Extension of the Ocarina Tool Suite to support Reliable Replication-Based Fault-Tolerance

Wafa GABSI — wafa.gabsi@redcad.org

Bechir ZALILA — bechir.zalila@enis.rnu.tn

Mohamed Jmaiel — mohamed.jmaiel@enis.rnu.tn

21st International Conference on Reliable Software Technologies

Ada-Europe'2016

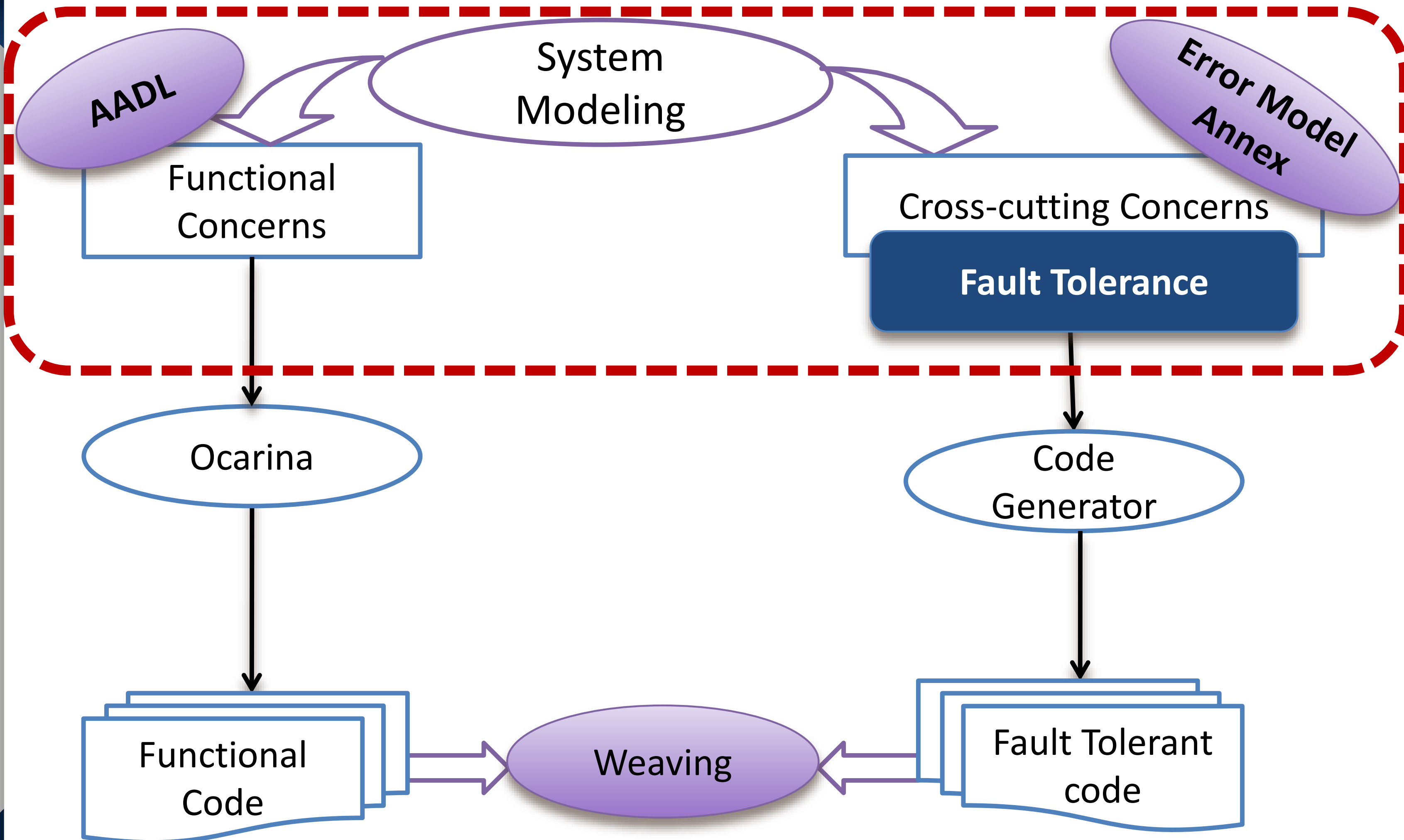
June 13-17 Pisa, Italy

Outline

Context (1/2)

