CATSY: Catalogue of System and Software Properties

Harold Bruintjes (RWTH)
Fondazione Bruno Kesseler (FBK)
Space Systems Finland (SSF)

Introduction

CSSP

Catalogue of System and Software Properties

- Objective: Define a catalogue of the properties used for early Verification and Validation activities;
- Provide a systematic way for derivation, specification and flow-down through different architectural levels and across different design phases, and
- provide technologies for a cohesive environment for the specification and validation activities

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

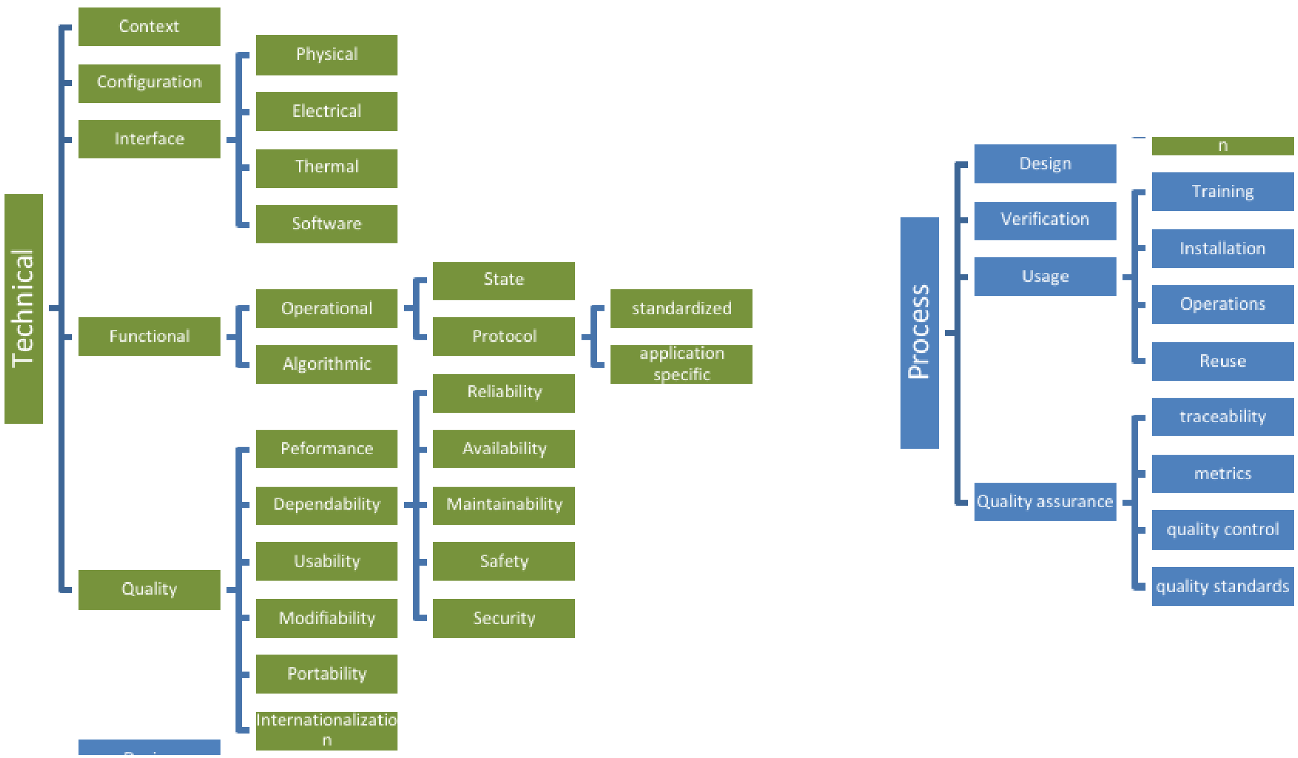
Methodology

Three phases:

1. Informal Analysis: Classification of requirements. Given a requirement taxonomy (in CATSY based on ECSS standards), determine class of individual requirements (not necessarily 1:1 mapping).
2. Formalization: Based on properties (design attributes) associated with taxonomy classes, determine formalization.
3. Formal Validation: Use formal techniques to validate the formalized requirements using formal verification engines. This may find errors in requirements specification or formalization.

