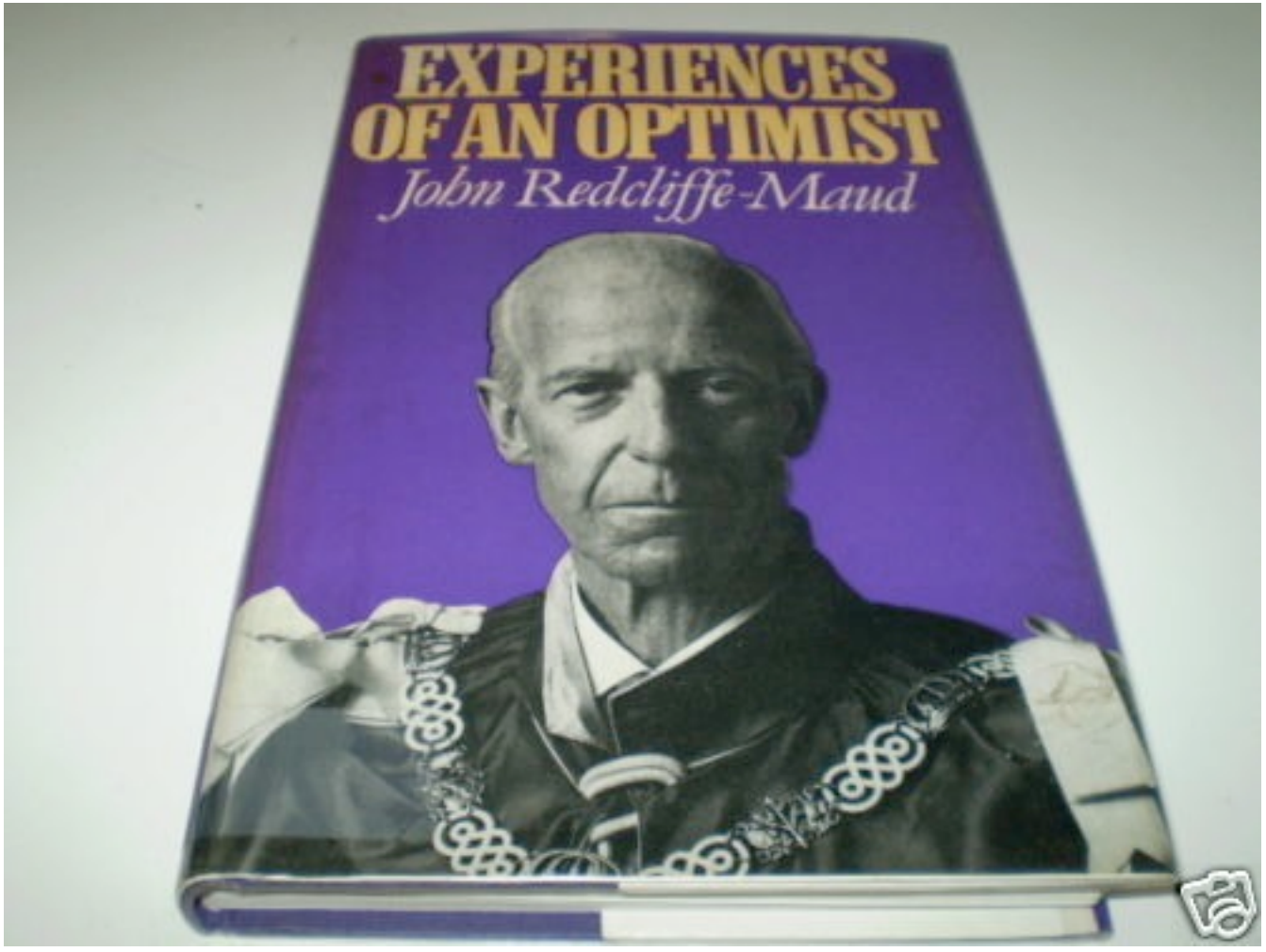
The Industrial Use of Formal Methods: Experiences of an Optimist

Prof. Jonathan P. Bowen

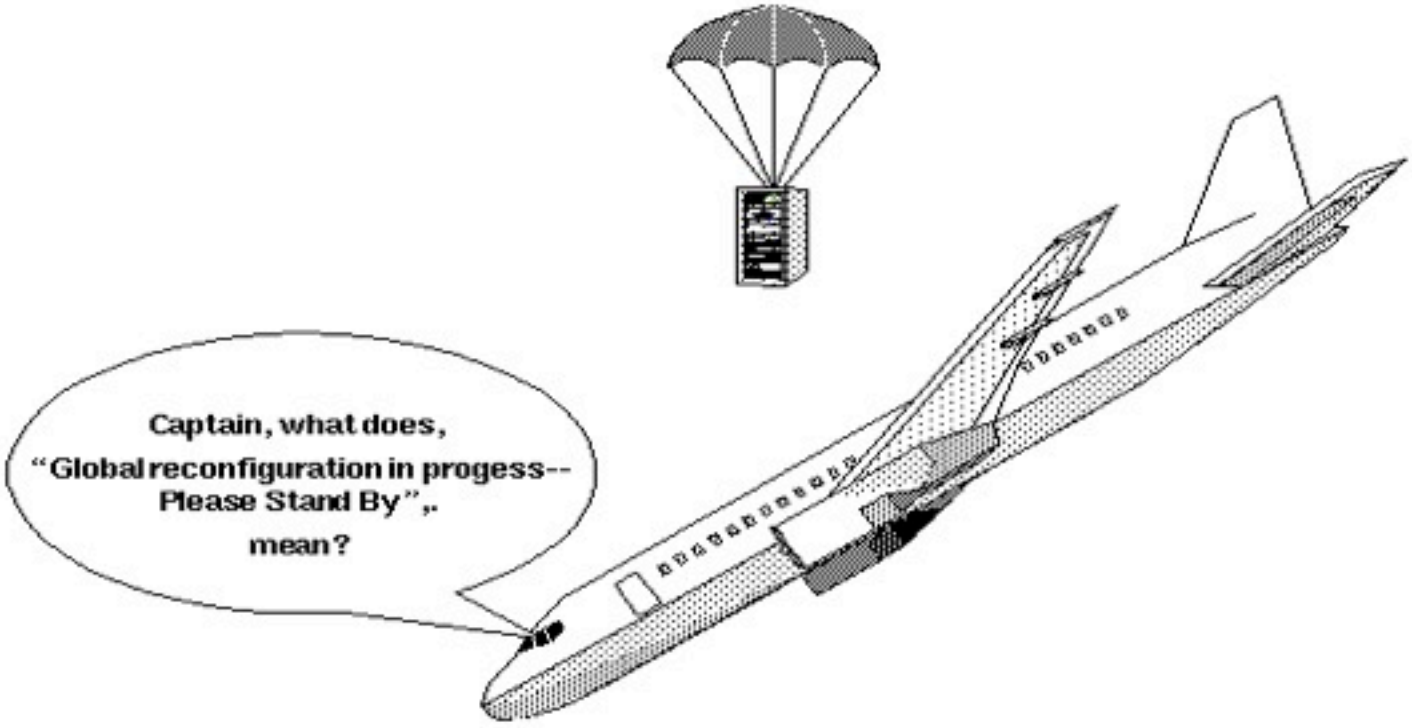
Museophile Limited, Oxford, UK
London South Bank University
www.jpbowen.com
jonathan.bowen@lsbu.ac.uk

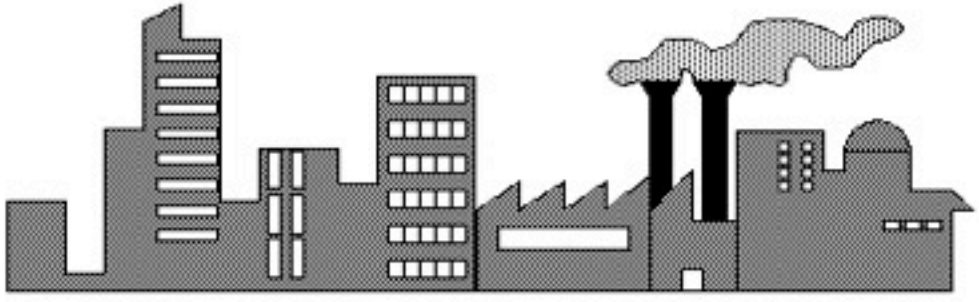
Experiences of an Optimist



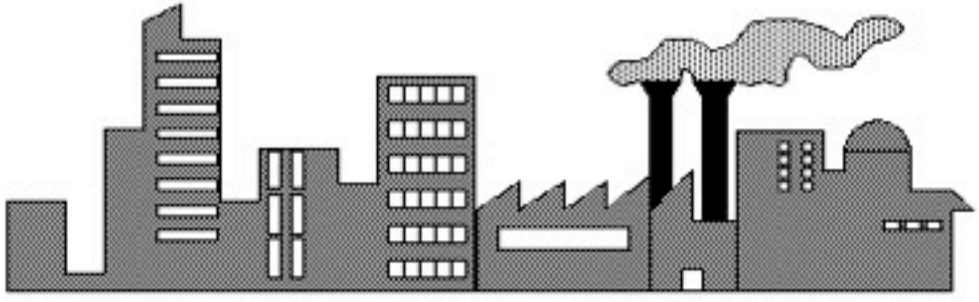
http://en.wikipedia.org/wiki/John_Redcliffe-Maud

Background: Safety and reliability





Background: Safety and reliability



Airbus A380 simulator

Emirates Aviation College

Dubai, 3 February 2011

Theory and Practice

"It has long been my personal view that the ***separation of practical and theoretical work is artificial and injurious****. Much of the* ***practical work*** *done in computing, both in software and in hardware design, is* ***unsound and clumsy*** *because the people who do it have not any clear understanding of the fundamental design principles of their work. Most of the abstract mathematical and* ***theoretical work*** *is* ***sterile*** *because it has no point of contact with real computing."*

