CATSY_

Catalogue of System and Software Properties

- ESA Contract No: 4000111828/14/NL/FE
- Consortium:
  - SSF, Finland
  - FBK, Italy
  - RWTH Aachen, Germany
- ESA Technical officer: Andreas Jung

Overview of Objectives_

- Main goal: improve the early verification and validation (V&V) activities
- Define approach to derive formal property from informal requirement
- Define taxonomy of system and software properties
- Evaluate using suitable case study

OUTLINE_

1. Methodology
2. The CSSP
3. COMPASS
4. Case Studies
5. Evaluation
6. Conclusions

METHODOLOGY_

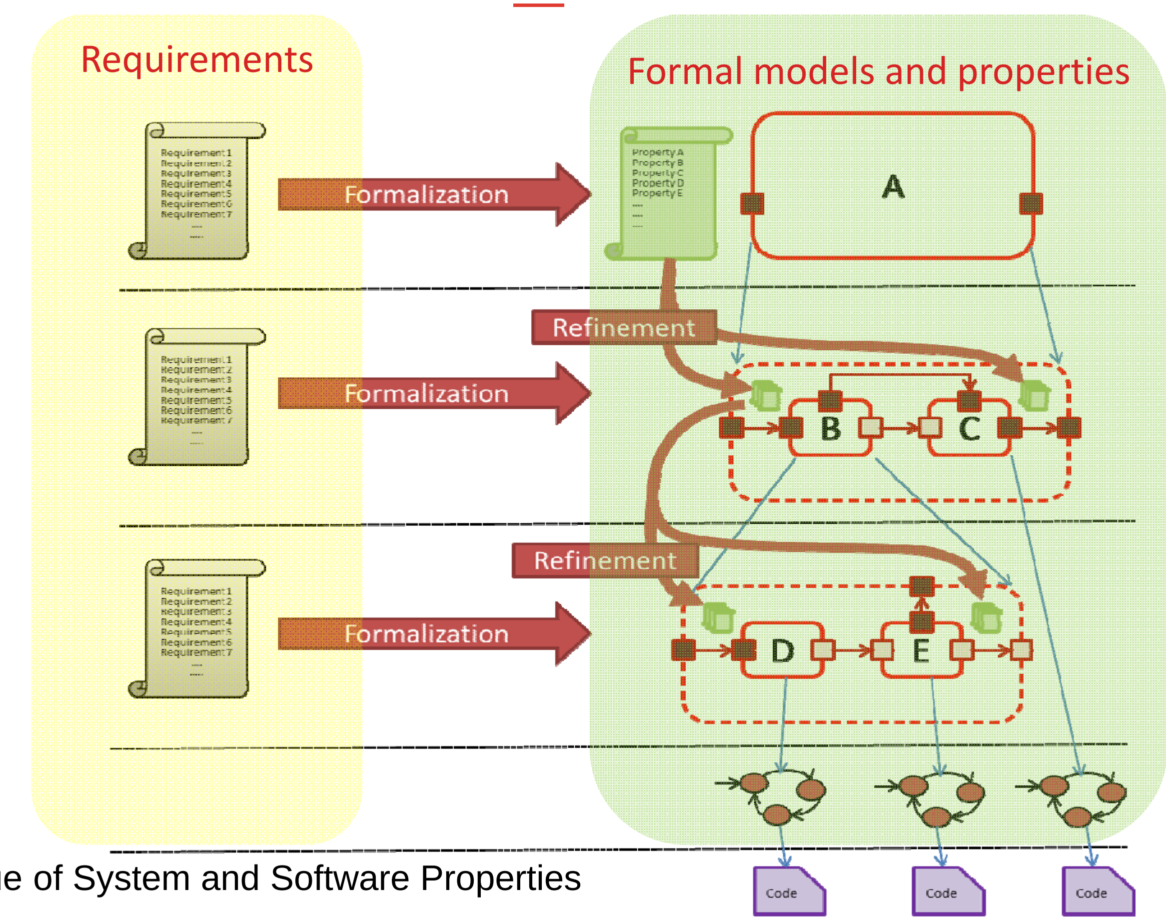
- Requirement Flowdown
- Taxonomy of Requirements
- Examples of Requirements Classes
- Requirements and Properties in a SW Development Project
- Example Properties

Requirement Flowdown

System level

Avionics level

Software level



Taxonomy of Requirements_

- **5 Abstraction levels**
  - **Mission**: mission objectives, products, and services
  - **System-of-Systems**: ground, space, and support segments
  - **System**: satellites, launchers
  - **Sub-system**: AOCS, electrical power control, software
    - Avionics
    - Software
  - **Equipment**: valves, batteries, electronic boxes

Taxonomy of Requirements_

- **Requirement classes**
  - Derived from space industry standards, focus on technical requirements
    - Hierarchy of requirement classes
    - 5 top-level classes
    - 13 classes in total
  - Requirements and properties

- **Property**: a requirement specified as a parameter with a certain value.

