AMASS

Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems

AMASS Usage Scenario 3: Architecture Refinement

2nd EAB Workshop
Västerås, September 17, 2018

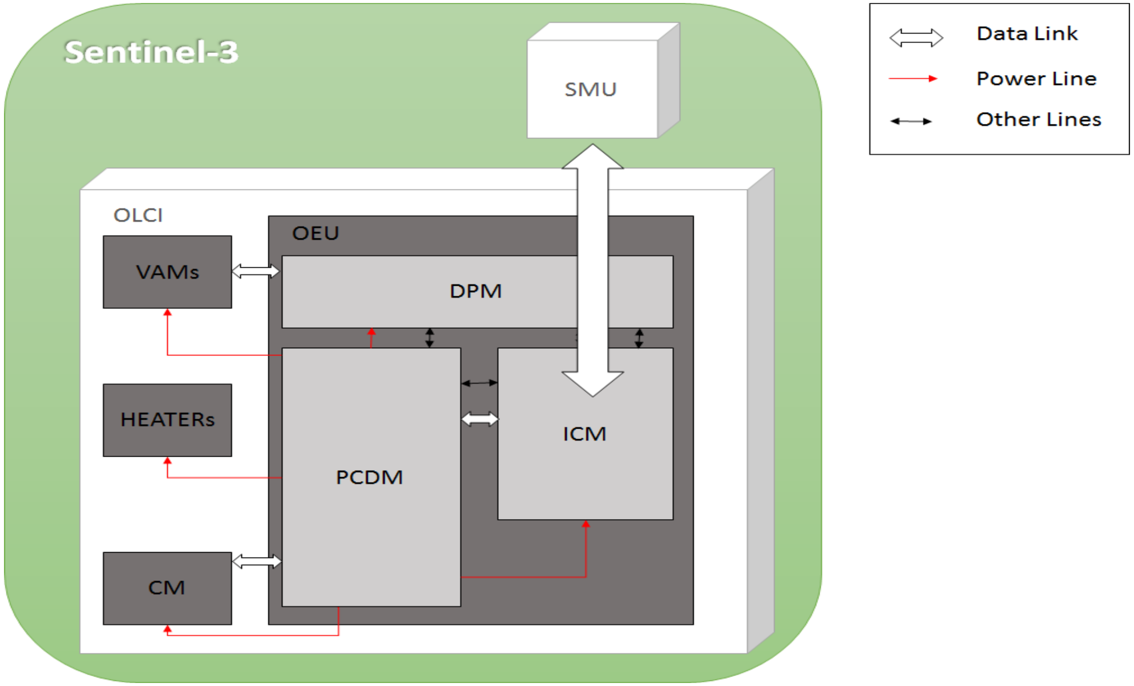
Stefano Puri
WP3 Leader

Introduction – Architecture Driven Assurance Areas

- Requirements specification
  - Support for formalization, quality evaluation
- System Architecture Modelling for Assurance
  - Exploit the system architecture in the assurance case
  - System architecture languages
  - Architecture trade-off and comparison
- Architectural Patterns for Assurance
  - Interaction between assurance and architectural patterns
  - Architectural patterns from standards
- Contract-based assurance
  - Assurance patterns for contract-based design
  - Enrich evidence produced by contract-based design
- V&V-based assurance
  - Enrich V&V techniques

Scenario

- To support system architecture design/refinement, allowing reuse and improvement of system assurance