— Christopher Strachey (1916–1975)

Formal Methods

Formal Methods

- Term established by late 1970s
  - Next stage from structured design
  - Mathematical basis

Formal Methods

- Term established by late 1970s
  - Next stage from structured design
  - Mathematical basis
- Formal specification and (optionally) proof:
  - Validation (correct specification)
  - Verification (correct implementation wrt spec)

Formal Methods

- Term established by late 1970s
  - Next stage from structured design
  - Mathematical basis
- Formal specification and (optionally) proof:
  - Validation (correct specification)
  - Verification (correct implementation wrt spec)
- But engineers *calculate* rather than prove

Formal Methods

- Term established by late 1970s
  - Next stage from structured design
  - Mathematical basis
- Formal specification and (optionally) proof:
  - Validation (correct specification)
  - Verification (correct implementation wrt spec)
- But engineers *calculate* rather than prove
- Please contribute to the *Formal Methods Wiki*:
  - http://formalmethods.wikia.com

Z notation

Z notation

- Formal specification – predicate logic, set theory, and schema boxes
  - Courses (academia & industry)
  - Textbooks (reasonable choice)
  - Tools (type-checkers, provers, …)

Z notation

- Formal specification – predicate logic, set theory, and schema boxes
  - Courses (academia & industry)
  - Textbooks (reasonable choice)
  - Tools (type-checkers, provers, …)
- Web resources – www.zuser.org

Z notation

- Formal specification – predicate logic, set theory, and schema boxes
  - Courses (academia & industry)
  - Textbooks (reasonable choice)
  - Tools (type-checkers, provers, …)
- Web resources – www.zuser.org
- Google group – comp.specification.z

Z notation

- Formal specification – predicate logic, set theory, and schema boxes
  - Courses (academia & industry)
  - Textbooks (reasonable choice)
  - Tools (type-checkers, provers, …)
- Web resources – www.zuser.org
- Google group – comp.specification.z
- Z User Group (meetings) & Z standard

Z Standard

Z Standard

- ISO/IEC 13568
  - Long process (1990s)
  - Inconsistencies found!

Z Standard

- ISO/IEC 13568
  - Long process (1990s)
  - Inconsistencies found!
- Final Committee Draft
  - accepted in 2001

Z Standard

- ISO/IEC 13568
  - Long process (1990s)
  - Inconsistencies found!
- Final Committee Draft
  – accepted in 2001
- Useful for tools and industrial application

Levels of Complexity – Abstraction

Levels of Complexity – Abstraction

- 25 lines of informal requirements

Levels of Complexity – Abstraction

- 25 lines of informal requirements
- 250 lines of specification (e.g., Z)

Levels of Complexity – Abstraction

- 25 lines of informal requirements
- 250 lines of specification (e.g., Z)
- 2,500 lines of design description

Levels of Complexity – Abstraction

- 25 lines of informal requirements
- 250 lines of specification (e.g., Z)
- 2,500 lines of design description
- 25,000 lines of high-level program code

Levels of Complexity – Abstraction

- 25 lines of informal requirements
- 250 lines of specification (e.g., Z)
- 2,500 lines of design description
- 25,000 lines of high-level program code
- 250,000 machine instructions of object code

Levels of Complexity – Abstraction

- 25 lines of informal requirements
- 250 lines of specification (e.g., Z)
- 2,500 lines of design description
- 25,000 lines of high-level program code
- 250,000 machine instructions of object code
- 2,500,000 CMOS transistors in hardware!

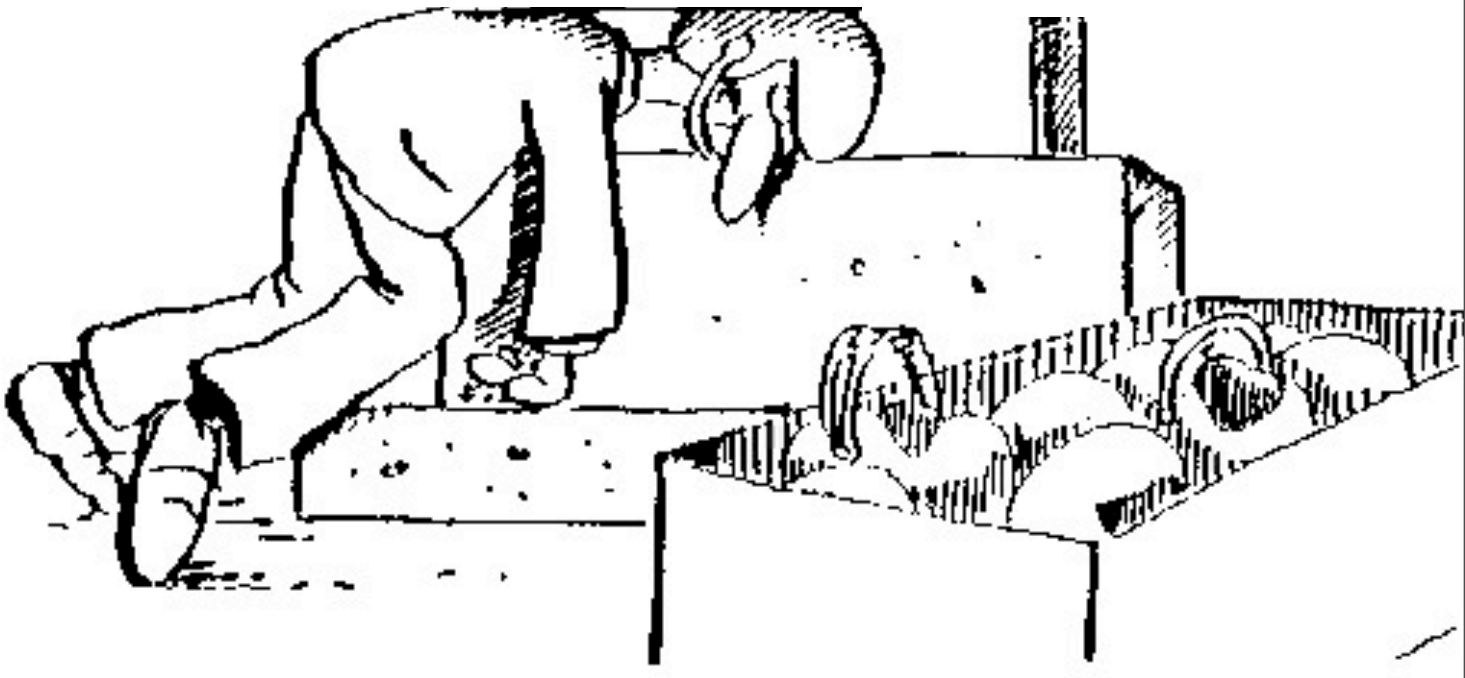
Technology transfer problems

The only thing harder to sell than formal methods.

Choosing a formal method – difficult

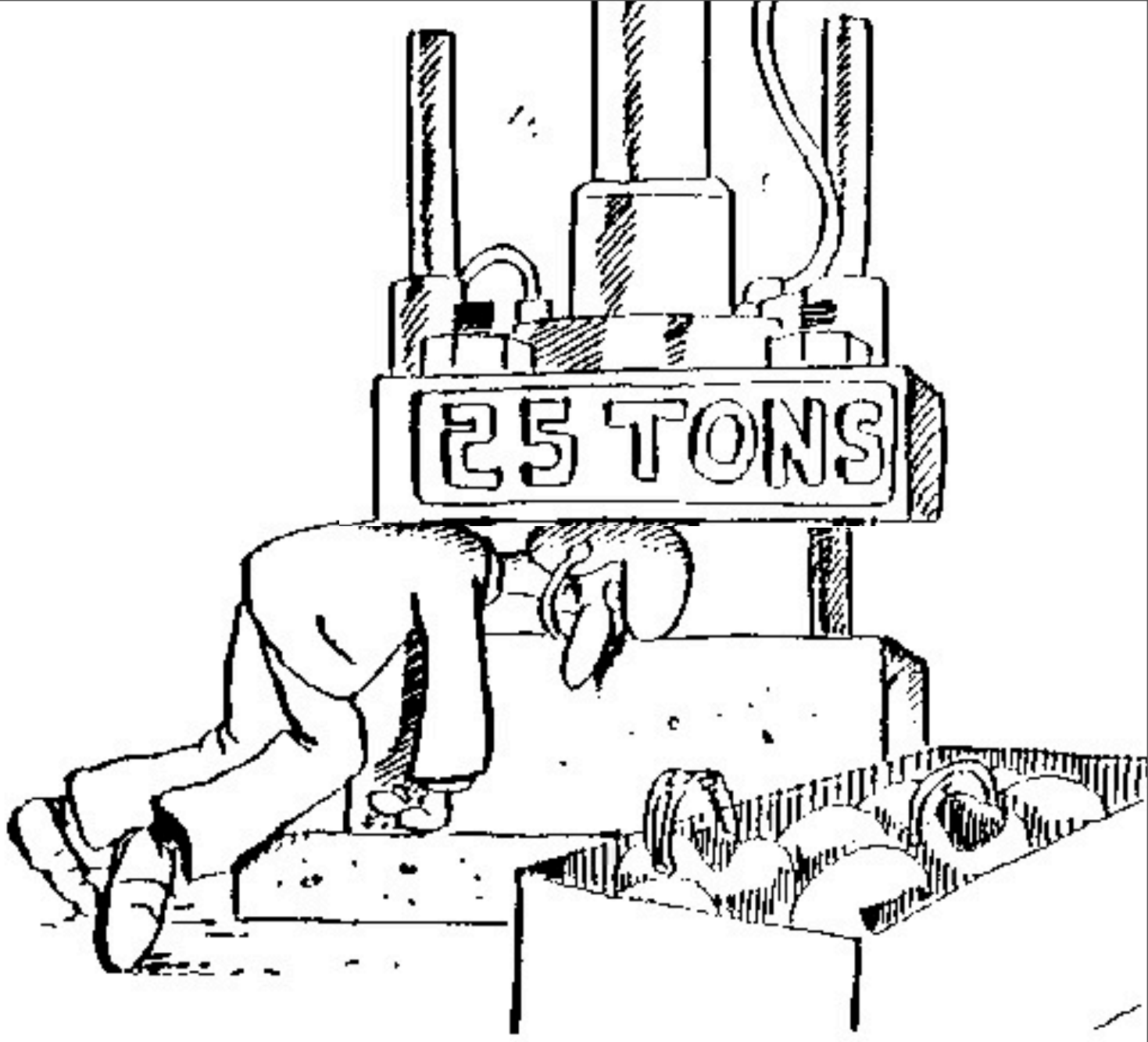
Choosing a formal method can be a fearful thing.

