

EE382C-3 Verification and Validation

khurshid@ece.utexas.edu

Lecture 1
January 14, 2025

Today

Introductions

Basics

ChatGPT

Next time

Alloy [<http://alloytools.org/>]

Introductions: this course

Advanced course on software testing and verification

- Based on research papers and tools
- Systematic, organized approaches for finding and eliminating “bugs”

Problem sets, quizzes, project (proposal, progress and final reports, presentation)

Introductions: teaching team

Instructor

- Sarfraz Khurshid <khurshid@ece.utexas.edu>
- Office hours: Wed 11:00—noon or by appointment
- Location: EER 7.880

TA

- TBD

Introductions: you

Programming?

Testing/verification?

Research?

ChatGPT?

Administrative info

Lectures: TTh 11:00-12:30pm in ECJ 1.306

Prerequisites: graduate standing

- Knowledge of data structures and object-oriented languages
- Programming experience

Evaluation

Project (40%)

Exams (30%) – **Exam 1 on Feb 11** and
Exam 2 on Mar 11

Homeworks (20%)

Class participation (10%)

- Present a research paper in class and lead its discussion
- Actively engage in discussions

Project

Research oriented or significant design / implementation (or both)

Done in teams of 2 or 3 students

Three reports

- Report 1 (due: Feb 25)
 - Members + proposal + related work
- Report 2 (due: Mar 25): initial results
- Report 3 (due: Apr 25)
 - Complete report with final results + conclusions

Presentation (last 3 classes): ~15 mins / team

Collaboration

No communication/texts during exams

You must individually write solutions for problem sets

You can:

- Discuss problem sets
- Collaborate among different project groups

Testing benefits from good communication skills

Textbooks – not required

Software Abstractions: Logic, Language, and Analysis by Daniel Jackson. ISBN: 0262017156

Introduction to Software Testing by Paul Ammann and Jeff Offutt. ISBN: 978-1107172012

Model Checking by Edmund M. Clarke, Orna Grumberg and Doron A. Peled. ISBN: 0262032708

Canvas

courses.utexas.edu

Course web-page

Slides

Handouts

Discussions

Problem sets

...

Focus: Bugs

Etymology [Wikipedia]

- Perhaps term was first used in hardware engineering for mechanical failures, e.g.,
“It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise — this thing gives out and [it is] then that "Bugs" — as such little faults and difficulties are called — show themselves and months of intense watching, study and labor are requisite before commercial success or failure is certainly reached.”

Thomas Edison, 1878

Warranties – 3 decades ago

ACM SIGSOFT Software Engineering Notes, Vol.
12, No. 3, July 87

Cosmotronic Software Unlimited Inc. does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error-free.

However, Cosmotronic Software Unlimited Inc. warrants the diskette(s) on which the program is furnished to be of black color and square shape under normal use for a period of ninety (90) days from the date of purchase.

NOTE: IN NO EVENT WILL COSMOTRONIC SOFTWARE UNLIMITED OR ITS DISTRIBUTORS AND THEIR DEALERS BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFIT, LOST SAVINGS, LOST PATIENCE OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES.

Warranties – 3 decades ago

ACM SIGSOFT Software Engineering Notes, Vol.
12, No. 3, July 87

We don't claim Interactive EasyFlow is good for anything--if you think it is, great, but it's up to you to decide. If Interactive EasyFlow doesn't work: tough. If you lose a million because Interactive EasyFlow messes up, it's you that's out the million, not us. If you don't like this disclaimer: tough. We reserve the right to do the absolute minimum provided by law, up to and including nothing.

This is basically the same disclaimer that comes with all software packages, but ours is in plain English and theirs is in legalese.

We didn't really want to include any disclaimer at all, but our lawyers insisted. We tried to ignore them but they threatened us with the attack shark at which point we relented.

Warranties – more recent

“**Apple** warrants the media on which the Apple Software is recorded and delivered by Apple to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of original retail purchase.”

“... THE **APPLE** SOFTWARE IS PROVIDED "AS IS", WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND ...”

Google: “... is provided "as is," with no warranties whatsoever.”

Microsoft: “... the Software will perform substantially in accordance with the accompanying materials for a period of ninety (90) days ...”

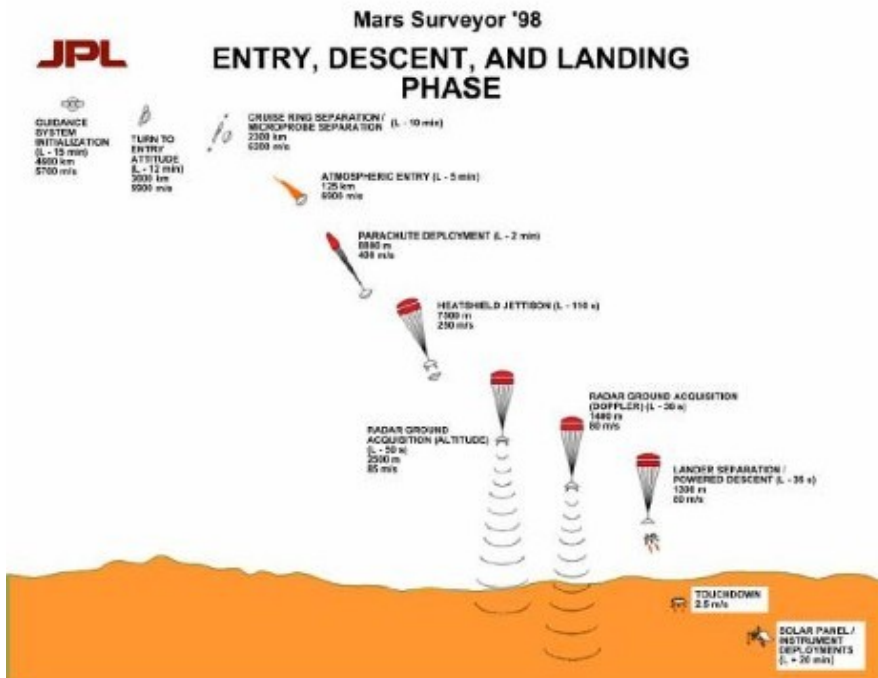
Ariane-5, 1996

Went off-course 37 sec into flight sequence – crashed



Mars Polar Lander, 1999

Premature shut down at 40 meters altitude – crashed



Photos: JPL/NASA

USS Yorktown, 1997

“Dead in the water” for 3 hours



Photo: navsource.org

Economic impact

NIST Report: “The Economic Impact of Inadequate Infrastructure for Software Testing” (May 2002)

