Towards an Integrated System Model for Testing and Verification

Benjamin Hummel and Peter Braun
Technische Universität München

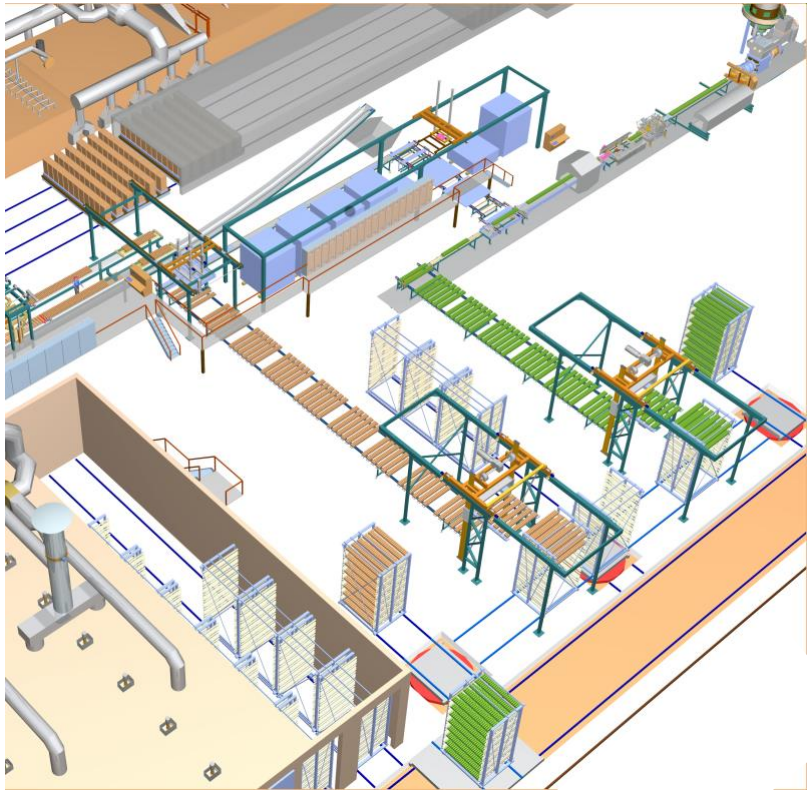
MiSE 2008

Domain

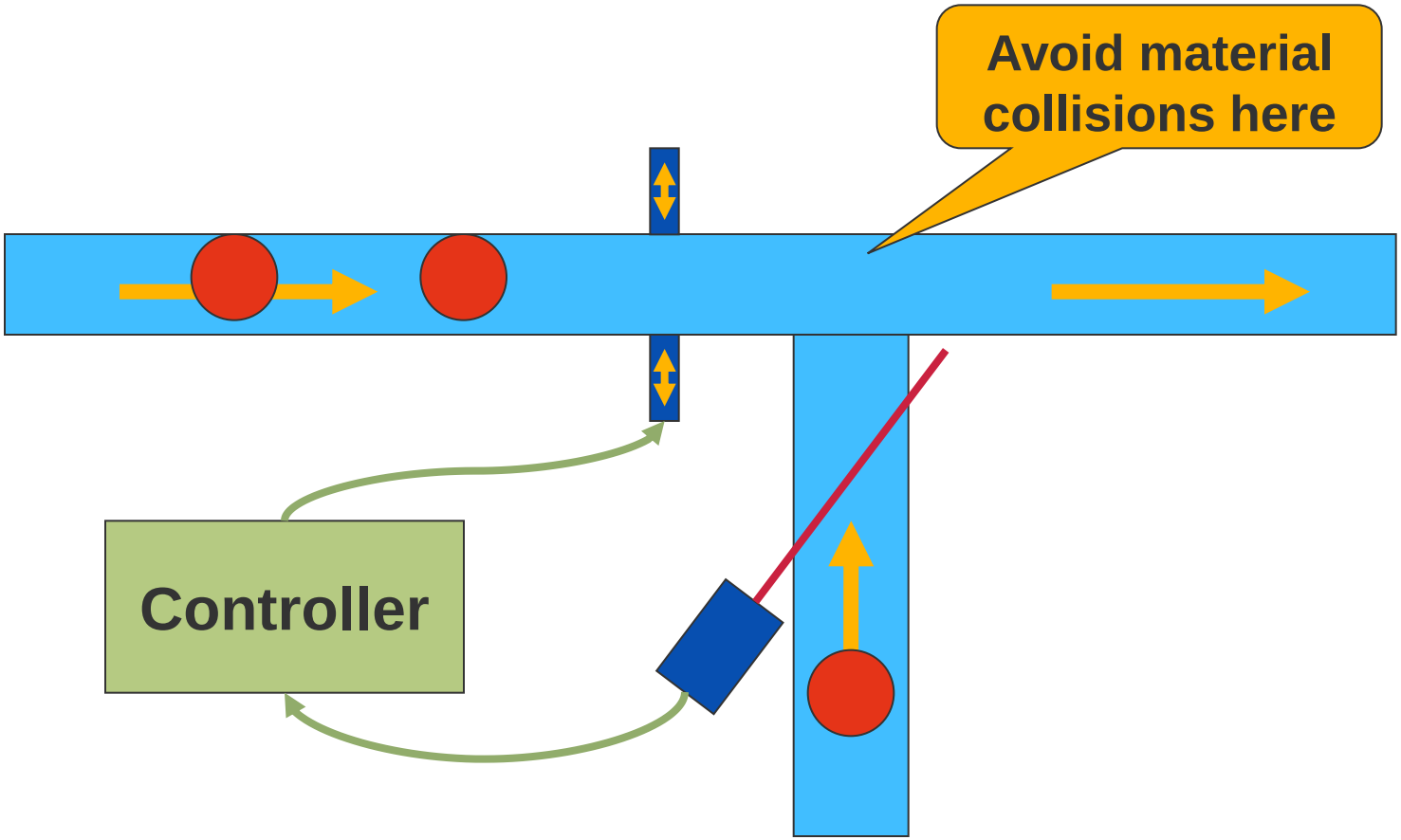
- Development of controller software for production machines