Tools – difficult to use



What it is like to use a mechanical theorem prover.

Tools – difficult to use



What it is like to use a mechanical theorem prover.

Applications of Formal Methods

Applications of Formal Methods

Examples:

Applications of Formal Methods

Examples:

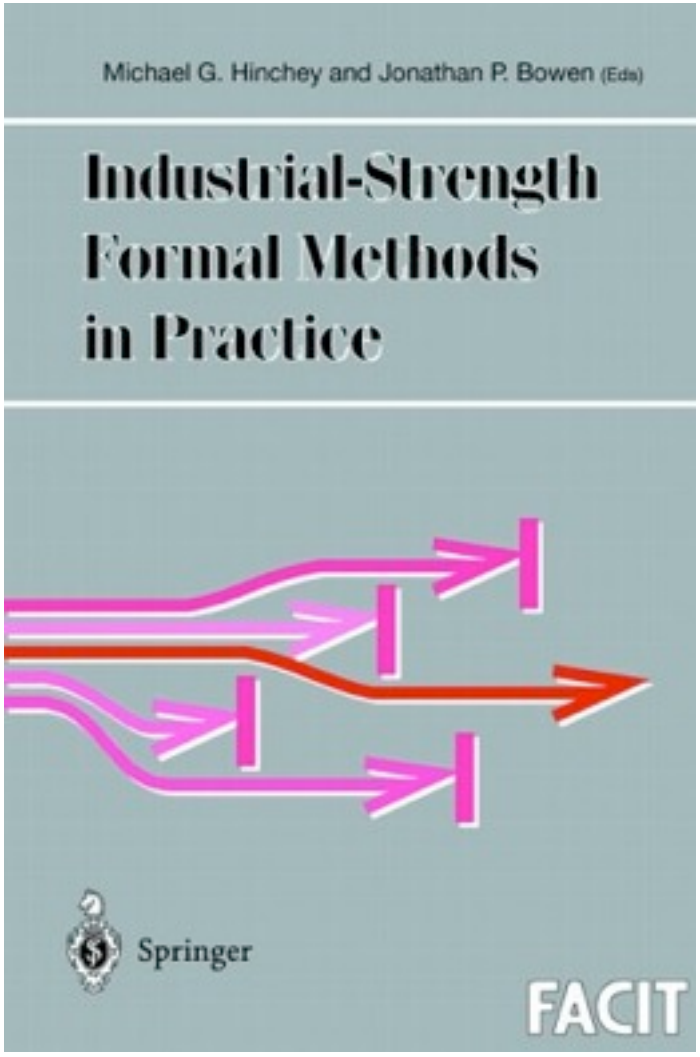
- Tektronix (Z)
- STV algorithm (VDM)
- IBM CICS (Z/B)

Applications of Formal Methods

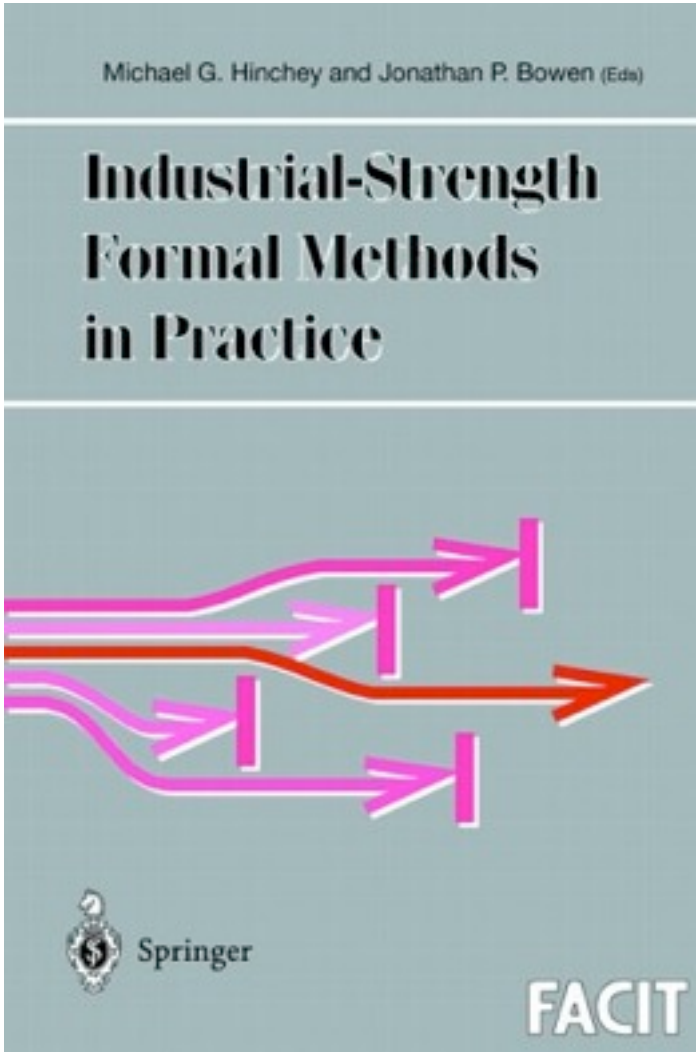
Examples:

- Tektronix (Z)
- STV algorithm (VDM)
- IBM CICS (Z/B)
- AAMP5 µprocessor (PVS)
- GEC Alsthom (B)
- A300/340 (Z)

Industrial-Strength Formal Methods in Practice

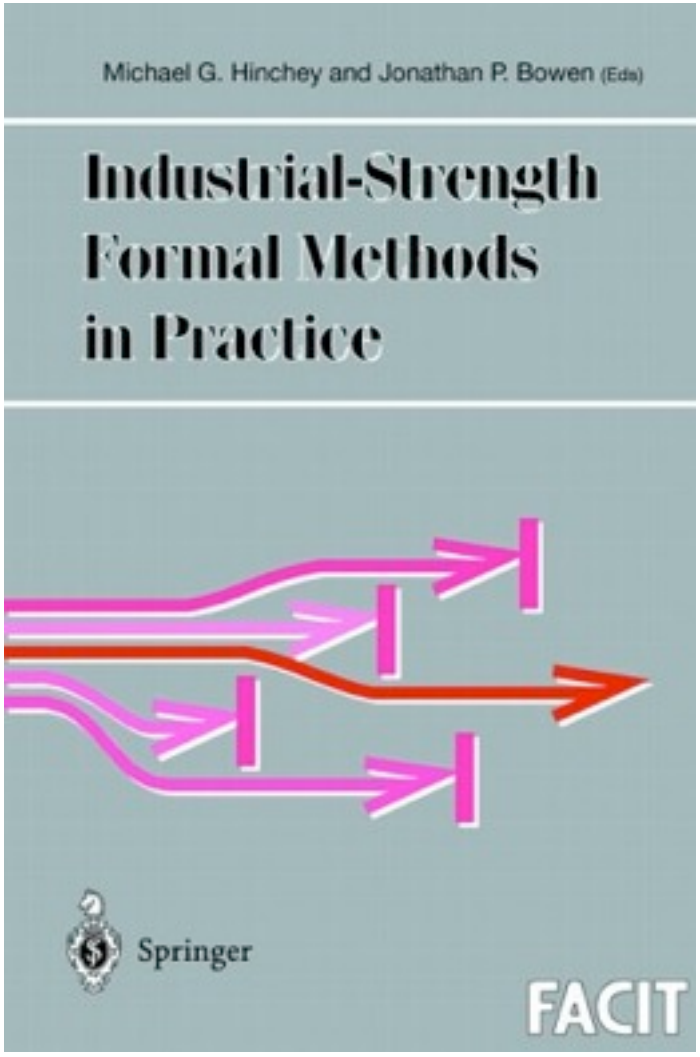


Industrial-Strength Formal Methods in Practice



Examples:

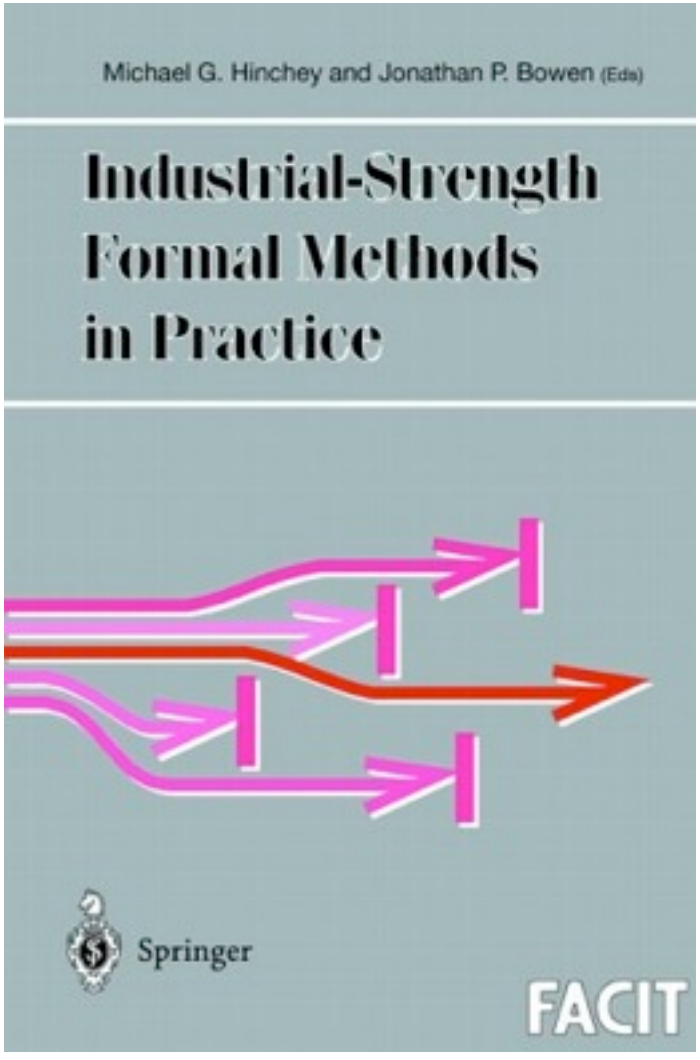
Industrial-Strength Formal Methods in Practice



Examples:

- Motorola CAP DSP (ACL2)
- Radiation Therapy Machine (Z)

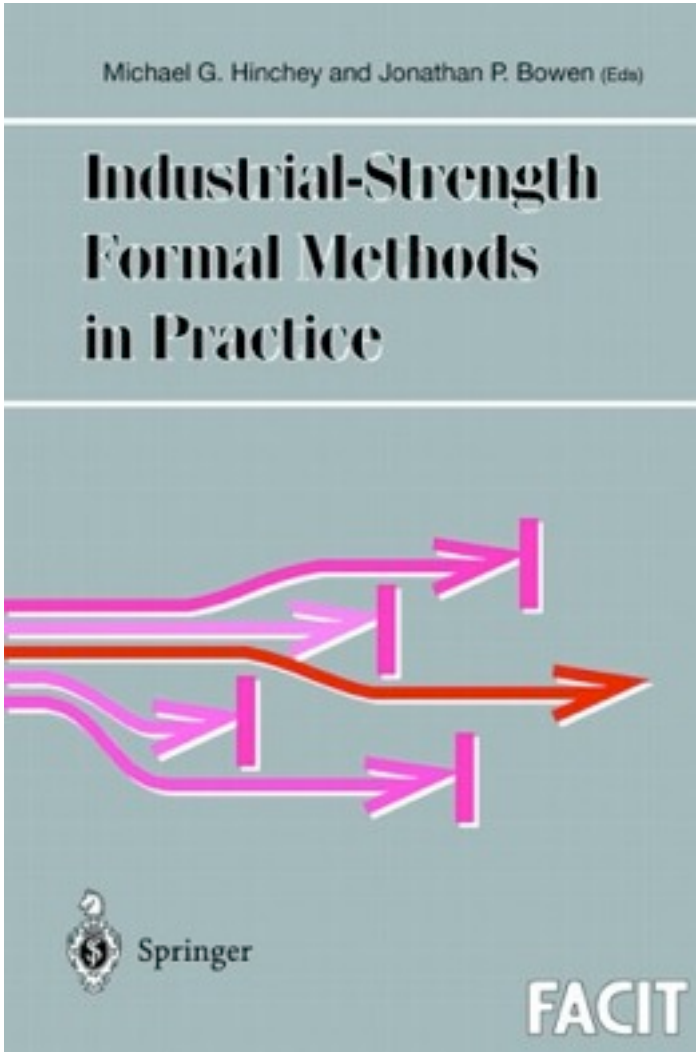
Industrial-Strength Formal Methods in Practice



Examples:

- Motorola CAP DSP (ACL2)
- Radiation Therapy Machine (Z)
- ATC system (VDM)
- Railways (Prover Technology)

Industrial-Strength Formal Methods in Practice



Examples:

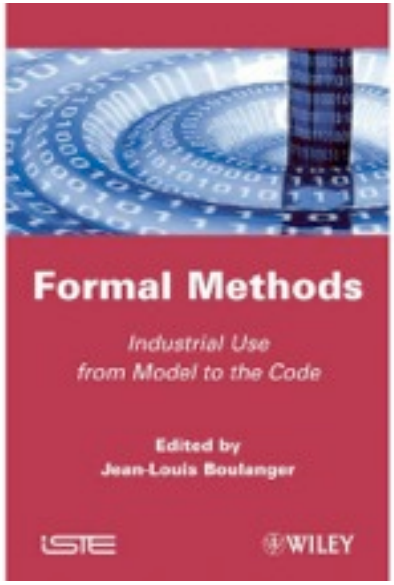
- Motorola CAP DSP (ACL2)
- Radiation Therapy Machine (Z)
- ATC system (VDM)
- Railways (Prover Technology)

And more recently:

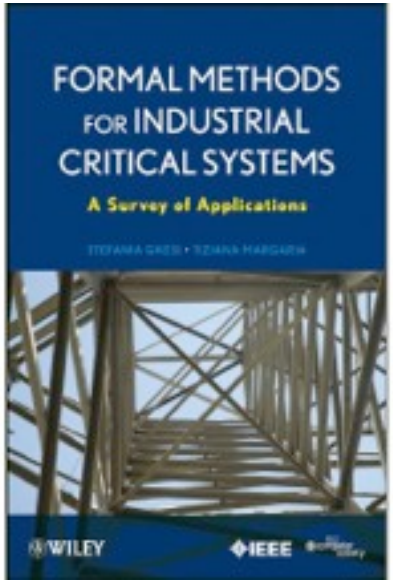
Microsoft

Recent books

- Boulanger, J.-L., ed. 2012.
  Formal Methods: Industrial Use from Model to the Code.
  ISTE, Wiley.
  ISBN 978-1848213623.



- Gnesi, S. and Margaria, T. 2012.
  Formal Methods for Industrial Critical Systems: A Survey of Applications.
  IEEE Computer Society Press, Wiley.
  ISBN 978-0470876183.



National Air Traffic Services

National Air Traffic Services

- Handled **2.2 million flights** (in 2009), covering the UK and eastern North Atlantic.

National Air Traffic Services

- Handled **2.2 million flights** (in 2009), covering the UK and eastern North Atlantic.
- And carried more than **200 million passengers** safely through some of the busiest and most complex airspace in the world.

National Air Traffic Services

- Handled **2.2 million flights** (in 2009), covering the UK and eastern North Atlantic.
- And carried more than **200 million passengers** safely through some of the busiest and most complex airspace in the world.
- Provides air traffic control from its centres at **Swanwick**, Hampshire and Prestwick, Ayrshire.

National Air Traffic Services

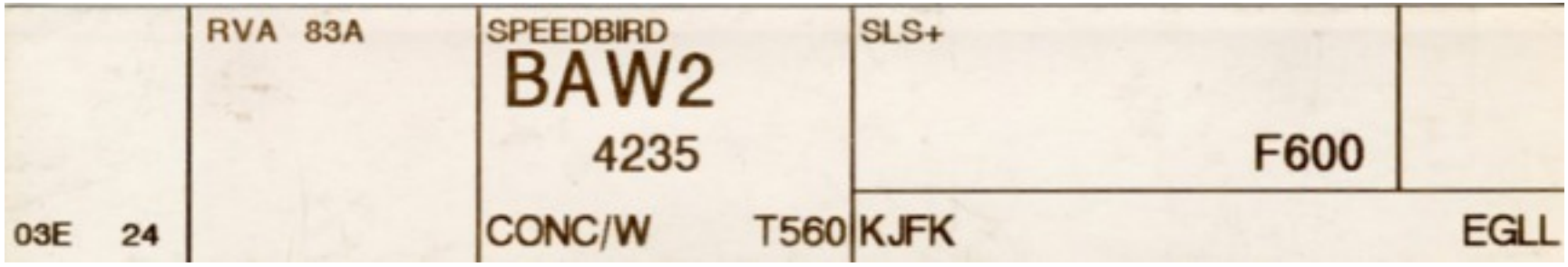
- Handled **2.2 million flights** (in 2009), covering the UK and eastern North Atlantic.
- And carried more than **200 million passengers** safely through some of the busiest and most complex airspace in the world.
- Provides air traffic control from its centres at **Swanwick**, Hampshire and Prestwick, Ayrshire.
- Also provides air traffic control services at 15 of the UK's major airports including **Heathrow**, Gatwick, Stansted, Birmingham, Manchester, Edinburgh, and Glasgow, together with air traffic services at Gibraltar Airport.

