Software Verification

Bertrand Meyer
Carlo A. Furia
Sebastian Nanz

ETH Zürich, Fall 2013

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/2013b_fall/sv/

Monday lectures
    10-12, RZ F21, **except** on 18.11 when in IFW C 42
    Classical lecture

Wednesday lecture (15-16, RZ F21):
    Variable slot: seminar by guest, or extra lecture

Exercise session: Wednesday, 16-18, RZ F21

Purpose of this course

To present available techniques for ensuring better software quality

Topics (see Web page for details)

Axiomatic semantics
Separation logic
Assertion inference
Graphical program specifications

Program proofs

Static analysis
Abstract interpretation

Program analysis

Model checking
Real-time systems

Model checking

Testing

Testing

Guest lecturers

Julian Tschannen (AutoProof)

Chris Poskitt (separation logic; graphical program specs)

Nadia Polikarpova (Boogie and Boogaloo)

Đurica Nikolić (static analysis / JULIA)

Yu Pei (AutoTest)

Kaue Soares da Silveira, Google (Testing at Google)

Grading

Project: 30%

Written exam (16 December): 70%

All material considered during <u>regular lecture slots</u> and <u>exercise classes</u> is examinable

A problem has been detected and Windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

Overview of software verification

The more general notion: software quality assurance

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):