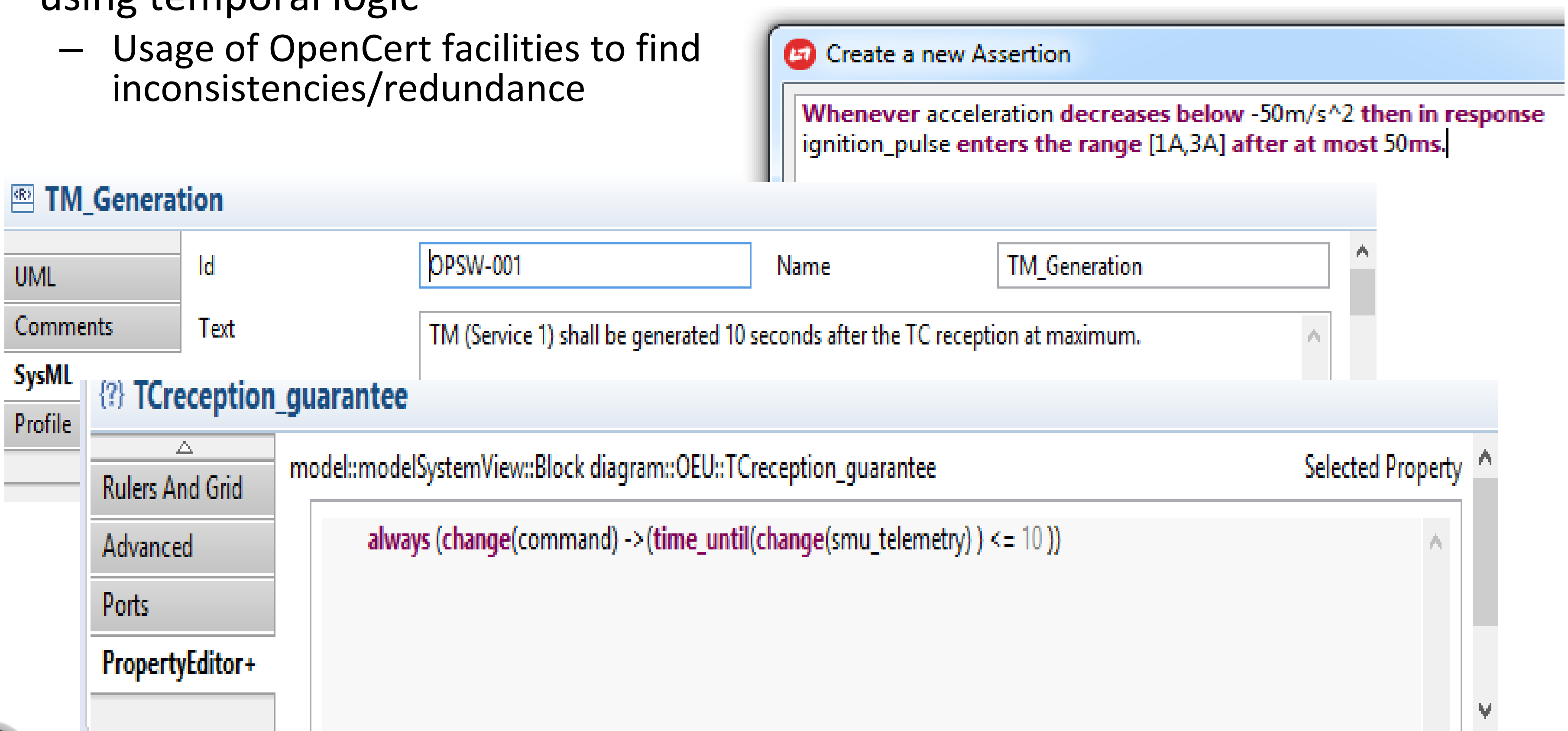


Higher-level objectives & expected gains

- **O1:** define a holistic approach for ***architecture-driven assurance*** to leverage the reuse opportunities in assurance and certification by directly and explicitly addressing current technologies and HW/SW architectures needs.
- Metrics (subset)
  - Effort for assurance and certification
  - Effectiveness in system architecture issues identification
  - Number of requirements formalized

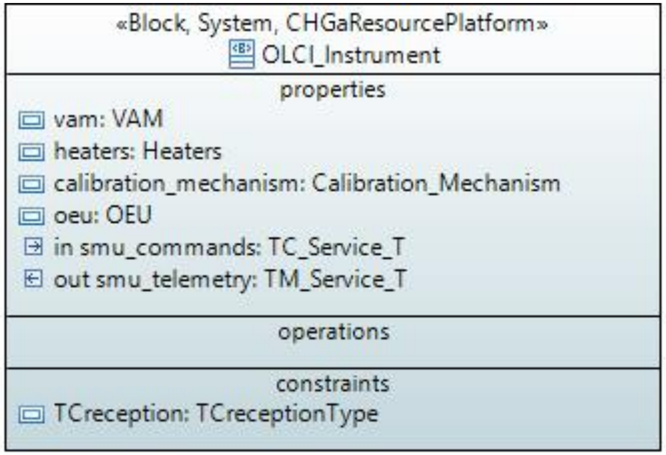
Scenario step: requirements specification

- Requirements can be written in informal language
  - Usage of OpenCert facilities to measure the quality of the requirements
- Templates for semi formal requirements specification are supported
- Formal definition of requirements is supported by using temporal logic
  - Usage of OpenCert facilities to find inconsistencies/redundance

