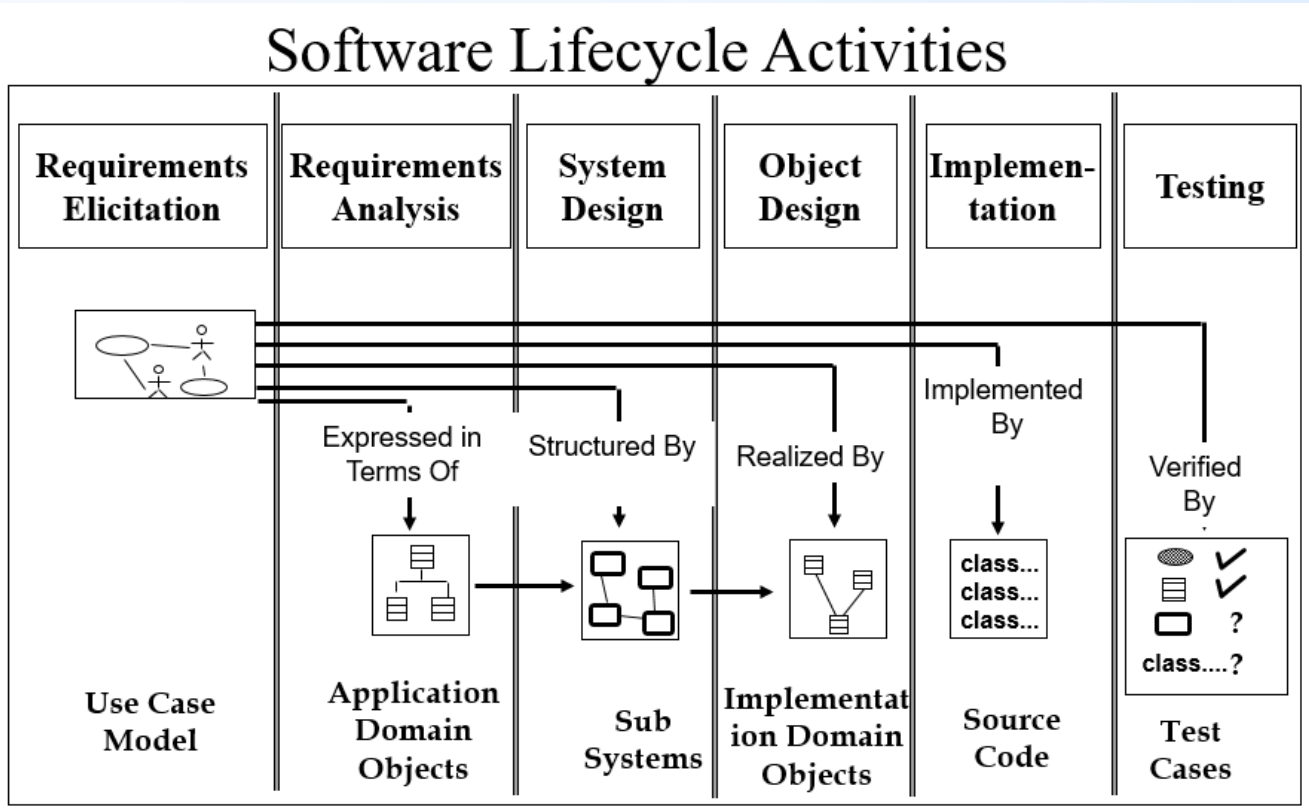
What is Formal methods ?

A Formal Method M consists of three components:

- **Syntax** : Gives a precise description of the form of objects (strings or graphs) belonging to M.
- **Semantics**: Describes the ‘meaning’ of the syntactic objects of M, in general by a mapping into some mathematical structure.
- **Method** : describes algorithmic ways of transforming syntactic objects, in order to gain some insight about them.

Formal Methods in SDLC

- SDLC refers a software process, describing requirement gathering, analysis, design, implementation, deployment, maintenance and support. or
- SDLC can be a process to describe development, deployment, maintenance and reverse engineering of a software product.
- Water fall model in SDLC steps are sequentially ordered
- Water fall is criticized for the completion of one phase to start the next which results in delays.
- Water fall assumes all requirements are known at the beginning in which new requirements has further cost to accommodate



- Validation and Verification at the end of SDLC, an error at later stages may further cause delays.