- With the evolution of distributed real-time embedded systems, new requirements for high dependability and fault tolerance are emerging
- These requirements have to be satisfied at design time
- Such systems must be highly dependable in order to increase their performance, effectiveness and reliability
- Some work provide design supports of fault tolerance techniques such as model weaving or passive replication applied to AADL models

Context (2/2)


AADL
System
Modeling
Error Model
Annex
Functional
Concerns
Cross-cutting Concerns
Fault Tolerance
Ocarina
Code
Generator
Functional
Code
Weaving
Fault Tolerant
code

Error Model Annex

Fault Tolerance

Strong support of several concepts related to fault tolerance

Formal Modeling

Formal specification of different types of errors, error propagation and error behaviors

Error Model Annex

Tool Support

State machines error behavior: Error events, error states and propagation conditions

Analysis Tool

Safety analysis tools implemented to support safety and reliability evaluation process

Outline

Replication Issues

It does not support automatic redundancy in distributed real-time embedded systems.

It relies on manually specified redundancy of components, connections and behaviors.

The more the replicas or replicated components we have, the more complex and error prone the model is.

Most of existing works consider only one replication style but not both.

Outline

1. Context & motivation
2. Issues
3. Objectives
4. Existing Work
5. Approach
6. Conclusion & Future work

Objectives

1 Assist the designer in modeling the fault tolerant system by integrating replication techniques from the design phase

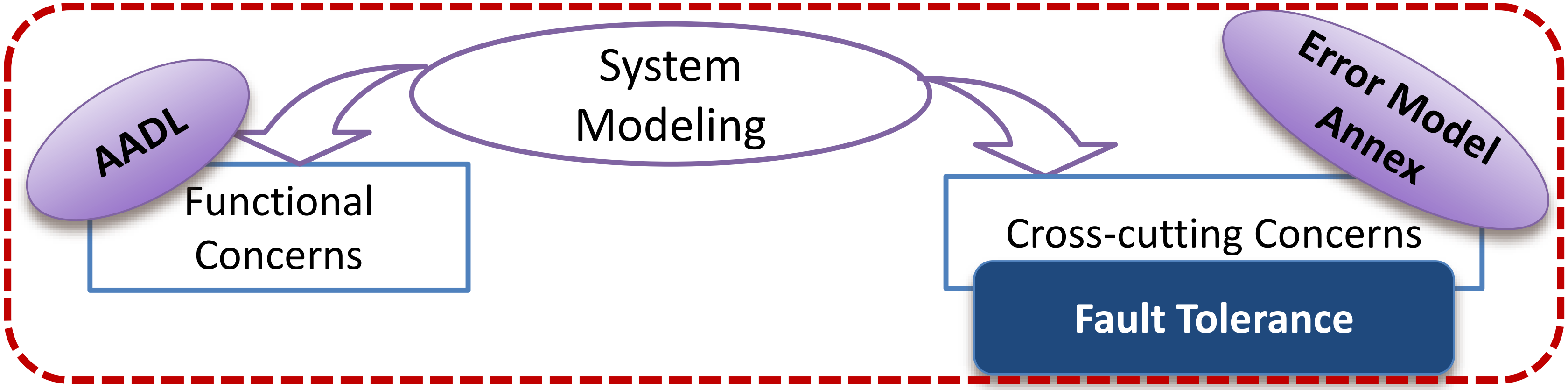
2 Support automatic replication of AADL components

3 Support automatic code generation of both active and passive replication

Outline

1. Context & motivation
2. Issues
3. Objectives
4. Approach
5. Case study
6. Conclusion & Future work

Approach (1/2)