National Air Traffic Services, UK

Swanwick
southern England

www.nats.co.uk

Flight strips on paper

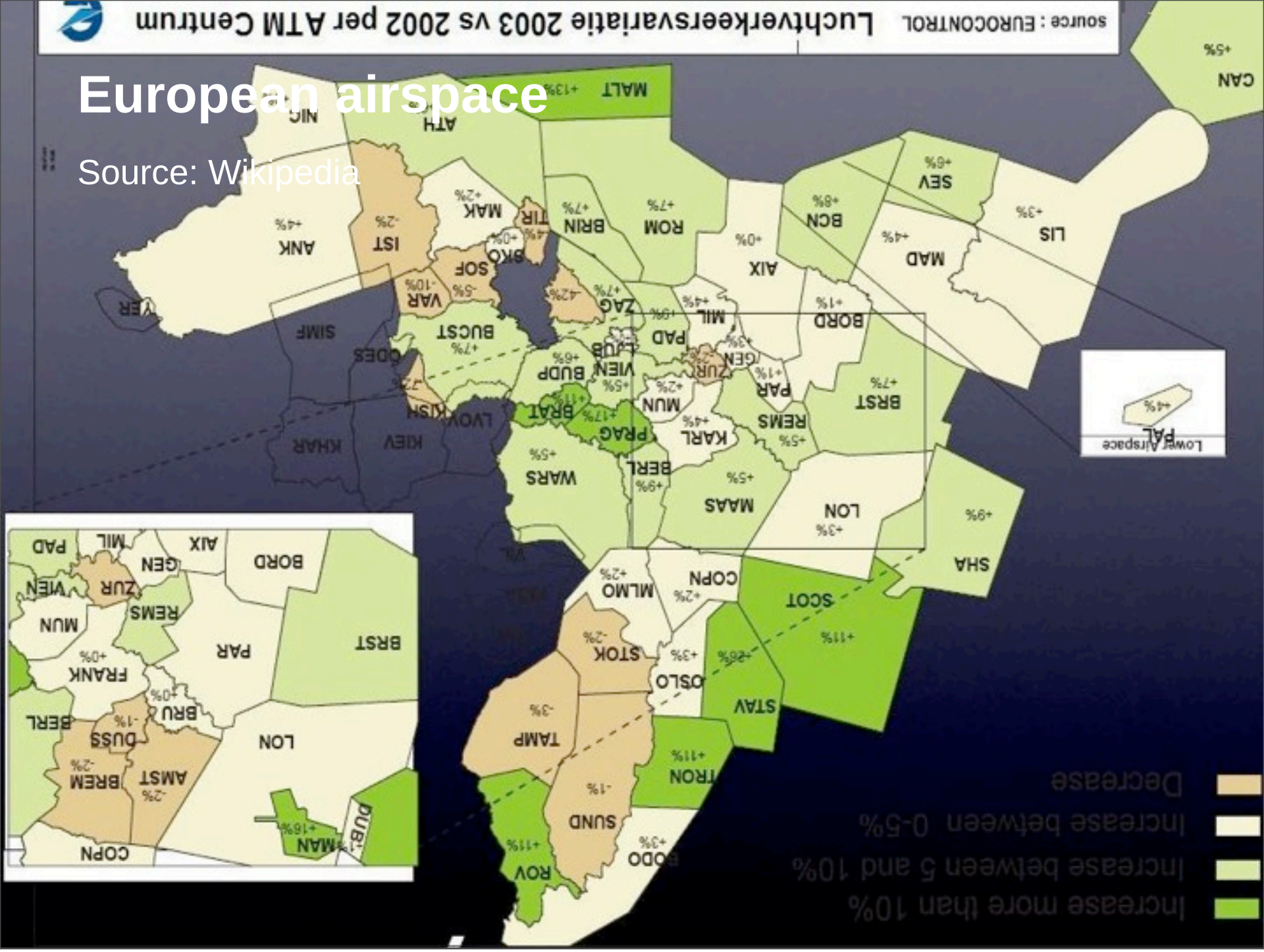


Flight strips on paper

Last flight of Concorde

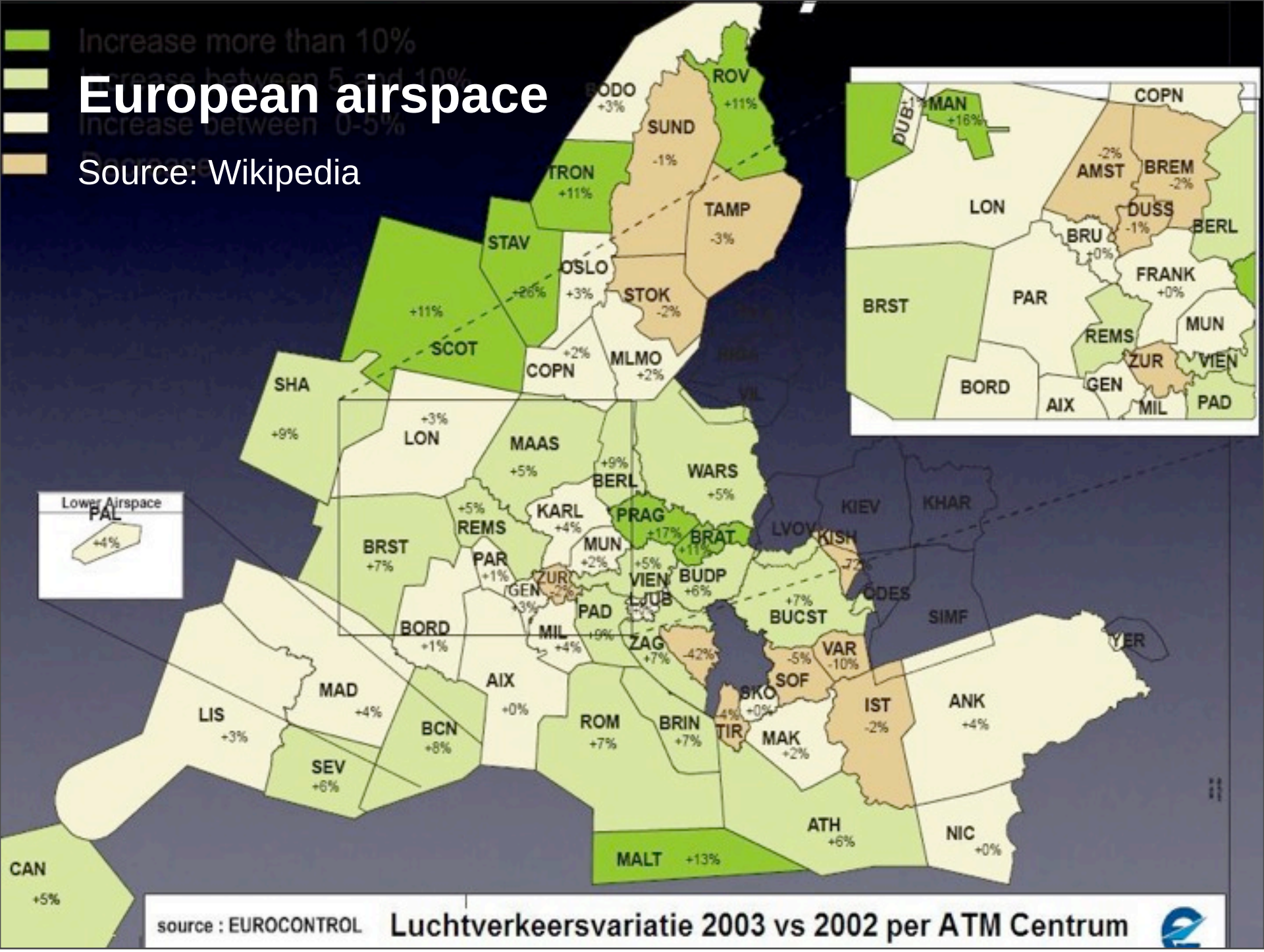
European airspace

Source: Wikipedia



European airspace

Source: Wikipedia



European airspace

Source: Wikipedia

London:
England
& Wales

NATS

National Air Traffic Services

- Advertisement & leaflet at Heathrow Airport →
- Air Traffic Management (ATM)
- Single European Sky ATM Research (SESAR)
- SESAR Joint Undertaking
- www.sesarju.eu
- SESAR project (2004–20)

NATS

National Air Traffic Services

- Advertisement & leaflet at Heathrow Airport →
- Air Traffic Management (ATM)
- Single European Sky ATM Research (SESAR)
- SESAR Joint Undertaking
- www.sesarju.eu
- SESAR project (2004–20)

www.altran-praxis.com

Open-DO

Formal Methods in Air Traffic Control

Slides by Neil White

www.slideshare.net/AdaCore/white-open-do
www.youtube.com/watch?v=IQMWVqQfm5A

Agenda

- A quick introduction
  - What is iFACTS?
- Formal methods for Specification
  - Z, State machines.
- Formal methods for Implementation
  - Implementation: SPARK.
- Formal methods for Test
  - Verification: more Z, Mathematica.