Examples of Requirement Classes

- Interface requirements
  - Interconnections
  - physical, thermal, electrical, or software
- Operational requirements
  - Communication protocols
  - Observability
- Performance requirements
  - Time
  - Space

Requirements and Properties in a SW Development Project_

Property: a requirement specified as a parameter with a certain value. Can be formalized.

- For each requirement/property:
  - Document type (SSS, IRD, SRD, ICD, …)
  - Abstraction level
  - How to formalize: formal model or formal property
  - Ensured by platform (that runs the SW)

Example properties

- Data format: SSS / TRD, System level, Formal model
- Command conditions: SSS, System level, Formal model and properties
- Access to shared state: SRD, Software level, Formal model and properties
- Clock synchronization: SSS, Avionics level, Formal model, ensured by platform

SPECIFICATION LANGUAGE AND THE CSSP_

- AADL
- Formal Properties and Patterns
- The CSSP
- Coverage of Design Attributes
- Example

Architecture Description Language_

- AADL: Architecture Analysis and Design Language
- SLIM: System Level Integrated Modelling
  - Extends AADL to support property specification
  - Input to COMPASS
- Components and sub-components
- Component types and component implementations
- Interconnected by (typed) ports

Example component_

```
system Car
  features
    battery_status : in data port enum(OK, DEAD);
end Car;

system implementation Car.Impl
  subcomponents
    battery: device Battery.Impl;
  flows
    battery_status := case battery.output > 0 : OK otherwise DEAD end;
end Car.Impl;
```


SSF

Formal Properties and Patterns_

- Requirements formalized into formal properties
- Formal properties: logical expressions over data/events/modes specified in the architecture
    - Example: “fdi.alarm implies sensor.temperature>=threshold”
- Different logics: propositional, first-order, temporal (LTL/CTL), probabilistic (CSL)
- Patterns (formulas with place-holders):
    - The atomic proposition P never holds
    - Whenever the atomic proposition P holds, this is eventually responded with S
    - P will eventually become true within Time1 and Time2 with a probability p.

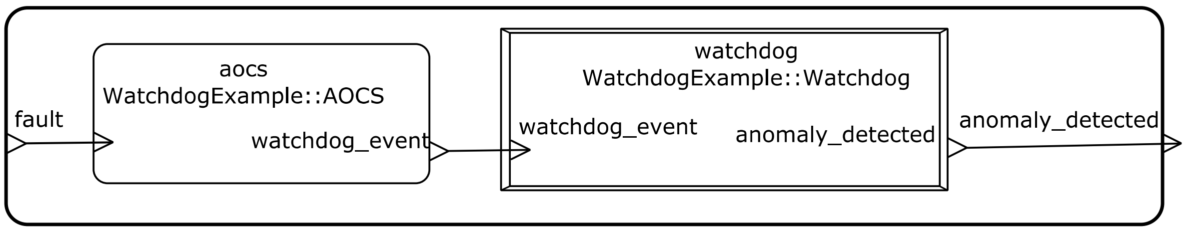
The CSSP_

- Detailed designs offer low-level attributes/properties
- Have a clear corresponding formal property
- Still required background in the corresponding logic
- Catalogue of System and Software Properties (CSSP): list of properties with a predefined formalization
- Example:
    - "PeriodInterval" and "PeriodOffset" associated to an event or event data port
    - They are *Time* attributes
    - Used to define the periodicity of an event e with period p and offset o
    - Implicit formal property: $F_{=o} e \wedge G\ (e \rightarrow \neg e\ U_{=p}\ e)$

Coverage of Design Attributes

Design Attributes	**CSSP** parameters or structure
processing capacity, clock frequency, endianness, memory size, addressable memory units	InvariantRange or Function property associated to corresponding connected feature
list of sub-systems, type of sub-systems, connections between sub-systems	Modeled as AADL subcomponents and port connections between components.
response to inputs and reaction time	Reaction and ReactionLatency associated to input event or event data ports.
events or specific functions	Modeled as AADL event ports
The input that is required to generate an output	PrecededBy associated to output event or event data ports.
. . .	. . .