caused by → Mistakes

result from → Faults
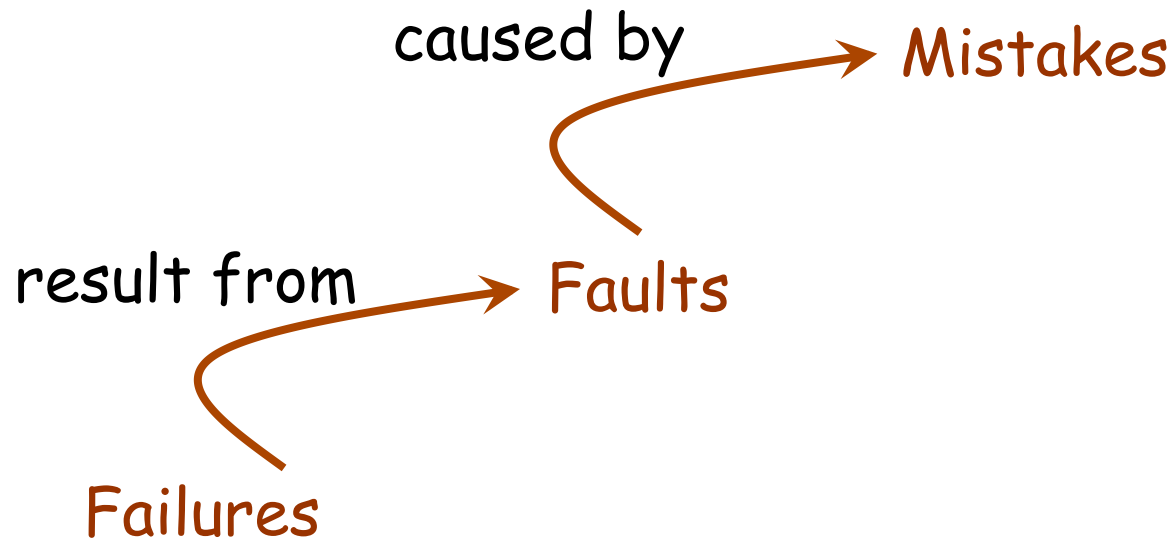
Failures

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

Reliability

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
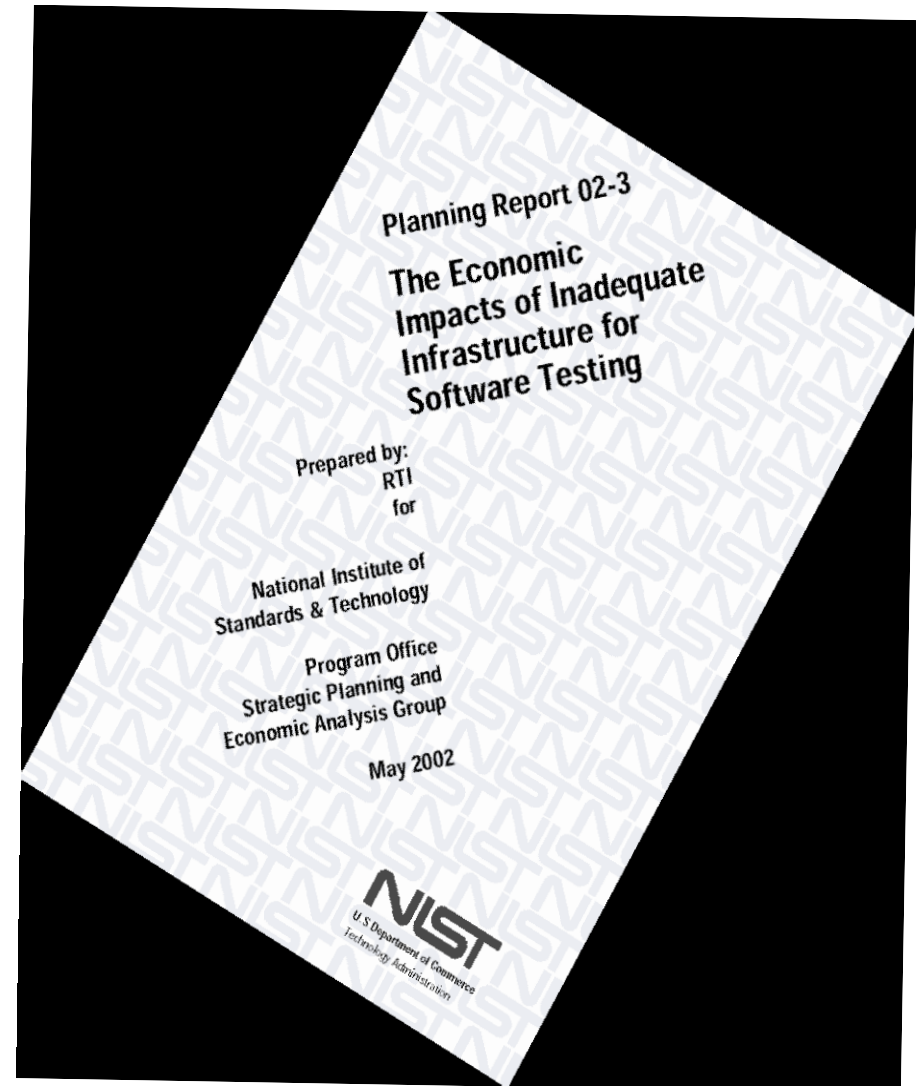
Security:
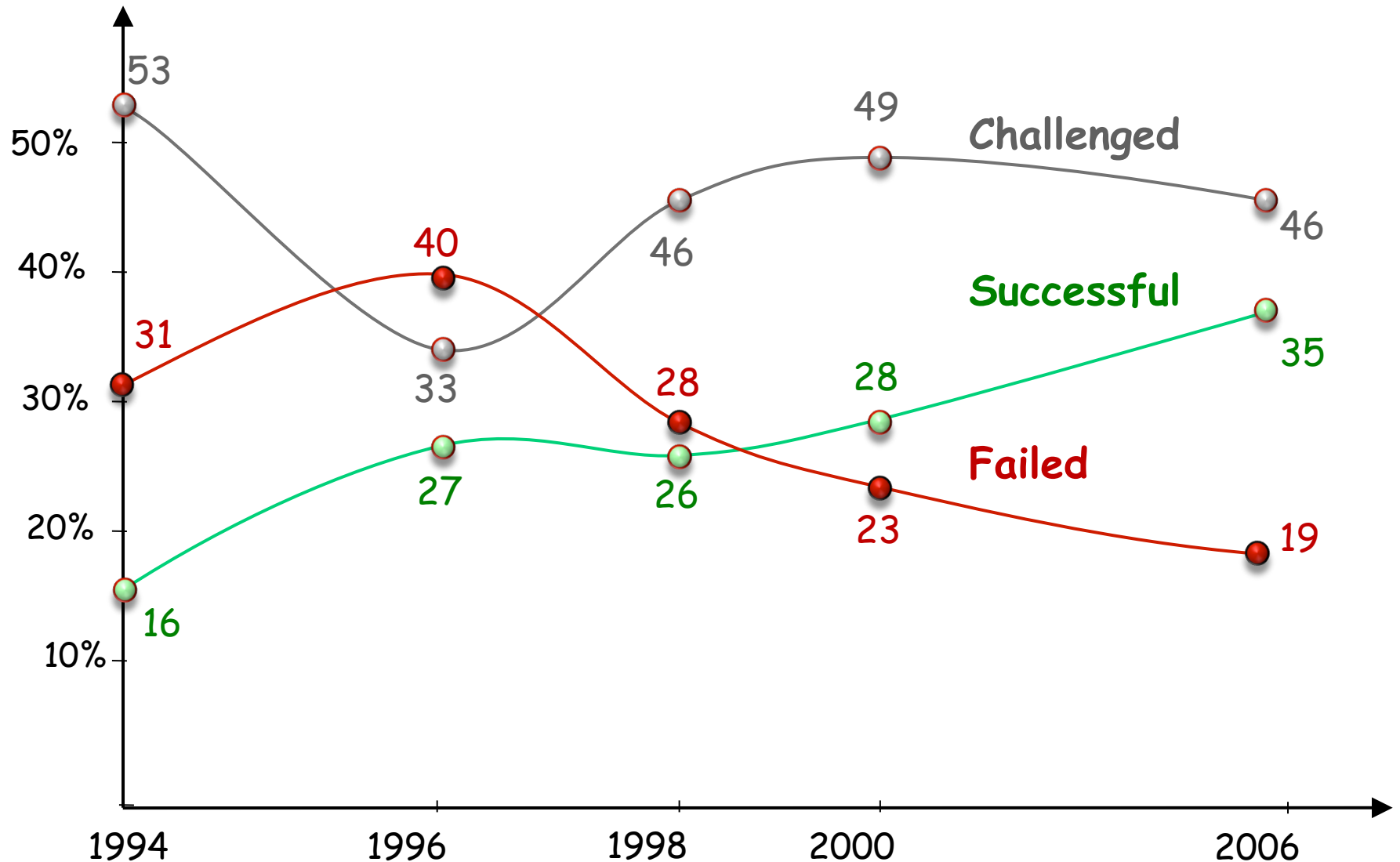The systems' ability to protect itself against hostile use

