Analysis of system behaviour using the mCRL2 toolset

Aad Mathijssen

Design and Analysis of Systems group
Laboratory for Quality Software (LaQuSo)

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

Bits&Chips 2008 Embedded Systemen
Evoluon, Eindhoven

9th October 2008

Introduction

Software analysis techniques

Software is inherently complex and therefore expensive.

Analysis techniques help to get software under control.

Analysis techniques used in software development:

- Structure: what *things* are in the system?
- Behaviour: what *happens* in the system?

The two techniques complement each other because they focus on *different aspects* of the system.

Behavioural analysis is less used in practice than structural analysis.

Analysis of system behaviour

What is it about?

What is analysis of system behaviour about?

- Modelling: create an *abstract* model of the *behaviour* of the system
  - *gain insight* in the behaviour
  - *reduce complexity* to allow for validation and verification
- Validation: are we building the right product?
  - *simulate* the model
  - *test requirements* on the model for a number of paths and configurations
- Verification: are we building the product right?
  - *verify requirements* on the model for all possible paths and configurations

Analysis of system behaviour

Use

Analysis of system behaviour is used to *assess* and *improve* the quality of software:

- Error prevention: *prove* that the design does not have fundamental flaws
- Error detection: find errors and their *causes* in the design or implementation of a system

Analysis of system behaviour

Tool support

For analysing system behaviour in *industry*,
tool support is essential.

Tools for analysing system behaviour:

- I-Mathic (Imtech ICT, The Netherlands)
- ASD (Verum, Waalre, The Netherlands)
- FDR (Formal Systems Limited, Oxford, UK)
- CADP (INRIA Rhone Alpes, France)
- mCRL2 (OAS group, TU/e, The Netherlands)
- Uppaal (Uppsala University, Sweden)
- SPIN (Bell Labs, USA)
- ...

mCRL2 toolset

Goals

Goals of the mCRL2 toolset:

- Generic basis for the analysis of system behaviour
- Research and development of verification techniques
- Industrial application of verification techniques

mCRL2 toolset

Overview

Overview of the mCRL2 toolset:

- 20 years of history:
  - Late 1980s: Common Representation Language (CRL)
  - From 1990: μCRL
  - During 1990s: μCRL toolset
  - From 2004: mCRL2 and mCRL2 toolset
- Collection of tools for modelling, validation and verification of system *behaviour*
- External languages and tools are supported: μCRL, CADP, χ, PNML, TorX, LySa, SystemC
- Multi-platform: Windows, Mac and UNIX variants
- Free software licence: Boost licence
- Release policy: fixed release cycle (January and July)

mCRL2 toolset

Modelling: ingredients

Ingredients for modelling:

- *Actions* (push_button, place_order, call_f)
- *Non-deterministic choice*
  (*either* push_button *or* place_order)
- *Sequence* (*first* push_button, *then* place_order)
- *Processes* (Client, WebShop)
- *Parallelism* (Client *in parallel with* WebShop)
- *Synchronous communication*
  (push_button *communicates with* place_order)
- *Data types*
  (push_button(on), Client(1), call_f($\{x|prime(x)\}$))