- Special case of mechatronic system

System, whose functions are realized by a combination of mechanics, electronics, and software



Example: Joining of Material Flows



Problem Statement

- SE provides methods for describing (modeling) and developing (implementing) the controller

Problems:

- What are the actual requirements for the controller?
  - Requirements usually describe behavior of mechanics
- How do we test (or even verify) the controller?
  - State-of-the-art: Perform testing on assembled machine
  - Sometimes construct physical simulation model for testing

Outline

- **Proposed Solution: Integrated System Models**
- Towards a Modeling Technique
- Conclusion

Integrated System Models

- The controller has little meaning without its environment
- Thus for understanding the controller we have to understand its environment

$\Rightarrow$ One possible solution is an integrated model describing both the controller and its environment (the machine)

$\Rightarrow$ Allows reasoning about the entire machine

Domain Characteristics

- Focus on material and material flow
- Collisions are very common
  - Negative effects ("Material may not collide")
  - Manipulation of material flow (e.g. by sliding barriers)
  - Activation of sensors (e.g. collision of material with light ray)
- Geometry and position are highly relevant

Requirements for the Modeling Technique

- Explicit material support
- Model includes the machine

- Implicit collision detection, explicit collision handling
- Geometric data and motion information

- Precise enough to be executable

$\Rightarrow$ Allows testing and verification

Outline

- Proposed Solution: Integrated System Models
- **Towards a Modeling Technique**
- Conclusion

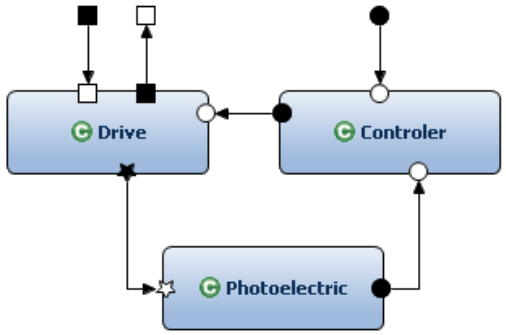
Modeling Overview

- Machine is modeled dataflow driven
  - Parts of the machine are modeled as components
  - Components may be connected using ports
- Different directed ports
  - Signal ports for exchanging logical data
  - Material ports for exchanging physical objects
  - Collision ports for defining valid collisions and delivering collision events

