CMPT 477/777

www.cs.sfu.ca/~ter/CMPT477-777-fall2015

Instructor: Eugenia Ternovska

email ter@sfu.ca

[CMPT 477/777]

Introduction to Formal Verification

About the course

This course is a general introduction into the field of formal verification through a particular but widely used technique of Model Checking.

It gives an introduction to the main methods and tools that you may use in your own field.

In particular, it is useful in

- security,
- software engineering,
- program synthesis,
- artificial intelligence,
- databases, business process modelling,
- medical systems

(anything that involves reasoning about dynamic systems)

(3)

The importance of system verification

It has been estimated that people are in contact with ≈ 25 "information processing devices" per day

- cell phones
- other mobile devices
- TV, remote controls
- calculators
- ATM machines
- elevators
- cars
- medical equipment
- electronic commerce sites

⇒ reliability is the key issue

especially for

1. Safety-Critical Systems

a) Communication systems
used by
- ambulances
- search and rescue
- military

b) Process control systems
- nuclear plants
- chemical industry
- radiation machines in hospitals

c) Transportation systems

- airplane navigation & control
- cars
- city traffic control

20% of development costs of such systems goes to information technology

2. Financial / Business systems

⑥ With the development of service-oriented economy, verification of business processes is becoming more and more important.

e.g IBM is developing business process management products for

- electronic commerce
- logistic services
- document handling / processing
- cloud computing & external Web services

Correctness of business processes is of crutial importance since big money is involved

example of a property to verify:
e.g. no product is being sent until the payment is made in full

There are scientific conferences on business process management / verification.

CMPT 882
⇒ (some undergrads will be allowed)

(8) System validation as part of design procen.

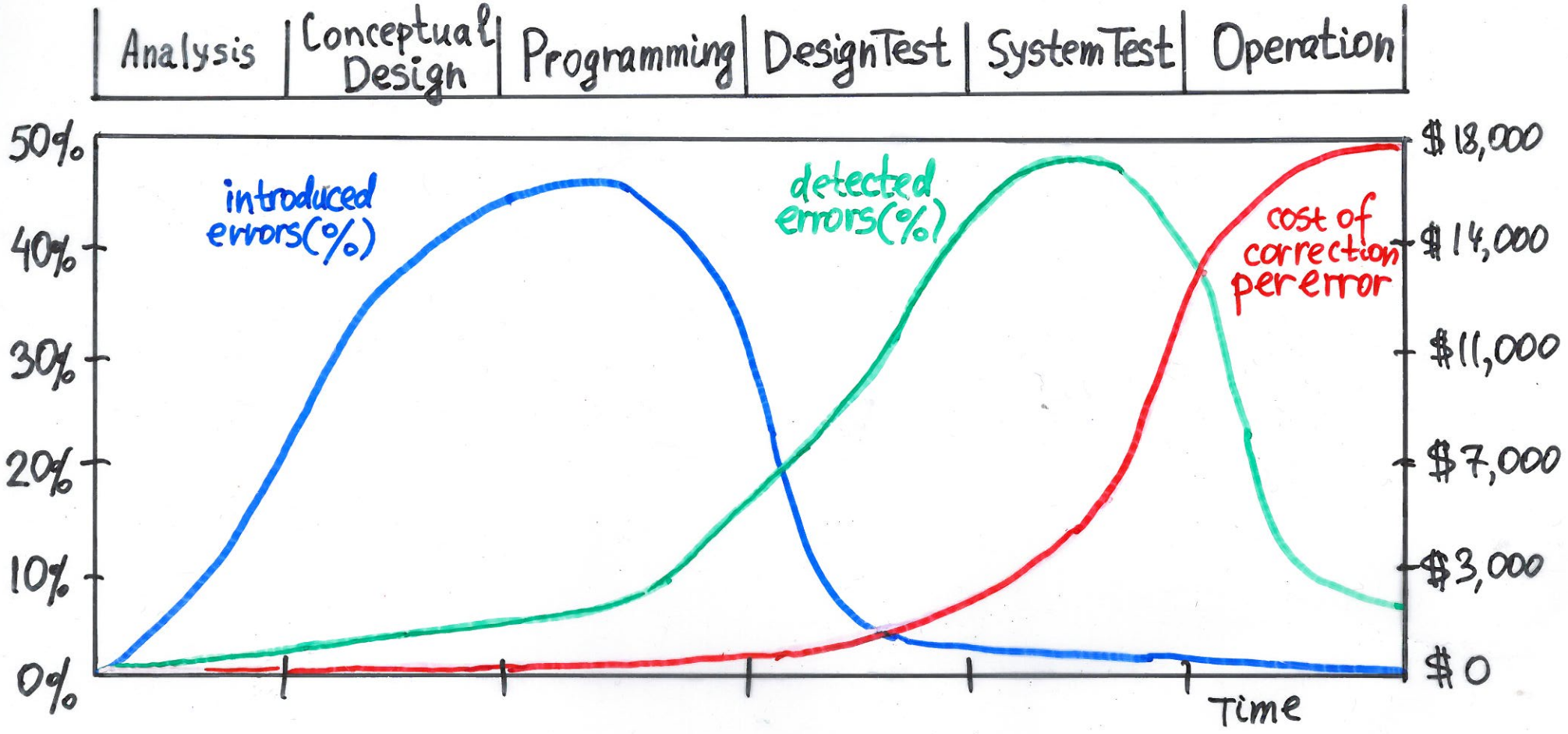
A study of several German software eng. companies shows:

- Cost of repair

Before the test phase:
$360 per error

In the design ~~testing~~ phase
$1,500 per error

During system ~~op~~ operation
up to $18,000 per error.



System life cycle: error introduction, detection and costs of repaire

Verification in the broader context of system validation techniques.

(11) Techniques to ensure system correctness (= validation techniques)

- Peer reviewing (completely manual)

Design is viewed by a team of developers that have not been involved in the design process

- Testing & simulation

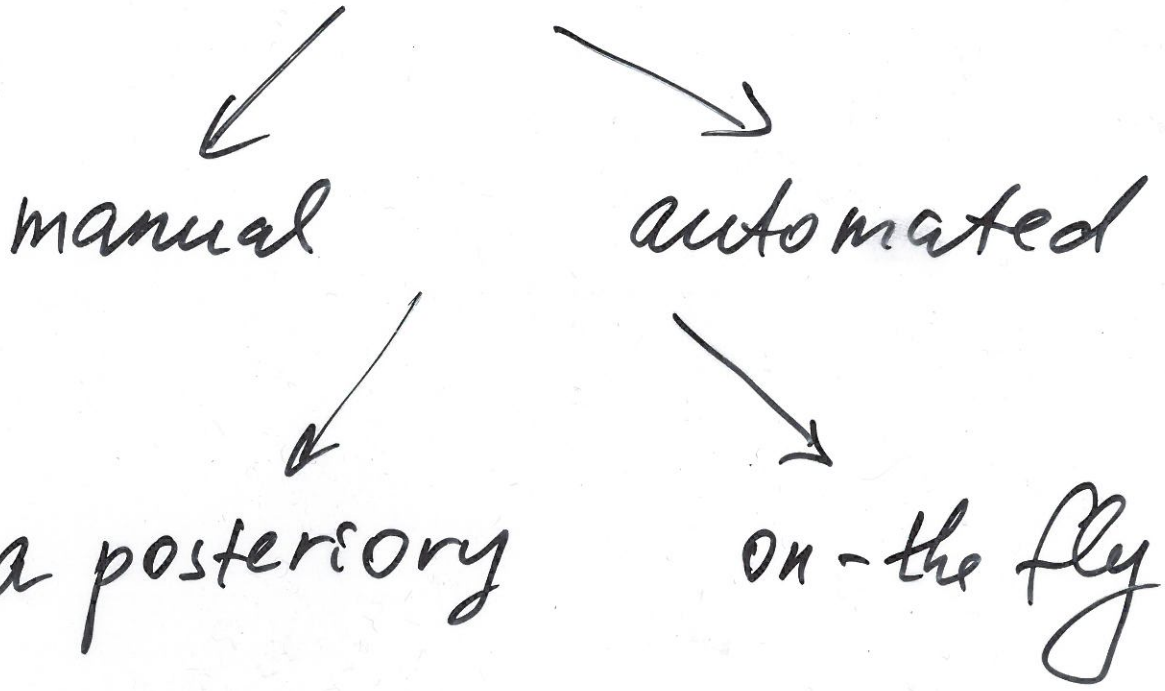
Testing is usually generated by hand, tool support is almost non-existent