Informal Analysis: Requirement Taxonomy

- Derived from ECSS standards
- For each class, design attributes (properties) are defined
- Focus is on Technical requirements



Informal Analysis: Requirement Taxonomy

- Derived from ECSS standards
- For each class, design attributes (properties) are defined
- Focus is on Technical requirements

4. 1 *Input/Output Functional Requirements* describe the relation between input/output of the system. Typical design attributes associated to input/output functional requirements are the output that is generated in *response* to an input of the component, the maximum *reaction time* that can elapse between the received input and the generated output, and the *input that is required to generate an output*.

Informal Analysis: Requirement Taxonomy

Three types of attributes

1. Non-formalized
2. Formalized in design model (modes/configuration, subcomponents)
   →SLIM
3. Formalized by property (behavior)
   → CSSP →AADL properties

Formalization

- Design attributes are encoded as property values from the CSSP property set
- One or more property values make up a formal property
- Formal property can be validated directly, or embedded into a contract

Formalization: CSSP Property set example

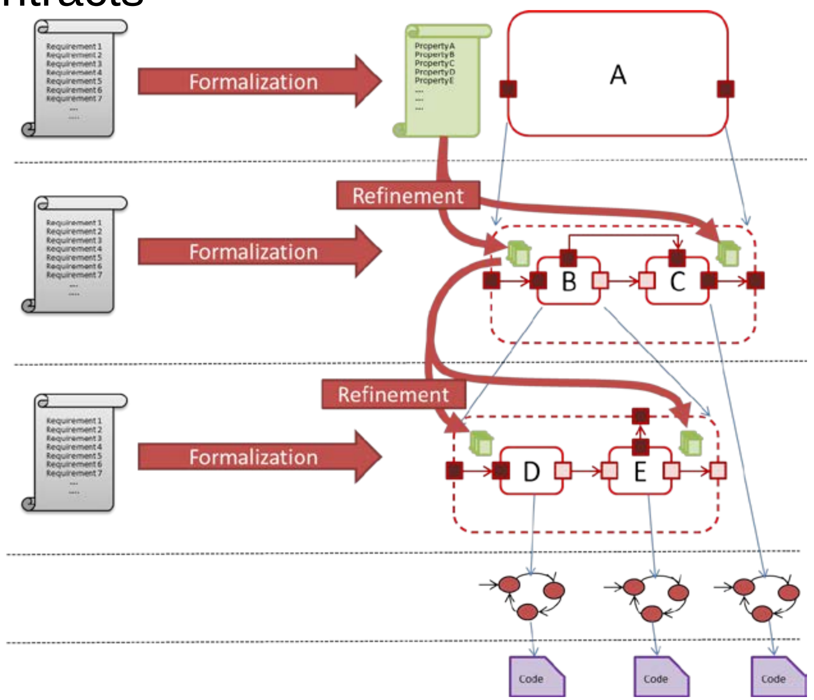
```
-- Monitoring properties.
-- for every input event data port p of numeric type
-- if MonitorRange(p) and MonitorResponse(p) are defined,
-- the following formal property is defined
-- MonitorProperty(p) := "G ((p & mode in MonitorEnabled(p) &
--  !(data(p) in MonitorRange(p))) -> F_I MonitorResponse(p))"
--  if MonitorEnabled(p) is defined
-- where I=[0,MonitorDelay(p)] if MonitorDelay(p) is defined
-- else I=[0,+infinity)
MonitorRange: range of aadlinteger
  applies to (event data port);
MonitorResponse: reference(event port, event data port)
  applies to (event data port);
MonitorDelay: Time
  applies to (event data port);
MonitorEnabled: list of reference(mode)
  applies to (event data port);
```


Software Modeling and Verification Chair

Formalization: Contract Based Refinement

Formal properties are specified at component interface level (event and data ports). This allows for a neat definition of refinement, which can be specified at the implementation in terms of subcomponent contracts

- Contracts:
  Pair of assumption and guarantee
- Contract refinements:
  List of subcomponent contracts



Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

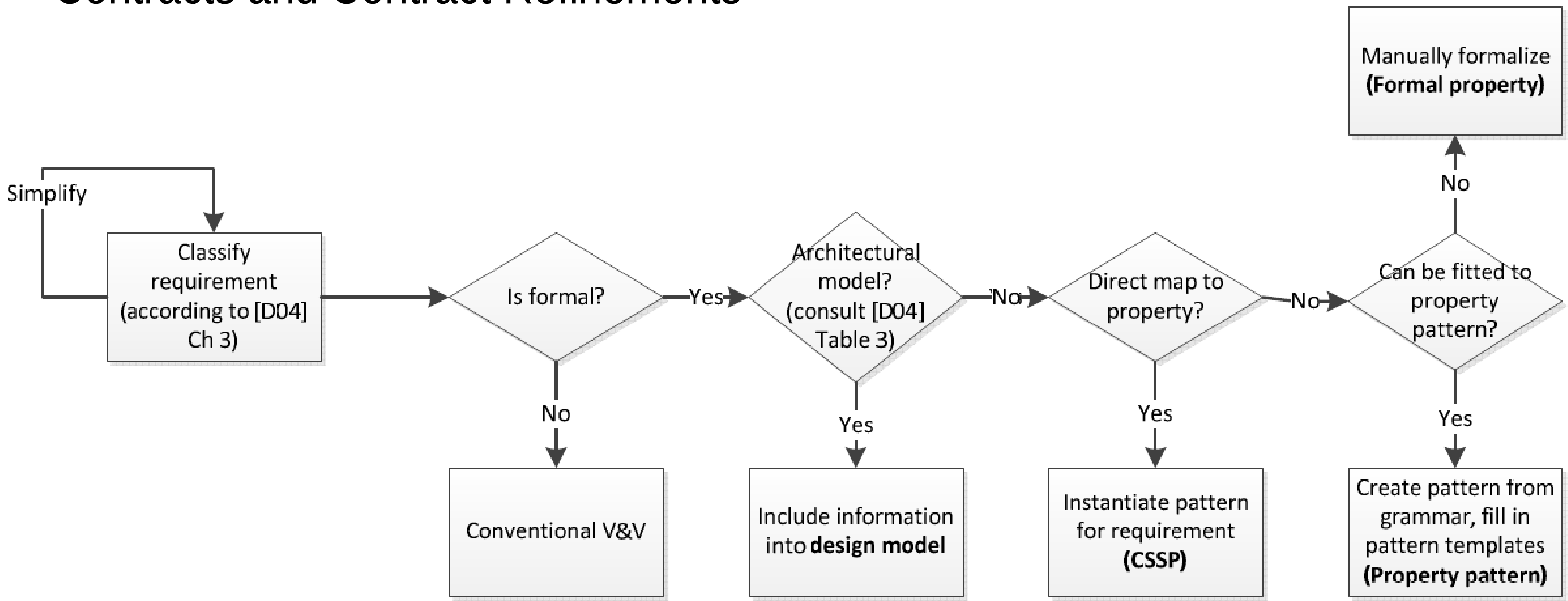
Formalization: Pattern Based & Generic Properties

- Pattern based properties: Formulate property based on patterns using 5 scopes and 8 classes
  - Scopes: Global, Existence, Before, After, Between, After-Until
  - Classes: Universality, Absence, Existence, Recurrence, Precedence, Response, Response Invariance, Until
  - Optionally timed
  - Optionally probabilistic
- Generic properties: Enter properties directly

Software Modeling and Verification Chair

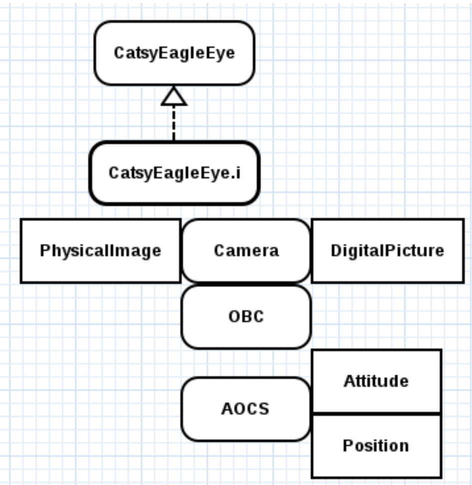
Formalization: Summary of Possible Properties