Watchdog example_




```
device Watchdog
  features
    watchdog_event : in event port;
    anomaly_detected : out event port;
end Watchdog;

system AOCS
  features
    watchdog_event : out event port;
    fault: in event port;
end AOCS;
```

Watchdog example

```
device Watchdog
 features
   watchdog_event : in event port;
   anomaly_detected : out event port;
 properties
   CSSP::Timeout => 1 Sec applies to anomaly_detected;
   CSSP::TimeoutReset => reference(watchdog_event) applies to anomaly_detected;
end Watchdog;
```

$$G(O_{\leq 1sec}\texttt{watchdog\_event} \rightarrow \neg\texttt{anomaly\_detected}) \wedge GF_{\leq 1sec}(\texttt{anomaly\_detected} \vee \texttt{watchdog\_event})$$

```
 features
   watchdog_event : out event port;
   fault: in event port;
 end AOCS;

 system implementation AOCS.impl
 modes
```

$$G\left(\texttt{alive} \rightarrow F(\texttt{watchdog\_event} \vee \neg\texttt{alive})\right)$$
$$\wedge\, G\left((\texttt{watchdog\_event} \wedge \texttt{alive}) \rightarrow (F_{\leq 1sec}\neg\texttt{alive} \vee \neg\texttt{watchdog\_event}\, U_{=1sec}\, \texttt{watchdog\_event})\right)$$

```
 properties
   CSSP::PeriodInterval => 1 Sec applies to watchdog_event;
   CSSP::PeriodEnabled => (reference(alive)) applies to watchdog_event;
   CSSP::ModeInhibited => (reference(watchdog_event)) applies to dead;
end AOCS.impl;
```

$$G(\texttt{dead} \rightarrow \neg\texttt{watchdog\_event})$$

COMPASS_

- Project
- Toolset
- Workflow

COMPASS Project_

- Started in 2008
- Aim: system-software co-engineering approach for evaluation of *correctness*, *safety*, *dependability* and *performability* of aerospace systems.
- Various follow-up projects over the years
  - AUTOGEF, HASDEL, FAME, CATSY (ESA funded)
  - DMILS, CITADEL (EU funded)