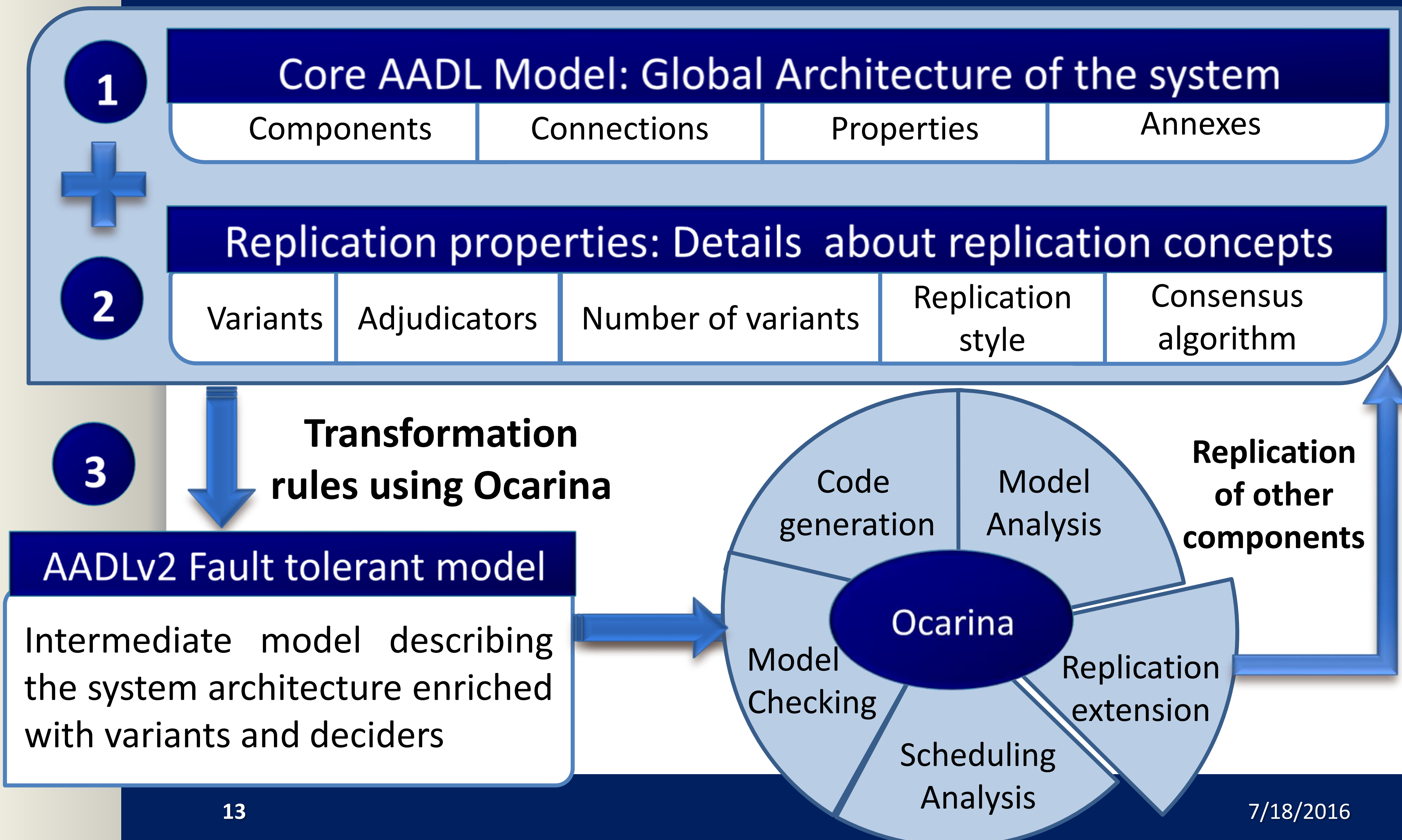


A model driven fault tolerance approach, relying on automatic replication of AADL components since the design level and automatic code generation for both active and passive replication

Approach (2/2)

- Assist the designer in modeling his system by automation of the replication design using properties
- Encapsulate the needed replication parameters into a property set and integrate them in the base model through a model transformation
- Define:
  - A set of ***properties*** to describe replication concepts
  - A set of ***transformation rules*** to generate a replicated model
- Implement these rules into the ***Ocarina*** tool suite

Replication process



Replication property set

PropertySet Replication_Properties is

Description : **aadlString**;

Replica_Number : **aadlInteger**;

Replica_Identifiers : list of **aadlString applies** to (system, process, thread, processor, device);

Replica_Type : Replication::ReplicationType;

ReplicationType: type enumeration(**Active, Passive**);

consensusAlgorithm_Source_Text: **aadlString**;

consensusAlgorithm _Class: **classifier** (subprogram classifier) **applies to** (system, process, thread, device, subprogram, event data port);

consensusAlgorithm_Ref: **reference** (subprogram classifier) **applies to** (system, process, thread, device, subprogram, event data port);

end Replication_Properties;

Replicate what ?