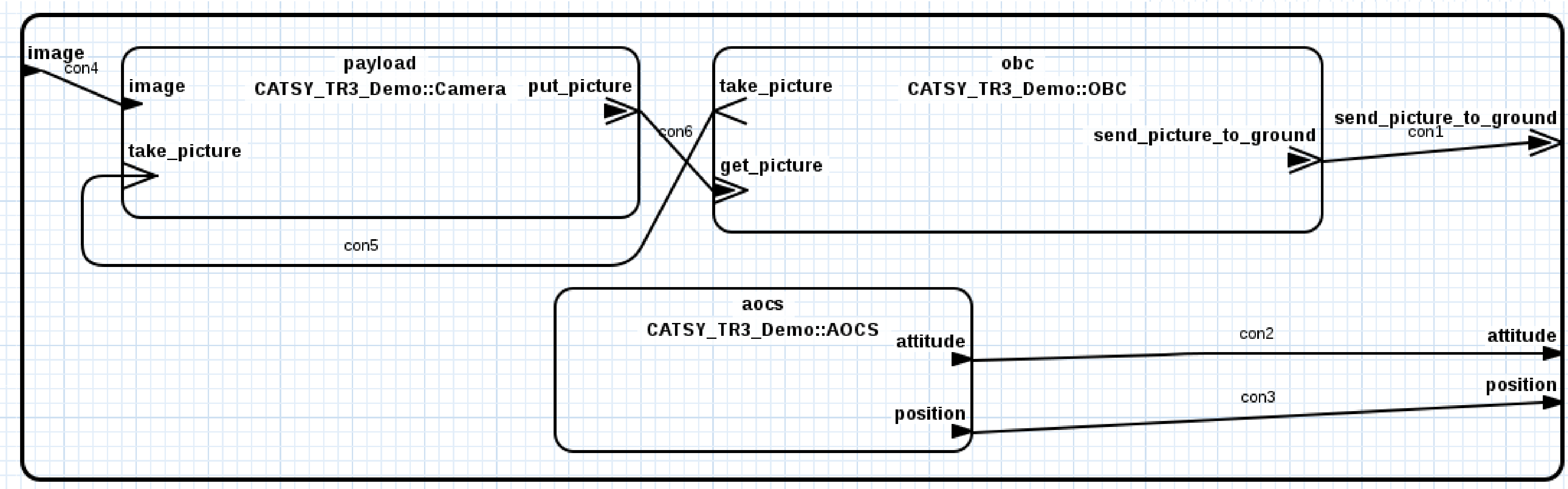
- CSSP property set
- Generic and pattern properties
- Contracts and Contract Refinements