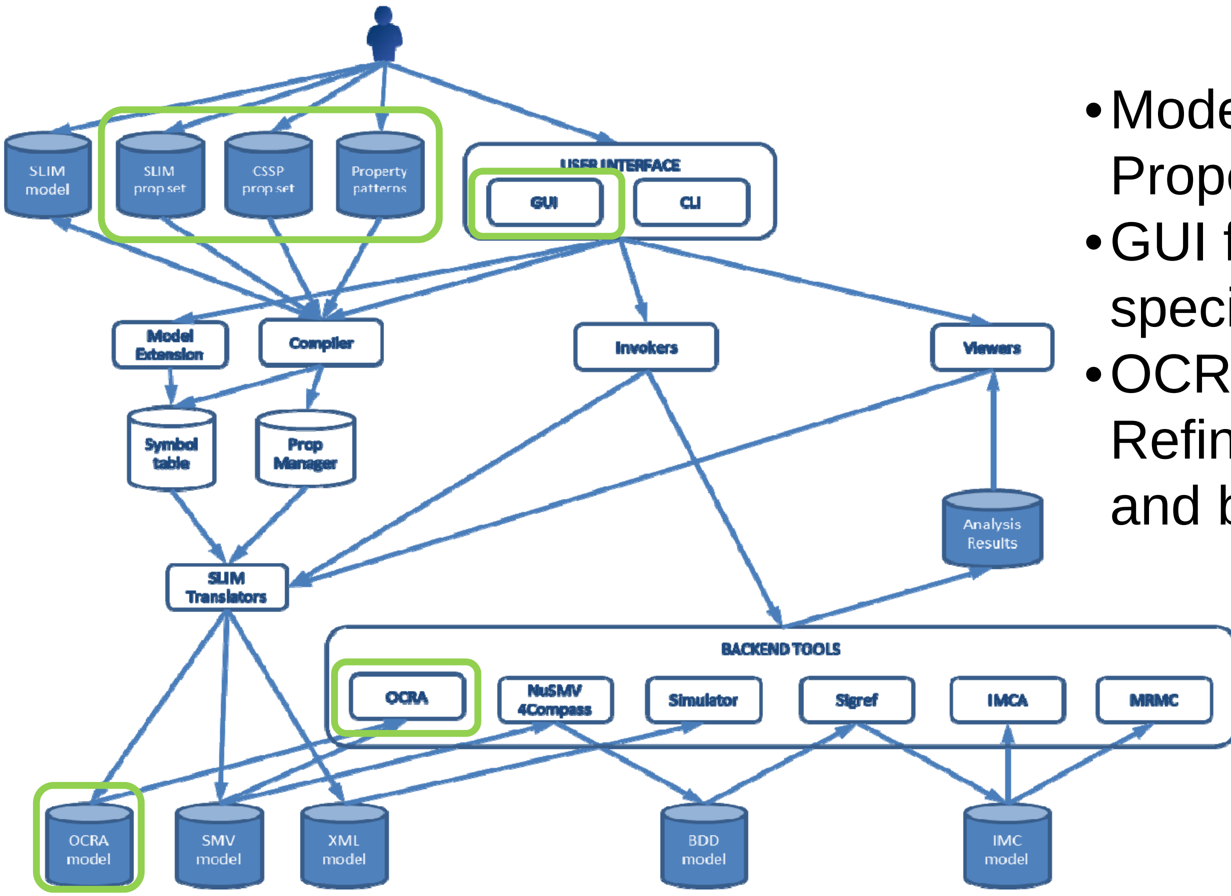
COMPASS Toolset_

- Integrated toolset for
- Input language based on AADL: SLIM (System Level Integrated Modeling)
  - Components
  - Ports
  - Modes
  - …
- Model extension: Embed error behavior into model

COMPASS Toolset_

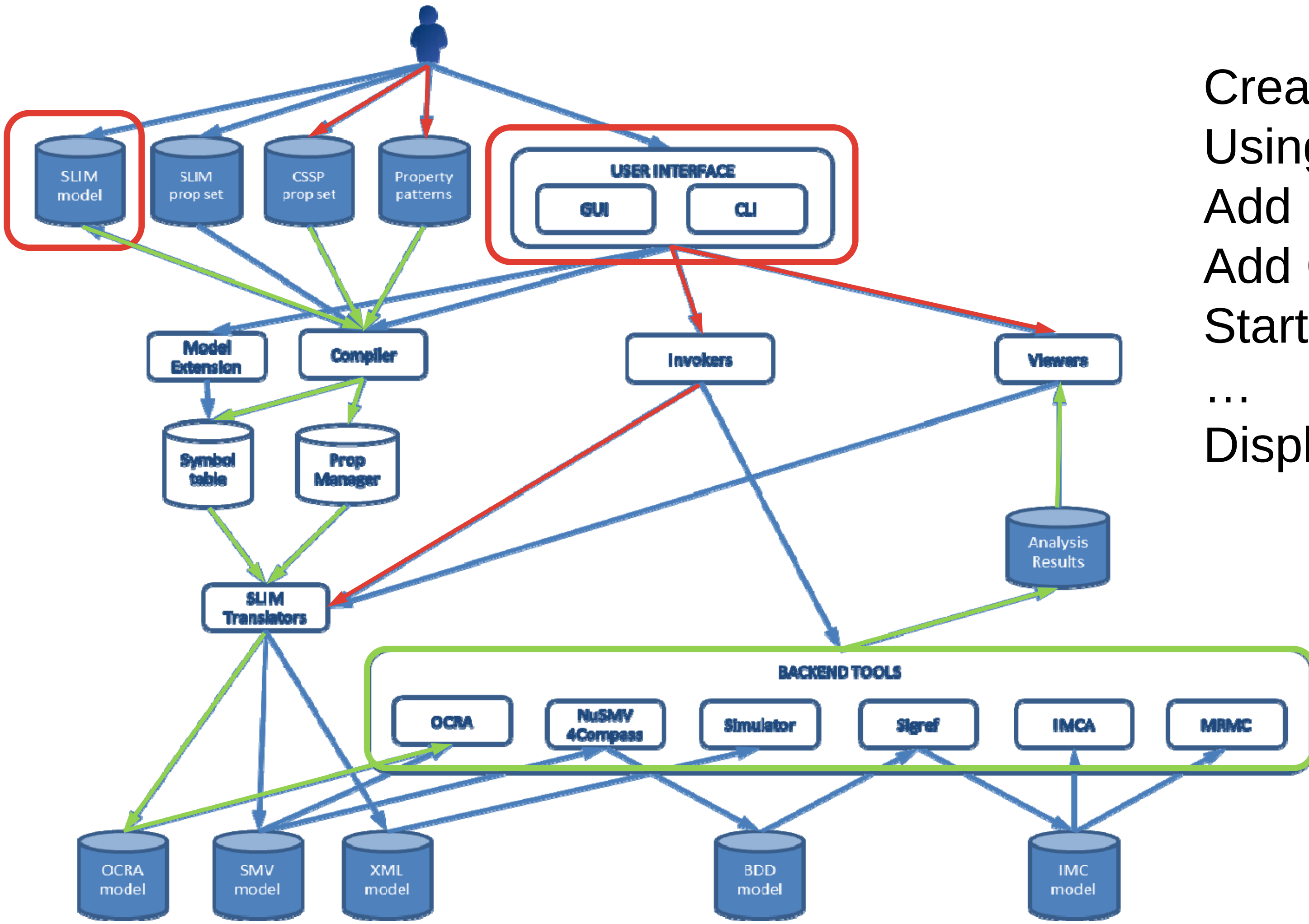
- Extensions developed during CATSY:
  - Model properties (AADL)
  - Upgraded pattern specification (GUI)
  - CSSP property specification (GUI)
  - New verification engines (OCRA)