- Supported AADL Components:
  - Components hierarchy
  - Properties and possible features of each component
  - Modes and mode transitions

AADL components		
Software	**hardware**	**Hybrid**
Process ✓	Device ✓	
Thread ✓	Memory X	System ✓
Data X	Processor ✓	
Subprogram X	Bus X	

Ocarina Extension (1/4)

1 VALIDATION OF THE PROPERTIES USE

- We check the validity of the use of our property set items using Ocarina
  - Replica number bounded between the minimal and the maximal number of replicas
  - Type of the replication: active or passive
  - Consensus algorithm specified to decide about replicas

→ Properties have to be coherent and not redundant

Ocarina Extension (2/4)

1 VALIDATION OF THE PROPERTIES USE

2 EXTRACTION OF THE LIST OF PROPERTIES

- we extract for each replicated component the list of properties
  - Collecting all replication properties as a record
  - Invoking the suitable transformation rules to apply the expansion of the AADL model

Ocarina Extension (3/4)

1. VALIDATION OF THE PROPERTIES USE
2. EXTRACTION OF THE LIST OF PROPERTIES
3. EXPANSION OF THE AADL MODEL

- Based on transformation rules, we expand the AADL base model with replicas by manipulating its AADL tree
- A replication algorithm is applied to extend the model by the replication mechanisms depending on the type of replicated component and on the selected replication strategy

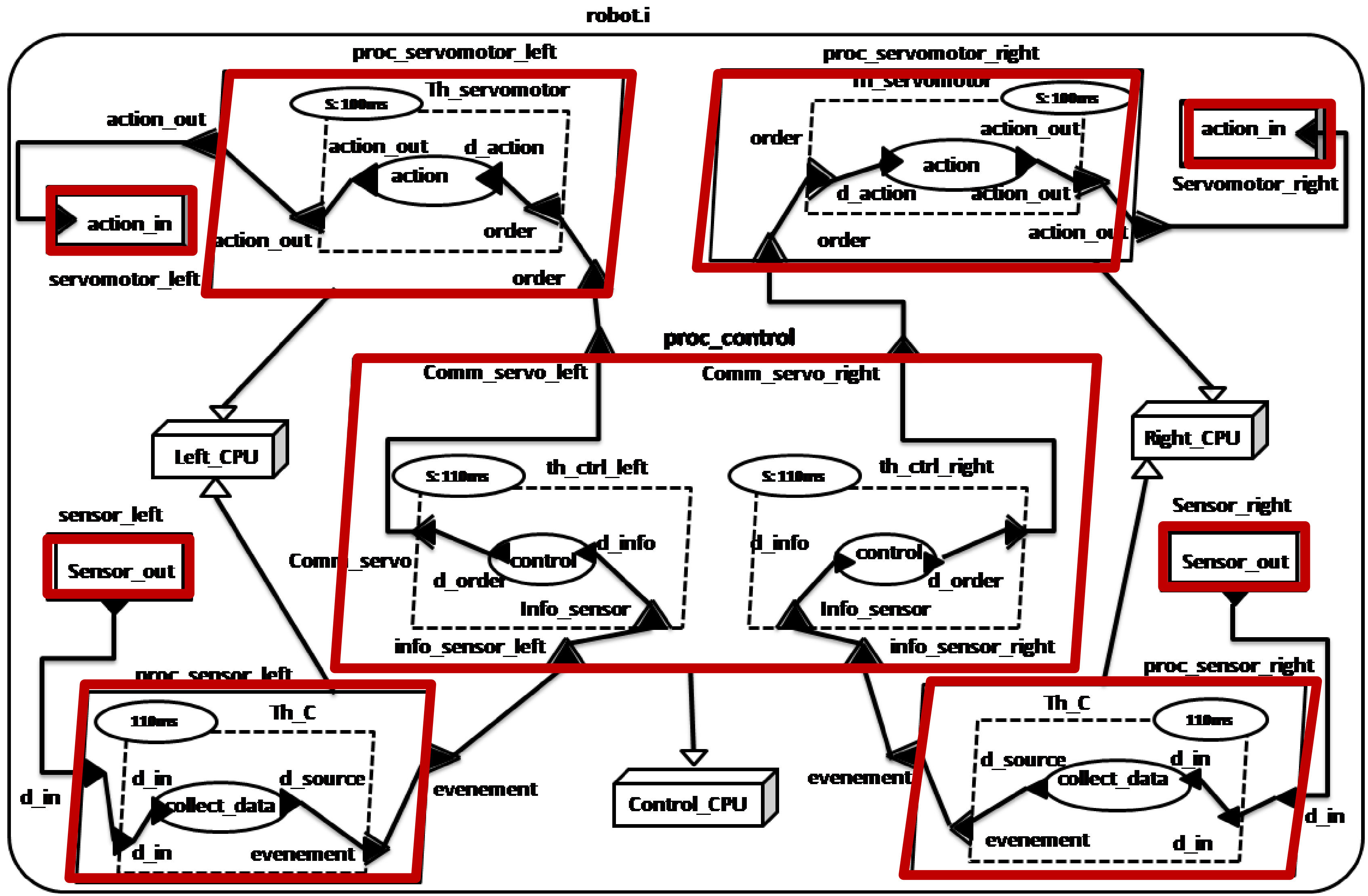
Ocarina Extension (4/4)

