Software Verification

Bertrand Meyer
Carlo A. Furia
Sebastian Nanz

ETH Zürich, Autumn 2015
Today

Aims of the course

Introduction to issues of software quality
Course organization

Lecturers: Bertrand Meyer, Carlo Furia, Sebastian Nanz
Assistant: Chris Poskitt

Webpage: [http://se.inf.ethz.ch/courses/2015b_fall/sv/](http://se.inf.ethz.ch/courses/2015b_fall/sv/)

Monday lectures

10-12, RZ F21  Classical lecture

Wednesday lecture (14-15, RZ F21):

Variable slot: seminar by guest, or extra lecture

Exercise session: Wednesday, 15-17, RZ F21
Purpose of this course

To present available techniques for ensuring better software quality
<table>
<thead>
<tr>
<th>Topics</th>
<th>Subtopics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiomatic semantics</td>
<td>Program proofs</td>
</tr>
<tr>
<td>Auto-active verification</td>
<td>Program analysis</td>
</tr>
<tr>
<td>Separation logic</td>
<td></td>
</tr>
<tr>
<td>Theory of Programs</td>
<td></td>
</tr>
<tr>
<td>Static analysis</td>
<td></td>
</tr>
<tr>
<td>Abstract interpretation</td>
<td></td>
</tr>
<tr>
<td>Model checking</td>
<td></td>
</tr>
<tr>
<td>Real-time systems</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
</tr>
</tbody>
</table>
Guest lecturers

**Chris Poskitt** (auto-active verification; separation logic)

**Scott West, Google** (topic TBA)

**Alexey Kolesnichenko** (AutoTest)

**Max (Yu) Pei** (AutoFix)
Grading

Project: 30%
Written exam (Monday 14th December): 70%

All material considered during regular lecture slots and exercise classes is examinable
Overview of software verification
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)

If you'd like to know more, you can search online later for this error: HAL_INITIALIZATION_FAILED
A set of policies and activities to:

- Define quality objectives
- Help ensure that software products and processes meet these objectives
- Assess to what extent they do
- Improve them over time
Verification

The Quality Assurance activity devoted to enforcing quality, in particular:

- Detecting deviations from quality
- Correcting them

Common distinction (“V & V”):

- **Validation**: assessment of any product relative to its specification (“checking that it is doing the right things”)
- **Verification**: assessment of internal quality (“checking that it is doing things right”)

In this course, “Verification” covers both
The product side

Quality is the absence of “deficiencies” (or “bugs”).

More precise terminology (IEEE):

Mistakes  
caused by  
Faults  
result from  
Failures
What is a failure?

For this discussion, a failure is any event of system execution that violates a stated quality objective.
Verification techniques

A priori techniques

- Build system for quality; e.g.: process approaches, proof-guided construction, Design by Contract

A posteriori techniques

- Static: from software text only
  - Program proofs
  - Program analysis / abstract interpretation
  - Model checking
- Dynamic: execute software
  - Testing
Software quality: external vs internal

External factors: visible to customers

(not just end users but e.g. purchasers)

- *Examples*: ease of use, extendibility, timeliness

Internal factors: perceptible only to developers

- *Examples*: good programming style, information hiding, documentation

Only external factors count in the end, but the internal factors make it possible to obtain them.
Software quality: product vs process

Product: properties of the resulting software

For example: correctness, efficiency

Process: properties of the procedures used to produce and “maintain” the software
Some external factors

Product quality (immediate):
- Reliability
- Efficiency
- Ease of use
- Ease of learning

Process quality:
- Production speed (timeliness)
- Cost-effectiveness
- Predictability
- Reproducibility
- Self-improvement

Product quality (long term):
- Extendibility
- Reusability
- Portability
Correctness:
The systems’ ability to perform according to specification, in cases covered by the specification

Robustness:
The systems’ ability to perform reasonably in cases not covered by the specification

Security:
The systems’ ability to protect itself against hostile use
Financial consequences, on developers and users, of “insufficient testing infrastructure”