Example

EagleEye: System structure



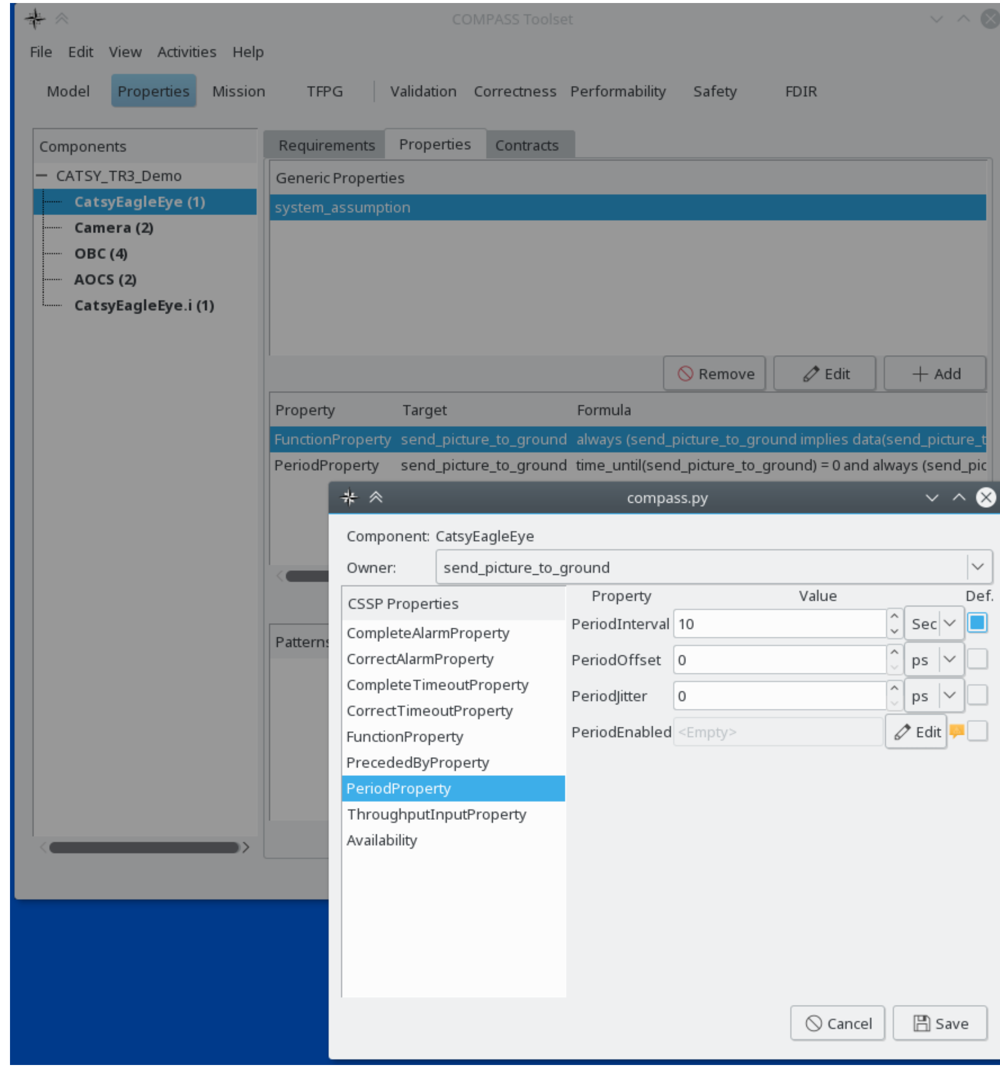


Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Example

CSSP Properties



Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Example

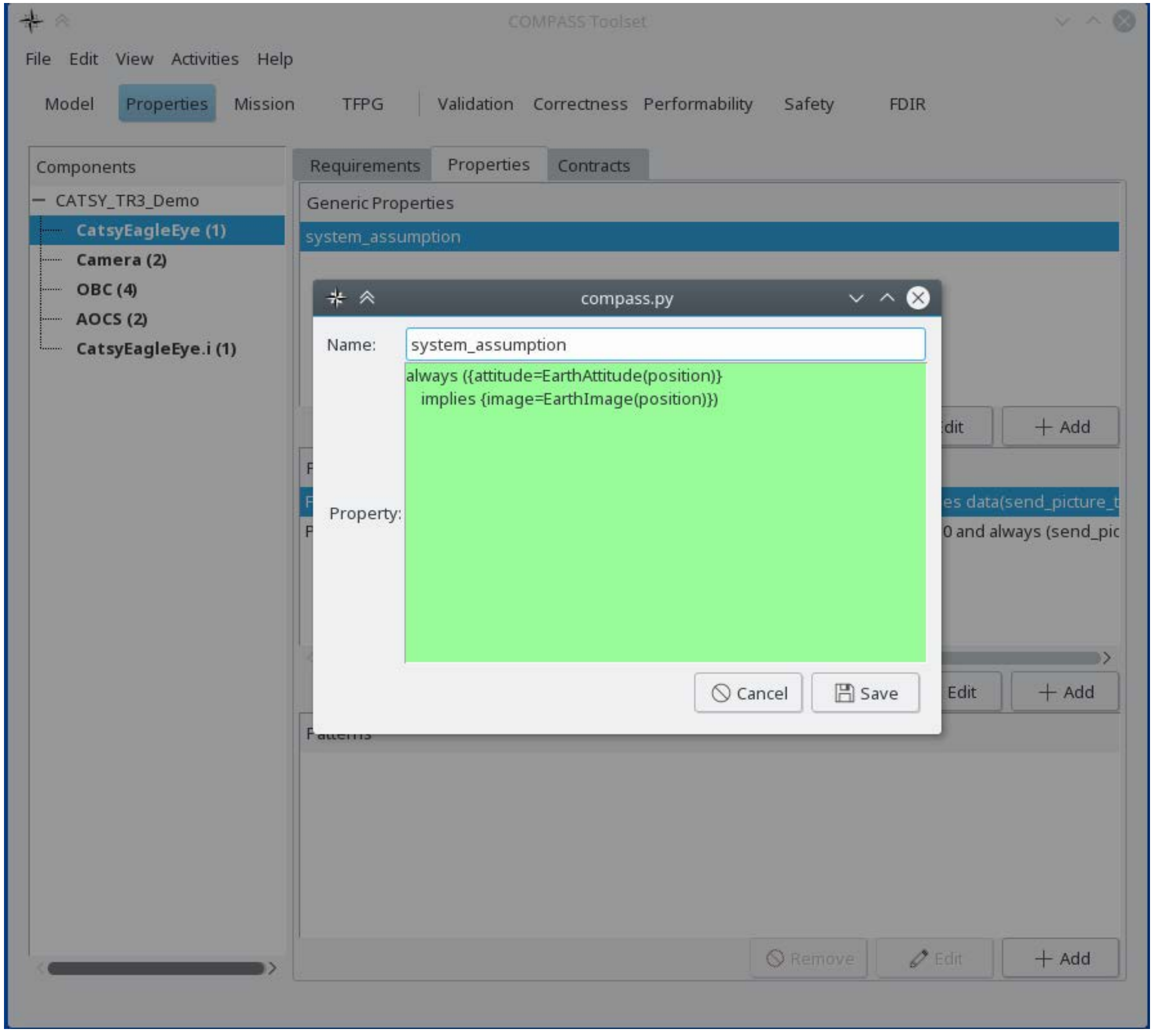
CSSP Properties

CSSP::PeriodInterval => 10 Sec applies to send_picture_to_ground;
CSSP::Function => "compress(picture(EarthImage(AboveEspoo)))"