COMPASS Toolset: Overview



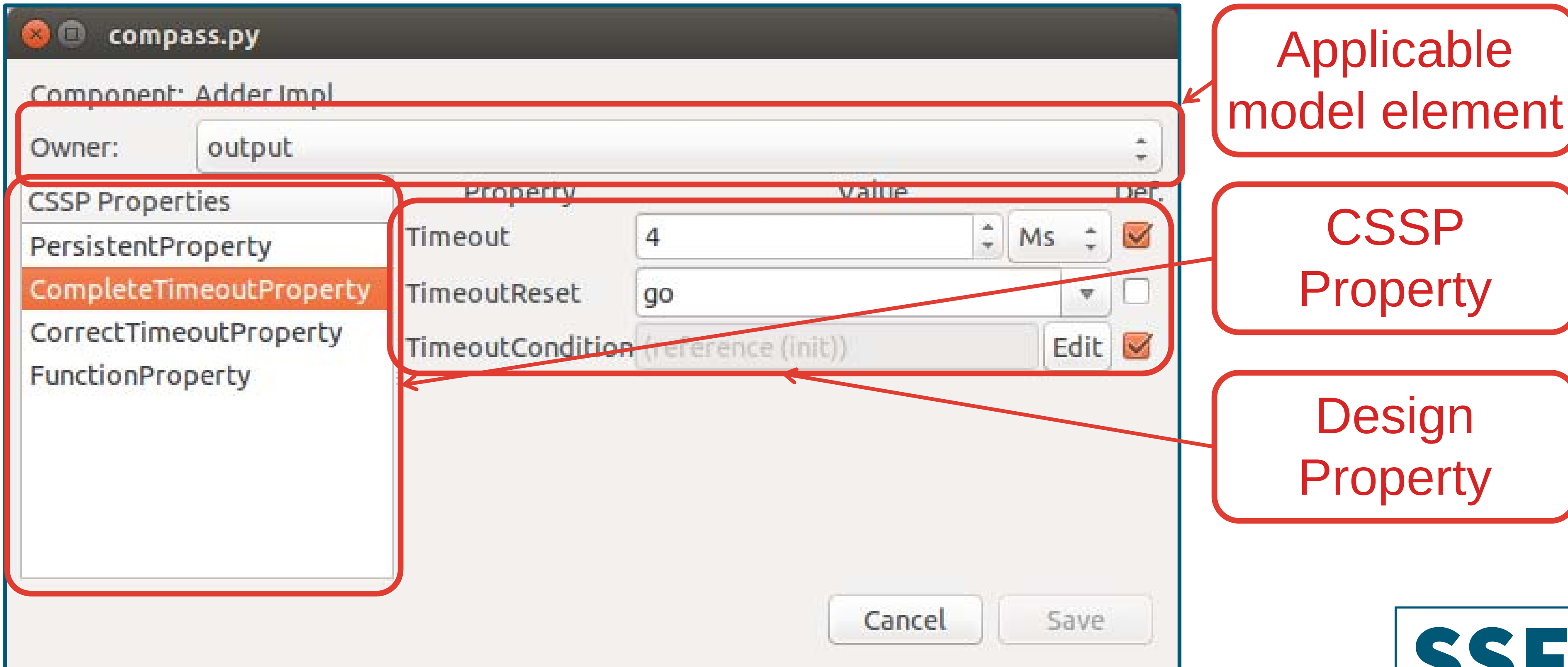
- Model Properties and Property Patterns
- GUI for property specification
- OCRA (Othello Contract Refinement Analysis) tool and backends