$ 59.5 B.$
Software projects according to Standish

- **Successful**
- **Failed**
- **Challenged**

<table>
<thead>
<tr>
<th>Year</th>
<th>Successful</th>
<th>Failed</th>
<th>Challenged</th>
</tr>
</thead>
<tbody>
<tr>
<td>1994</td>
<td>31</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1996</td>
<td>40</td>
<td>27</td>
<td>33</td>
</tr>
<tr>
<td>1998</td>
<td>28</td>
<td>26</td>
<td>46</td>
</tr>
<tr>
<td>2000</td>
<td>28</td>
<td>23</td>
<td>49</td>
</tr>
<tr>
<td>2006</td>
<td>35</td>
<td>19</td>
<td>46</td>
</tr>
</tbody>
</table>
Some famous failures

Ariane 5
Therac
Patriot
London Ambulance System
Mars Orbiter Vehicle
Buffer overflows

...
Mars Climate Orbiter

Mishap Investigation Board

Phase I Report

November 10, 1999
Mars Polar Lander

Launch
- Delta 7425
- Launch 1/3/99
- 576 kg Launch Mass

Cruise
- RCS attitude control
- Four trajectory correction maneuvers,
  Site Adjustment maneuver 9/1/99,
  Contingency maneuver up to Entry – 7 hr.
- 11 Month Cruise
- Near-simultaneous tracking w/ Mars Climate
  Orbiter or MGS
during approach

Entry, Descent, and Landing
- Arrival 12/3/99
- Jettison Cruise Stage
- Microprobes sep. from Cruise Stage
- Hypersonic Entry (6.9 km/s)
- Parachute Descent
- Propulsive Landing
- Descent Imaging [MARDI]

Landed Operations
- 76° S Latitude, 195° W Longitude
- Ls 256 (Southern Spring)
- 60–90 Day Landed Mission
- MVACS, LIDAR Science
- Data relay via Mars Climate
  Orbiter or MGS
- Commanding via Mars
  Climate Orbiter or
direct-to-Earth high-gain antenna
On September 27, 1999, the operations navigation team consulted with the spacecraft engineers to discuss navigation discrepancies regarding velocity change ($\Delta V$) modeling issues. On September 29, 1999, it was discovered that the small forces $\Delta V$'s reported by the spacecraft engineers for use in orbit determination solutions was low by a factor of 4.45 (1 pound force = 4.45 Newtons) because the impulse bit data contained in the AMD file was delivered in lb-sec instead of the specified and expected units of Newton-sec.
Ariane-5 maiden launch, 1996

37 seconds into flight, exception in Ada program not processed; order given to abort mission. Loss estimated to $10 billion.

Exception was caused by an incorrect conversion: a 64-bit real value was incorrectly translated into a 16-bit integer.

Systematic analysis had “proved” that the exception could not occur – the 64-bit value (“horizontal bias” of the flight) was proved to be always representable as a 16-bit integer!

It was a REUSE error:

- The analysis was correct – for Ariane 4!
- The assumption was documented – in a design document!

Security example: the buffer overflow

System expects some input from an external user:

First name: ___________________________

Last name: ___________________________

Address: ______________________________
Getting the input

\[
\begin{align*}
\text{from } i & := 1 \text{ until } i > \text{input\_size} \\
\text{loop} & \\
\text{buffer} [i] & := \text{input} [i] \\
i & := i + 1 \\
\text{end}
\end{align*}
\]
Overflowing a buffer!

The buffer array (overflowing)

Programs

Memory

Max

Data

Main

Routine 1

Routine 2

... 

Routine n

My return address

My nasty code

Code of routine n-1

Return address, arguments, locals

“The stack” (activation records)
Getting the input

from $i := 1$ until $i > \text{input}\_\text{size}$ or $i > \text{buffer}\_\text{size}$

loop

buffer [$i$] := input [$i$]

$i := i + 1$

end
Verification in the software lifecycle
# Quality assurance techniques

## Process
- Manual
- Technology-generic
- Phase-generic
- Product-generic

## Product
- Tool-supported
- Technology-specific
- Phase-specific
  - (analysis, design, implementation...)
- Product-specific
  - (code, documentation...)

### Build (a priori)
- Static
- Informal
- Complete

### Assess (a posteriori)
- Dynamic
- Mathematical
- Partial
Quality assurance throughout the process

“Software” is not just code!

Quality affects code, documentation, design, analysis, management, the software process, and the software quality policy itself.

Most of the techniques presented will, however, be for code.
Process-based approaches to quality assurance

- Lifecycle models
- Inspections
- Open-source process
- eXtreme Programming (XP)
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (0% complete)

If you'd like to know more, you can search online later for this error: HAL_INITIALIZATION_FAILED