Design and Analysis of Embedded Software

Aad Mathijssen

Design and Analysis of Systems Group
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

TU/e and Philips Healthcare meeting
Technische Universiteit Eindhoven

July 2nd, 2008

Design and Analysis of Embedded Software

The group

Some information about the group:

- Design and Analysis of Systems group
- Headed by prof.dr.ir. Jan Friso Groote
- Personnel: 11 staff, 6 PhD students

Design and Analysis of Embedded Software

Group focus

Main analysis techniques used in hardware/software development:

- Structural analysis: what *things* are in the system
  - Class diagrams
  - Component diagrams
  - Package diagrams
  - ...
- Behavioural analysis: what *happens* in the system
  - State diagrams
  - Message sequence charts
  - Petri nets
  - Process algebra
  - Temporal logic
  - ...

Design and Analysis of Embedded Software

Behavioural analysis

What is behavioural analysis about?

- Modelling:
  - Create an abstract model of the behaviour
- Validation and Verification:
  - Validation: does the model roughly behave as expected?
  - Verification: does the model satisfy the requirements in all states?

Design and Analysis of Embedded Software

Behavioural analysis (2)

Behavioural analysis is applicable to all phases of the software lifecycle:

- Requirements Analysis and Design:
  Prove that the design satisfies the requirements before anything is built.
- Implementation to Maintenance:
  Prove that the software satisfies the requirements in a rigorous way.

Design and Analysis of Embedded Software

Modelling

Why modelling?

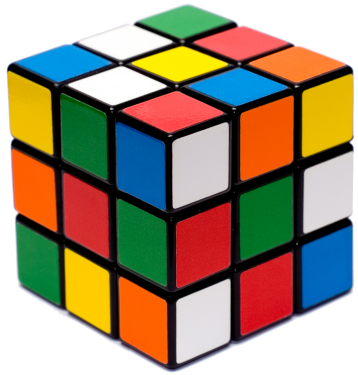
Design and Analysis of Embedded Software

Modelling

Why modelling?

To reduce complexity:

- Direct verification of all states of the software is impossible due to the huge number of states.
- Much more complex than e.g. Rubik's cube:



$43,252,003,274,489,856,000$ (4.3×10^{19}) states

Design and Analysis of Embedded Software

Modelling (2)

From our experience:

- Without proper modelling it is impossible to get a system right.
- Implementing a model does not introduce substantial flaws.
- Modelling an implementation nearly always reveals flaws.

Design and Analysis of Embedded Software

Tool support

For verification of *industrial* systems, tool support is essential.

Main toolsets for modelling, validation and verification of behaviour:

- CADP (INRIA Rhone Alpes, France)
- SPIN (Bell Labs, USA)
- FDR (Formal Systems Limited, Oxford, UK)
- Uppaal (Uppsala University, Sweden)
- mCRL2 (OAS group, TU/e)

Design and Analysis of Embedded Software

The mCRL2 toolset

The mCRL2 toolset:

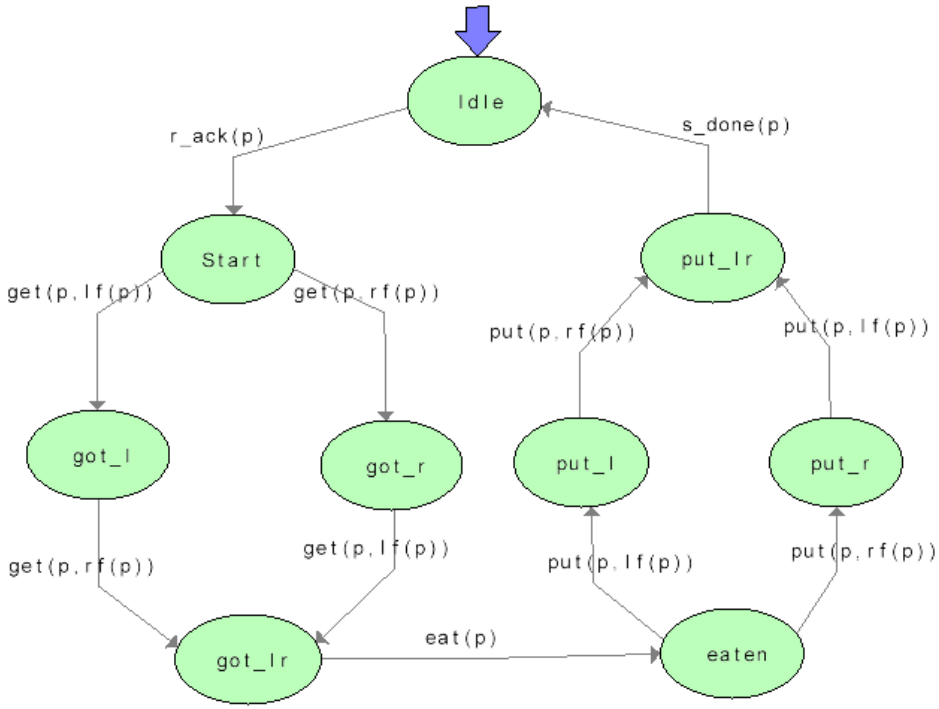
- mCRL2: micro Common Representation Language 2
- Developed since 2004
- Built on the ideas of the μCRL toolset (since 1990)
- Supports the complete behavioural analysis approach:
  - from modelling
  - through validation
  - to verification