COMPASS Toolset: Workflow_

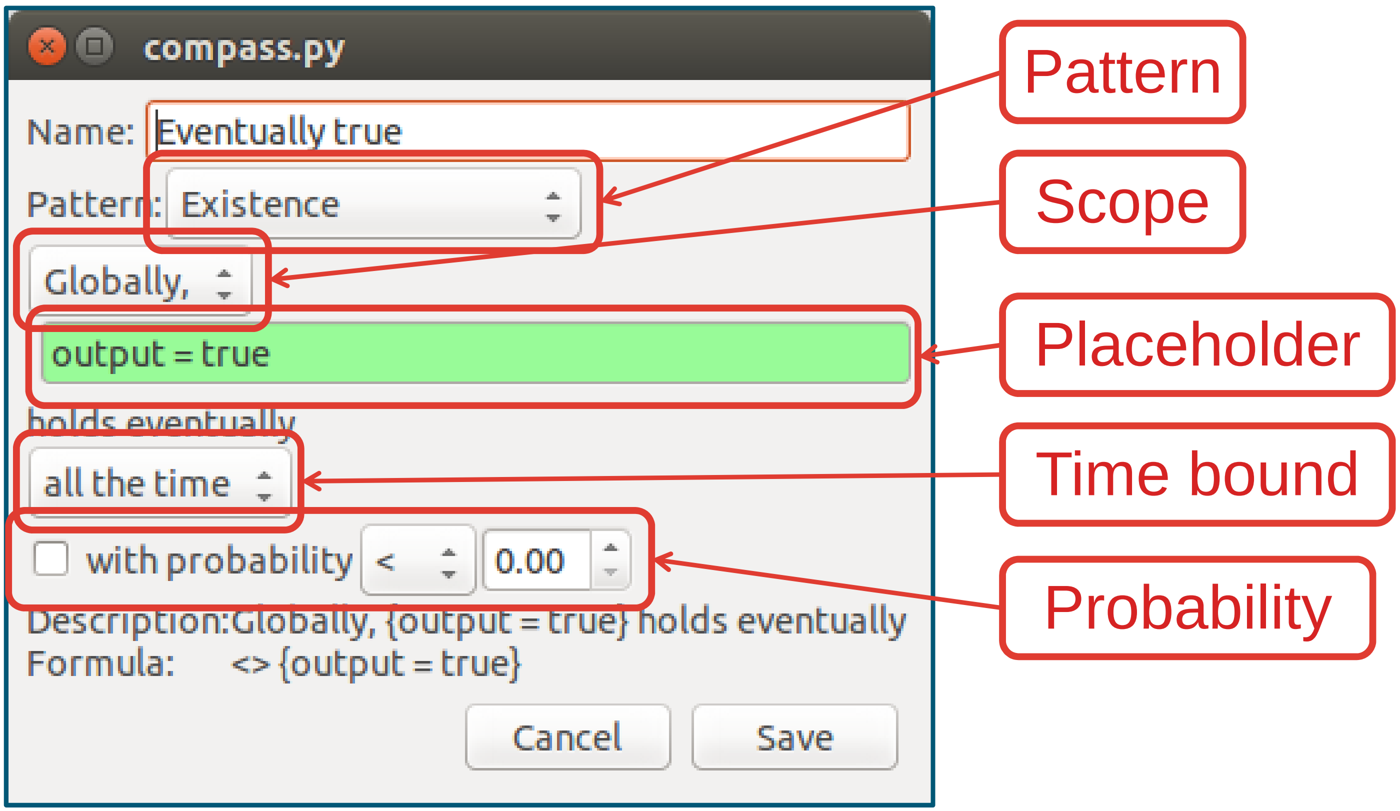


Create Design Model
Using GUI:
Add properties
Add CSSP Attributes
Start analysis
…
Display result