Formal Methods in SDLC

Three Levels of Formal Methods

LEVEL 0

Formal Specification

1. Requirements Only
2. No Analysis/Proof
3. Cost effective

LEVEL 1

Formal Verification

1. Produce a program in a more formal manner
2. Use proofs of properties or refinements from formal specification
3. Costly

LEVEL 2

Theorem Provers

1. Use theorem prover
2. Fully formal machine-checked proofs.
3. Expensive, hard and often costly
4. Formally prove the entire system.

Why Formal Methods ?

1. History of software encountered notorious (infamous) bugs that were the cause of financial and life losses.

- Therac-25 is a system for computerized radiation therapy machine killed many people due to the controller bug which couldn't stop therapy radiation.
- AT & T long distance breakdown bug 1990 caused by ill placed break statement in the code which stops whole USA network connection for 9 hours. Loses to banks, for ATM and transaction.
- Patriot Missile Failure Gulf war causes the missile hit own American troops due to the software defect in 1991which killed and injured many troops

Why Formal Methods ?

2. The International Space Station (ISS) Practical Byzantine Fault Tolerance consensus algorithm proved correct was found to have two problems using formal method.

ISS Fault Tolerant Computer : – Findings

- ✓Seven deadlock situations were uncovered
- ✓About five live locks were detected in the ISS Fault Tolerant Computer fault management layer

3. Text Processing

Example 2.1: Tool Experiments for Text Processing

What happens if we replace the wildcard sign '*' by the single character 'x'? As original text we take the string 'abc'.

- In Word, the result of replacing '*' by 'x' in 'abc' is 'xxxx';
- in Word, the same result is returned when taking the wildcard symbol '?@';
- in Emacs, replacing '*' by 'x' in 'abc' gives 'abc';
- in Emacs, replacing '.*' by 'x' in 'abc' gives 'x'.

This might come as a surprise.

The problem is that both for Word and Emacs, there is no formal semantics of "replacement of regular expressions". The syntax is correct and the semantics is only formally explained without its implementation

Why Formal Methods ?

4. Software Complexity

– Why has software complexity increased so much?
  - Exponentially increasing computing power.
  - Complexity in solutions to improve life styles.
— Consequences of software complexity
  - +ve is large functionalty e.g. AlphaGo
  - -ve side is complexity increases risk of failure
— Our reliance on complex software

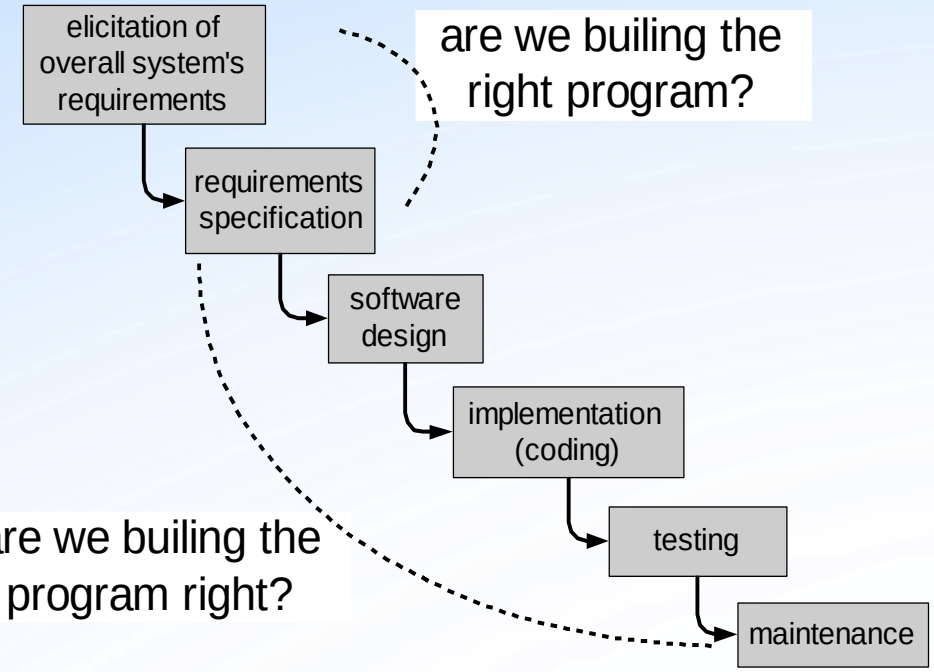
	Lines of code (approx.) (in million)
Operating Systems	
Windows NT 3.1 (1993)	0.5
Windows 95	11
Windows 2000	29
Windows XP (2001)	35
Windows Vista (2007)	50
Windows 7	40
Mac OS X	85
Android OS	12
Automobiles	
1981	0.05
2005	10
2015 (high end)	100
Miscellaneous	
Pacemaker	0.1
Mars Curiosity Rover	5
Firefox	10
Intuit Quickbook	10
Boeing 787	14
Airbus 380	120
F-35 fighter jet	24
Large Hadron Collider	50
Facebook	60
Google (gmail, maps, etc)	2,000