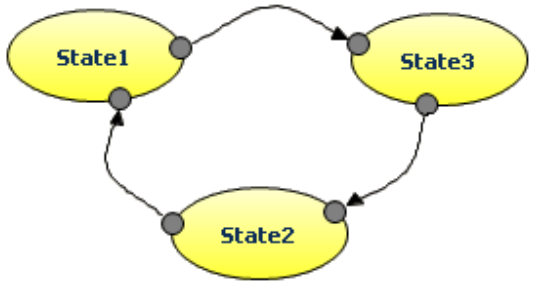
Component Specification

- Components may be described as networks of other components

⇒ Hierarchic decomposition of the system

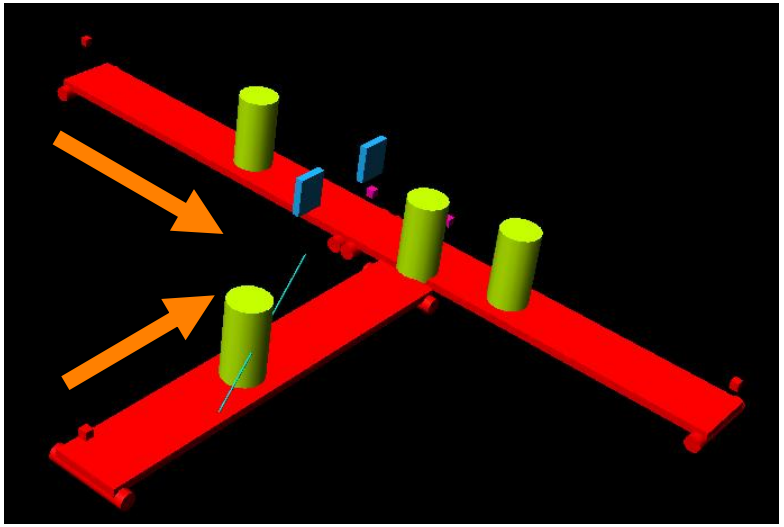


- Atomic components are specified using variant of hybrid I/O automata
  - Communication and calculations happen on transitions
  - Continuous changes happen in states (differential equations)
  - Extensions for dealing with material and collisions



Geometry and Collisions

- Simplified geometry and motion information may be linked to components
- Collisions are detected based on this data (implicitly)
- Collisions are delivered as events to the affected components



Material

- Material is modeled as geometry (shape)
- Material objects are instances of material
- Each material object is managed by one component
- Component influences material during execution
  - Changing position and orientation
  - Handing material over to other component
- Collisions with material are reported to component
  - Used for modeling sensors

close?true/position=0.3

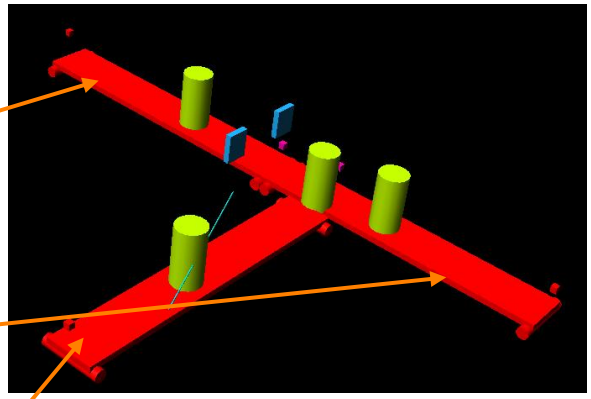
open

closed

close?false/position=0

close

coll



JoiningExample

m_out

sense

coll

coll?x/
sense!true

main

coll?-/
sense!false

push(m_out,
loc_out)

main
m_in: accept any
x'=0.2

m_in

Outline

- Proposed Solution: Integrated System Models
- Towards a Modeling Technique
- **Conclusion**

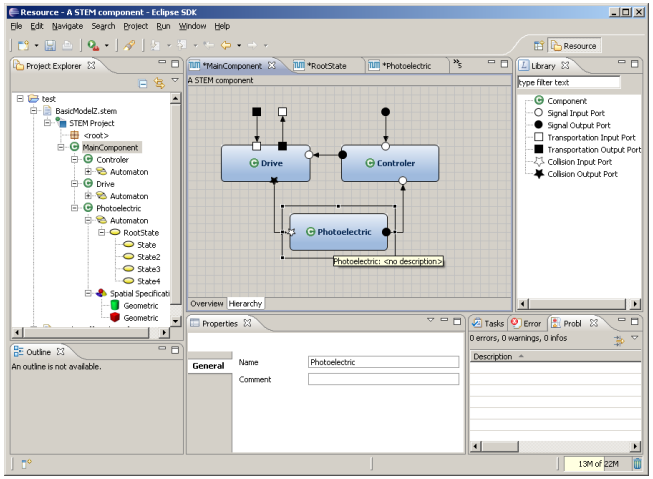
Contribution

- Identification of aspects of production machines, which are relevant for testing and verification of controller software

- First steps towards integrated machine models which are accessible to testing and verification procedures

Future Work

- Improve modeling technique
  - Check applicability to other domains
  - Extend towards general model for mechatronic systems
- Actually perform testing and verification
  - Test-case generation
  - Formal verification
    (e.g. Bounded Model Checking)
- Complete prototypical tool support



Discussion

- Do we really need these integrated models for SE?
- Is this still SE or should we leave it to other engineers?

- What could be improved about the current approach?
- Are there completely different ways of modeling this?