Behavioural visualization of model based systems

Frank Stappers

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

Model based development
Track: Modelling and Simulation
Evoluon, Eindhoven, The Netherlands

October 09, 2008
Personalia

The speaker

- Name: Frank Stappers
- History: Studied Computer Science
- Current position: PhD Student at Technische Universiteit Eindhoven
- Research Area: Formal verification of multi-disciplinary systems.
- Project: ITEA TWINS: Optimizing HW-SW Co Design Flow for Software Intensive System Development
Outline

1. Introduction
2. Preliminaries
3. Approach
4. Show Case
5. Conclusions
Outline

1. Introduction
2. Preliminaries
3. Approach
4. Show Case
5. Conclusions
Introduction

Problems

Current trends show that industrial systems are being development in a multi-disciplinary environment.

- Different disciplines = different methods, techniques and tools

Developing a multi-disciplinary system is a difficult trajectory

- Incompatible information resources,
- Hard to exchange information between disciplines,
- Difficult to predict system behaviour:
  - Concurrent behaviour
  - Exceptional cases can easily be missed
  - Forgotten behaviour
Introduction
Range of Solutions

Guard coherence and tune mono-disciplinary designs.

- Designs need to be kept **native**, as they can be **exchanged** among other disciplines with a clear and intuitive **method**.

This can be accomplished by:

- Translating commonality in mono-disciplinary models between disciplines.
- Merging mono-disciplinary models to multi-disciplinary models.
Introduction
Suggested Solution

Method for virtual model based design or virtual prototyping:

*Simulation of a modelled object or system with a high degree of realism compatible with:*

1. physical and  
2. logical functionality  

*by combining different kind of models*

A (visual) choice:

1. Physical design: Drawing - preferably original CAD design  
2. Logical functionality: Behavioural specification language - mCRL2
Preliminaries

mCRL2

- The mCRL2 language is a discrete-event modelling language that allows multi-actions, data and time.
- The mCRL2 toolset can be used for modelling, validation and verification of concurrent systems and protocols.
- Developed at the department of Mathematics and Computer Science of the Technische Universiteit Eindhoven, in collaboration with LaQuSo and CWI.
- The mCRL2 toolset is available for:
  - Microsoft Windows
  - Linux
  - Mac OS X
  - FreeBSD
  - Solaris
- Available at http://mcrl2.org
Preliminaries

Computer-aided design (CAD)

- CAD is the use of computer technology to aid in the design and engineering drafting of a part or product.
- Drafting can be done in two dimensions ("2D") and three dimensions ("3D").
- CAD software packages range from 2D vector-based drafting to 3D solid and surface modellers.
- Application fields:
  - Design and manufacturing of tools and machinery
  - Drafting and design of buildings
  - Engineering process from conceptual design to layout of products (PCB)
  - Strength and dynamic analysis of components (FEM)
  - Film industry
Preliminaries
Justify Choice

Advantages mCRL2:
- Formal unambiguous behavioural model
- Verification for functional properties

Disadvantage mCRL2:
- High level of abstraction

Advantages CAD:
- Precise 3D detailed drawing
- Scalable (Vector graphics, LOD)
- Easier to see the characteristics
- Animatable: Behavioural sense prior to build

Disadvantage CAD:
- Animation does not capture incorrect behaviour
Contribution

Joined Advantages

Take out the disadvantages of mCRL2 models and CAD models
Contribution

Joined Advantages

Take out the disadvantages of mCRL2 models and CAD models

- Easier to explain certain behaviour
- Test time reduction
Contribution

Joined Advantages

Take out the disadvantages of mCRL2 models and CAD models

- Easier to explain certain behaviour
- Test time reduction
- Early virtual component integration
- Validation of the behavioural model
- Reduction on modelling mistakes (mCRL2/CAD)
Outline