SPECIFICATION   ERRORS   HOSTILE USE

Correctness

Robustness

Security
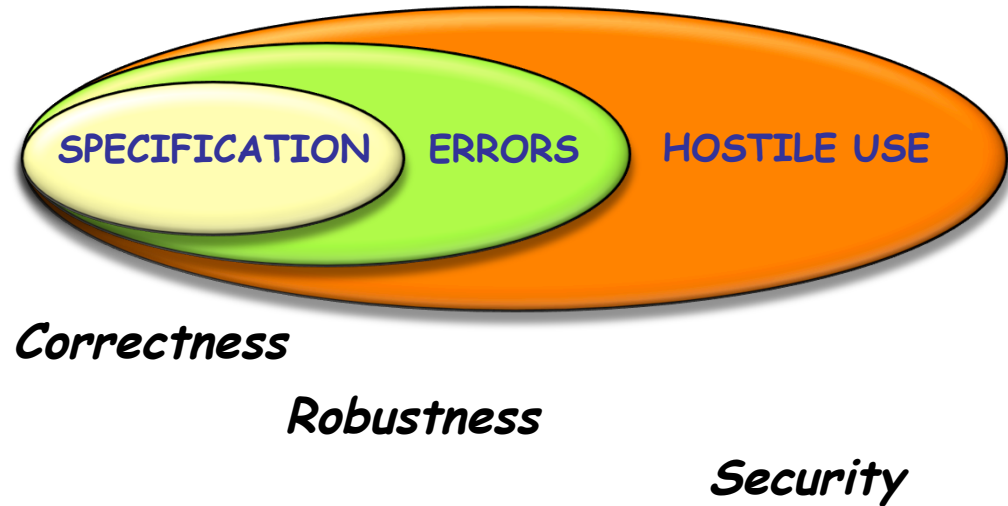
NIST report on testing (May 2002)

Financial consequences, on developers and users, of "insufficient testing infrastructure"

$ 59.5 B.

Planning Report 02-3

The Economic Impacts of Inadequate Infrastructure for Software Testing

Prepared by:
RTI
for

National Institute of Standards & Technology

Program Office
Strategic Planning and Economic Analysis Group

May 2002

NIST
U.S Department of Commerce
Technology Administration

Software projects according to Standish

Some famous failures

Ariane 5

Therac

Patriot

London Ambulance System

Mars Orbiter Vehicle

Buffer overflows

…

Mars Climate Orbiter Vehicle

Mars Climate Orbiter
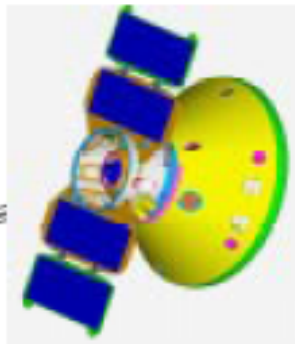
Mishap Investigation Board