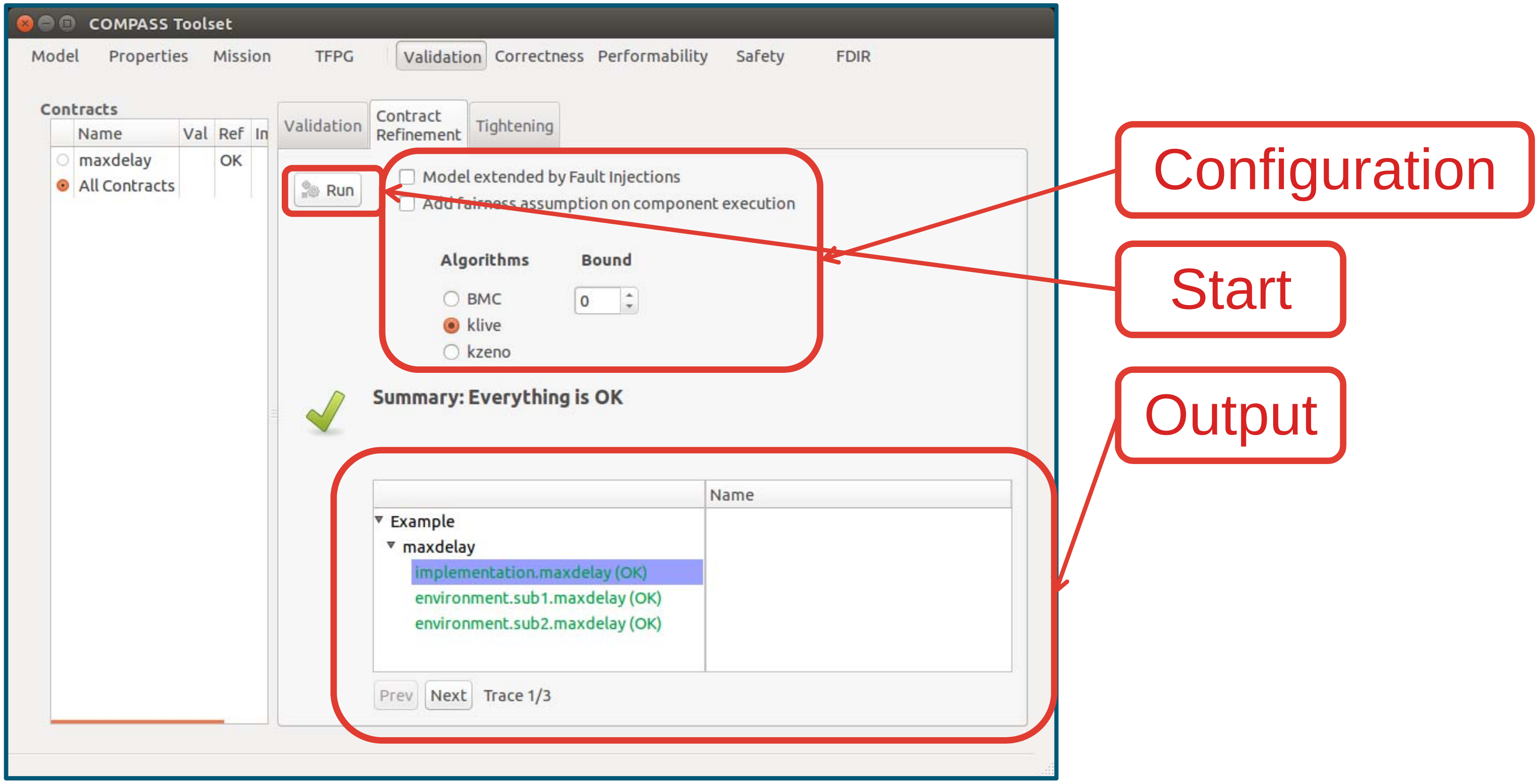
COMPASS GUI


compass.py
Component: Adder.Impl
Owner:
output
CSSP Properties
PersistentProperty
CompleteTimeoutProperty
CorrectTimeoutProperty
FunctionProperty
Property
Value
Def.
Timeout
4
Ms
TimeoutReset
go
TimeoutCondition
(reference (init))
Edit
Cancel
Save
Applicable model element
CSSP Property
Design Property

COMPASS GUI _


compass.py
Name: Eventually true
Pattern: Existence
Globally,
output = true
holds eventually
all the time
with probability
<
0.00
Description:Globally, {output = true} holds eventually
Formula: <> {output = true}
Cancel
Save
Pattern
Scope
Placeholder
Time bound
Probability

COMPASS GUI _


COMPASS Toolset
Model
Properties
Mission
TFPG
Validation
Correctness
Performability
Safety
FDIR
Contracts
Name
Val
Ref
maxdelay
OK
All Contracts
Validation
Contract Refinement
Tightening
Run
Model extended by Fault Injections
Add fairness assumption on component execution
Algorithms
Bound
BMC
klive
kzeno
Summary: Everything is OK
Name
Example
maxdelay
implementation.maxdelay (OK)
environment.sub1.maxdelay (OK)
environment.sub2.maxdelay (OK)
Prev
Next
Trace 1/3
Configuration
Start
Output