" Testing can only show the presence of errors, never their absence "

Edsger W. Dijkstra (1930-2002)

- Formal verification

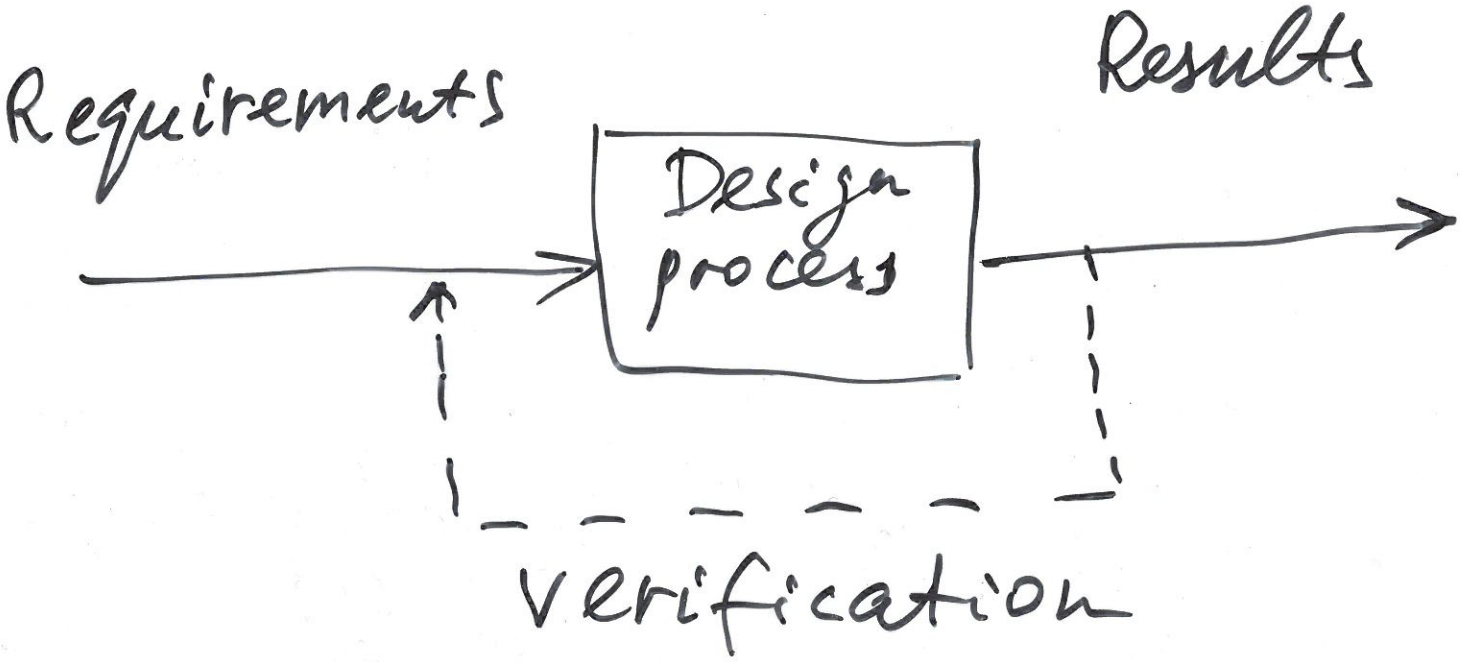
(complements peer reviewing, testing & simulation)



"Manual verification is at least as likely to be wrong as the program itself."