Why Formal Methods ?

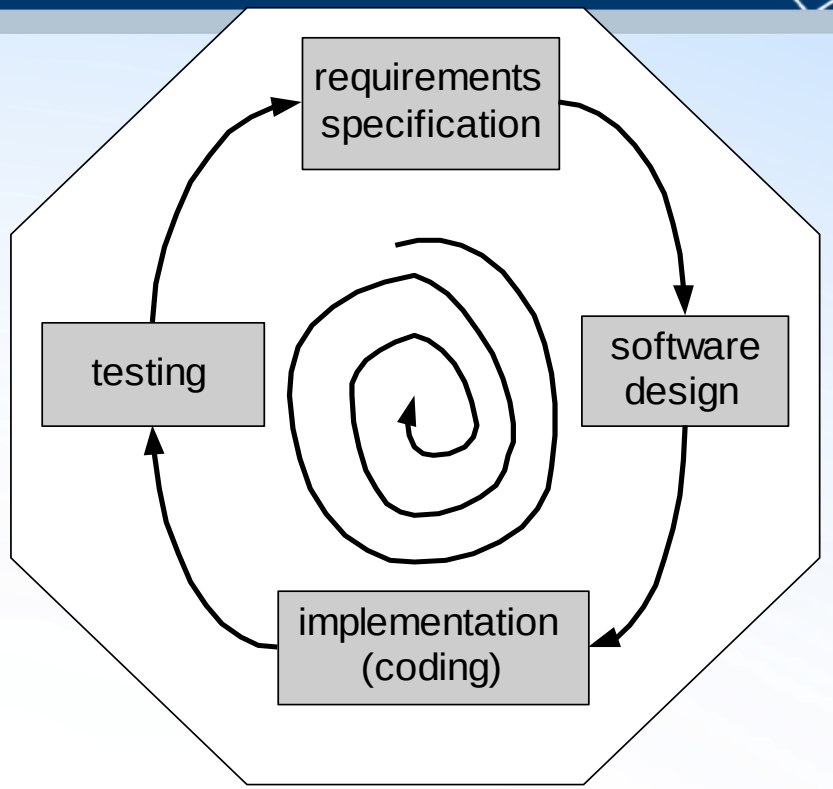
- **How to deal with software complexity** ?
  - Decomposition, abstraction, reuse, automation, testing and analysis→ SE model.
  - Formal methods→ mathematical model.
  - Formal methods→ Abstraction.

- The principal benefits of formal methods are in reducing the number of faults in systems.
- Consequently, their main area of applicability is in critical systems engineering. There have been several successful projects where formal methods have been used in this area.
- In this area, the use of formal methods is most likely to be cost-effective because high system failure costs must be avoided.

