UNIVERSITY OF TWENTE.

Design and Use of CSP Meta-Model for Embedded Control Software Development

Communicating Process Architectures 2012

Maarten Bezemer, Robert Wilterdink and Jan Broenink

Robotics and Mechatronics, University of Twente, The Netherlands

Outline

- Introduction
- CSP Meta-Model
- Use Case
- Example
- Conclusions and Future Work

Demo afterwards!

Introduction - Context

- Model-Driven Design tools help with development of complex software
- CSP helps with development of concurrent software
- The combination helps with complex concurrent software
  - Eg embedded control software for large(r) robots

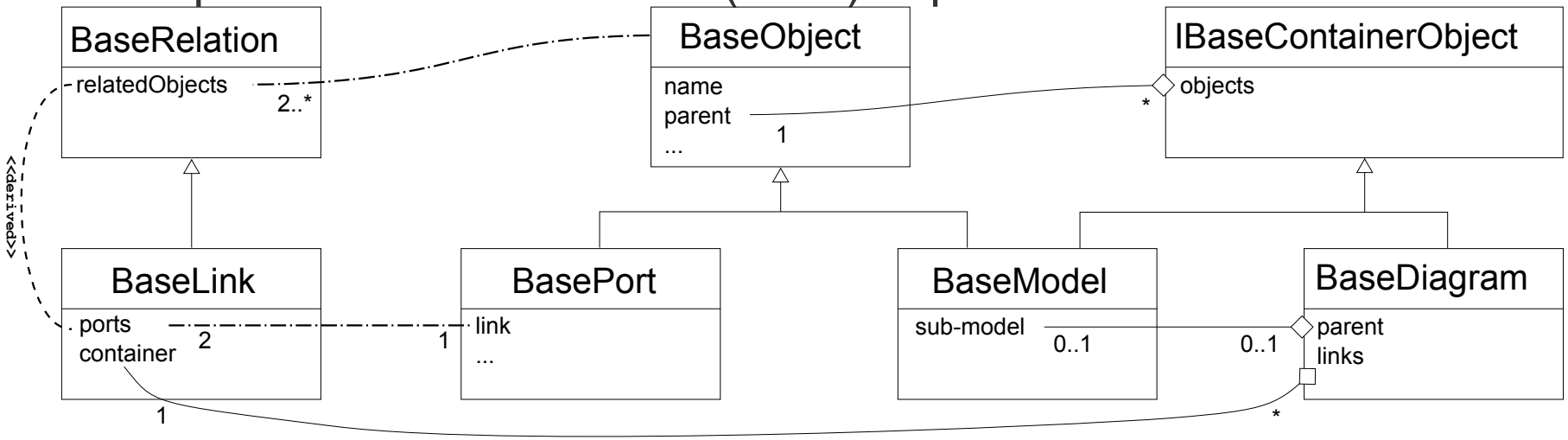
Introduction - Meta-Models

- Constituting elements of models is required
  - Multiple tools (of a tool suite) need knowledge of the models
  - Interchangeable models within a community
- A meta-model describes the format of a model
  - A meta-meta-model describes the format of a meta-model
    - etc.
- Models that conform to a meta-model use format that is known on before-hand
- No existing CSP meta-model available