1 Introduction
2 Preliminaries
3 Approach
4 Show Case
5 Conclusions
Approach

Architecture

mCRL2 Model

Meta-Language

CAD Model

Behavioral Model

Merge

Visualization

3D Studio Max

department of mathematics and computer science
Approach

Meta-language explained

- Maps relevant observable actions to a behavioural model via tool
- A mapping consists of:
  - an mCRL2 action
  - a visualization primitive
  - a time range value
- An action must have an ID that relates to an animatible object.
- Visualization primitives are:
  - Movement
  - Rotation
  - Scaling
  - Color with Transparency
- No real-time simulation
- Input: XML structure
- Output: MaxScript
Outline

1. Introduction
2. Preliminaries
3. Approach
4. Show Case
5. Conclusions
Show Case
Substrate handler for a printer

- $f = 1$ substrate / 30 sec.
- Every substrate must be accepted at “State 0”
- Output states are “State 0” and “State 4”
- Every substrate needs to reside inside the purple rectangle for at least 60 seconds
- Every “Move” action takes 3 seconds to complete
- Every “Turn” action takes 3 seconds to complete
- “Move”s and “Turn”s are mutual exclusive executed
Show Case
Substrate handler for a printer

State 0: Not Turned → Move
        Turned → Move

State 1: Turn

State 2: Not Turned → Move
        Turned → Move

State 3: Move

State 4: Move

f = 1 substrate / 30 sec.
Every substrate must be accepted at “State 0”
Output states are “State 0” and “State 4”
Every substrate needs to reside inside the purple rectangle for at least 60 seconds
Every “Move” action takes 3 seconds to complete
Every “Turn” action takes 3 seconds to complete
“Move”s and “Turn”s are mutual exclusive executed

Question: Deadlock?
Show Case

The statespace

- Number of states: 463859
- Number of transitions: 634794
Show Case

The statespace

- Number of states: 463859
- Number of transitions: 634794
- Deadlock states: 1
Show Case
The statespace

- Number of states: 463859
- Number of transitions: 634794
- Deadlock states: 1
Show Case
A trace to deadlock

A counter example:
- Create(1) · s0s1(1) · tick$^{18}$
- s1s2(1) | rs1s2(1) · tick$^{3}$ · s2s1(1) | rs1s2(1)
- Create(2) · s0s3(2)
- s3s2(2) · s1s2(1) | s2s1(2) | rs1s2(1) | rs1s2(2)
- s2s3(1) · tick$^{18}$ · Create(3)
Show Case
A trace to deadlock

A counter example:
\[
\text{Create}(1) \cdot s0s1(1) \cdot \text{tick}^{18} \\
\cdot s1s2(1) \mid rs1s2(1) \cdot \text{tick}^{3} \cdot s2s1(1) \mid rs1s2(1) \\
\cdot \text{Create}(2) \cdot s0s3(2) \\
\cdot s3s2(2) \cdot s1s2(1) \mid s2s1(2) \mid rs1s2(1) \mid rs1s2(2) \\
\cdot s2s3(1) \cdot \text{tick}^{18} \cdot \text{Create}(3)
\]

Imagine:
- You had to explain such a counter example to a stranger
- Explain the system and problem within a minute
- A more difficult counter example
- etc…
Show Case
Demonstration

Demo
Outline

1. Introduction
2. Preliminaries
3. Approach
4. Show Case
5. Conclusions
Conclusions

- Demonstrated a practical method for combining a modelling language with an industrial standard into an industrial application
- Validate the correctness of an mCRL2 model
- Verify the correctness of the system
- A visual approach enables information sharing between developers from different disciplines
- Easy to deploy
- Applicable for demonstrators
- Meta-model mapping can also be used for other (generated) data
- Other visualizers can be used as well (Ogre, OpenGL, Maya, Blender, etc . . . )
Acknowledgements

Special thanks to:

- E. (Eugen) Schindler
- K. (Klemens) Schindler

Contact Information:

F.P.M (Frank) Stappers
Eindhoven University of Technology
Department of Mathematics and Computer Science
OAS Group, HG 6.82
P.O.Box 513, 5600 MB Eindhoven
The Netherlands

Phone: +31-40-2472714
E-mail: f.p.m.stappers@tue.nl
URL: http://www.win.tue.nl/~fstapper