Formal Methods in Software Engineering

Luís S. Barbosa

lsb@di.uminho.pt

Departamento de Informática

Universidade do Minho

Braga

Software Engineering

Fact:

The recent increase on both the availability of **processor power** and the **complexity of the problems** computers are requested to solve is unprecedented in other technologies

Software Engineering

Fact:
The recent increase on both the availability of **processor power** and the **complexity of the problems** computers are requested to solve is unprecedented in other technologies

But still software remains

- hard to develop

Software Engineering

Fact:
The recent increase on both the availability of **processor power** and the **complexity of the problems** computers are requested to solve is unprecedented in other technologies

But still software remains

- hard to develop
- unreliable ($\equiv$ '*faulty goods over budget and behind schedule*')

Software Engineering

Fact:
The recent increase on both the availability of **processor power** and the **complexity of the problems** computers are requested to solve is unprecedented in other technologies

But still software remains

- hard to develop
- unreliable ($\equiv$ '*faulty goods over budget and behind schedule*')
- difficult to re-use

Software Engineering

Fact:
The recent increase on both the availability of **processor power** and the **complexity of the problems** computers are requested to solve is unprecedented in other technologies

But still software remains

- hard to develop
- unreliable ($\equiv$ '*faulty goods over budget and behind schedule*')
- difficult to re-use
- excessively costly to maintain

Software Engineering

Fact:
The recent increase on both the availability of **processor power** and the **complexity of the problems** computers are requested to solve is unprecedented in other technologies

But still software remains

- hard to develop
- unreliable ($\equiv$ '*faulty goods over budget and behind schedule*')
- difficult to re-use
- excessively costly to maintain
- ... the larger the project, the worse the picture ...

Some Development Catastrophes

Denver Airport baggage handling system	$ 200 million
CONFIRM travel information system	$ 160 million
London Ambulance Service dispatching	£ 9 million

Some Development Catastrophes

Denver Airport baggage handling system	$ 200 million
CONFIRM travel information system	$ 160 million
London Ambulance Service dispatching	£ 9 million

Average schedule slips by 50%

[Sci. American, Sep. 1994]

Some Development Catastrophes

Denver Airport baggage handling system	$ 200 million
CONFIRM travel information system	$ 160 million
London Ambulance Service dispatching	£ 9 million

Average schedule slips by 50%

25% of all large systems are cancelled

[Sci. American, Sep. 1994]

Anatomy of a Disaster

The London Ambulance Service

- developers inexperienced in safety-critical systems

Anatomy of a Disaster

The London Ambulance Service

- developers inexperienced in safety-critical systems
- users excluded from the design process

Anatomy of a Disaster

The London Ambulance Service

- developers inexperienced in safety-critical systems
- users excluded from the design process
- incomplete and flawed system design

Anatomy of a Disaster

The London Ambulance Service

- developers inexperienced in safety-critical systems
- users excluded from the design process
- incomplete and flawed system design
- extreme time pressure, with no realistic testing

Anatomy of a Disaster

The London Ambulance Service

- developers inexperienced in safety-critical systems
- users excluded from the design process
- incomplete and flawed system design
- extreme time pressure, with no realistic testing
- deficient management determined to push through

Critical Software Failures

US Telephone Network	[switching software failures, 1991]
Bank of New York	[90 minutes failure, Nov. 20 1985 lost information on $ 32 b in transactions forced to borrow $ 23.6 b from the FR at a cost in interest of $ 5 m]
GM Detroit Automated Factory	[1 year working at half its capacity]
THERAC-25 Radiotherapy	[2 killed by overdose]

[*Digital Woes*, Wiener, L., 1993]

Some Challenges

- A main economic challenge is the **cost of rework**:
  How much software gets used as delivered?

Some Challenges

- A main economic challenge is the **cost of rework**:
  How much software gets used as delivered?
- This cost is related to the ‘distance’ between the commission and the discovery of the error.

Some Challenges

- A main economic challenge is the **cost of rework**:
  How much software gets used as delivered?
- This cost is related to the ‘distance’ between the commission and the discovery of the error.
- Improved requirements analysis and rigorous design could reduce some of the most critical costs.

Some Challenges

- A main economic challenge is the **cost of rework**:
  How much software gets used as delivered?
- This cost is related to the ‘distance’ between the commission and the discovery of the error.
- Improved requirements analysis and rigorous design could reduce some of the most critical costs.
- But besides costs ... software industry has to deal **mainly** with **safety-critical** and **mission-critical** applications.

Some Challenges

- A main economic challenge is the **cost of rework**:
  How much software gets used as delivered?
- This cost is related to the ‘distance’ between the commission and the discovery of the error.
- Improved requirements analysis and rigorous design could reduce some of the most critical costs.
- But besides costs ... software industry has to deal **mainly** with **safety-critical** and **mission-critical** applications.