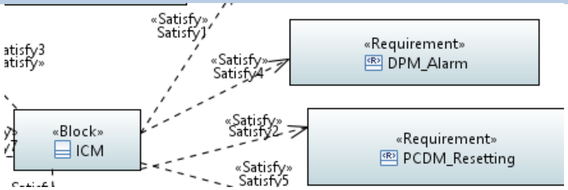


Scenario step: architectural modeling

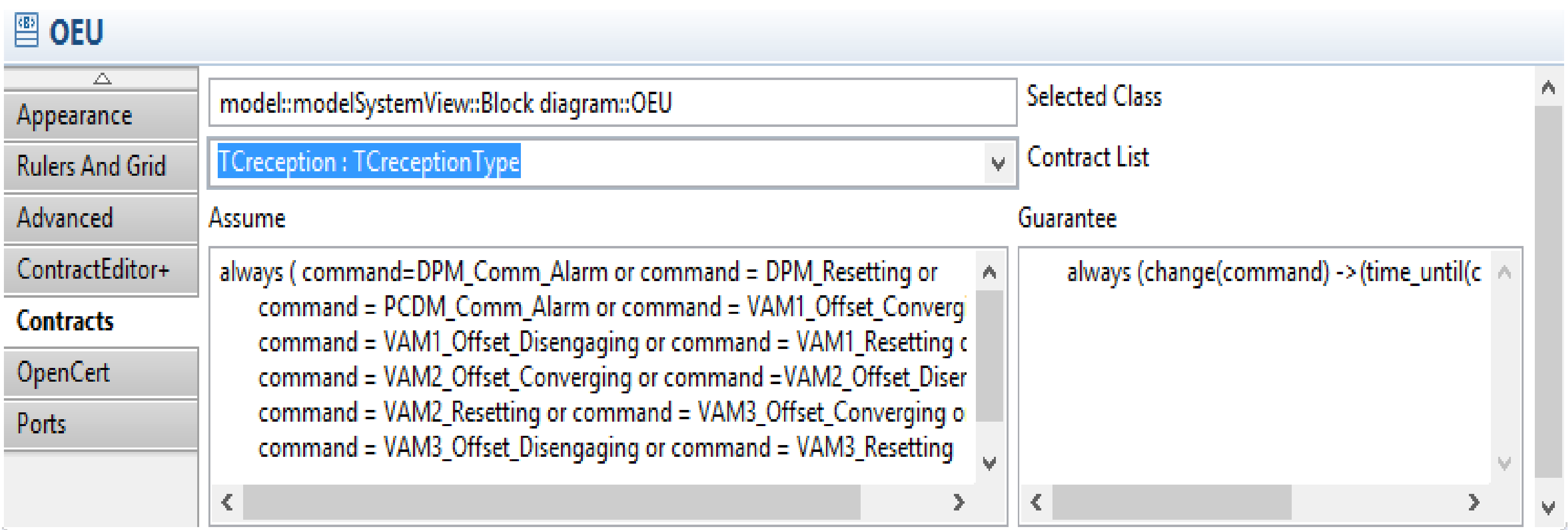
- System architecture can be modeled by using Papyrus SysML tool (part of OpenCert) or by using external tool (e.g. Rhapsody, Medini, SafetyArchitect)
- Several importers are available to connect external modelling tool to Papyrus
- System components are defined out of any context, with their properties and then instantiated in the given context



Scenario step: contracts definition

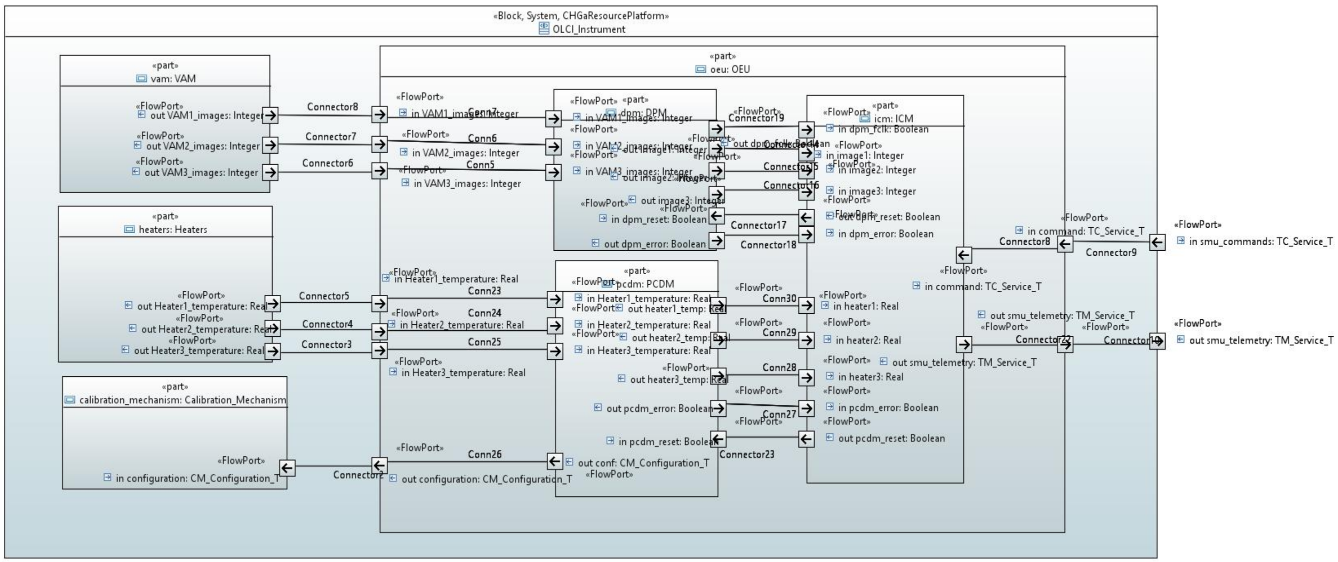


- Requirements are assigned to components
- Contracts are created for a component
  - Pair of assumption and guarantee formal properties
  - A contract covers one or more requirements
  - The assumption and guarantee elaborate upon the component properties
  - Usage of weak and strong contracts
    - E.g., weak contracts are used to specify timing behaviour in different environments, or safety behaviour under different failure conditions



Scenario step: architectural refinement

- System components with high complexity are decomposed by using fine-grained components (parts)
  - Top-down or bottom-up process
  - The implementation of a composite component is completely delegated to its parts
    - The interfaces of the composite component have to be realized/required by the parts
  - Sub-Requirements are associated to the parts
  - Components parts are connected together via their interfaces