Design and Analysis of Embedded Software

The mCRL2 toolset: modelling

Two types of modelling:

- Textual specification
- Graphical specification (individual component)

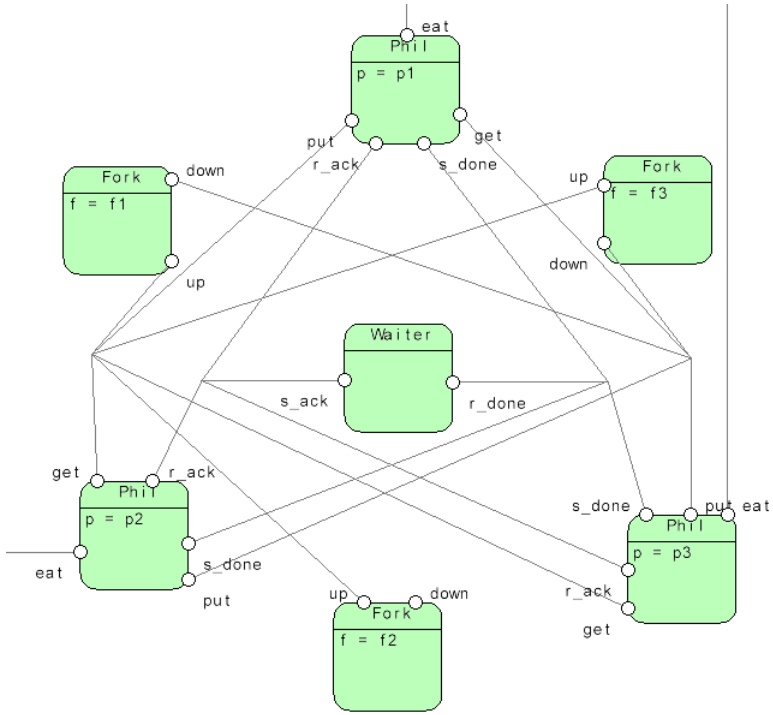


Design and Analysis of Embedded Software

The mCRL2 toolset: modelling

Two types of modelling:

- Textual specification
- Graphical modelling (communicating components)