Formal Methods in SDLC



Water fall model

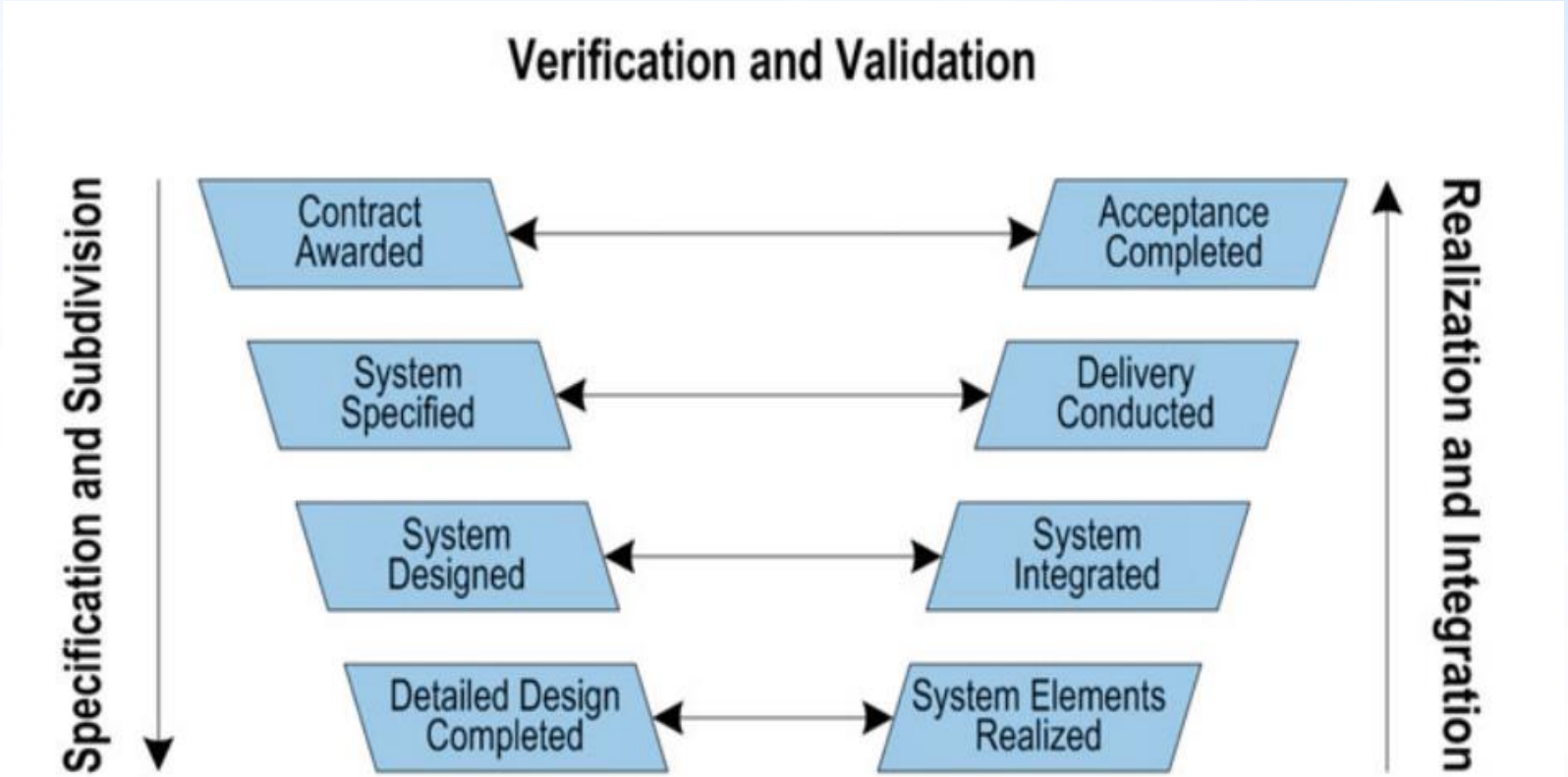


Spiral model

- Single point software testing and validation
- Testing can only verify the presence of software bug but not its absence

Formal Methods in SDLC

- **The V-Model SDLC** instead of describing phases, the V-model describes states in the development process. It refrains from prescribing a specific order to development phases.
- **The V-Model** encourages validation and verification of different aspects of a software earlier.
- **Validation** proves whether users' need is incorporated whereas, **verification** proves if those needs are correctly implemented.