mCRL2 toolset

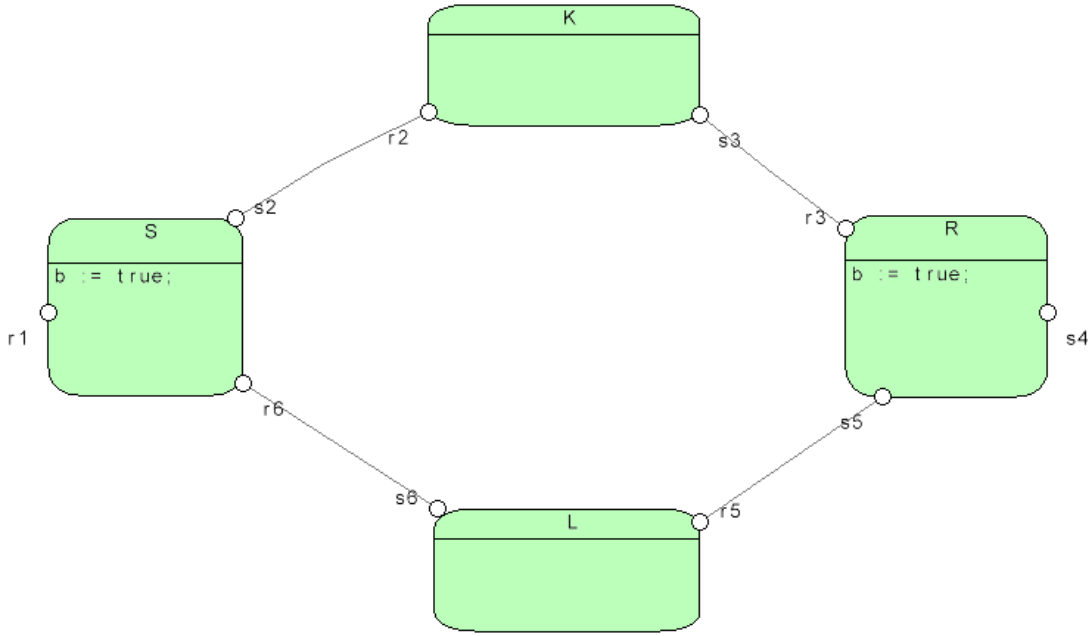
Modelling: textual and graphical

The toolset supports two kinds of modelling:

- *Textual*:

  init $\nabla_{\{r1,s4,c2,c3,c5,c6,i\}}(\Gamma_{\{r2|s2\rightarrow c2,r3|s3\rightarrow c3,r5|s5\rightarrow c5,r6|s6\rightarrow c6\}}($
  $\mathsf{S}(true) \parallel \mathsf{K} \parallel \mathsf{L} \parallel \mathsf{R}(true)$
  $));$

- *Graphical*:



mCRL2 toolset

Validation

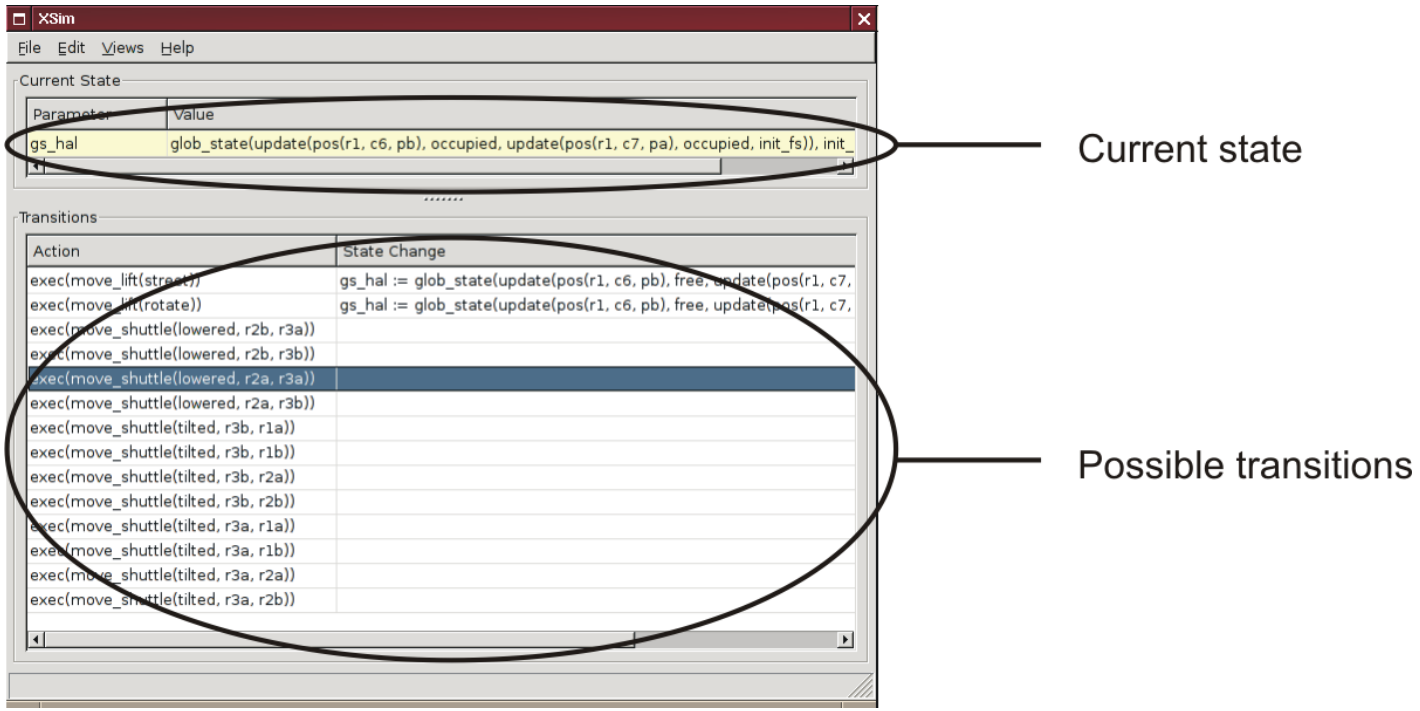
Validation of models supported by the toolset:

- Manual or semi-automatic simulation
- Automated testing using the TorX test tool
- Different types of visualisation

mCRL2 toolset

Validation: simulation

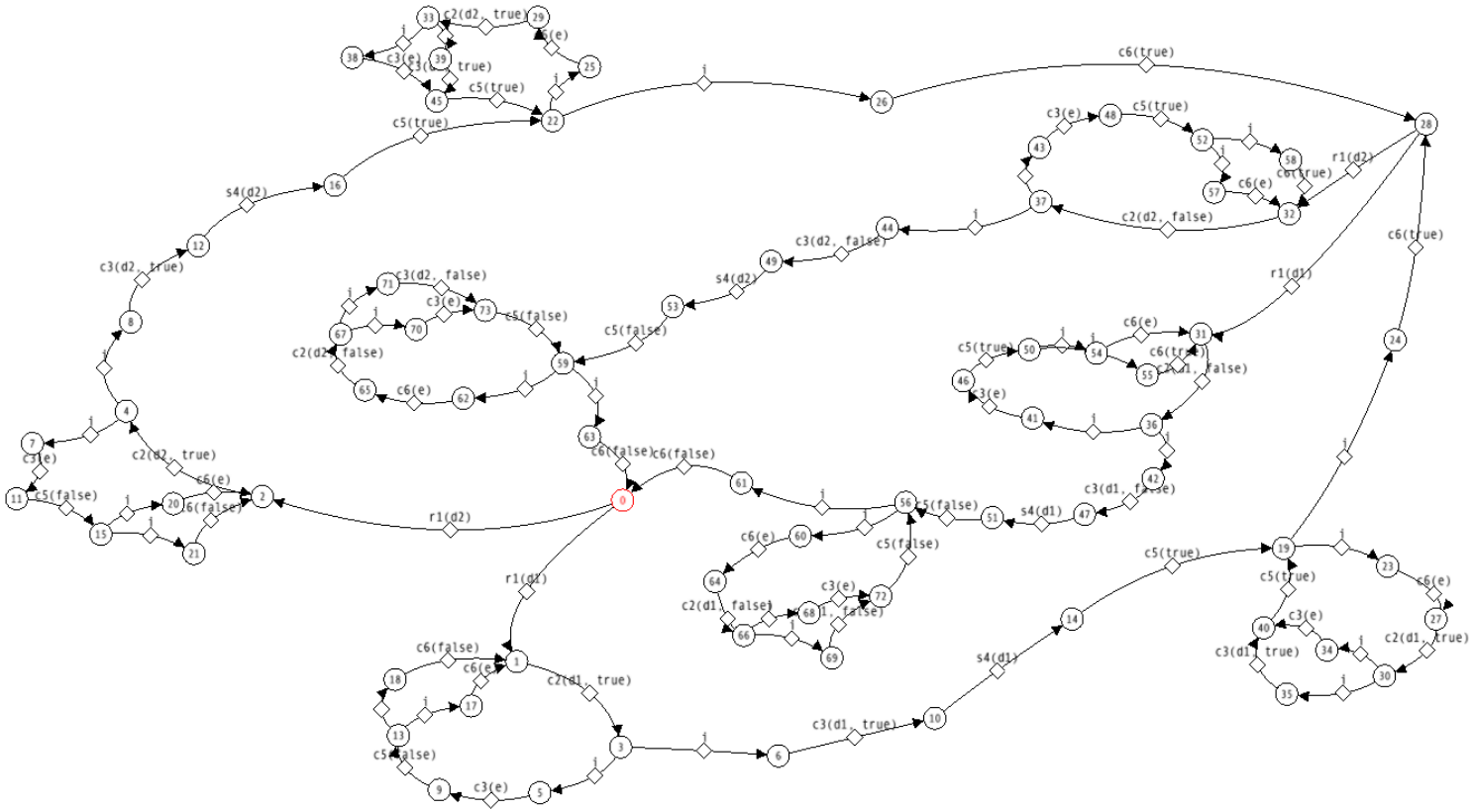
Simulation:



mCRL2 toolset

Validation: visualisation

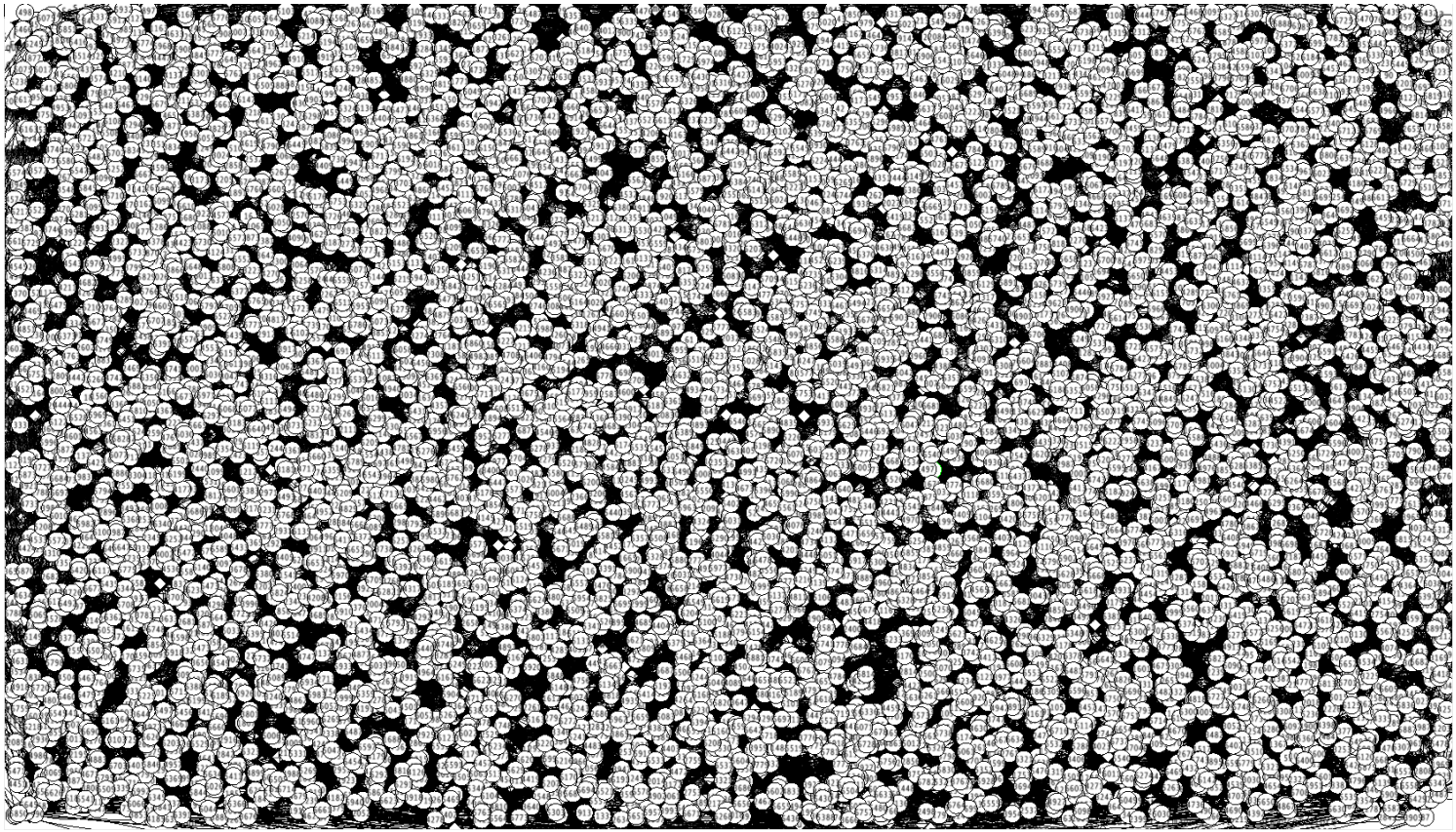
Visualisation as a directed graph using *automatic positioning*:



mCRL2 toolset

Validation: visualisation

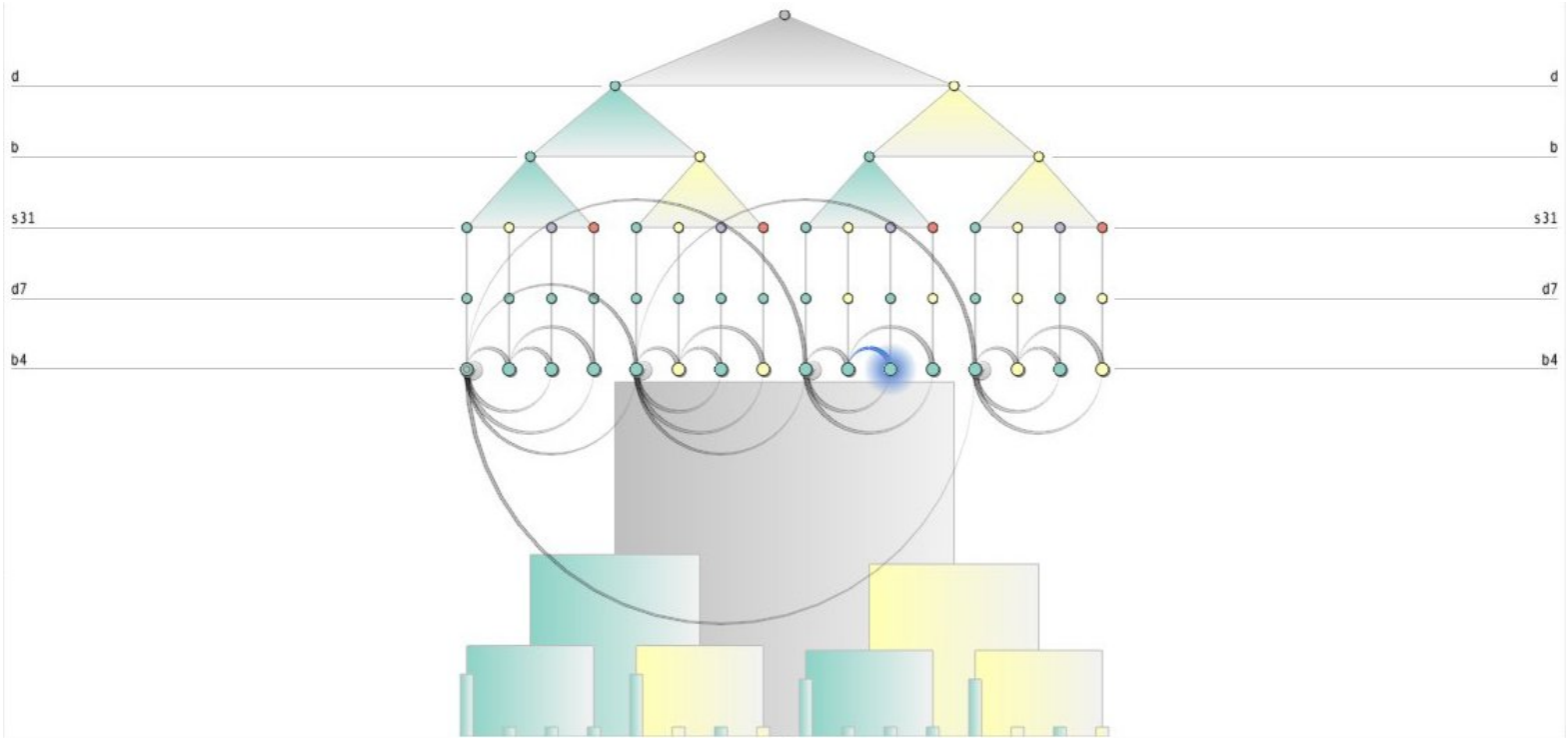
Visualisation as a directed graph is limited to *small models*:



mCRL2 toolset

Validation: visualisation

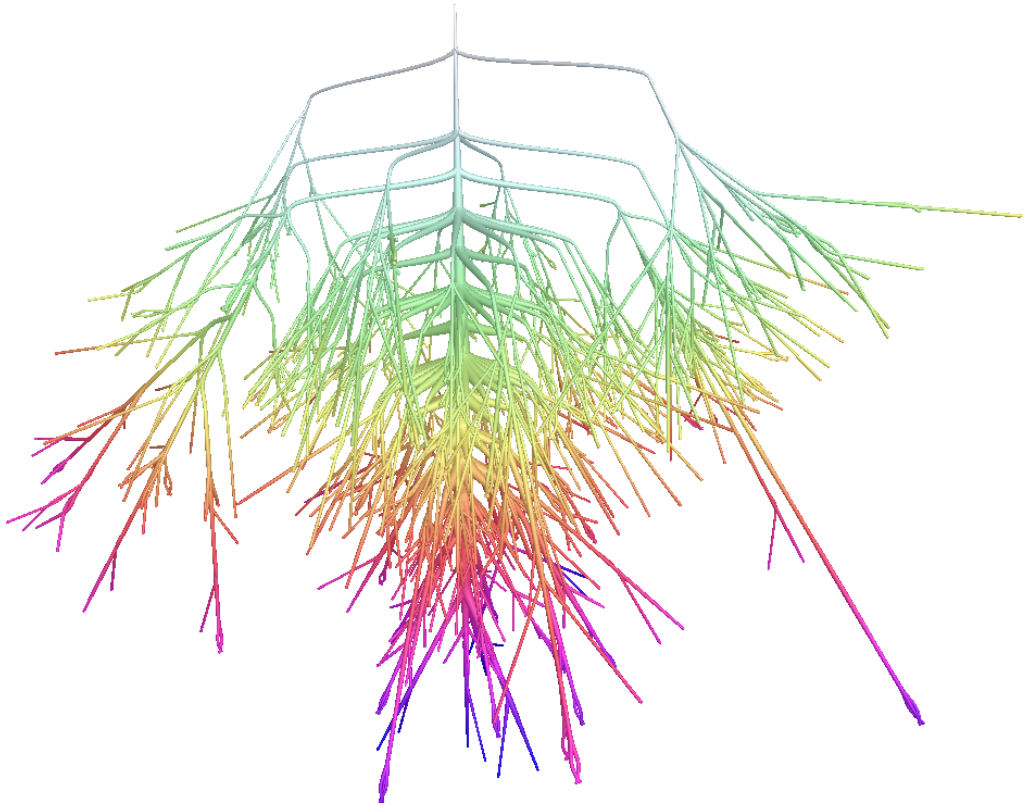
Visualisation as a graph of clusters of states:



mCRL2 toolset

Validation: visualisation

Visualisation as a 3D tree of clusters of states:



mCRL2 toolset

Verification

Toolset support for automated verification
of requirements on the complete model:

- Occurrences of *deadlocks*
- Occurrences of specific *actions*
- *Equivalence* of models
- *Formula checking*:
  - express requirements as *temporal logic formulas*
  - check these formulas on the model

Industrial case studies

Selection of industrial case studies performed using the μCRL and mCRL2 toolsets:

- *Error prevention*:
  - Vitatron: artificial *pacemaker*
- *Error detection*:
  - Add-controls: distributed system for *lifting trucks*
  - Océ: automatic *document feeder*
  - AIA: ITP *load balancer*