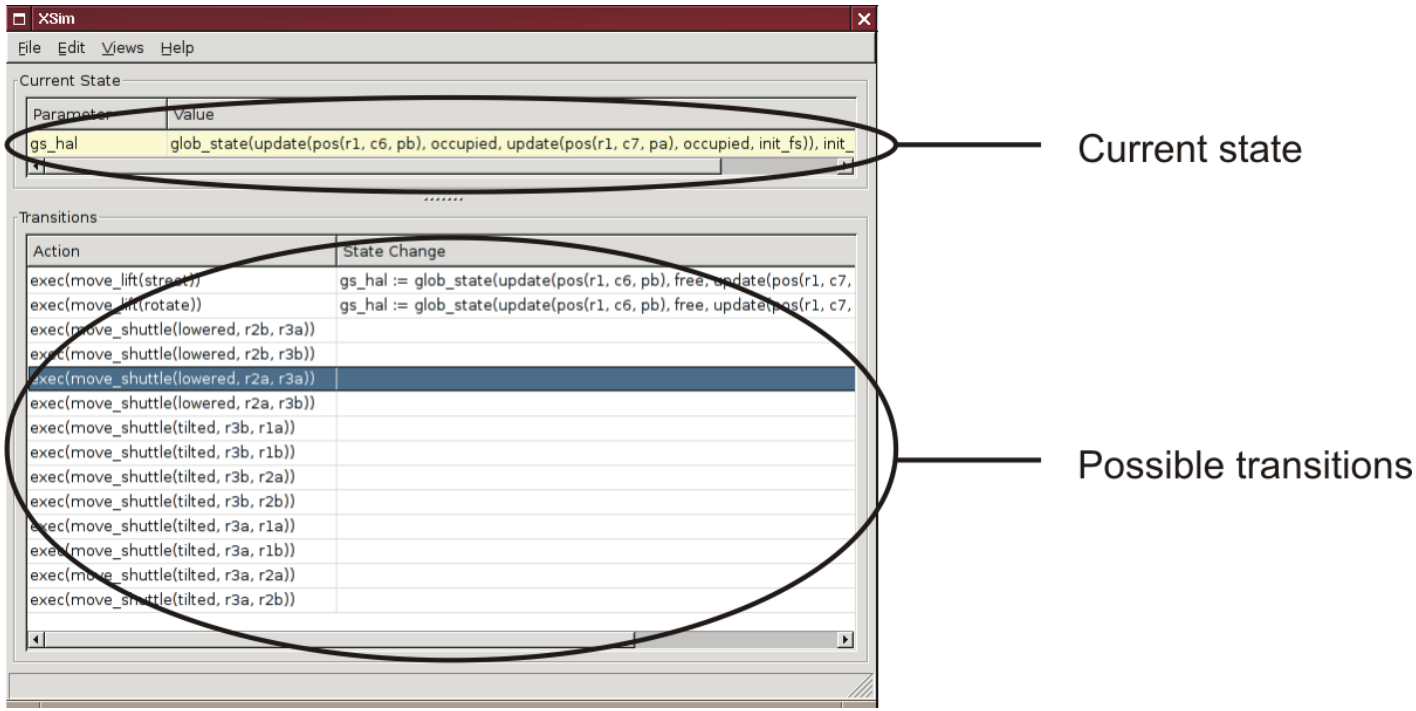


Design and Analysis of Embedded Software

The mCRL2 toolset: validation and verification

Validation and verification tools:

- Simulation

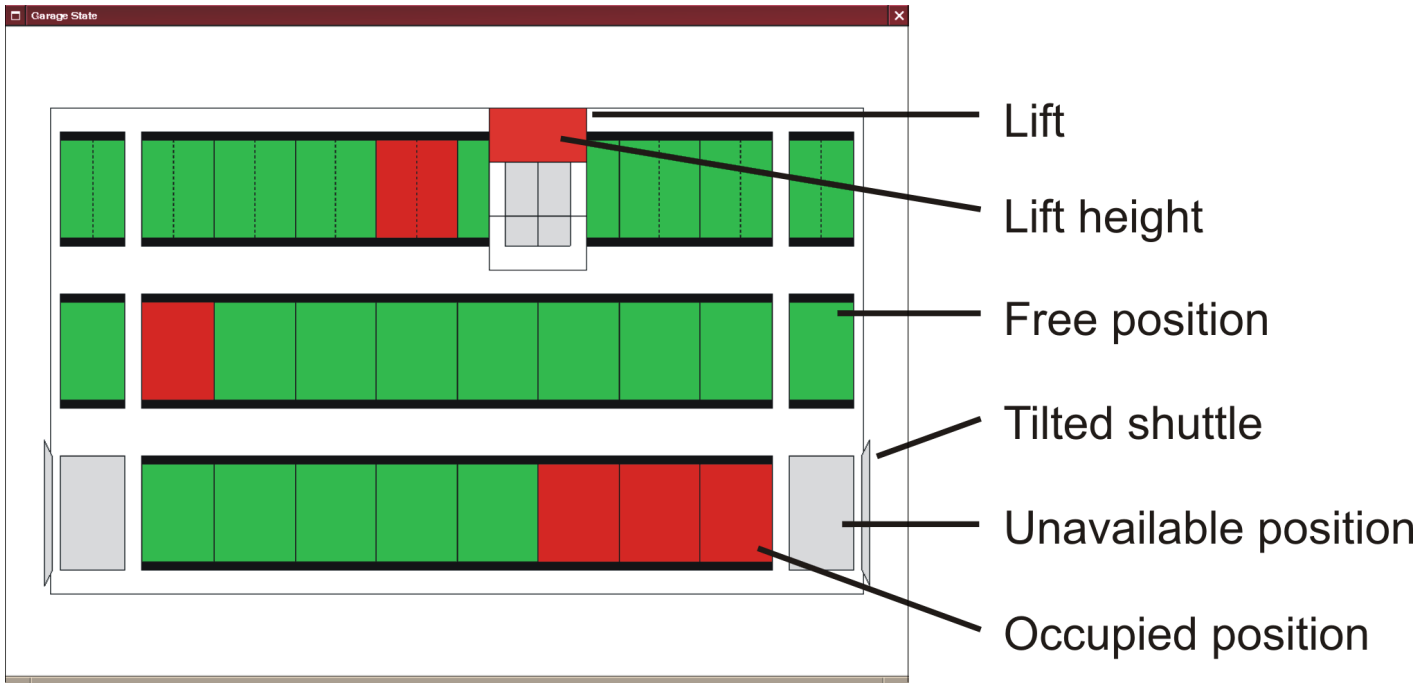


Design and Analysis of Embedded Software

The mCRL2 toolset: validation and verification

Validation and verification tools:

- Simulation with plugins



Design and Analysis of Embedded Software

The mCRL2 toolset: validation and verification (2)

Validation and verification tools:

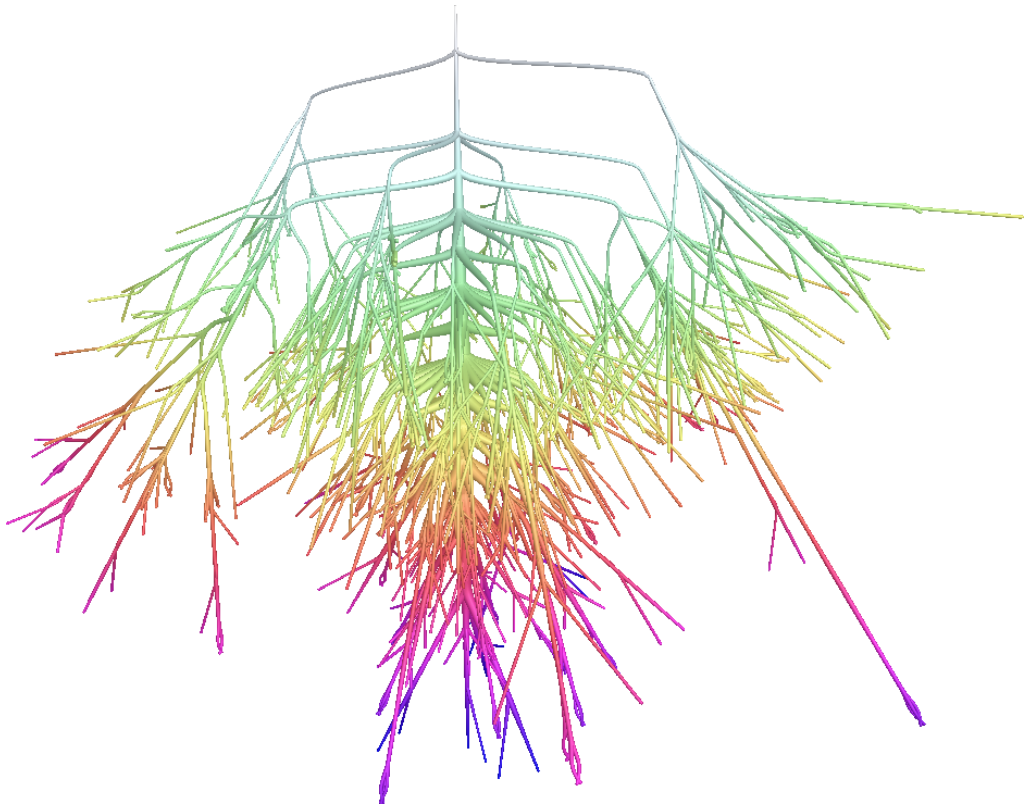
- Model checker:
  prove that the requirements hold for all states of the model

Design and Analysis of Embedded Software

The mCRL2 toolset: validation and verification (3)

Validation and verification tools:

- State space visualisation:



Design and Analysis of Embedded Software

The mCRL2 toolset: industrial case studies

Some industrial case studies carried out using the μCRL and mCRL2 toolsets:

- Add-controls: distributed system for lifting trucks
- Vitatron: pacemaker
- Philips Healthcare: patient support platform

Design and Analysis of Embedded Software

The mCRL2 toolset: industrial case studies (2)

A distributed system for lifting trucks (Add-controls):

- Each lift has a controller
- Controllers are connected via a circular network
- 3 errors found after testing by the developers

Analysis:

- Model: μCRL
- Verification: CADP
- 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

Design and Analysis of Embedded Software

The mCRL2 toolset: industrial case studies (3)

Embedded software of a pacemaker (Vitatron):

- Controlled by firmware
- Must deal with all possible rates and arrhythmias
- Firmware design

Analysis:

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

Design and Analysis of Embedded Software

The mCRL2 toolset: industrial case studies (4)

Patient support platform (Philips Healthcare):

- Verified design of the control software
- Convertor and Motion Controller
- Implemented in Python

Analysis:

- Model: mCRL2
- Verification: CADP
- Requirements:
  - 4 checked
  - 1 did not hold but was very unlikely to occur
- Size: 45 million states

Design and Analysis of Embedded Software

The mCRL2 toolset

More information on the mCRL2 toolset:

`http://mcrl2.org`