Formal Methods in SDLC

- **Agile SDLC** develop software in a manner where teams are small, the phases are not clearly identified, and the user is represented in the whole process of software development.
- agile is all about speed and flexibility whereas, formal methods is all about correctness and method.
- Agile development follows as test driven development approach at each iteration key to proper **validation and verification.**

Issue	Agile	Formal
Identify the requirements for the software	Y?	Y
Ensure that the software meets these requirements	Y?	Y
Provide usable prototypes rapidly	Y	Y?
Minimise the costs of development	Y?	N?
Ensure that code is high quality	N	Y
Ensure that software is easy to use	N	N
Ensure that changes can be made easily	Y	N
Be easily applied without extensive training	N	N

Formal Methods in SDLC

- ***Formal Methods: When and Where***
- **Requirements Specifications**. Formal methods are a more rigorous form of the specifications systems that software engineers have been developing since the 1960's.
- **Software Testing**. The goal of the formal approach is the same as testing: verifying the proper behavior of a system. While formalists have sometimes claimed that formal methods can replace testing, a more realistic approach is to say that formal verification complements testing.
- **Design Verification**. Ultimately, formal methods are another verification tool. Formal verification is a way of ensuring the correctness of the theory behind the design

Formal Methods in SDLC

- Formal specification is part of a more general collection of techniques that are known as ‘formal methods’.
- These are all based on mathematical representation and analysis of software.
- Formal methods include
  - Formal specification;
  - Specification analysis and proof;
  - Transformational development;
  - Program verification.