Scenario step: contracts refinement

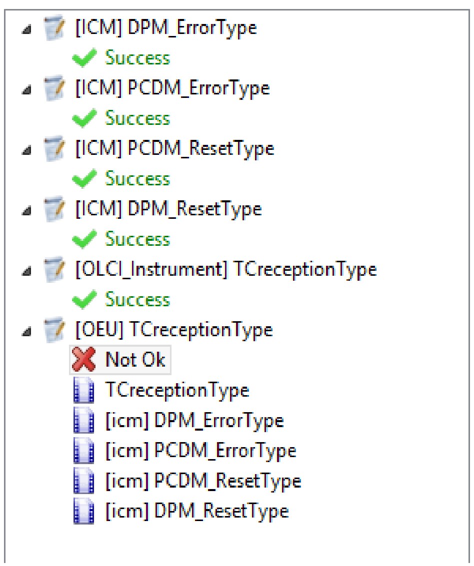
- Contracts covering the sub-requirements are defined for the sub-components
- Contracts decomposition follows the requirements refinement

System Architectures	Number of Subcomponents and Contracts
OLCI_Instrument	5
calibration_mechanism:Calibration_Mechani:	0
heaters:Heaters	0
oeu:OEU	4
dpm:DPM	0
icm:ICM	6
pcdm:PCDM	0
TCreceptionType	
vam:VAM	0
TCreceptionType	

Refined Contracts	Number of sub-contracts
OLCI_Instrument.TCreception	1
oeu.TCreception	4
icm.DPM_Error	1
icm.DPM_Reset	1
icm.PCDM_Error	1
icm.PCDM_Reset	1

Scenario step: apply early analysis

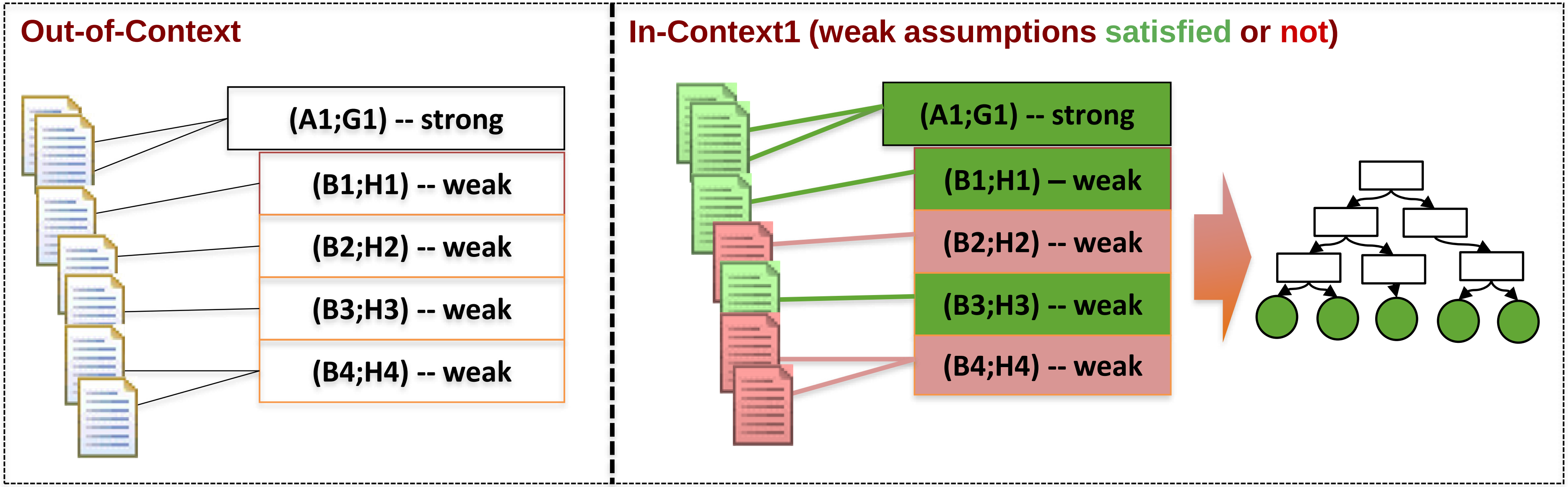
- Usage of the CHESS feature and available integration with external tool OCRA to
  - verify the components assembly is correct wrt the associated contract assumption-guarantee
  - verify that the contracts decomposition is correct
    - E.g., if the refinement is not correct, then contracts/requirement has to be changed and the analysis reexecuted



	Step 1	Step 2	Step 3	Step 4
Ports	State Ports	State Ports	State Ports	State Ports
OEU_inst				
pcdm				
heater3_temp	5.0		5.0	
heater2_temp	3.0		3.0	
heater1_temp	4.0		4.0	
pcdm_error	FALSE		TRUE	
conf	CM_Usa...itted		CM_Usa...itted	
dpm				
image1	2		2	
image2	0		0	
image3	1		1	
dpm_error	FALSE		TRUE	
dpm_fclk	FALSE		FALSE	
Heater3_temperature	7.0		7.0	
icm				

Scenario Step: Weak assumptions validity check

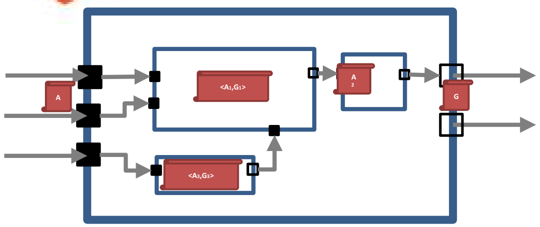
- Automatic selection/filtering of the weak contracts applicable in the given environment



Scenario step – apply safety analysis

...