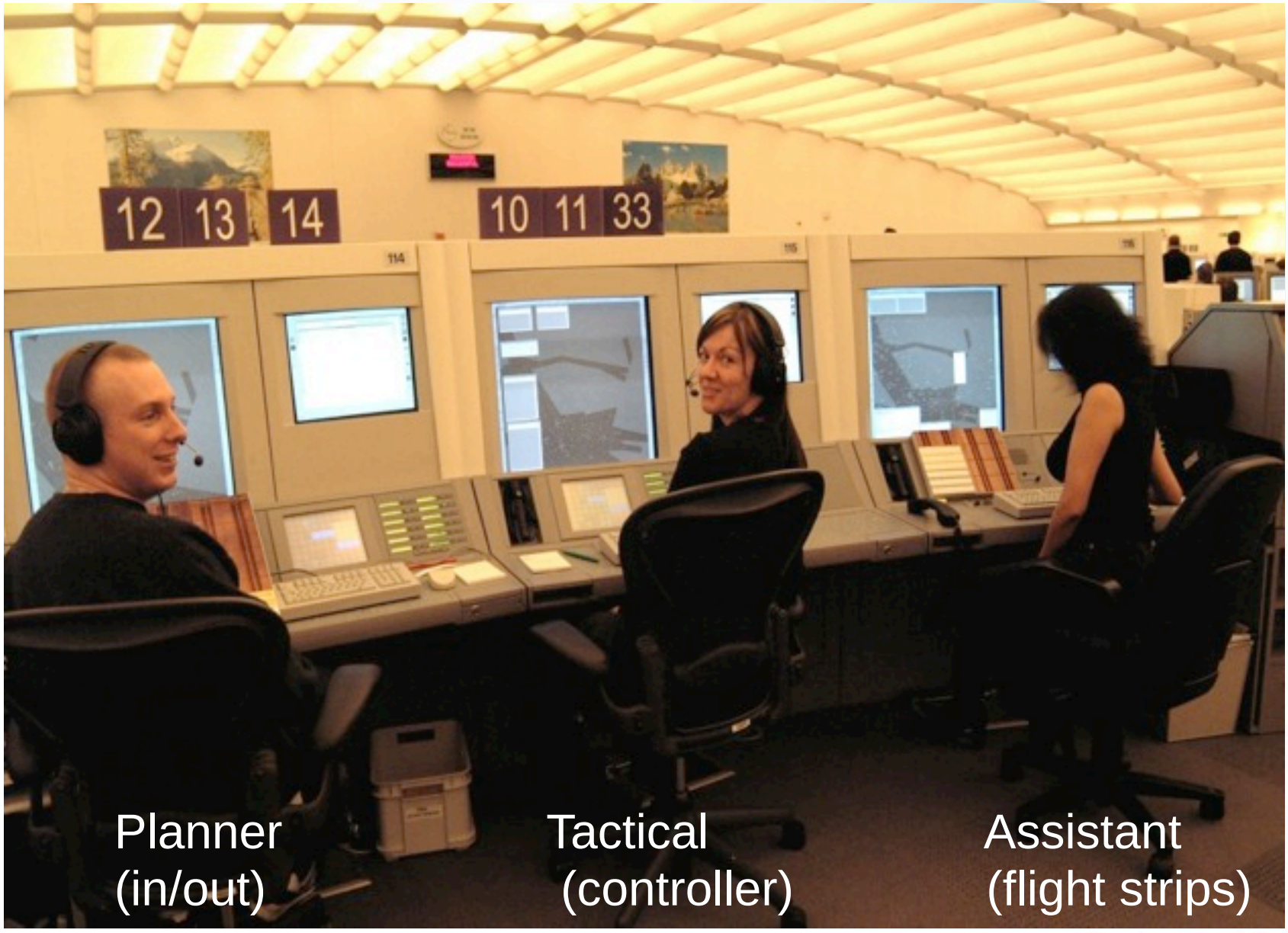
Context

- NATS, the UK's leading air traffic services provider, has pioneered research and development of advanced air traffic control tools for several years from its simulator and research centre. The iFACTS project will deliver a subset of these tools onto the system at the company's main en-route Control Centre at Swanwick.
- Further early information is available at: www.computerweekly.com/Articles/2007/03/07/222258/Nats-claims-the-biggest-air-traffic-control-innovation-since.htm

UK Air Traffic Control

ATC team

ATC team



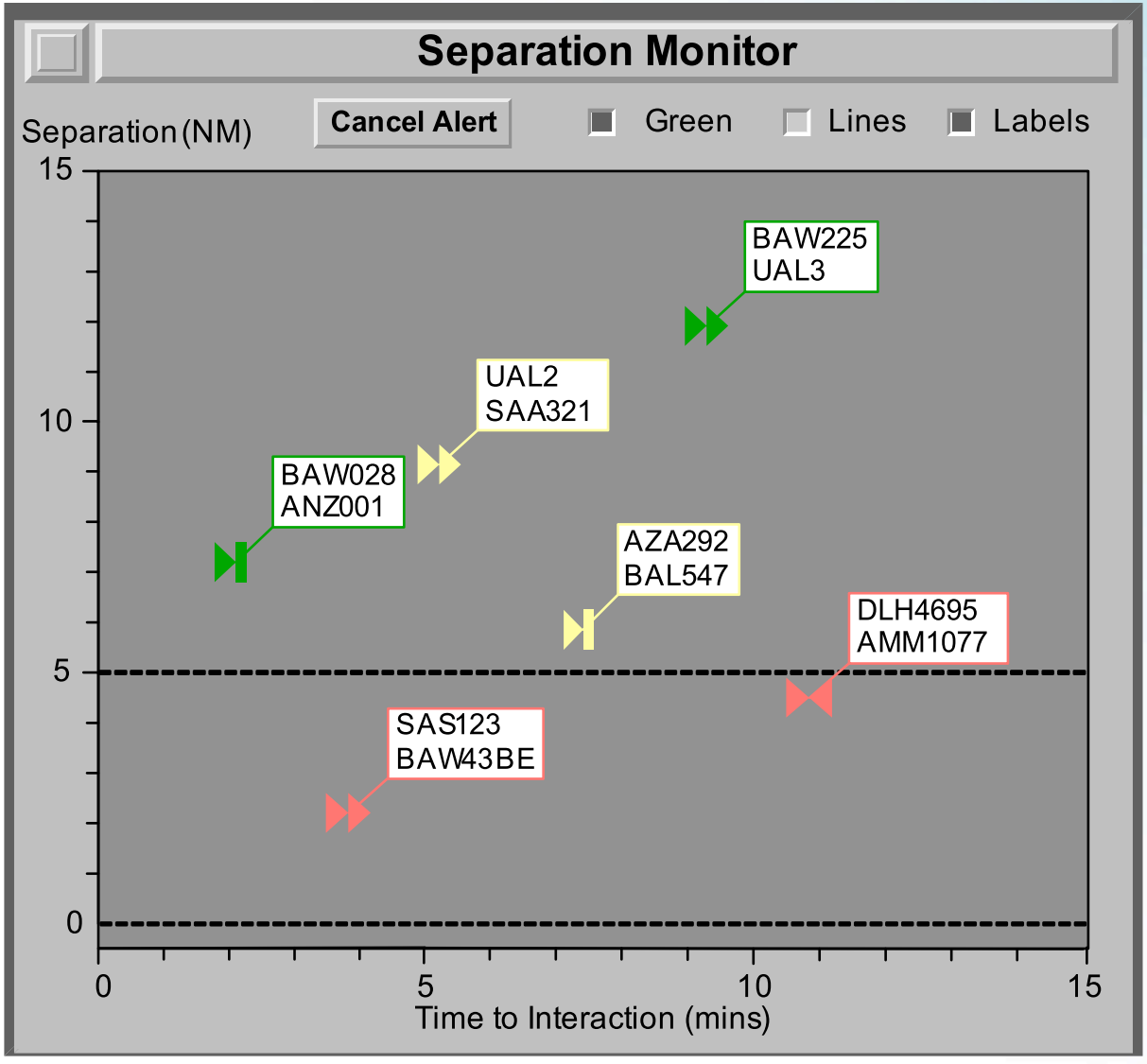
Why iFACTS?

- iFACTS – Interim Future Area Control Tools Support – to further improve safety and provide Controllers with a set of advanced support tools, which enables them to increase the amount of traffic they can comfortably handle. In trials, the system delivered significant capacity increases.

What is iFACTS?

- iFACTS provides tools to support the controllers
  - Electronic flight strips replace the paper flight strips.
  - Trajectory tools – including prediction, deviation alerts, and conflict detection – are added.
- iFACTS is not an Air Traffic control system
  - Integrated with, but sits alongside, the existing system.

Medium Term Conflict Detection:



Agenda

- A quick introduction
  - What is iFACTS?
- Formal methods for Specification
  - Z, State machines.
- Formal methods for Implementation
  - Implementation: SPARK.
- Formal methods for Test
  - Verification: more Z, Mathematica.

The complete iFACTS specification

- The functional specification
  - Z
- The algorithm specification
  - Maths
- The HMI specification
  - State tables
- The rest of the specification!
  - English

The Z specification

Every flight is associated with an aircraft type if its aircraft type name matches.

Every flight is associated with a performance model. If there is no model corresponding to the aircraft type, then this is a default model. If there is a filed speed up to *maxPistonSpeed*, then this is *unknownPiston*; if there is a filed speed above this but no greater than *maxTurbopropSpeed*, it is *unknownTurboprop*; otherwise it is *unknownJet*.

The set *fwpNASDeletedFlights* is those flights that have been NAS deleted. It is used by Recognised Flights.

$$
\begin{array}{l}
\textit{FWPFlightsDerivedAssociations} \\
\hline
\textit{FWPFlightsAssociations} \\
\textit{fwpFlightAircraftType} : \textit{FLIGHT} \rightarrow\!\!\!\!\rightarrow \textit{AIRCRAFTTYPENAME} \\
\textit{fwpFlightPerformanceModel} : \textit{FLIGHT} \rightarrow\!\!\!\!\rightarrow \textit{PERFORMANCEMODEL} \\
\textit{fwpNASDeletedFlights} : \mathbb{F}\ \textit{FLIGHT} \\
\hline
\textit{fwpFlightAircraftType} = \{ f : \textit{fwpFGFlights} ; a : \textit{aircraftTypes} \\
\quad \mid (\textit{fwpFlightState}\ f).\textit{aircraftTypeName} = a \bullet f \mapsto a \} \\
\textit{fwpFlightPerformanceModel} = \\
\{ f : (\textit{fwpFGFlights} \setminus \textit{fwpBlockers}); \textit{model} : \textit{PERFORMANCEMODEL} \mid \\
\quad (\mathbf{let}\ \textit{speed} == (\textit{fwpFlightState}(f)).\textit{filedSpeed} \bullet \\
\quad\quad \textit{model} = \mathbf{if}\ \textit{the speed} \leqslant \textit{maxPistonSpeed}\ \mathbf{then}\ \textit{unknownPiston} \\
\quad\quad\quad\quad \mathbf{else\ if}\ \textit{the speed} \leqslant \textit{maxTurbopropSpeed}\ \mathbf{then}\ \textit{unknownTurboprop} \\
\quad\quad\quad\quad \mathbf{else}\ \textit{unknownJet}) \} \\
\quad \oplus (\textit{fwpFlightAircraftType} \mathbin{;} \textit{typePerformanceModel}) \\
\textit{fwpNASDeletedFlights} = \{ f : \textit{fwpFGFlights} \mid (\textit{fwpFlightState}\ f).\textit{nasDeleted} = \textit{True} \}
\end{array}
$$

Z training

- Z reader training
  - 3 day course; fluency then comes after 1 week on the job.
  - We have trained 75 people to read Z.
  - Engineers, domain experts, ATCOs.
- Z writer training
  - 3 day course, fluency then comes after 3 months on the job.
  - We have trained 11 people to write Z.
  - All engineers.