applies to send_picture_to_ground;

Trigger send_picture_to_ground every 10 seconds.
The value of send_picture_to_ground matches a compressed picture

Example

Generic Properties



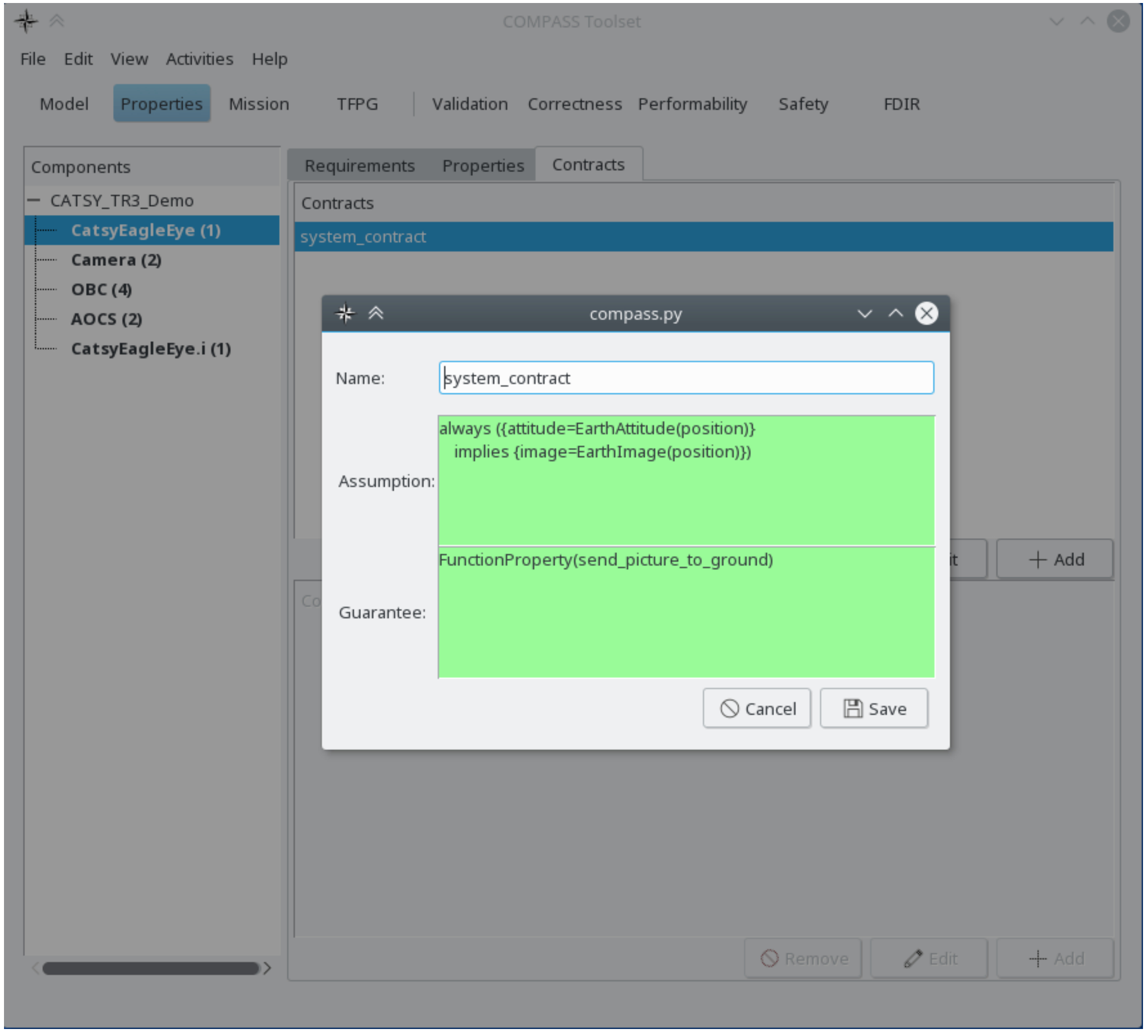
Example

Generic Properties

```
SLIMpropset::GenericProperties => ([Name => "system_assumption";
   Formula => "always (attitude=EarthAttitude(position) implies
image=EarthImage(position))";
   ]);
```