CHESS



CHESS-FLA

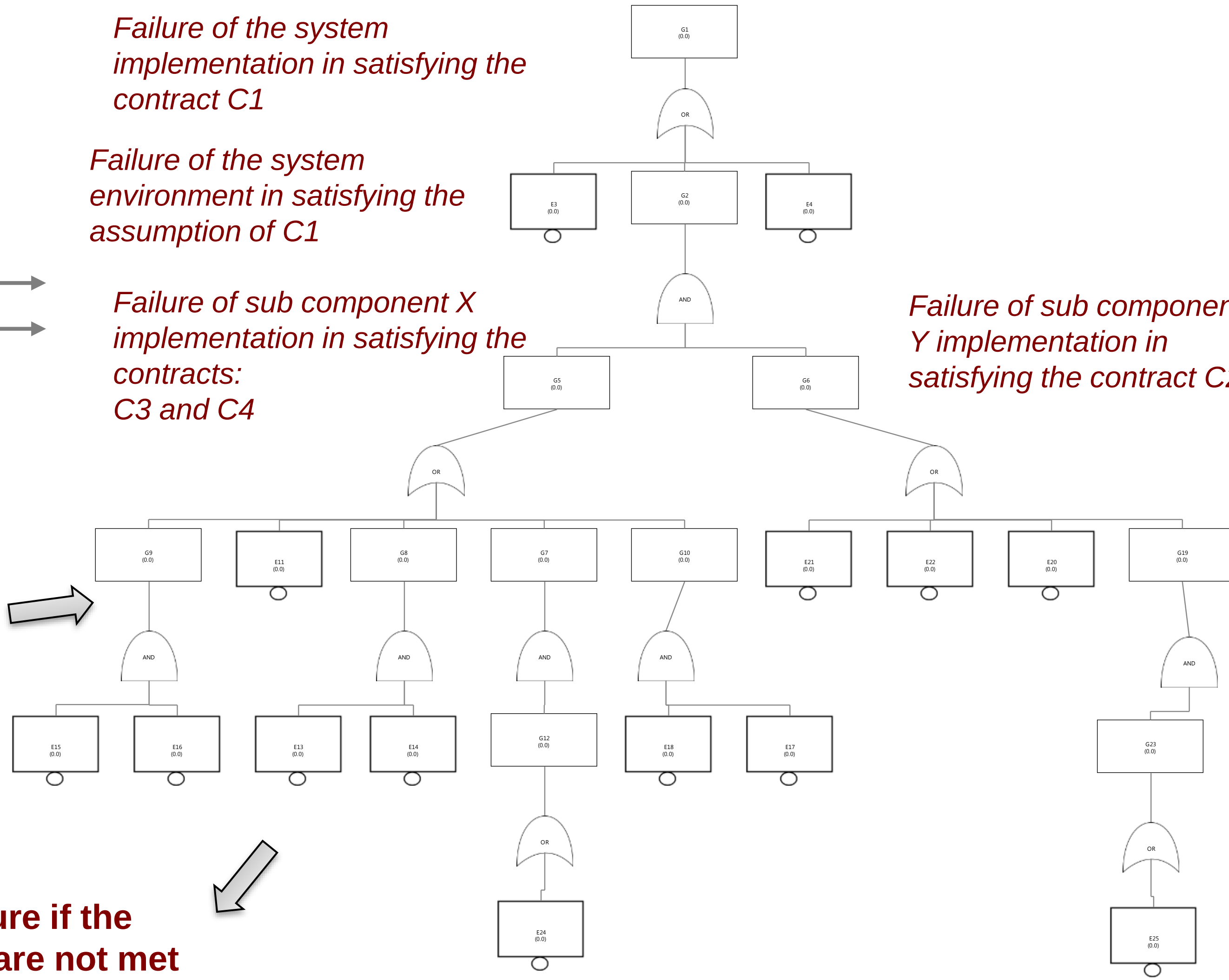
OCRA

Failure of the system implementation in satisfying the contract C1

Failure of the system environment in satisfying the assumption of C1

Failure of sub component X implementation in satisfying the contracts:
C3 and C4

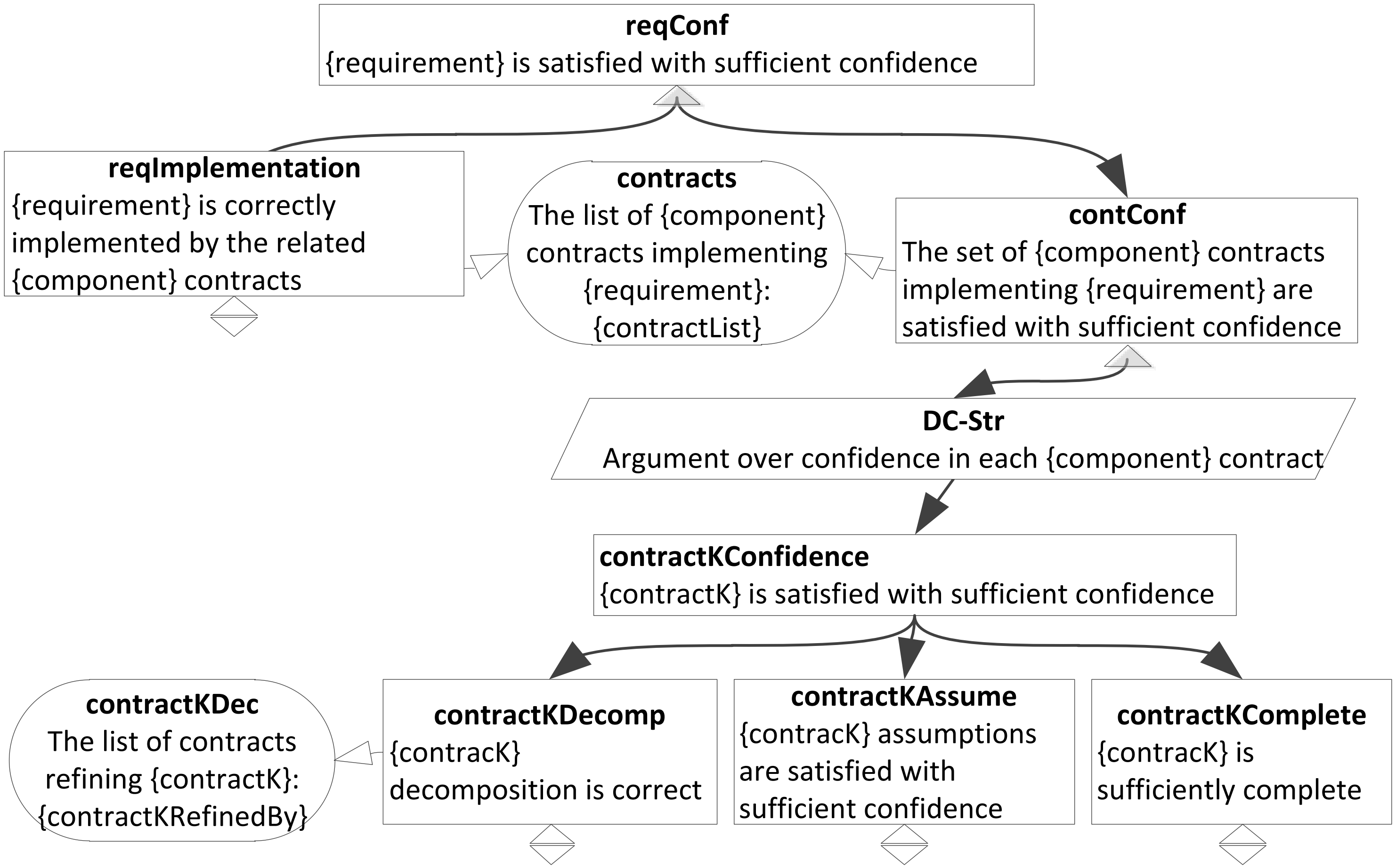
Failure of sub component Y implementation in satisfying the contract C2



Update the architecture if the safety requirements are not met (e.g. by adding redundance)

Scenario step: link to assurance

- Automatic generation of argumentation fragments
  - Usage of traceability links between system architecture, assurance case and evidence entities



Scenario Outcome

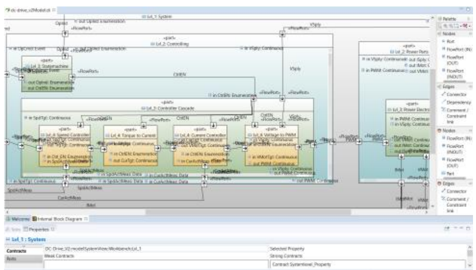
- Number of requirements formalized
  - Requirements template, semi-formal to formal requirement transformation, ad-hoc LTL editor assistance allow to improve this metric
  - Good quality of requirements and requirement traceability can be assured
- Effort for assurance and certification
  - With tools like CHESS, Savona and using SysML and contracts in comparison to conventional approaches, it is possible to achieve a higher number of automated assurance objectives and hence an improvement of this metric
  - Using formal proof decreases the cost of issue correction by detecting them earlier and raise the assurance
  - Using contracts we can reuse the assurance results for a subsystem in another context or system
    - Usage of strong and weak contracts formalism
  - Evidences about contracts fullfillment have to be provided for the leaf components only
    - By providing the contract refinement verification results as evidence
  - System assurance is improved by collecting the automatically generated evidences
- Effectiveness in system architecture issues identification
  - By using Component+Contract based design and connection to V&V formal verification tools it is possible to improve this metric
    - E.g. the guarantee that the components assembly /decomposition is correct reduces system design and integration errors

OpenCert P2 prototype

Early Safety Assessment

- Combine simulation-based fault injection, together with the contract-based approach and the insertion of monitors.

