UTA

Today's Agenda

- Mailing list
- Syllabus
- Introduction

UTA

Introduction

- Software Reliability
- Formal Methods
- Overview of the Course
- Summary

UTA

Software matters …

- Software is one of the most complex man made artifacts.
  - Windows NT has about 16 million lines of code
- Software controls many aspects of our lives
  - Daily life: banking, telephone, emails
  - Safety-critical environments: medical devices, space shuttles, ballistic missiles, nuclear plants
- Software failure can cause severe consequences
  - Loss of investments, environmental damage, and even loss of human life

UTA

Software Reliability

- In safety-critical environment, reliability is a top concern
  - More than 70% effort spent on quality assurance for safety-critical software development
- Priority is also changing for less critical applications
  - As the software industry matures, reliability is replacing features as the distinguishing factor
  - Microsoft is launching a trustworthy computing campaign to improve the reliability of Windows & its applications.

From your perspective …

- Everyone can write software, but not everyone can write "good" software!
  - There is no lack of programmers, but only "good" programmers.
- Software is what you will be building after graduation.
  - Your job performance depends on your ability to write "reliable" software

Introduction

- Software Reliability
- Formal Methods
- Overview of the Course
- Summary

UTA

What are formal methods?

- Formal methods are a collection of notations and techniques for describing and analyzing systems in a rigorous manner.
- In other words, formal methods view programs and their execution as mathematical objects and apply mathematical techniques to specify and analyze their properties and behaviors.

UTA

Why use formal methods?

- force one to think about issues in a systematic way.
  - Leads to better design
  - Earlier detection of inconsistencies and flaws
- remove ambiguity in requirement and design.
  - Makes it possible to automatically reason the properties and behavior of software systems.
  - Provides precise documentation within a team of developers
- help to remove bugs in the implementation

How to use formal methods?

Formal methods can be used in almost every stage of a software development project:

- Requirements can be formally specified to obtain a precise description of the system properties.
- Designs can be formally verified to prevent bugs in them from entering implementations.
- Test cases can be generated from formal spec. and design.
- Formal spec. and design can also be used to facilitate software maintenance.

Misconceptions

- Formal methods can be used only by mathematicians
- Using formal methods will slow down projects
- The verification process itself is prone to errors, so why bother at all?
- Testing is the only method used in practice.

UTA

The truth

- Formal methods are based on some math, but the user does not have to care.
- Early discovery of bugs can often speed up the project.
- The use of formal methods can reduce errors, if it cannot eliminate them.
- In many domains, formal verification and testing methods are used together.

UTA

The reality

In reality, the use of formal methods is spotty, due to several factors:

- Formal methods are relatively new.
  - Some of the methods are suggested only recently.
- The benefits of formal methods are often discounted under tight deadlines.
- Research in this area has focused on improving expressiveness and efficiency, while neglecting human interface issues.

UTA

Research questions

- How to minimize the human intervention?
- How to increase the scalability of formal methods?
- Are there good heuristics that work better than standard methods in many practical cases?
- What is the common notation used by software developers? How to integrate it?

UTA

Introduction

- Software Reliability
- Formal Methods
- Overview of the Course
- Summary

The overall structure

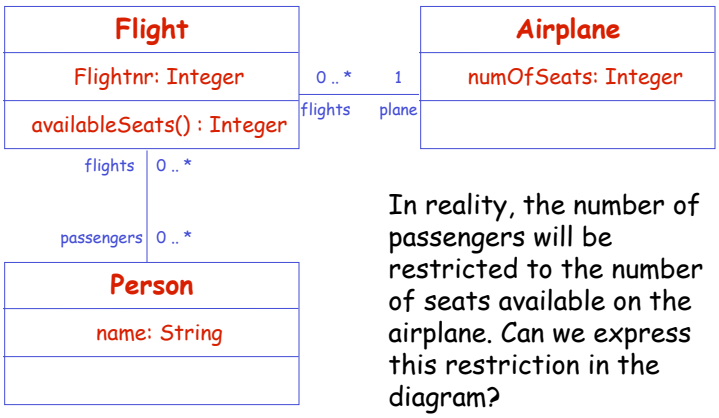
- We will emphasize "strong formal methods", i.e., formal methods with tool-supported semantic analysis.
- Topics covered in this course include Model Driven Architecture, Design by Contract, Temporal Logic, Büchi Automata, Model Checking, and so on.
- These topics are embedded into three modules: The Object Constraint Language (OCL), Java Modeling Language (JML) and The SPIN Model Checker.

OCL (1)

OCL, often used in conjunction with UML, is a language that can add necessary and vital information about the object-oriented models and other artifacts.

One beneficial feature of OCL is that it has a mathematical foundation but maintains the easy of use of natural languages.

OCL (2)



In reality, the number of passengers will be restricted to the number of seats available on the airplane. Can we express this restriction in the diagram?

```
context Flight
inv: passengers -> size () <= plane.numOfSeats
```


JML (1)

JML is a notation for formally specifying the behavior and interfaces of Java classes and methods.

JML specifications can be used to aid reasoning about the correctness of Java programs as well as help debugging and testing of these programs.

UTA

JML (2)

```
package org.jmlspecs.samples.jmlrefman;
public abstract class IntHeap {
//@ public model non_null int [] elements;
/*@ public normal_behavior
  @ requires elements.length >= 1;
  @ assignable \nothing;
  @ ensures \result
  @ == (\max int j;
  @ 0 <= j && j < elements.length;
  @ elements[j]);
  @*/
public abstract /*@ pure @*/ int largest();
//@ ensures \result == elements.length;
public abstract /*@ pure @*/ int size();
};
```


UTA

SPIN

Model checking involves checking desired properties over a system (or a model of the system) by searching through its state space.

SPIN is one of the most widely used tools for software model checking. It is particularly suited for checking concurrent software.

Introduction

- Software Reliability
- Formal Methods
- Overview of the Course
- Summary

Summary (1)

- Reliability has always been a primary concern in a responsible software development process.
- Formal methods use logical and mathematical techniques to specify and analyze the behavior of software systems.
- Formal methods can be used to improve software development at all stages, including requirements analysis, design, coding, testing and maintenance.

UTA

Summary (2)

- This course covers formal specification and verification, both at the design and coding level.
- The topics of this course are embedded into three modules: (1) OCL; (2) JML ; (3) SPIN.