Z tools

- Z written in Microsoft Word
  - To get acceptance, you need to work with what people know.
  - Supported by Word Add-ins.
    - A Z character set.
    - A simple interface to the fuzz type checker.
    - A graphical representation tool.

Z tools

- Advantages
  - Easy to develop commentary and Z together.
  - Hyper linking of fuzz errors back to source.
  - Cross-referencing of Z names in final document.
- Disadvantages
  - All the problems of large word documents.
  - Tools can be slow on 1000 page documents.
  - Merging branches is painful.

The state machine specification

	Button 1	Checkbox 1
State 1	State 2	N/A
State 2	State 1	State 3
State 3	State 1	State 2

Transition Actions

State 1 -> State 2 : De-select Checkbox 1

State machine training & tools

- Training
  - So trivial that we don't train!
  - People "just get it".
- Tools
  - Err …. None.

Agenda

- A quick introduction
  - What is iFACTS?
- Formal methods for Specification
  - Z, State machines.
- Formal methods for Implementation
  - Implementation: SPARK.
- Formal methods for Test
  - Verification: more Z, Mathematica.

The SPARK Implementation

- SPARK Ada
  - An annotated subset of Ada.
- 150 KSLOC (Logical)
- RTE (Run-Time Exception) Proof
  - Formal partial correctness proof against specification not considered cost-effective.

Code

```
function Segment_Group_FL_Occupancy
  (Segs_Group : PIO_Data.Segment_Group_Array_T;
   Quantity   : PIO_Data.Trajectory_Index_T)
   return Altitudes.Level_Range
is
   The_Range  : Altitudes.Level_Range;
   Temp_Range : Altitudes.Level_Range;
begin

   --  By virtue of the fact that this procedure has been called means
   --  that the level ranges must be populated so set to a senseless
   --  null value guaranteed to be overwritten
   The_Range.Lower := Altitudes.Flight_Level_T'Last;
   The_Range.Upper := Altitudes.Flight_Level_T'First;

   for Idx in PIO_Data.Trajectory_Index_T range 1 .. Quantity loop
      --# assert Quantity = Quantity%;

      --  Must have a standard occupancy at the very least so check for that
      if MTCD_Types.Get_Standard_Occupancy (Segs_Group (Idx)).Exists then

         Temp_Range := Segment_FL_Occupancy (Segs_Group (Idx));

         The_Range.Lower :=
           Altitudes.Flight_Level_T'Min (The_Range.Lower,
                                         Temp_Range.Lower);

         The_Range.Upper :=
           Altitudes.Flight_Level_T'Max (The_Range.Upper,
                                         Temp_Range.Upper);
      end if;

   end loop;

   return The_Range;

end Segment_Group_FL_Occupancy;
```

SPARK Training

- 57 people trained in SPARK
  - Mostly contractors and clients.
  - Diverse programming background.
  - All SPARK coders are also Z readers.
- Effective as SPARK coders immediately
- Picking up RTE proof takes longer.
  - About 2 months.
- How long to pick up formal correctness proofs?
  - No data, but I suspect longer again.

SPARK Tools

- The SPARK toolset
  - Examiner.
  - Proof Simplifier.
  - Proof Checker.

Agenda

- A quick introduction
  - What is iFACTS?
- Formal methods for Specification
  - Z, State machines.
- Formal methods for Implementation
  - Implementation: SPARK.
- Formal methods for Test
  - Verification: more Z, Mathematica.

Test Design

2.2.1.18 TPDeviationRequests

Summary

Requests the required deviation trajectories.

This is a non-conditional schema.

Partitions

There are two equivalence classes:

1 Flight is not radar supported, so no information.

2 Flight is radar supported.

The output condition in the first equivalence class is that there is no request. This can also occur when there are no deviation trajectories, so that input condition should be tested as well.

It is stated within the FPM process specification that the number of deviation requests will be either none, one or two ([4] section 13.2.13.2). We should test for each of these conditions separately (since 0 and 2 are boundary conditions).

Test Conditions

TPDeviationRequests	1	2	3	4
$fpmData!? = nil$	●	○	○	○
$(the\ fpmData!?).fpmDeviationTrajectories = \varnothing$		●	○	○
$deviationReqs = \varnothing$	●	●	○	○
$\#deviationReqs = 1$	○	○	●	○
$\#deviationReqs = 2$	○	○	○	●

The Challenge of Test Design

$$
\begin{array}{l}
\textit{TPRemoveMultiplePIOs} \\
\hline
\Delta TP \\
tpFlights!? : \mathbb{P}\,FLIGHT \\
piosToRemove : \mathbb{P}\,PIO \\
\hline
\exists\, deletedDirectPIOs, deletedGroupedPIOs : \mathbb{P}\,PIO \mid \\
\quad deletedDirectPIOs = flightPIO \llparenthesis\, tpFlights!? \,\rrparenthesis \cap piosToRemove \\
\quad \wedge\, deletedGroupedPIOs = pioPIOGroup^{\sim} \llparenthesis\, deletedDirectPIOs \,\rrparenthesis \bullet \\
\qquad flightPIO' = flightPIO \rhd deletedDirectPIOs \\
\qquad \wedge\, pioPIOGroup' = pioPIOGroup \rhd deletedDirectPIOs \\
\qquad \wedge\, pioGroupDisplayPIO' = deletedDirectPIOs \lhd pioGroupDisplayPIO \\
\qquad \wedge\, pioState' = (deletedDirectPIOs \cup deletedGroupedPIOs) \lhd pioState \\
nominalVerticalProfiles' = \\
\quad \mathbf{if}\ fwpHookedFlight \cap tpFlights!? \neq nil \\
\quad \mathbf{then}\ nil\ \mathbf{else}\ nominalVerticalProfiles
\end{array}
$$

How many potential tests for this fragment?

The Challenge of Test Design

- If you just turn the handle there are 1134 conditions to test.
- But if you work at it hard enough you can cover the required subset in just 6 test scripts.
- Formal methods are not a substitute for initiative.

Test reference models

- Algorithms are specified in pure mathematics.
  - Working out the expected answer for test cases is very difficult and error prone.
- We generate test cases as usual.
- We create a test reference implementation in Mathematica.
- We do back-to-back testing of iFACTS against the reference.
  - Diverse tools and implementers reduce the possibility of a common failure.

Mathematica tools & training

- Small team – only 5 trained.
- Reference model has similar defect density to SPARK implementation.
- Limited conclusions to draw from such a small activity.

Conclusions

- Formal methods are applicable to all phases of the lifecycle.

- Training engineers is not a barrier
  - It's a one-off cost
  - Our data shows that training is easy and cheap.

- Tool support is vital
  - The Achilles heel of formal methods
    - Except the SPARK Examiner!

Altran Praxis Limited

20 Manvers Street
Bath BA1 1PX
United Kingdom

Website: www.altran-praxis.com

Tracing

- Completeness of coverage
  - e.g., testing all parts of a Z specification
- DOORS tool
  - Integrate Systems Engineering
- Link all specification components with test case(s) or argument for safety case
- Flag unlinked components
- Also visualization of schema structure

www.integrate.biz/casestudies/BusinessGoalAlignment.aspx

More recent developments

- iFACTS in maintenance phase
- Traffic Load Prediction Device (TLPD)
- Forecast air traffic load up to 4 hours ahead
- Plan workloads for optimum traffic flows