  with a focus on graphical modelling

Base Meta-Model

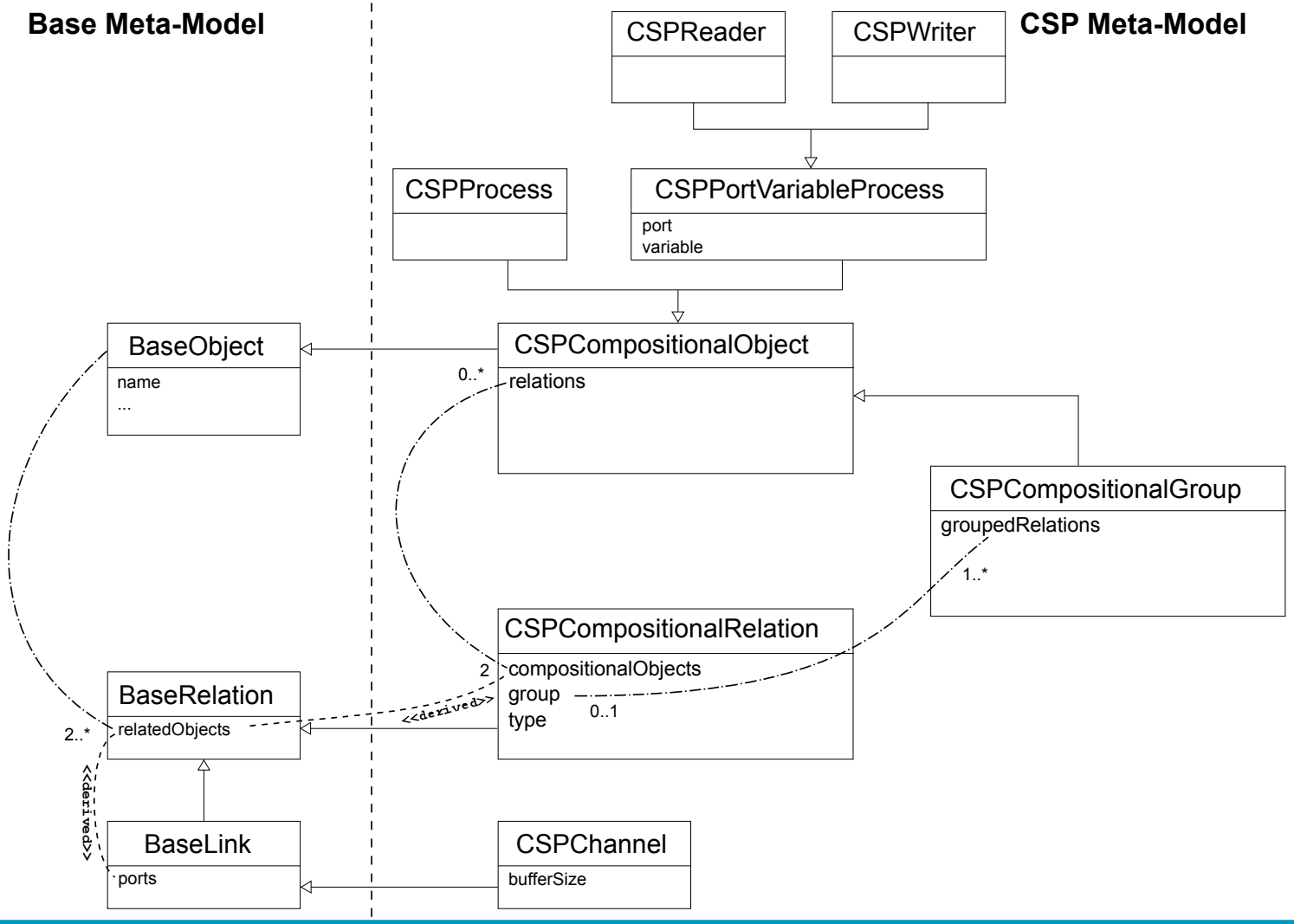
- Modular design
- Base Meta-Model
  - Component-Port-Connector (CPC) implementation



- Usable for multiple types of models
  - Graph based models
  - Communication based interfaces to external tools
  - Basically any model having 'communicating' objects