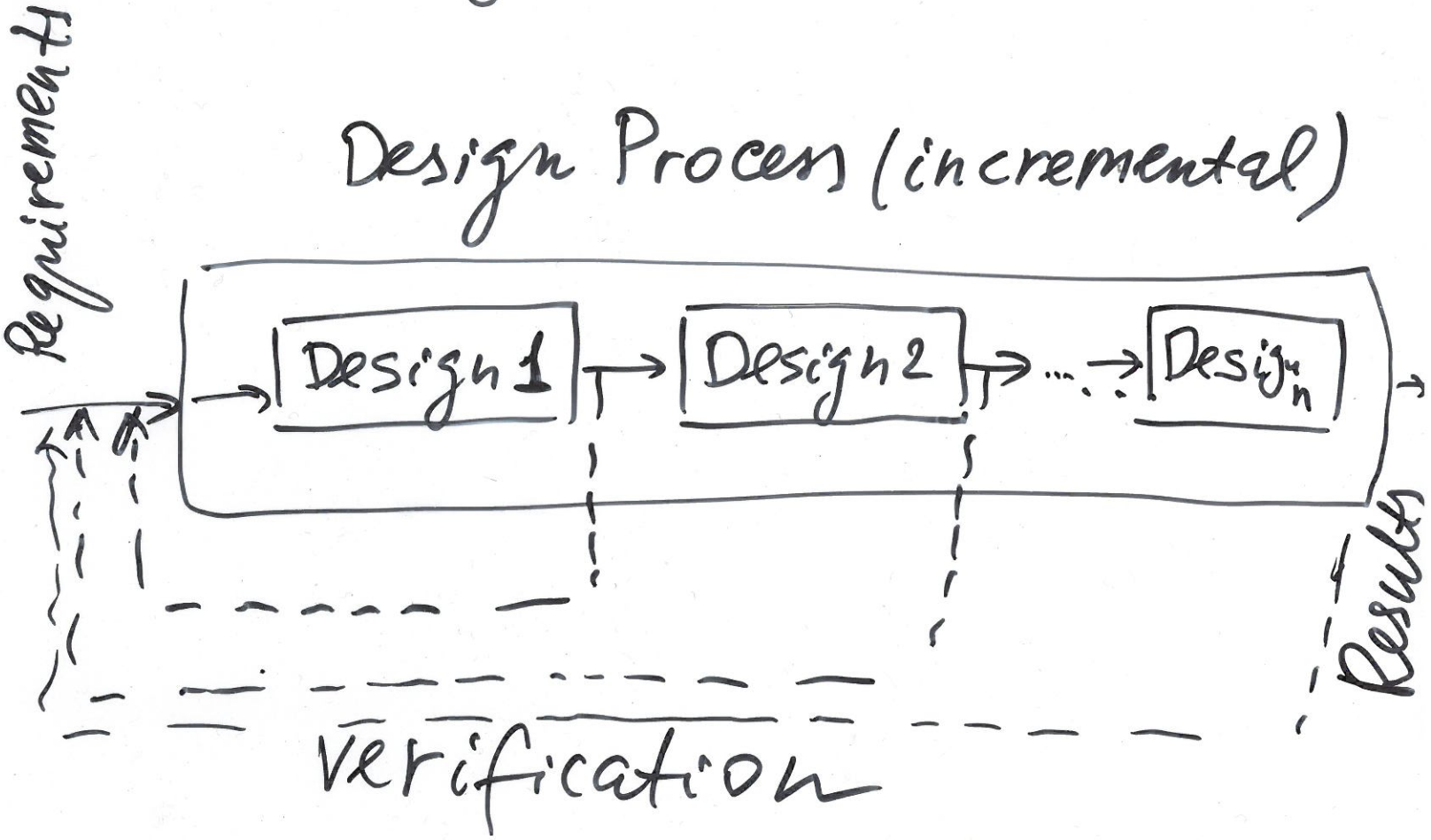
P. Wolper

a posteriory system verification



may be rather costly:
re-design from scratch

On-the fly system verification



Reduces the overall costs significantly.

In general, in most designs, more time is spent on validation than on construction.

Since the eary 80's:
a combination of Logical and Algorithmic techniques in proving program correctness

$\Rightarrow$ Model Checking (MC)
(one of the most important techniques)

- widely used in hardware industry
- used in software industry
  - for verification
  - for testing (40-70% of cost of software development)

usually, used in conjunction with other methods

Microsoft: develops its own methods, often using model checking techniques in combination with other tools such as Satisfiability Modulo Theory (SMT) solvers.

Microsoft hired Ken McMillan, the author of the SMV tool we are going to use.

Key idea of model checking (MC)

- Construct a formal mathematical model of the system which represents the possible behaviour (usually a transition system)

M

- Formalize the correctness requirement in a formal specification (usually a logical formula)

φ

- Check that the possible behavior "agrees with" the desired behaviour.

formally, $M \models \varphi$

A typical situation where MC is needed.

A complex software system (e.g. a flight control system)
- dozens of concurrent processes on multiple CPUs.

=> concurrency bug:

Events X and Y happen concurrently, say every 10^{10} cycles

Programmer did not anticipate X and Y could happen concurrently.

Current MC techniques can deal with such complex systems.