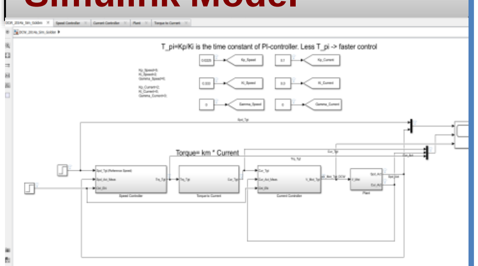
CHESS Model+safety contracts and faulty behaviour

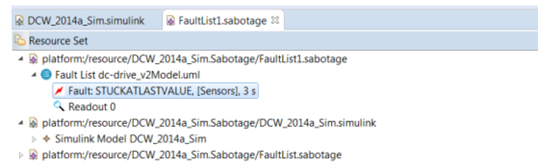


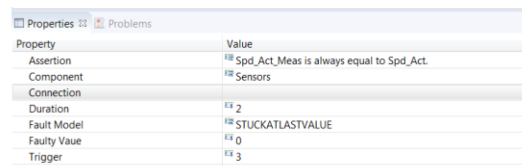
Extend Simulink Model with faulty behaviour (saboteurs) and monitors

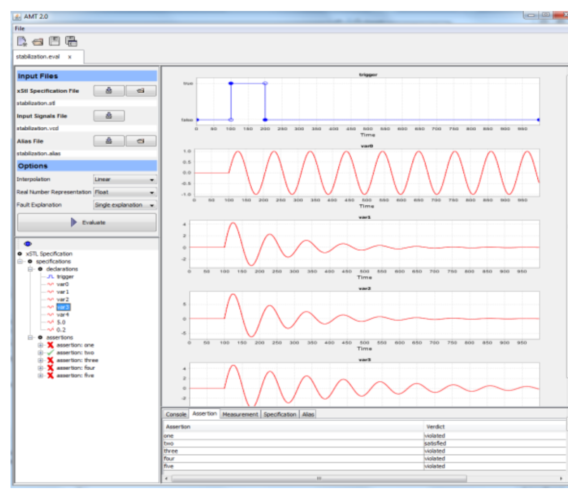
- Include **saboteurs** at component inputs which represent a violation of the assumption (omission, commission, true when false, false when true, too low, too high)
- Include **monitors** at component outputs and see if the guarantee still holds when the assumption is violated