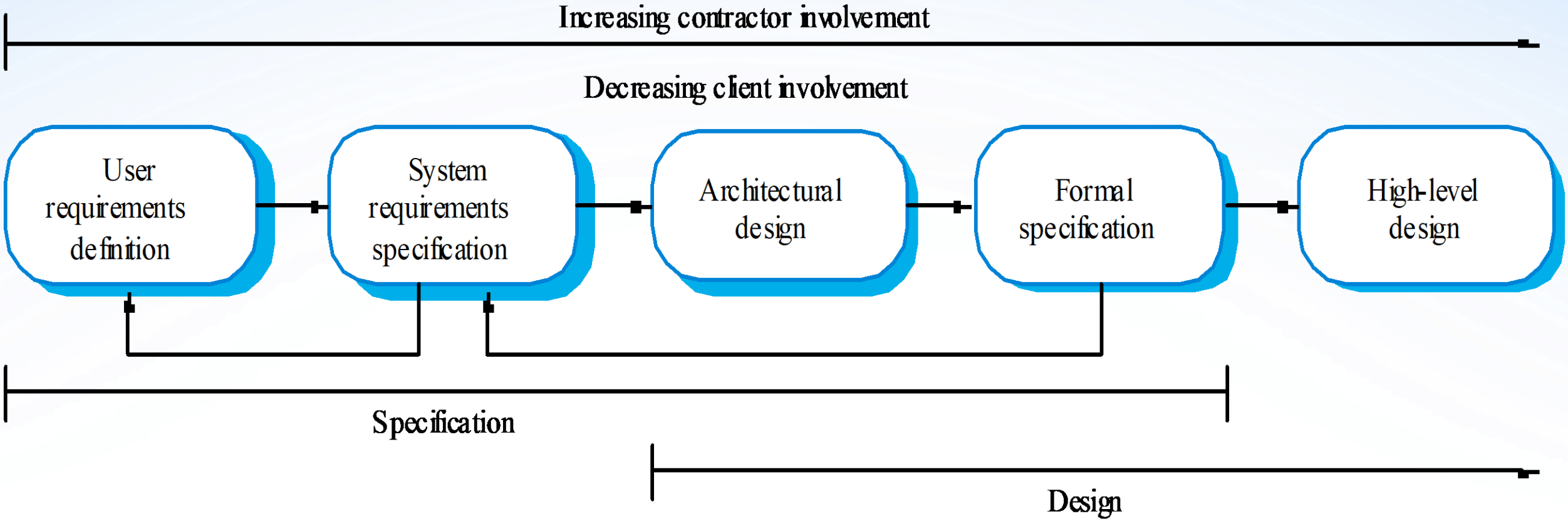
Formal Methods in SDLC

Specification in the software process

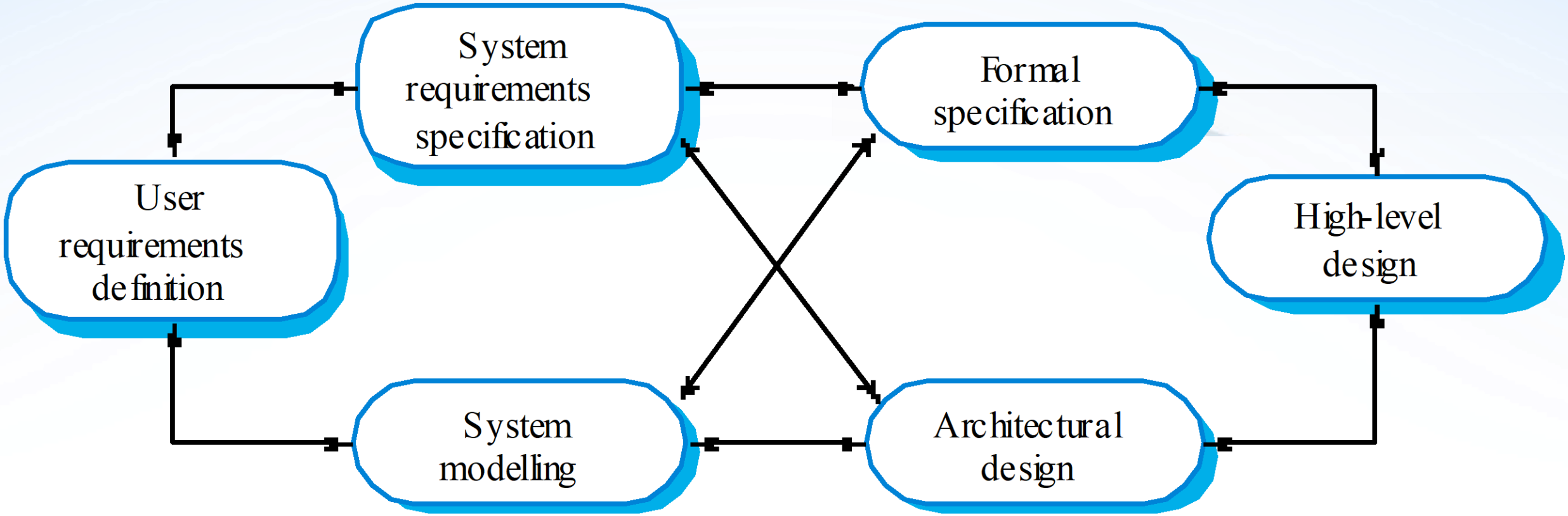
- Specification and design are inextricably intermingled.
- Architectural design is essential to structure a specification and the specification process.
- Formal specifications are expressed in a mathematical notation with precisely defined vocabulary, syntax and semantics.