Phase I Report

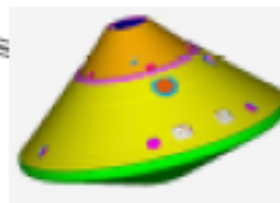**November 10, 1999**

Mars Polar Lander

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
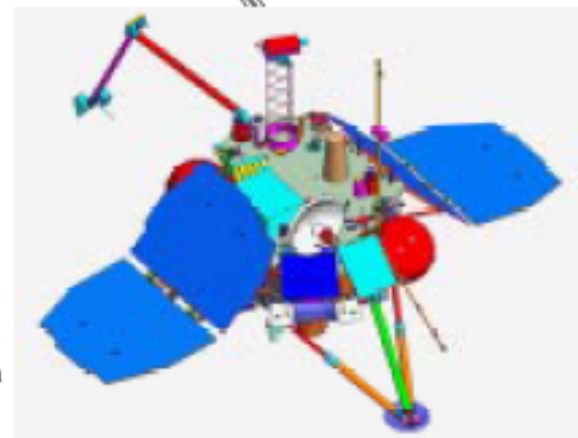- Propulsive Landing
- Descent Imaging [MARDI]

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

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

On September 27, 1999, the operations navigation team consulted with the spacecraft engineers to discuss navigation discrepancies regarding velocity change (ΔV) modeling issues. On September 29, 1999, it was discovered that the small forces ΔV'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.
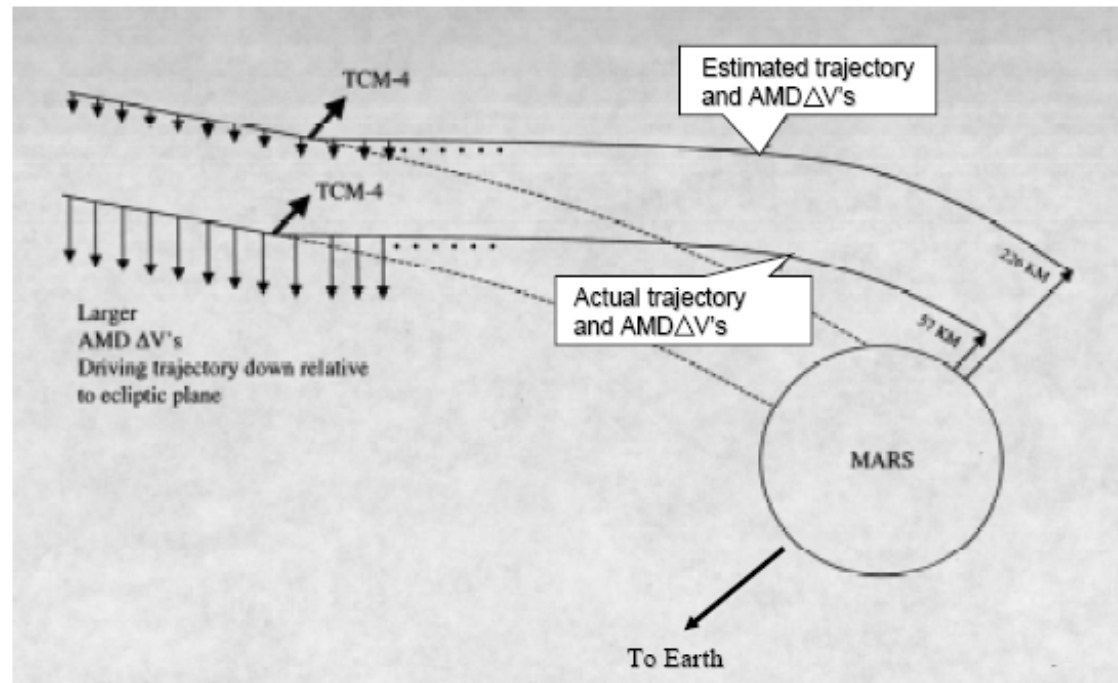
Ariane-5 maiden launch, 1996

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 !