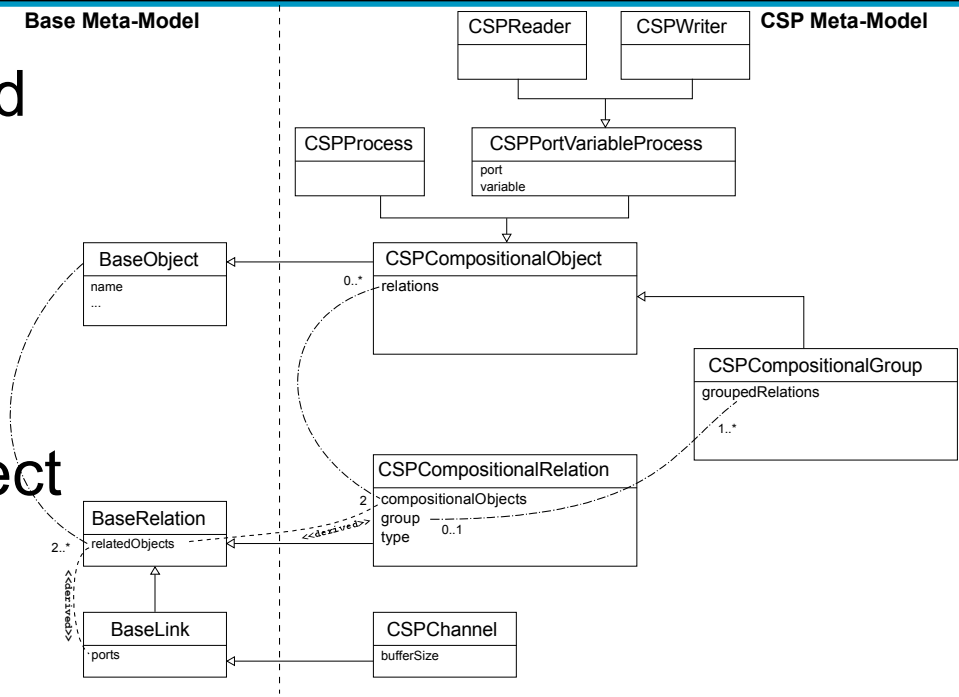
CSP Meta-Model

- Base Meta-Model + CSP constructs



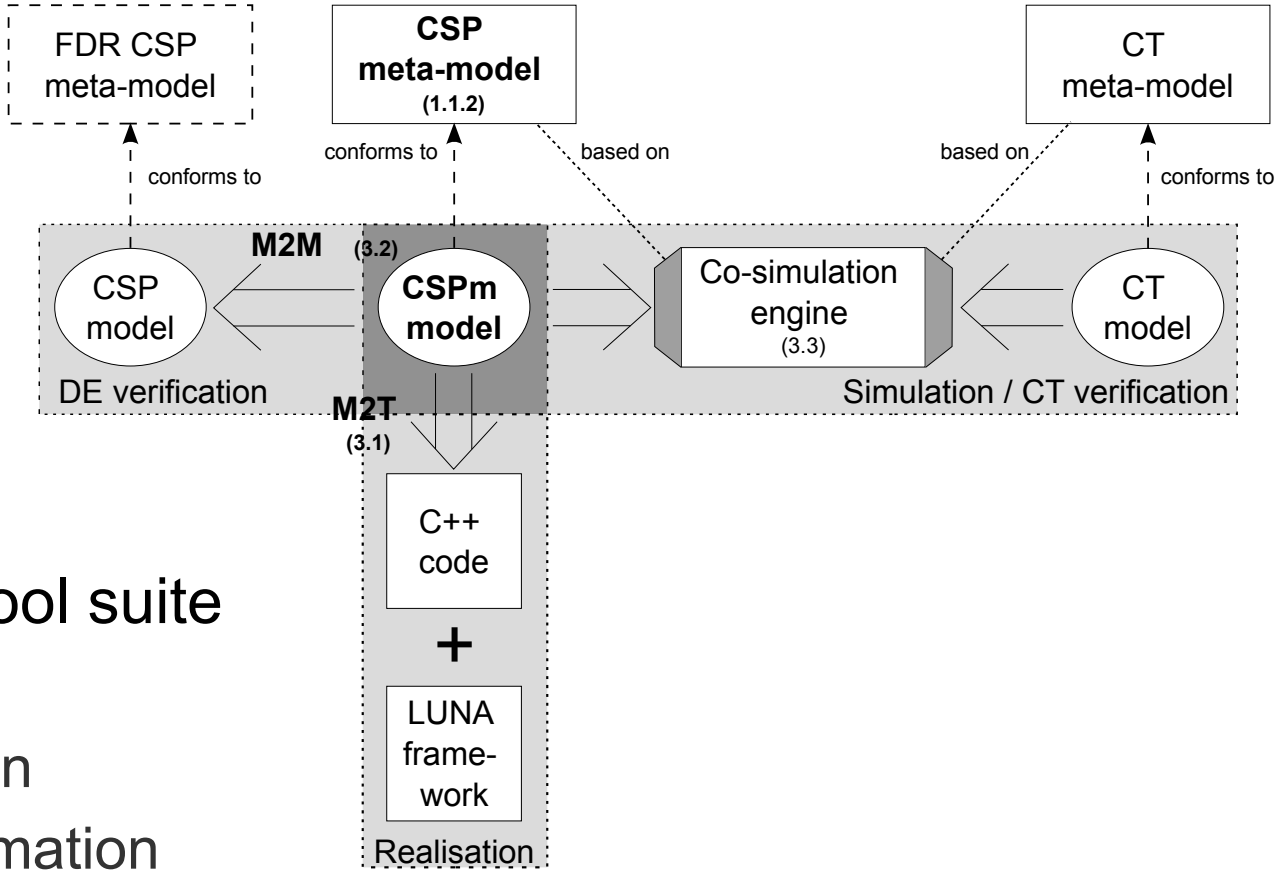
CSP Meta-Model details

- Strict border between Base and CSP Meta-Models
  - Convenient for tools/editors
- All CSP objects/processes are based on same CSP base object
  - Even groups
- BaseRelation defines ANY relation
  - Used to define compositional relations
- BaseLink defines a communicational relation
  - Used to define communication channels