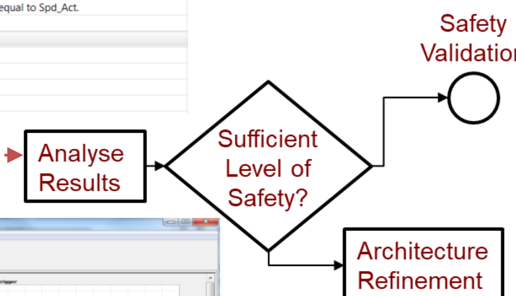
Simulink Model





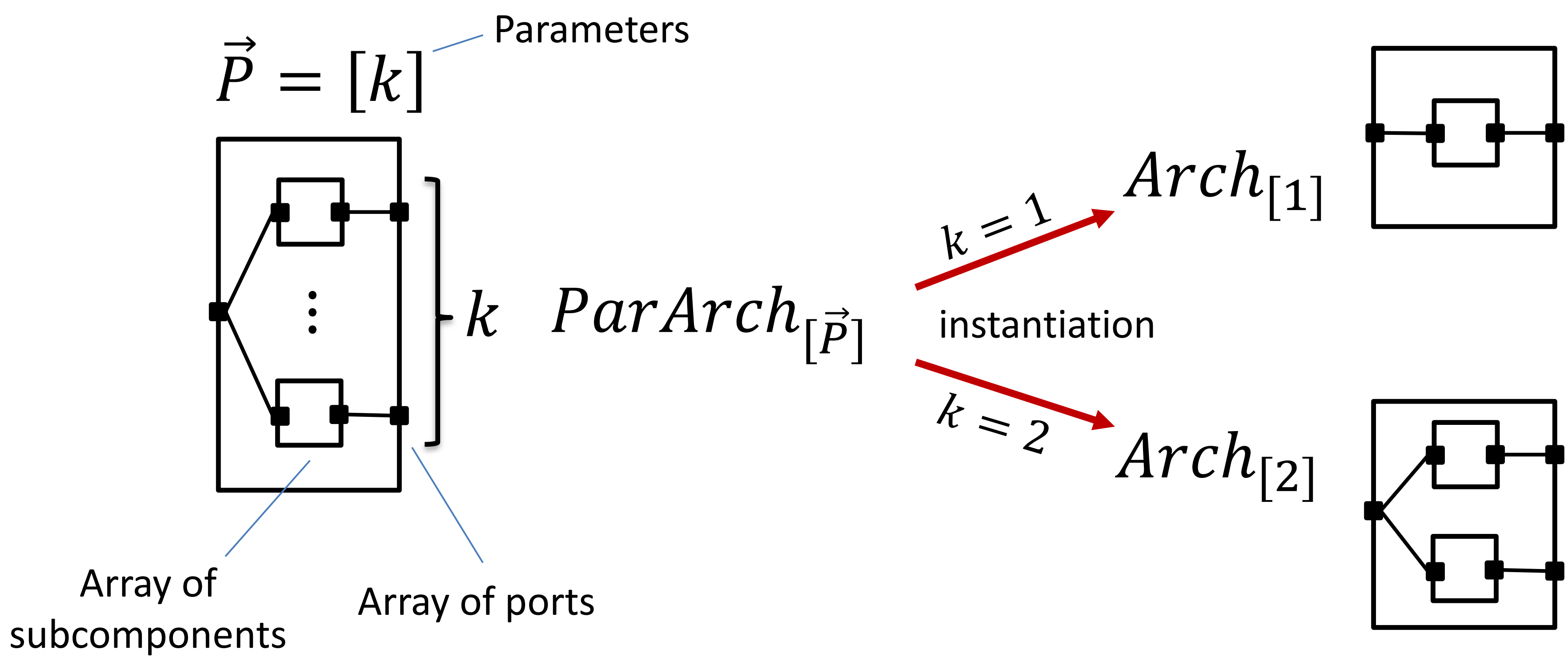




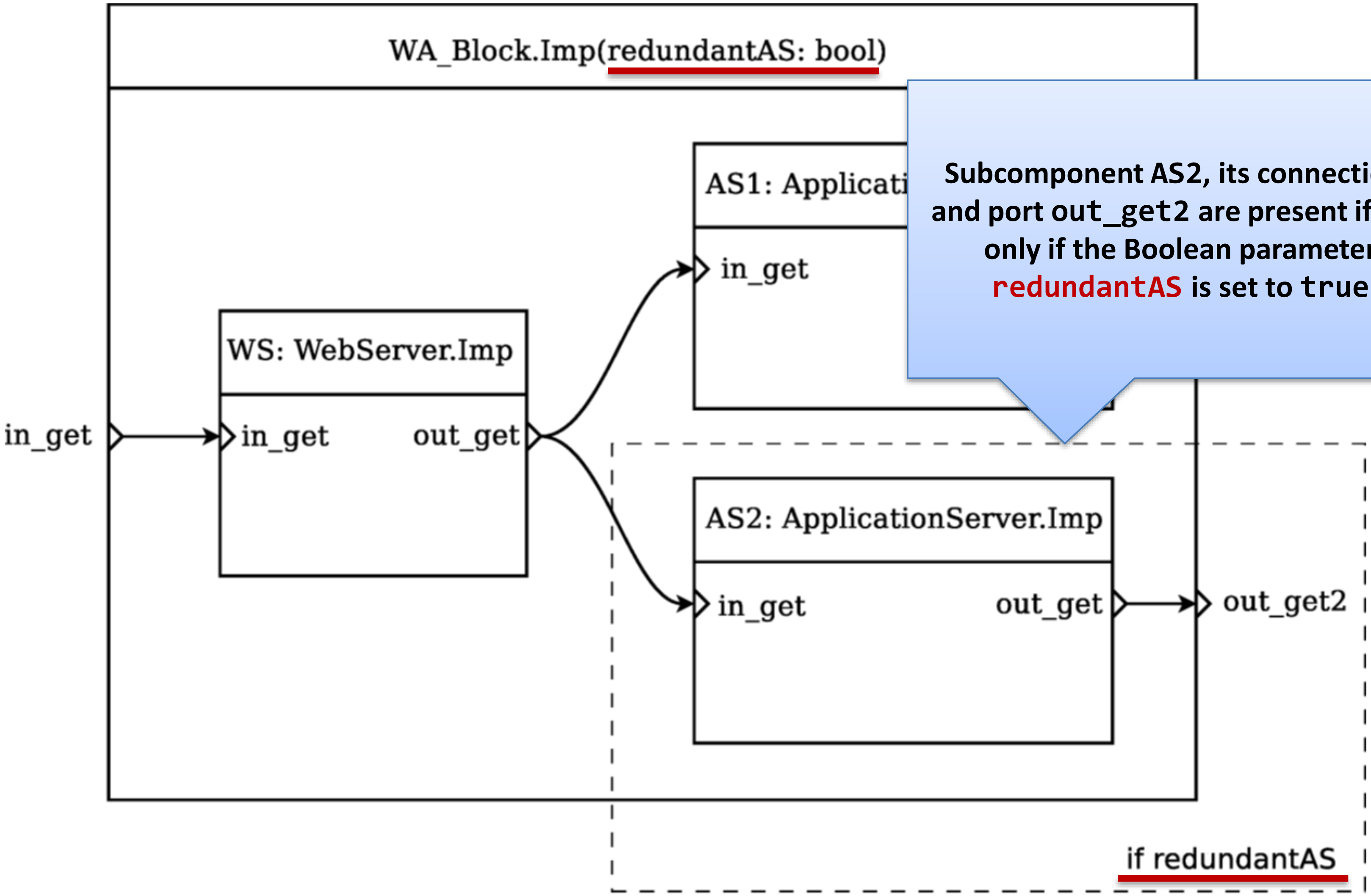


Support for Parameterized Architecture (reuse oriented)

- Parametric number of components/ports



Load balancer (LB) example: boolean parameters



Subcomponent AS2, its connections and port `out_get2` are present if and only if the Boolean parameter `redundantAS` is set to `true`

Others P2 features

- Extended support for metrics
  - About requiremens and architecture
- Extended integration with external modelling/analysis tools
  - Scade, Savona, SafetyArchitect
- FMEA generation from CHESS models
- Support for verification and validation of behavioural models
  - CHESS+external validation tools

Conclusions

- Several mode-based features and methodology guidelines have been provided, to support the different steps of CPSs development and feed the assurance case
  - Requirements specification, architectural design, V&V
  - Usage of Papyrus/CHESS tool integrated within OpenCert and external tools
- Currently we can provide claims stating why the AMASS architecture-driven assurance solution can improve the identified metrics
- Final iteration of AMASS case studies will be run in the next period by using the final prototype iteration (P2)
  - Values for identified metrics will be collected

Thank you for your attention!

?