See Jean-Marc Jézéquel & Bertrand Meyer, "Design by Contract: The Lessons of Ariane, IEEE *Computer*, January 1997, available at se.ethz.ch/~meyer/publications/computer/ariane.pdf

Security example: the buffer overflow
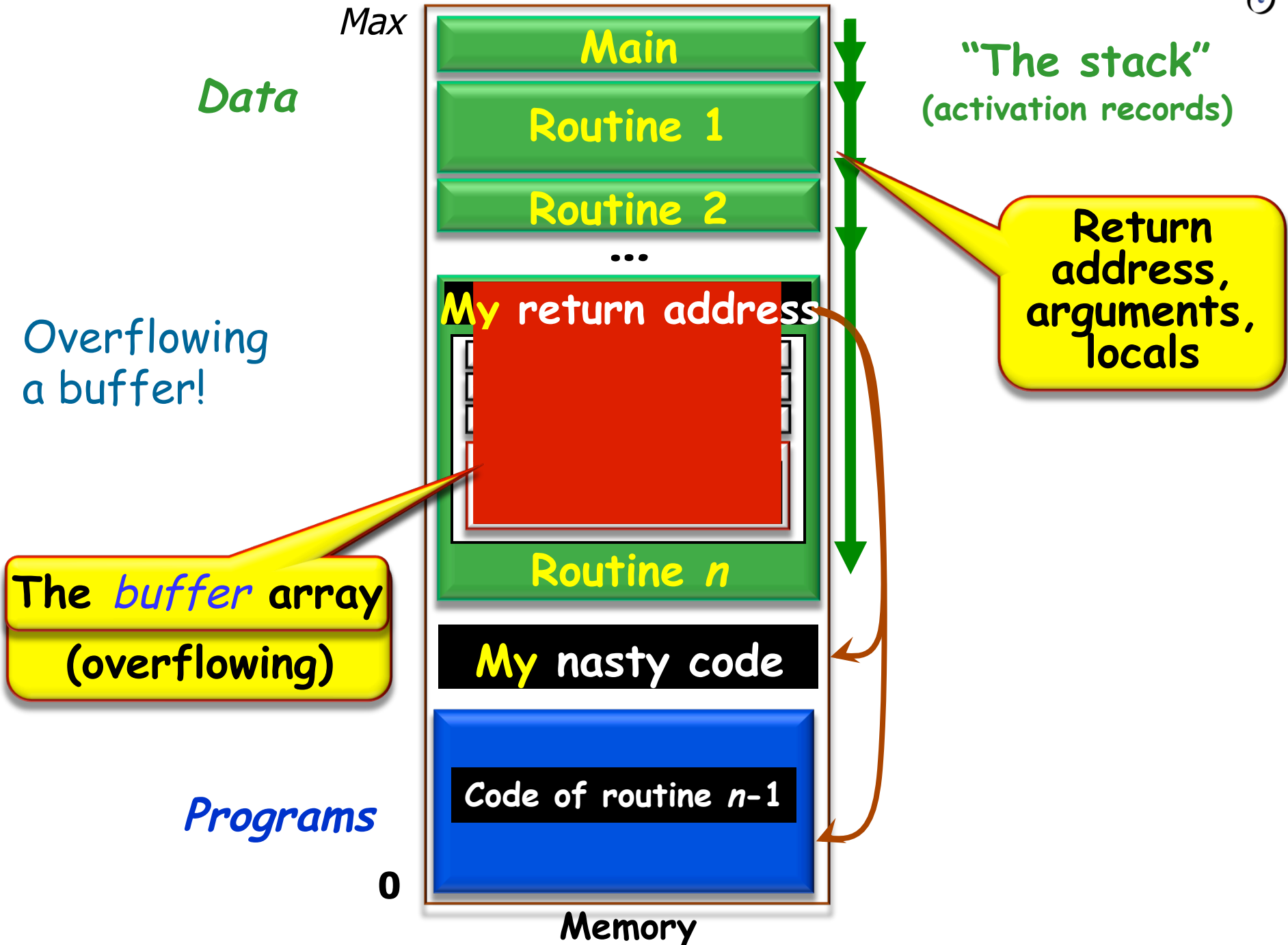
System expects some input from an external user:

First name: _____

Last name: _____

Address: _____

from *i* := 1 **until**

    i > *input_size*

loop

    buffer [*i*] := *input* [*i*]

    i := *i* + 1

end

```
from i := 1 until
        i > input_size     or i > buffer_size
loop
        buffer [i ] := input [i ]
        i := i + 1
end
```

Verification in the software lifecycle

Quality assurance techniques

	vs	
Process		*Product*
Manual		*Tool-supported*
Technology-generic		*Technology-specific*
Phase-generic		*Phase-specific* *(analysis, design, implementation…)*
Product-generic	**vs**	*Product-specific* *(code, documentation…)*
Build (a priori)		*Assess (a posteriori)*
Static		*Dynamic*
Informal		*Mathematical*
Complete		*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

➢ Process models: CMMI, ISO 9001:2000

➢ Inspections

➢ Open-source process

➢ eXtreme Programming (XP)