But is Software Engineering (SE) ... Engineering?

Hardware *vs* Software Engineering

Reusable Components	"Flat" & Unrelated Tons of Code
Catalogue of specifications	... formal?
Refinement Methods	Why reuse, if it is easy to begin again?
Development stations	Debug = (Edit;Compile;Test)*
Standard Methodologies and Notation	Proliferation ...
Production and Maintenance: • Production Plans & Data Bases • Components' Families — classification, equivalences, etc.	Cubic meters of • unreadable programs • unhelpful manuals

Software Engineering

 Traditional “paper & pencil” development has created the illusion that Software Engineering was little more than a balanced compromise between intuition and craft

Software Engineering

- Traditional “paper & pencil” development has created the illusion that Software Engineering was little more than a balanced compromise between intuition and craft
- *Informal* design methods emphasise textual descriptions.

Software Engineering

- Traditional “paper & pencil” development has created the illusion that Software Engineering was little more than a balanced compromise between intuition and craft
- *Informal* design methods emphasise textual descriptions.
- CASE tools are mostly oriented toward the *production* process (*e.g.*, goal analysis, planning, version control, etc.), but have modest success in addressing the *product* itself.

Software Engineering

- Traditional “paper & pencil” development has created the illusion that Software Engineering was little more than a balanced compromise between intuition and craft
- *Informal* design methods emphasise textual descriptions.
- CASE tools are mostly oriented toward the *production* process (*e.g.*, goal analysis, planning, version control, etc.), but have modest success in addressing the *product* itself.

> *There is a need for a software technology with a sound mathematical basis, coping with composition and refinement, in which a program would be unacceptable unless accompanied by a* guarantee *that it respects its* specified *behaviour.*

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...

systems' descriptions

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...

systems' descriptions

diagramatical or textual,

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...

systems' descriptions

diagramatical or textual,

vague or precise,

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...

systems' descriptions

diagramatical or textual,

vague or precise,

well documented or unreadable,

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...
systems' descriptions

diagramatical or textual,
vague or precise,
well documented or unreadable,
formal or informal,

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...
systems' descriptions

diagramatical or textual,
vague or precise,
well documented or unreadable,
formal or informal,
even executable

SE as Mathematical Modelling

SE is not concerned with **physical artifacts**, but with...
systems' descriptions

diagramatical or textual,
vague or precise,
well documented or unreadable,
formal or informal,
even executable

Precise, **abstract** descriptions $\equiv$ **mathematical models**

SE as Mathematical Modelling

From school physics, recall a basic **problem solving strategy**:

SE as Mathematical Modelling

From school physics, recall a basic **problem solving strategy**:

- Understand the problem

SE as Mathematical Modelling

From school physics, recall a basic **problem solving strategy**:

- Understand the problem
- Create a **mathematical model**

SE as Mathematical Modelling

From school physics, recall a basic **problem solving strategy**:

- Understand the problem
- Create a **mathematical model**
- Reason within the model

SE as Mathematical Modelling

From school physics, recall a basic **problem solving strategy**:

- Understand the problem
- Create a **mathematical model**
- Reason within the model
- Calculate a **solution** ($\equiv$ an **implementation**)

SE as Mathematical Modelling

Suitable **models** are:

- **abstract** ($\equiv$ *concise and precise descriptions*)

SE as Mathematical Modelling

Suitable **models** are:

- **abstract** ($\equiv$ *concise and precise descriptions*)
- **formal** ($\equiv$ *mathematical*, thus suitable for formal analysis)

SE as Mathematical Modelling

Suitable **models** are:

- **abstract** ($\equiv$ *concise and precise descriptions*)
- **formal** ($\equiv$ *mathematical*, thus suitable for formal analysis)

$\rightsquigarrow$ **Formal Specification & Development Methods**

SE as Mathematical Modelling

Suitable **models** are:

- **abstract** (≡ *concise and precise descriptions*)
- **formal** (≡ *mathematical*, thus suitable for formal analysis)

⇝ **Formal Specification & Development Methods**

- *Model-Oriented*: VDM, Z, B, RAISE, CAMILA, ...
- *Property-Oriented*: CLEAR, OBJ, ...

SE as Mathematical Modelling

Suitable **models** are:

- **abstract** (≡ *concise and precise descriptions*)
- **formal** (≡ *mathematical*, thus suitable for formal analysis)

⇝ **Formal Specification & Development Methods**

- *Model-Oriented*: VDM, Z, B, RAISE, CAMILA, ...
- *Property-Oriented*: CLEAR, OBJ, ...

Links:

`www.fmeurope.org` (includes an applications database)
`www.comlab.ox.ac.uk/archive/formal-methods` (huge FM Archive)
`shemesh.larc.nasa.gov/fm.html` (NASA FM Page; generalistic)

SE as Mathematical Modelling

The target of Formal Methods is to drive software production into solid engineering standards.