Formal Methods in SDLC

Specification and design

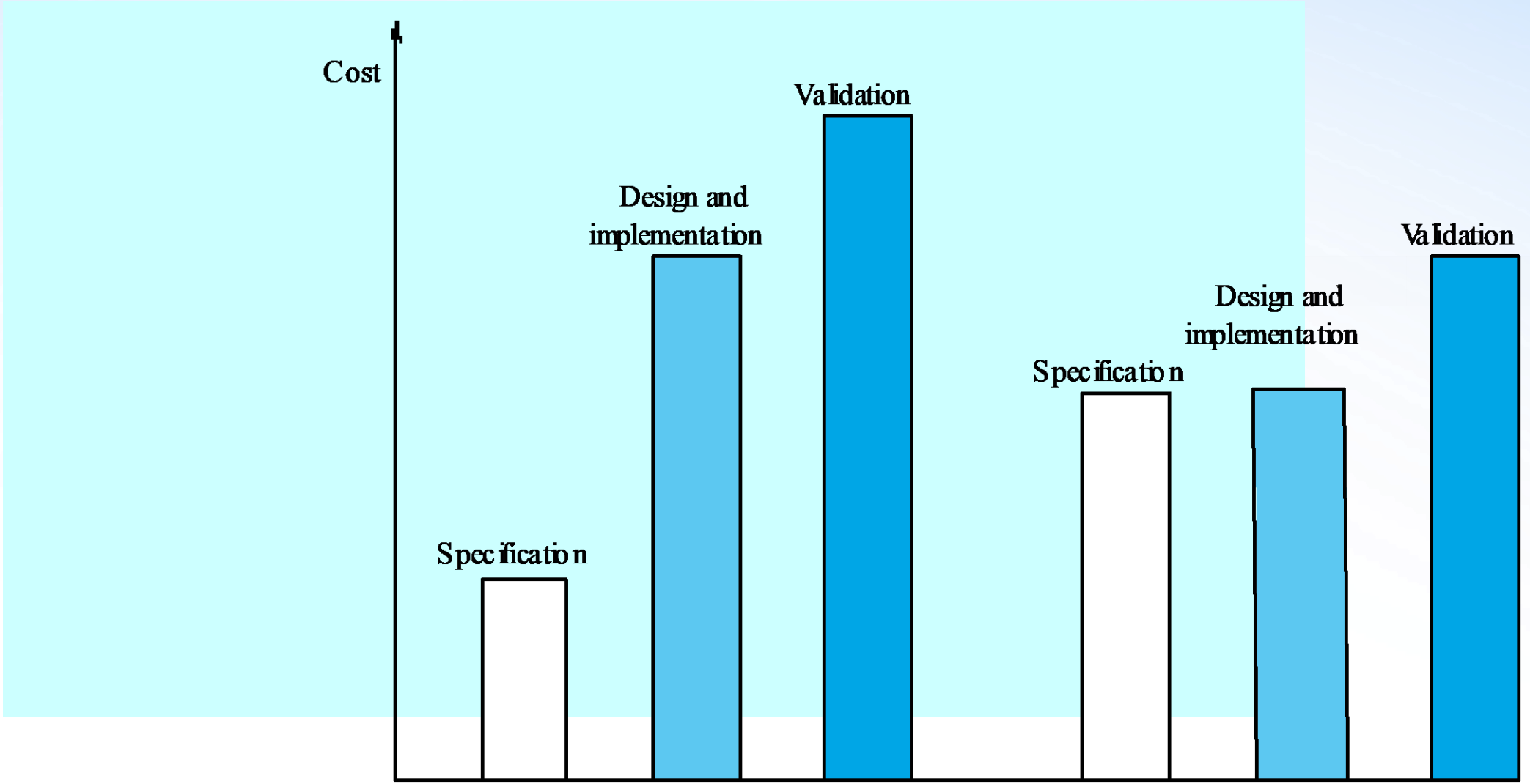


Specification in the software process



Formal Methods in SDLC

Development costs with formal specification in SDLC stages



Formal Method Specification techniques

- Algebraic specification
  - The system is specified in terms of its operations and their relationships.
- Model-based specification
  - The system is specified in terms of a state model that is constructed using mathematical constructs such as sets and sequences. Operations are defined by modifications to the system’s state.

Formal specification languages

	Sequential	Concurrent
Algebraic	Larch (Guttag et al., 1993) }, OBJ (Futatsugi et al., 1985)}	Lotos (Bolognesi and Brinksma, 1987)},
Model-based	Z (Spivey, 1992)} VDM (Jones, 1980)} B (Wordsworth, 1996)}	CSP (Hoare, 1985)} Petri Nets (Peterson, 1981)}

Interface specification

- Large systems are decomposed into subsystems with well-defined interfaces between these subsystems.
- Specification of subsystem interfaces allows independent development of the different subsystems.
- Interfaces may be defined as abstract data types or object classes.
- The algebraic approach to formal specification is particularly well-suited to interface specification as it is focused on the defined operations in an object.

Critical software

- There are basically three (3) different types of critical systems;

1. **Business Critical System** refers to a system where the honesty and integrity of the business is paramount.
   - ✓All data kept in the system must be accurate at all times.
   - ✓If a fault is found the entire process must be stoped to allow correction.
   - ✓ Most government, business and manufacturing company that requires payment are business critical.
2. **Mission Critical System** refers to a system where the continuous running of the system is paramount.
   - ✓ Accuracy takes a lower priority compare to the continuity (running) of the system.
   - ✓Alarm system, ATM, Car Ticketing
3. **Safety Critical System** refers to safety and health of actors. Medical, construction, oil rig system, military systems, nuclear radiation control system and safety critical

Critical software…

- ✓Many organizations today require a combination of the above as such you may have a
  - ✓ **business mission critical system,**
  - ✓**a business safety critical system.**

Integrity Level

- ✓ Integrity level refers to how much cost an organization is willing to spend and how much risk an organization is willing to take when developing software.