Example

Contracts



Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Example

Contracts

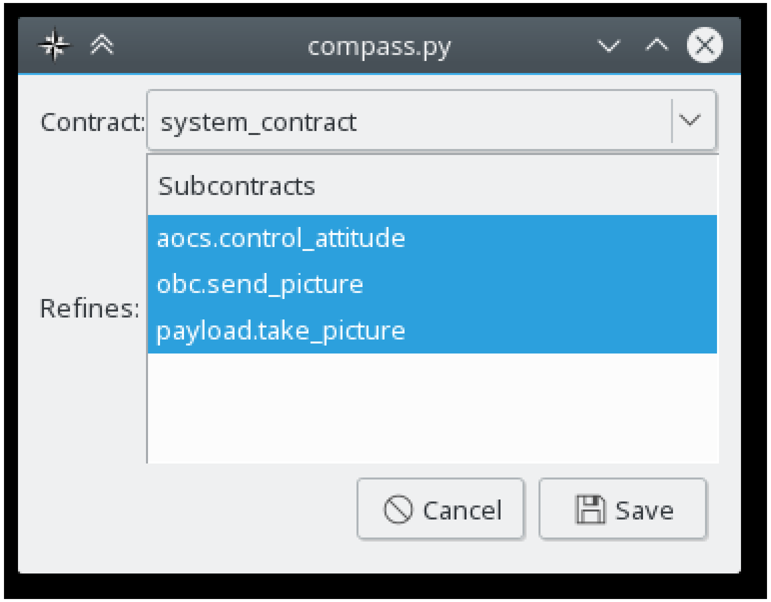
```
SLIMpropset::Contracts => ([Name => "system_contract";
   Assumption => " system_assumption";
   Guarantee => "FunctionProperty(send_picture_to_ground)";
   ]);
```

Reuse of both the generic property and CSSP property

Example

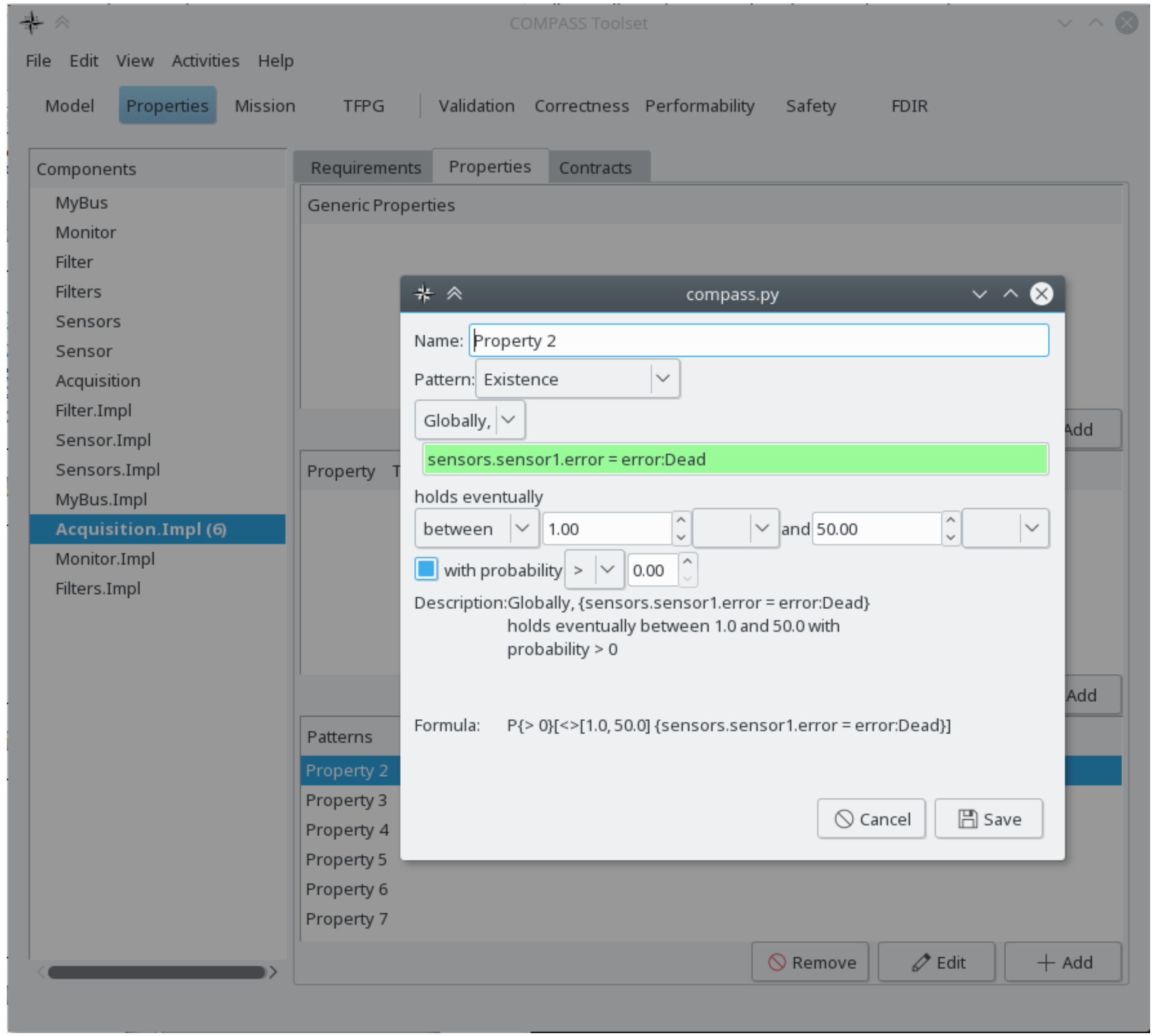
Contract Refinements

```
SLIMpropset::ContractRefinements => ([Contract => "system_contract";
  SubContracts => ("aocs.control_attitude", "obc.send_picture",
  "payload.take_picture");
  ]);
```