CASE STUDY_

- BepiColombo SSMM
- Requirement Selection
- Requirement Modelling
- Analysis and Results

BepiColombo Solid State Mass Memory_

- Concurrent data storage facility for
  - 9 payloads
  - On-board computer
  - Telemetry format generator
- Implemented in FPGAs and memory modules
- Controlled by SSMM Application software (ASW)

Requirements Selection_

Abstraction levels: System, Avionics, Software

Tracing between different levels

Focus on requirements specifying:

- Communication SSMM and other on-board systems
- FDIR approach in SSMM
- Booting and rebooting of processor modules
- File transfer session protocol

SSF_

Requirements Modelling_

- 25 system level requirements
- 24 avionics level requirements
- 9 software level requirements

Percentage of total number of requirements: 3.2%

Case Studies – Analysis and Results_

- Formal property validation
  - Satisfiability of guarantee expressions
  - Compatibility of sets of properties and components
  - Implication of properties by other properties
- Validation of refinement relations
- Calculation of the probability of mass memory corruption
- Analysis of coverage of requirements classification

SSF_

Achievements_

- Formalisation of representative space system requirements at different abstraction levels
- Use of the CSSP to capture properties
- Automated validation of properties
- Automated validation of “implemented-by” relations between different abstraction levels
- Demonstrated COMPASS performance in general very good

Not demonstrated by case study_

- Interaction between conventional requirements engineering process and the CATSY approach
- Benefits of the requirements classification
- Benefits of all property types in the CSSP

EVALUATION_

- Benefits
- Limitations
- Lessons learned

Evaluation – Potential Benefits_

- Harmonisation of requirements
  at one abstraction level and across abstraction levels
- Verification of requirements
  - Mutual consistency of requirements
  - Compatibility: property does not contradict requirements
  - Entailment: property is consequence of requirements
- COMPASS more reliable than manual verification
- Early verification: before implementation phase

Evaluation – Potential Benefits (cont'd.)

- Verification of traceability
  - Automated checking of formal contracts
- COMPASS enables systematic dependability and safety analysis:
  - Generation of failure probability data
  - FTA, FMEA, FDIR

Evaluation – Limitations

- CATSY does not address all requirement types
- Some analyses by COMPASS take too much time
- A lot of human effort needed
  - Formalizing requirements
  - Specifying formal properties and contracts

  Comparable to efforts for static analysis of source code

Evaluation – Lessons Learnt

Value of the CSSP has not been demonstrated in the case study

- Case study uses “since” and “until” operators a lot, but these are not covered by CSSP property patterns

Way forward:

1. Case study with more focus on existing CSSP properties
2. extension of the CSSP

Evaluation – Lessons Learnt

Interaction of CATSY approach with requirements engineering has not been demonstrated

- Ad hoc selection of requirements to formalize
- Requirements not updated due to formalization

Way forward: application of CATSY in on-going requirements engineering phase

Evaluation – Lessons Learnt_

SLIM modes and states were avoided as much as possible

- Limited COMPASS functionality

Way forward:

- extend SLIM for refinement of modes and states
- Extend COMPASS to analyse this refinement

Evaluation – Lessons Learnt_

Complicated temperal logic formulas

- Capturing many things in single formulas

Way forward: first check if requirements can be simplified, then introduce more CSSP property patterns

Possible Research and Development for Industrialisation_

- Extension of the CSSP
- Extension of SLIM
- Support for algorithmic requirements
- Support for test case specification
- Support for redundancy ASIL algebra
- Scalability of COMPASS

Relation to the other CSSP project_

- Similar architecture-based design:
  - Main differences between SLIM and BIP:
    - SLIM allows hybrid modeling (continuous variables)
    - BIP allows more complex interactions and priorities
- Similar approach to formalization:
  - Ontology only in CSSP(AUoT)
  - Boilerplates only in CSSP(AUoT)
  - Catalogue of predefined properties only in CSSP(SSF)
  - More expressive logic (fragments of FO-LTL, MTL, CSL) used in CSSP(SSF)
  - Richer validation analysis used in CSSP(SSF)

Relation to the other CSSP project_

- Tool support
  - Similar engines:
    - nuXmv, DFinder in CSSP(AUoT)
    - nuXmv, xSAP, OCRA, MRMC in CSSP(SSF)
  - Single front-end in CSSP(SSF) fully integrated with other COMPASS analysis
  - More loosely integrated in CSSP(AUoT): ontology, BIP, nuXmv

End of presentation

http://www.compass-toolset.org/catsy