www.altran.co.uk/uksolutions/ecs/sectors/air-traffic-management.html

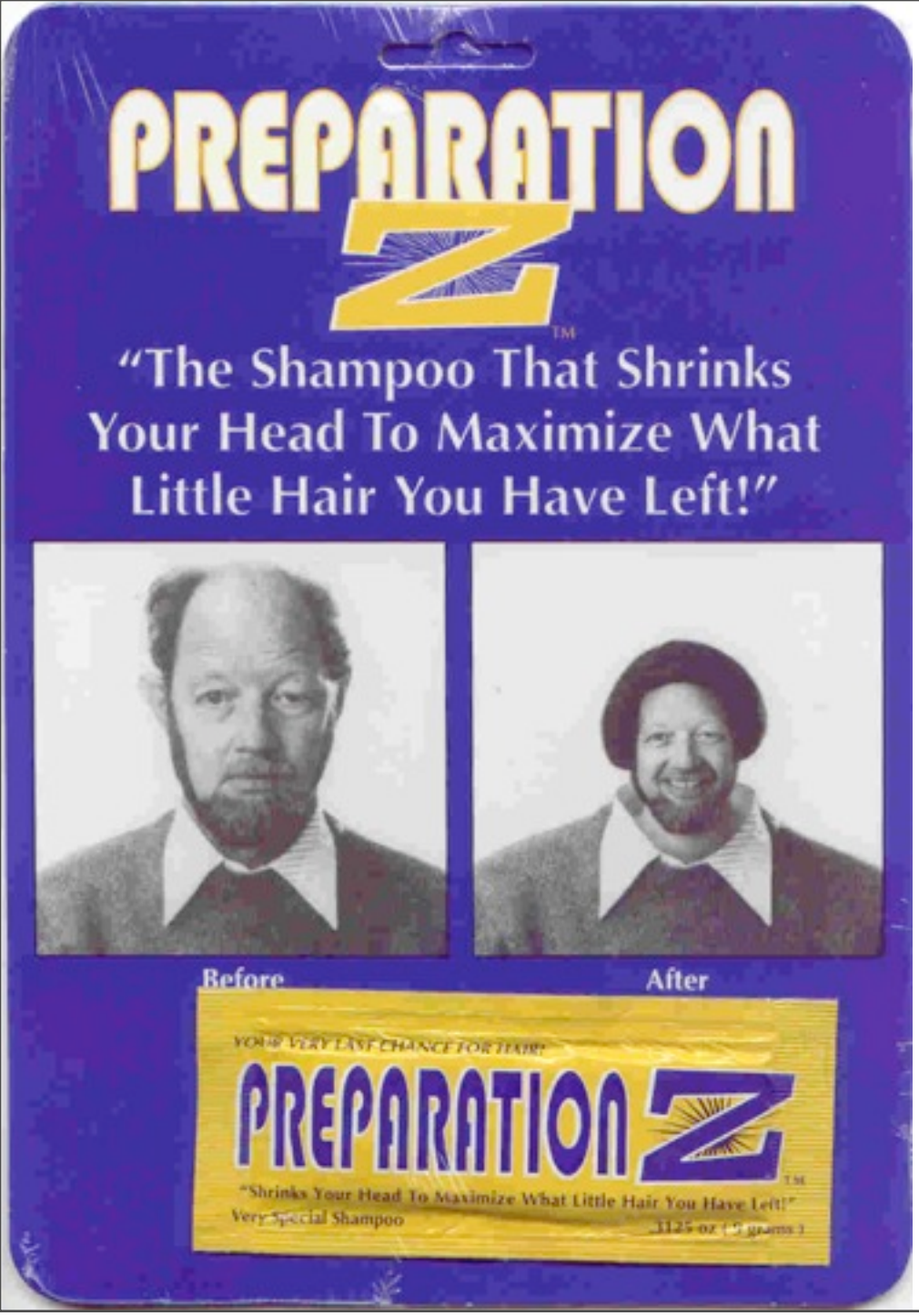
Reflection

Oui, l'ouvre sort plus belle
D'une forme au travail
Rebelle,
Vers, marbre, onyx, émail.

[Yes, the work comes out more beautiful from a material that resists the *process*, verse, marble, onyx, or enamel.]

— Théophile Gautier (1811–1872) *L'Art*

Beware Panaceas!

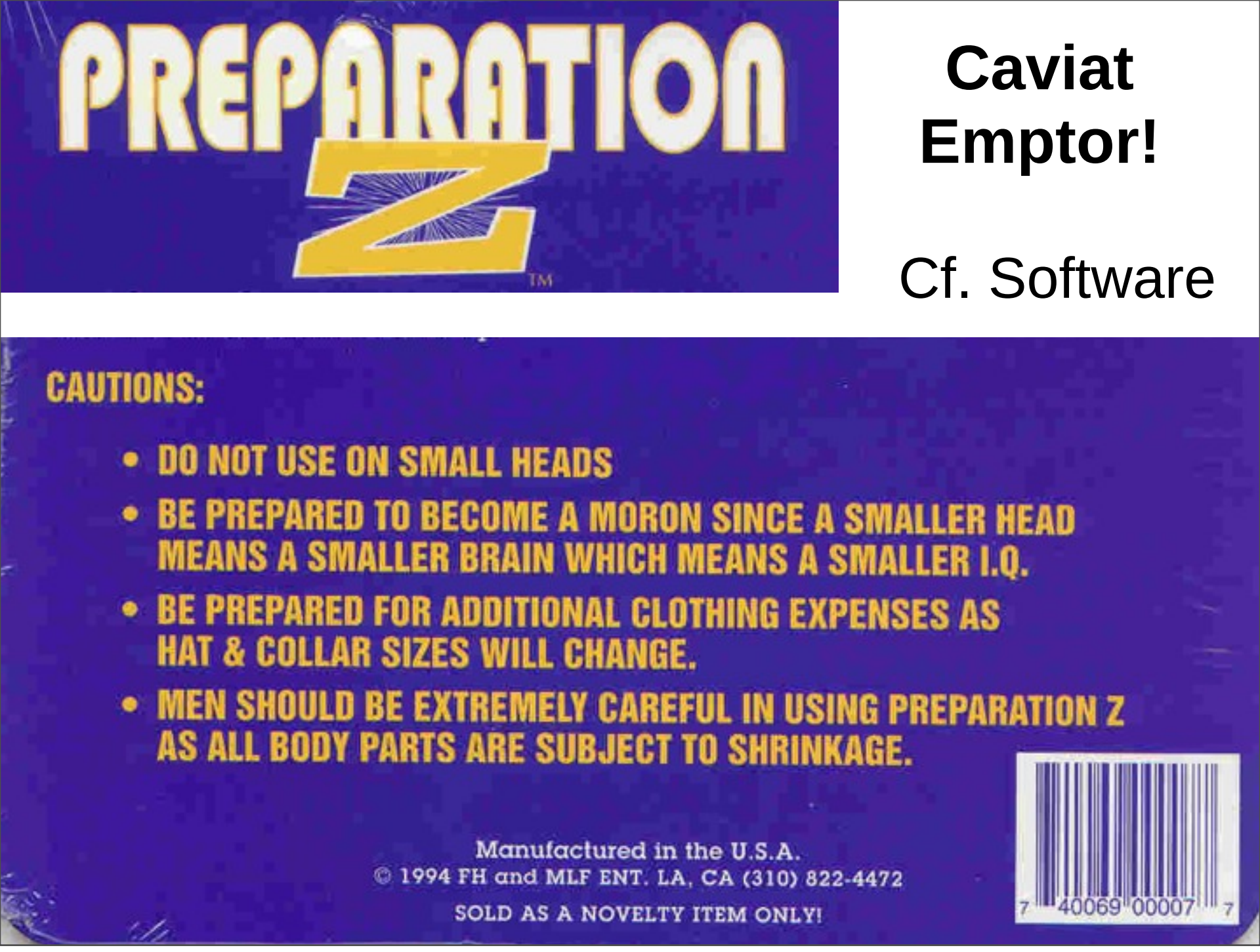


Beware Panaceas!

Cf. Formal methods

Caviat Emptor!

Caviat Emptor!

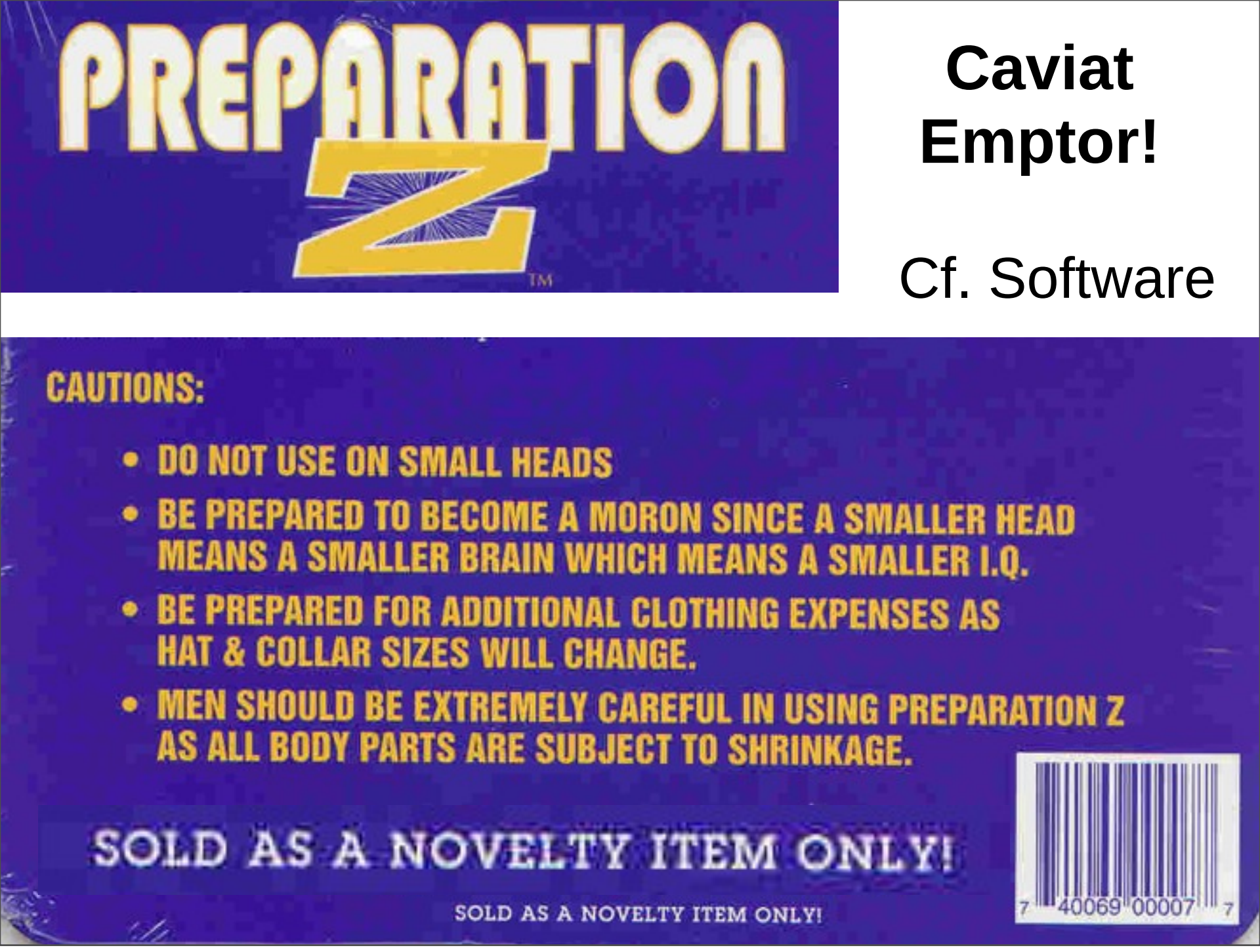


Caviat Emptor!

Cf. Software

Caviat Emptor!

Cf. Software



The Industrial Use of Formal Methods: Experiences of an Optimist

Prof. Jonathan P. Bowen

Museophile Limited, Oxford, UK
London South Bank University
www.jpbowen.com
jonathan.bowen@lsbu.ac.uk