SE as Mathematical Modelling

The target of Formal Methods is to drive software production into solid engineering standards.

Engineering means:

- Standard modelling notation (with an unambiguous semantics)

SE as Mathematical Modelling

The target of Formal Methods is to drive software production into solid engineering standards.

Engineering means:

- Standard modelling notation (with an unambiguous semantics)
- Formal calculi (to reason about and validate designs)

SE as Mathematical Modelling

The target of Formal Methods is to drive software production into solid engineering standards.

Engineering means:

- Standard modelling notation (with an unambiguous semantics)
- Formal calculi (to reason about and validate designs)
- Re-use (of both models and calculations)

SE as Mathematical Modelling

The target of Formal Methods is to drive software production into solid engineering standards.

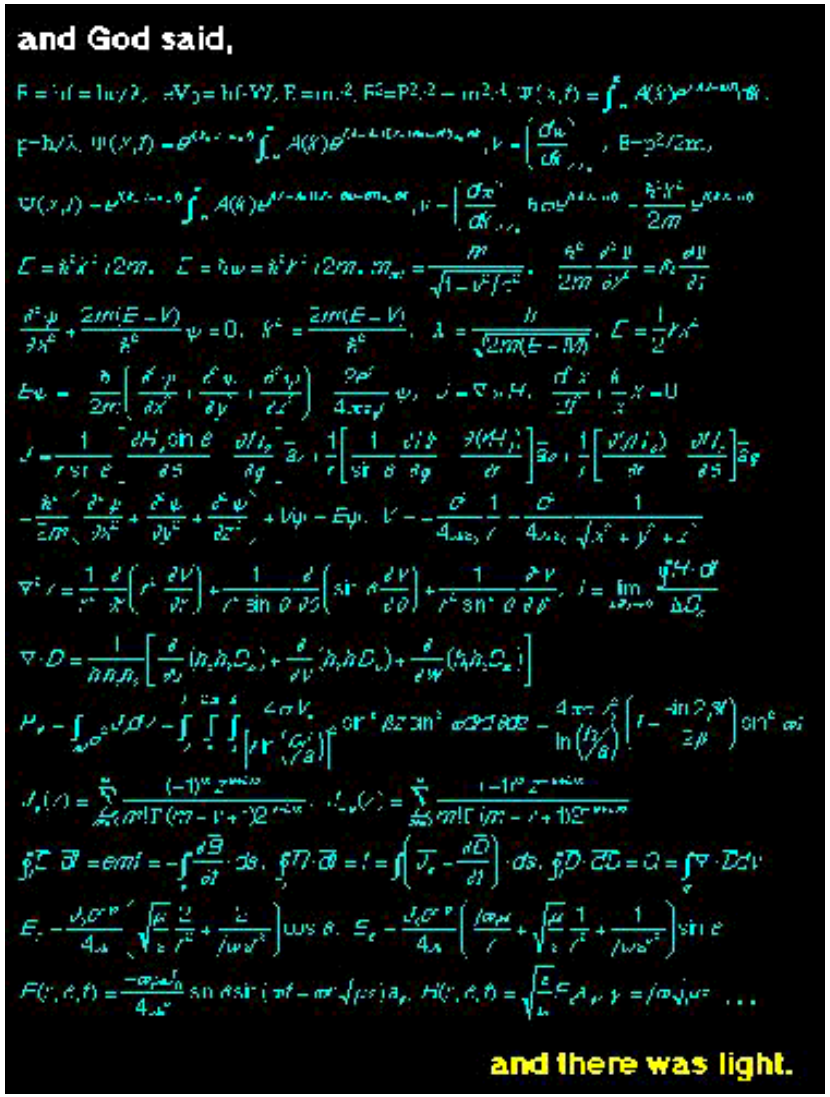
Engineering means:

- Standard modelling notation (with an unambiguous semantics)
- Formal calculi (to reason about and validate designs)
- Re-use (of both models and calculations)

> *'There is a big difference between good, sound reasons, ...*
> *... and reasons that sound good'* (Haldane)

Why Maths?

Why Maths?



Need for Tool Support

Most systems grow up from an unstructured collection of informal requirements

Need for Tool Support

- Most systems grow up from an unstructured collection of informal requirements
- ‘Going formal’ requires more and more precision from people

Need for Tool Support

- Most systems grow up from an unstructured collection of informal requirements
- ‘Going formal’ requires more and more precision from people
- Industry is slow in adopting mathematical based design methods

Need for Tool Support

- Most systems grow up from an unstructured collection of informal requirements
- 'Going formal' requires more and more precision from people
- Industry is slow in adopting mathematical based design methods

There is a need for tools to validate software formal descriptions: *type checkers*, *syntax-oriented editors*, *theorem provers*, *prototyping environments*...