Integrity Level	Cost	Risk	Example of System
1	Low	Low	Address Book System
2	Low	High	Global Tsunami Warning System
	High	Low	Waste Water System
3	High	High	Nuclear Reactor System

Integrity Level

- The integrity level determine how deep to go into the Formal Method stage.

Integrity Level	Cost	Risk	Formal Method Stages
1	Low	Low	**Formal Specification**
2	Low	High	**Formal Proof**
	High	Low	**Model Checking**
3	High	High	**Abstraction**

Pros and Cons of formal methods

Pros

- **Measure of correctness**: The use of formal methods provides a measure of the correctness of a system, as opposed to the current process quality measures.
- **Early defect detection**: Formal Methods can be applied to the earliest design artifacts, thereby leading to earlier detection and elimination of design defects.
- **Guarantees of correctness**: Formal analysis tools such as model checkers consider all possible execution paths through the system. If there is any possibility of a fault/error, a model checker will find it. In a multithreaded system where concurrency is an issue, formal analysis can explore all possible interleaving and event orderings. This level of coverage is impossible to achieve through testing.
- **Error Prone**: Formal description forces the writer to ask all sorts of questions that would otherwise be postponed until coding. This helps to reduce the errors that occur during or after coding. Formal methods have the property of completeness, i.e. it covers all aspects of the system.

Pros and Cons of formal methods...

- **Pros**
- **Abstraction**: A formal specification, on the other hand, is a description that is abstract, precise and in some senses complete. The abstraction allows a human reader to understand the big picture of the software product easily.
- **Rigorous Analysis**: Formal descriptions are generally written from different points of view, by which one can determine important properties such as satisfaction of high level requirements or correctness of a proposed design.
- **Trustworthy**: Formal methods provide the kind of evidence that is needed in heavily regulated industries such as aviation. They demonstrate and provide concrete reasons for the trust in the product.
- **Effective Test Cases**: From formal specification, we can systematically derive effective test cases directly from the specification. It’s a cost effective way to generate test cases.

Pros and Cons of formal methods...

Cons

- **Correctness of Specifications**: Generally, actual user requirements might be different from what the user states, and will usually vary with time. While using formal methods, there is no way to guarantee correctness and completeness of a specification with respect to the user's informal requirements.
- **Correctness of Implementation**: It is often impossible to prove the correctness of an existing program that has not been written with the correctness proof in mind. Correctness proofs are only feasible if programming and proof go simultaneously.
- **Correctness of Proofs may fail due to**:
  a) The program is incorrect and needs to be modified.
  b) The program is correct, but the correctness proof has not been found yet.
  c) The program is correct, but there is no correctness proof.
- **Dealing with complex language features**: complex data structures, pointers, human-computer interface (HCI) and error messages are difficult to prove with existing formal methods.

Pros and Cons of formal methods...

Cons

- **The Technical Environment**: A formal description of the program should contain a description that a program is to work in coordination with hardware and under the specification of operating system in order to prove the correctness of the program.
- **Rounding errors** in computations with floating point numbers. These are the reason why formal methods are not usually applied to numerical algorithms.

Pros and Cons of formal methods...

Acceptance of formal methods

- Formal methods have not become mainstream software development techniques as was once predicted
  - Other software engineering techniques have been successful at increasing system quality. Hence the need for formal methods has been reduced;
  - Market changes have made time-to-market rather than software with a low error count the key factor. Formal methods do not reduce time to market;
  - The scope of formal methods is limited. They are not well-suited to specifying and analysing user interfaces and user interaction;
  - Formal methods are still hard to scale up to large systems.

Summary

- Formal Method can detect defects earlier in life cycle.
- Formal Method can be applied at various levels of SDLC
- Formal Method can be integrated within existing project process models
- Formal Method can improve quality assurance when applied to appropriate software projects

Sources for this Slides

- Lecture Notes
- Internet Sources