Example

Pattern Properties



Software Modeling and Verification Chair

Example

Pattern Properties

Patterns => (

[Name => "Property 2"; Pattern => "Globally, {sensors.sensor1.error = error:Dead} holds eventually between 1 and 50 with probability > 0";],

[Name => "Property 3"; Pattern => "Globally, {sensors.sensor2.error = error:Dead} holds eventually between 1 and 5 with probability > 0";],

[Name => "Property 4"; Pattern => "Globally, {sensors.sensor2.error = error:OK} holds without interruption until {sensors.sensor2.error = error:Glitched} holds between 0 and 10 with probability > 0";],

[Name => "Property 5"; Pattern => "Globally, it is always the case that {sensors.sensor2.error = error:OK} holds between 0 and 1 with probability > 0";],

[Name => "Property 6"; Pattern => "Globally, if {sensors.sensor2.error = error:OK} holds then it must be the case that {sensors.sensor2.error = error:Glitched} has occurred before between 0 and 10 with probability > 0";],

[Name => "Property 7"; Pattern => "Globally, if {sensors.sensor1.error = error:Dead} has occurred then in response {sensors.sensor2.error = error:OK} eventually holds between 0 and 10 with probability > 0";]

);

Changes in SLIM

Modes and States

- Separation of configuration and behavior
- Modes closer to AADL (no invariants,
- No guards on transitions)
- States closer to BA

```
system Car
end Car;

system implementation Car.Impl
  subcomponents
    -- subcomponent configuration determined by modes
    battery : device Battery.Impl in modes (nominal);
    battery2 : device Battery.Impl in modes (backup);
  modes
    -- mode transitions describe configuration changes
    nominal : initial mode;
    backup : mode;
    nominal –[ battery.discharged ]-> backup;
end Car.Impl;

device Battery
  features
    discharged : event port;
end Battery;

device implementation Battery.Impl
  subcomponents
    charge : data continuous default 100.0;
  states
    -- states describe behaviour
    discharge : initial state while charge’ := -1 and charge >= 0;
    empty : state;
  transitions
    discharge –[ discharged when charge == 0 ]-> empty;
end Battery.Impl;
```

Changes in SLIM

Abstract components

- Input enabled
- Provide any possible output
- (Can also be selected as root)

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

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

 device Battery
   features
     output : data port real {Default => "12.8";};
 end Battery;
```

End of Presentation

See also http://compass.informatik.rwth-aachen.de