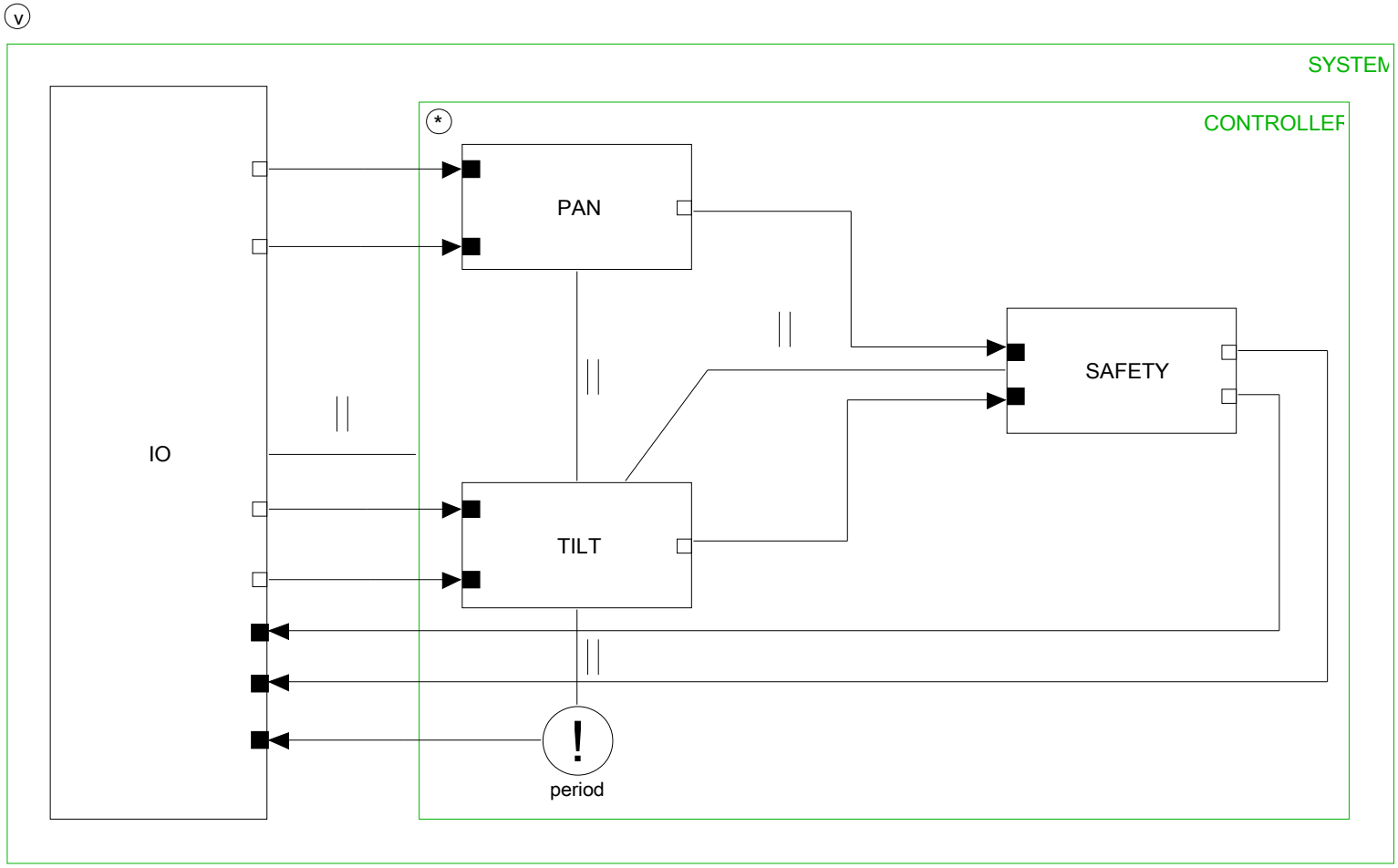
Use Case - TERRA

- Twente Embedded Real-time Robotic Application



- Eclipse based tool suite
  - CSP Editor
  - Model validation
  - Model transformation
    - CSP/FDR → Formal verification of model
    - C++/LUNA → Towards on target execution of model

Use Case – CSP Editor



- Graphical representation similar to gCSP
  - More compact models, due to some changes

Use Case – Model Validation

- Validation in editor
  - Rules are defined by Meta-Model
    'What to fill in where'
  - Addition rules for transformations

- Keeps models compliant for model transformation
  - Compatible with specific target transformation rules
    - Object names must be correct and unique (CSP and C++ LUNA)
    - Sub-models must be defined (CSP)

```
context CSPProcess {
  critique hasSubmodel {
    guard : CodeGenDescription.isUndefined() or CodeGenDescription == "CSP"
    check : self.submodel.isDefined()
    message : "CSP code generation will fail: " + self.name + " does not
have a sub-model set"
  }
}
```

Use Case – Model transformation

- Possible to use external expert tools
  - FDR for formal verification of the models
- No need to reinvent these external tools!
- Put the models to actual use
  - C++/LUNA code generation to execute model on target
- Prevent errors due to manual conversion of correct models