1. VALIDATION OF THE PROPERTIES USE
2. EXTRACTION OF THE LIST OF PROPERTIES
3. EXPANSION OF THE AADL MODEL
4. GENERATION OF THE ENRICHED AADL MODEL FROM ITS EXPANDED TREE

Outline

1. Context & motivation
2. Issues
3. Objectives
4. Approach
5. Case study
6. Conclusion & Future work

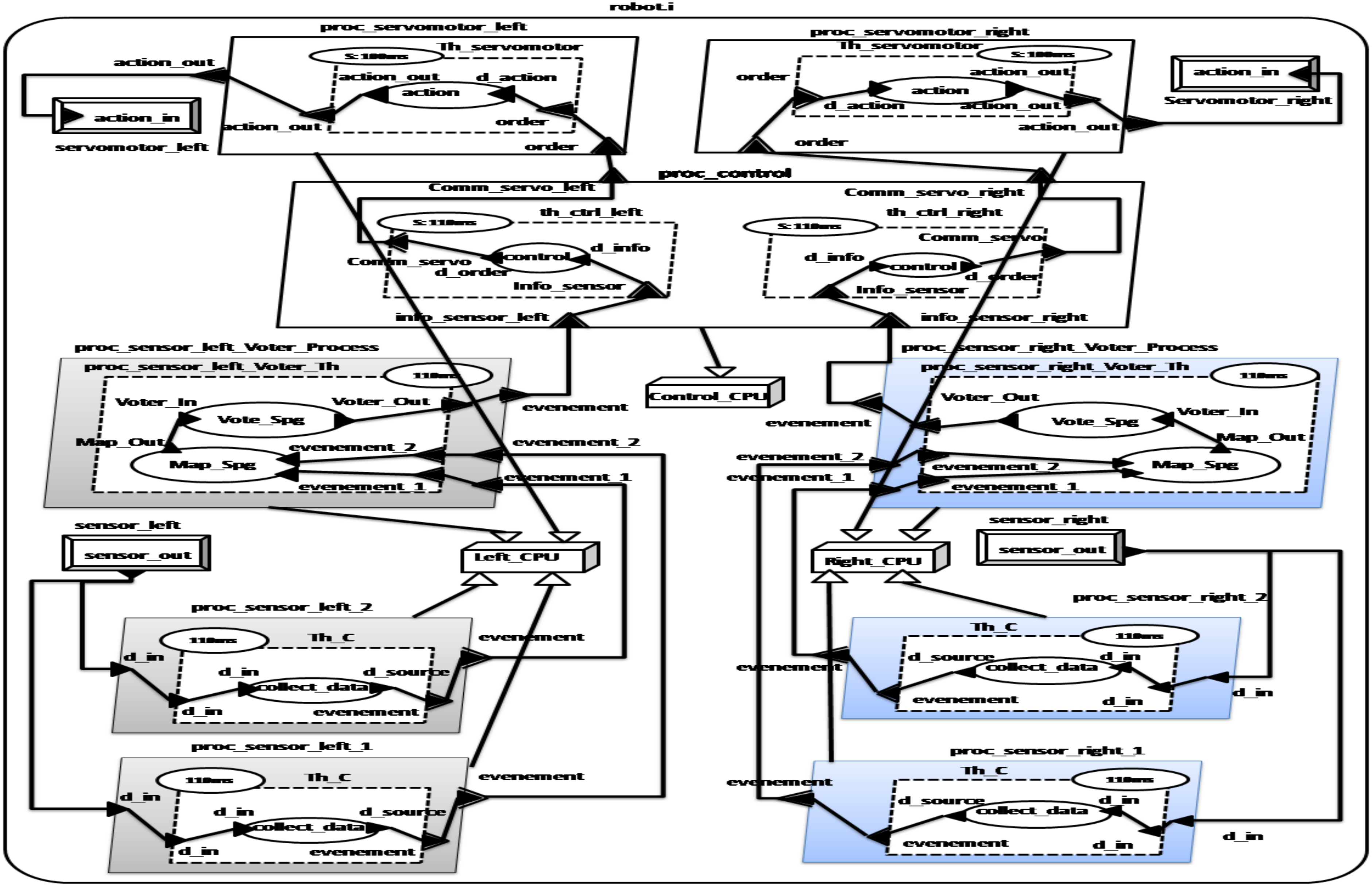
Case study: Robot system



Case study: Replication properties

```
system implementation robot.i
...
properties
  Replication_Properties::Description => "Replication_of_the_process_
      component_proc_sensor_right" applies to proc_sensor_right;
  Replication_Properties::Replica_Number => 2 applies to
      proc_sensor_right;
  Replication_Properties::Replica_Type => ACTIVE applies to
      proc_sensor_right;
  Replication_Properties::Replica_Identifiers => ("proc_sensor_right_1",
      "proc_sensor_right_2") applies to proc_sensor_right;
  Replication_Properties::Consensus_Algorithm_Source_Text => "robot.
      Do_Vote" applies to proc_sensor_right.evenement;
  Replication_Properties::Description => "Replication_of_the_process_
      component_proc_sensor_left" applies to proc_sensor_left;
  Replication_Properties::Replica_Number => 2 applies to
      proc_sensor_left;
  Replication_Properties::Replica_Type => ACTIVE applies to
      proc_sensor_left;
  Replication_Properties::Replica_Identifiers => ("proc_sensor_left_1","
      proc_sensor_left_2") applies to proc_sensor_left;
  Replication_Properties::Consensus_Algorithm_Source_Text => "robot.
      Do_Vote" applies to proc_sensor_left.evenement;
end robot.i;
```

Case study: generated Model



Case study: Results

- The generated model is complicated with respect to the initial one
- We help the designer to generate it while reducing the risk of errors and decreasing the number of lines of code in a meaningful way (up to **50%**)
- To validate the consistency of the generated intermediate model, we used Ocarina again to parse it and to generate Ada code using the PolyORB-HI middleware, GNATforLEON to compile it and TSIM to simulate it

Outline

1. Context & motivation
2. Issues
3. Objectives
4. Approach
5. Case study
6. Conclusion & Future work

Conclusion

- A new approach based on automation of replication of AADL components
- An approach supporting both active and passive replication
- Enabling designers to choose different number of variants even in a single model
- The workload of the designer minimized to only indicating the original component subject to replication, specifying the replication style, setting the number of variants and defining the consensus algorithm for each replicated component

Future work

- Accomplish the extension of this tool by passive replication
- Extend the POLYORB-HI middleware with fault tolerant concepts
- Enrich Ocarina with various consensus algorithms that are well used to ensure software fault tolerance including all agreement, weak validity, strong validity and termination
- Formally verify our approach

Thank you

For more questions:

wafa.gabsi@redcad.org