- \$59.5B – Annual cost of inadequate software testing infrastructure
- \$22.2B – Annual potential cost reduction from feasible infrastructure improvements

Finding/fixing bugs cost >300B USD annually
[Cambridge'13]

Terminology

Anomaly

Bug

Crash

Defect

Error, exception

Failure, fault, flaw, freeze

Glitch

Hole

Issue

...

“Bugs” in IEEE 610.12-1990

Fault – incorrect lines of code

Error – faults cause incorrect (unobserved) state

Failure – errors cause incorrect (observed) behavior

These terms are not always used consistently in literature

- But we'll (try to) follow these standard definitions

Correctness

Expected correctness properties have two basic forms

- Common properties that we expect of a large class of programs
 - No segfaults, core dumps, deadlocks, memory leaks, etc.
- Specific properties that we expect of the specific program under test
 - May be derived from requirements or specs
 - E.g., method under test sorts correctly

Approaches to tackle bugs

Software testing

Model checking

(Static) program analysis

Software testing (1)

Conceptually, testing is simple:

- Create tests, run them, and check for failures

In practice, testing is ad hoc and expensive

Creating tests is a crucial and typically labor intensive part of testing

- Automation can help!

Software testing (2)

Testing is a **dynamic** approach for finding bugs

- Checks correctness for some – typically a small number compared to total number of – executions
- No test failures does not imply no bugs in code
- In contrast, **static analysis** can prove certain properties for all inputs

Testing is **not** the same as **debugging**, i.e., locating and removing specific faults

Topics in testing

Basic questions

- How to create test inputs?
- How to check outputs?
- How many tests to create?

There are many additional topics

- Selection, minimization, prioritization, augmentation, evaluation, ...

Testing is not just about finding faults!

Testing effort

Reported to be >50% of development cost [eg. Beizer'90]

Microsoft: 75% time spent testing

- 50% testers who spend all time testing
- 50% developers who spend half time testing

Model checking

Typically hybrid dynamic/static approach

Checks correctness for all executions

Some techniques

- Explicit-state model checking
- Symbolic model checking
- Abstraction-based model checking

Static analysis

Static approach

Checks correctness for all executions

Some techniques

- Abstract interpretation
- Dataflow analysis
- Verification-condition generation

Soundness / completeness

Do we detect all bugs?

- Impossible for dynamic analysis

Are reported bugs real bugs?

- Easy for dynamic analysis

Most practical techniques and tools are both unsound and incomplete

- Give false positives / negatives
- Combine static and dynamic analyses

Current status

Testing remains the most widely used approach for bug finding

- **Validation**: are we building the right system?
- **Verification**: are we building the system right?

A lot of progress (especially during last 25 years) on model checking and static analysis

- Model checking: from hardware to software
- Static analysis: from sound to practical

Vibrant research in the area

Gap between research and practice

What is ChatGPT?

SA

Hello! Can you tell me what is ChatGPT?



Hello! Of course, I'd be happy to explain.

ChatGPT is a language model developed by OpenAI, and it's based on the GPT (Generative Pre-trained Transformer) architecture. The "GPT" part of the name stands for "Generative Pre-trained Transformer," which describes its key features:

1. **Generative:** It can generate text based on the input it receives. This means it can be used to write coherent and contextually relevant paragraphs of text.
2. **Pre-trained:** Before being fine-tuned for specific tasks, models like ChatGPT are trained on massive amounts of text data from the internet. This training helps the model learn grammar, facts, reasoning abilities, and some level of common sense understanding.

3. **Transformer:** The "Transformer" architecture is a type of neural network architecture specifically designed for handling sequential data, like text. It has proven highly effective for tasks involving natural language processing (NLP).

ChatGPT is designed for generating human-like text in a conversational manner. It can understand and generate responses to prompts, questions, and other forms of input text. It's often used for tasks like text completion, content generation, answering questions, and even simulating conversations.

It's important to note that while ChatGPT can produce impressive and coherent text, it doesn't possess real understanding, consciousness, or awareness. It operates based on patterns learned during its training, and its responses are generated based on statistical associations in the data it has seen.

https://chat.openai.com/

ChatGPT

Tips for getting started

Ask away

ChatGPT can answer questions, help you learn, write code, brainstorm together, and much more.

Don't share sensitive info

Chat history may be reviewed or used to improve our services. Learn more about your choices in our [Help Center](#).

Check your facts

While we have safeguards, ChatGPT may give you inaccurate information. It's not intended to give advice.

Okay, let's go

How did you use ChatGPT?

Compose an email

Write an essay

Implement a code snippet

Play a game

Study a new topic

Have a casual conversation

Seek answers to tough questions in life

...

How about we test ChatGPT?!

We are in a testing class after all :-)

But testing a system like ChatGPT is hard

- What does its correctness even mean?
- How to define useful questions to test it?
- How to check answers it gives?

Still, let's ask it some questions and see if we find some “bugs”!

?/!