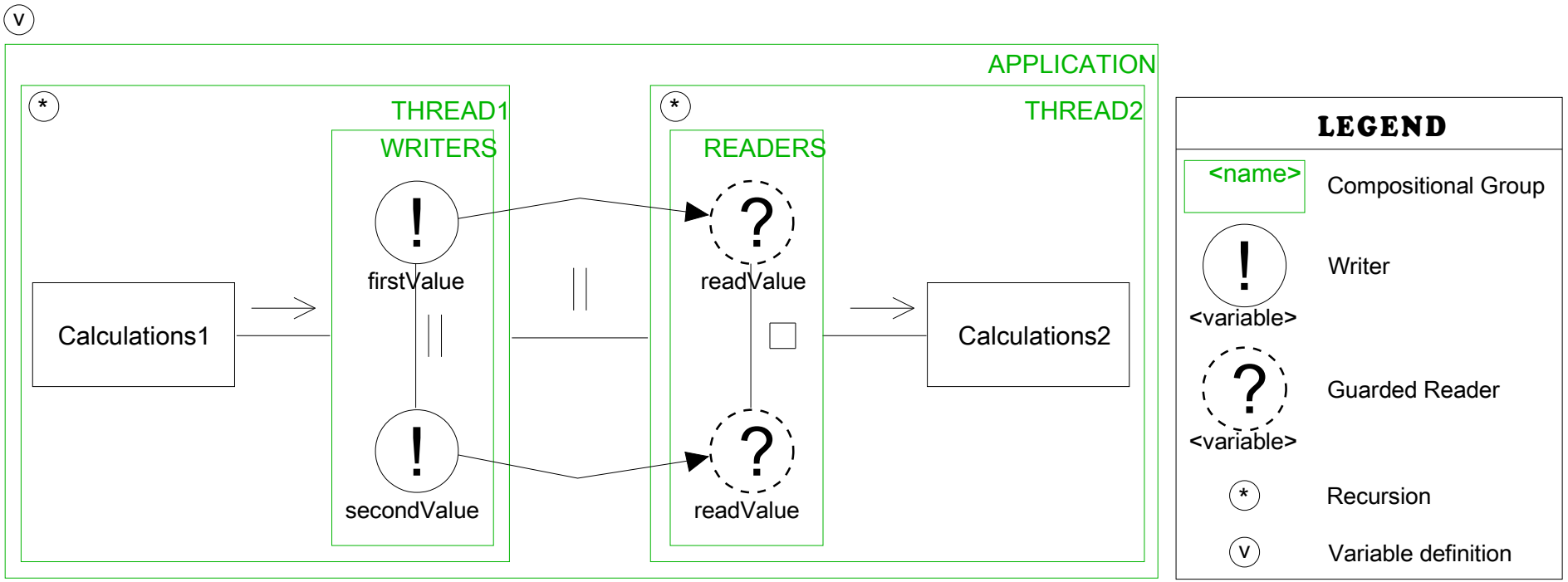
(The demo afterwards will show actual examples)

Use Case – Model transformation

- Representation of model in CSP (FDR)

CSPm	Meta-model representation	Additional Attributes
p = ...	CSPProcess	
	CSPCompositionalGroup	*groupedRelations*: relations that are grouped
channel c	CSPChannel	*ports*: two connected ports / processes
datatype <type> = <name>	CSPVariableDescription	*name*: name of the variable *type*: boolean, integer...
c ! <var>	CSPWriter	*variable* contains data *link*: channel to write to / read from
c ? <var>	CSPReader	
p ; q	CSPCompositionRelation	*type* = SEQ or PAR or ALT *compositionalObject*: the two related objects
p || q	CSPCompositionRelation	
p [] q	CSPCompositionRelation	
if-statement	CSPRecursionProperty	*expression*: true when another loop is required

Example



- Corresponding CSP (FDR) code available in paper
- Real-life examples
  - JIWY and Production Cell available
    (JIWY will be showed in the demo afterwards)

Conclusions

- CSP Meta-Model usable to store CSP related models
- TERRA tool suite makes use of CSP Meta-Models
  - Editors
  - Validation
  - Transformations to plain text
- Actual suitable for Embedded Control Software Development
  - JIWY and Production Cell models are designed
  - Used by students for their assignments

Future Work

- Better integration between external modelling tools and TERRA
  - 20-sim suitable for physics modelling
  - Directly integrate 20-sim models as CSP sub-models
- Building blocks
  - Provide standard Control Software functionality
  - Generic Architecture Component as a template/skeleton block
- Simulation and co-simulation capabilities
  - Better understand of models
  - Able to test a control model using a dynamics model of the plant

"Hopefully a standard meta-model will emerge that is suitable for the needs of the community and helping to improve interaction between multiple disciplinaries within the community."

- Standard storage format
- Interchangeable with the tools developed within our community

- Tighter cooperation!