Industrial case studies

Vitatron: artificial pacemaker

Artificial pacemaker:

- Must be able to deal with:
  - all possible *heart rates*
  - all possible *arrhythmias*
- *Design* of the firmware

Analysis:

- Model: mCRL2 (also Uppaal)
- Verification: formula checking
- Size:
  - full model: 500 million states
  - suspicious part: 714.464 states
- Actual errors found: 1 (known)

Industrial case studies

Add-controls: distributed system for lifting trucks

Distributed lifting system for trucks:

- Each lift has its *own controller*
- Controllers are connected via a *ring network*
- 3 errors found after *testing* by the developers

Analysis:

- Model: μCRL
- Verification: formula checking
- Actual errors found: 4

Lifts	States	Transitions
2	383	716
3	7.282	18.957
4	128.901	419.108
5	2.155.576	8.676.815

Industrial case studies

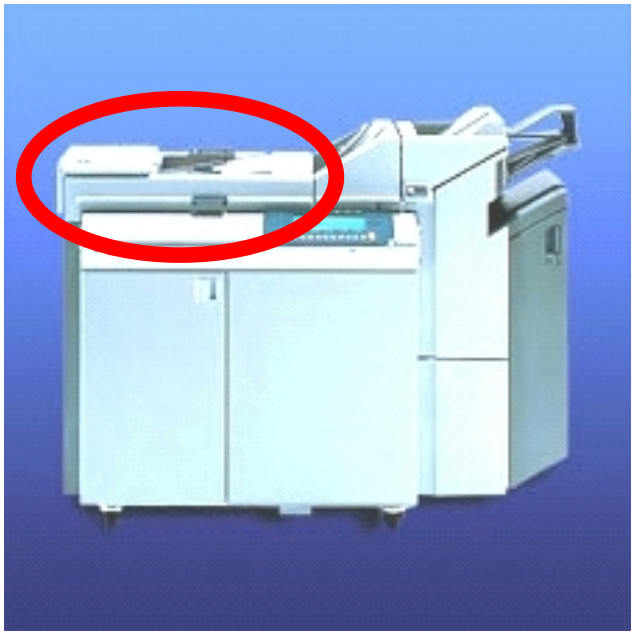
Océ: automatic document feeder

Automatic document feeder:

- Feeds documents to the scanner automatically
- 1 sheet at a time
- Prototype design

Analysis:

- Model: μCRL
- Verification: formula checking
- Size:
  - 350.000 states
  - 1.100.000 transitions
- Actual errors found: 2



Industrial case studies

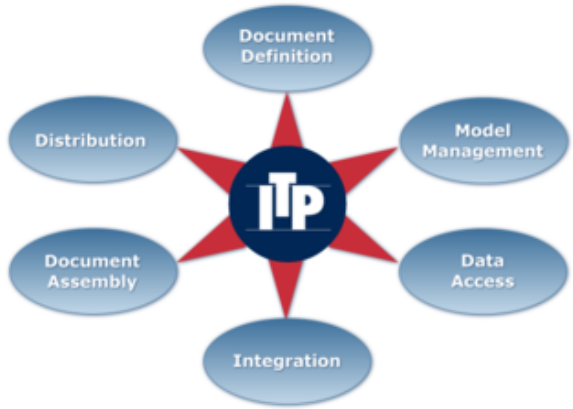
AIA: ITP load balancer

Intelligent Text Processing (ITP):

- Distribution of print jobs over document processors
- 7.500 lines of C code

Analysis:

- Focus: load balancing
- Model: mCRL2
- Verification: formula checking
- Actual errors found: 6
- Size:
  - 1,9 billion states
  - 38,9 billion transitions
- LaQuSo certification



Conclusions

Behavioural analysis *complements* structural analysis.

The mCRL2 toolset:

- *supports* many aspects of the analysis of system behaviour
- can be *used* to:
  - *detect errors* in the design or implementation of software
  - *prevent errors* already in the design of software

Preventing errors in the design *shortens* time spent on software development:

- *more* time spent on design
- *less* time spent on implementation and maintenance

Thank you for your attention

More information can be found on `mcrl2.org`.