European Space Agency

Aristotle University of Thessaloniki

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

ThalesAlenia Space

A Thales / Finmeccanica Company

Catalogue of System and Software Properties (CSSP)

Final Presentation of ESA Contract No. 4000112344/14/NL/FE

ESA/ESTEC, December 7, 2016

Panagiotis Katsaros
Aristotle Un. of Thessaloniki (GR)

Simon Bliudze
École Polytechnique Fédérale de Lausanne (CH)

Main objective of the CSSP

- A model-based requirements specification approach in the System & Software Development Lifecycle.
- Early discovery and resolution of design correctness and consistency issues
  - verification of design models against formal properties derived from the requirements
- Ultimate Aim:
  - to reduce the high cost of corrective measures applied in the late phases of the lifecycle.

Requirements, properties & design models

- System and software requirements
  - conditions or capabilities in natural language to be met by the system or a software component under design
- Design model
  - abstract representation of the "physical" system in a modelling language with formal semantics
- Each requirement can be formally captured by properties:
  - specifications for entities and events of a design model that constrain the structure and the behaviour of the system; ensure that the corresponding requirements are properly covered

Example (natural language requirements)

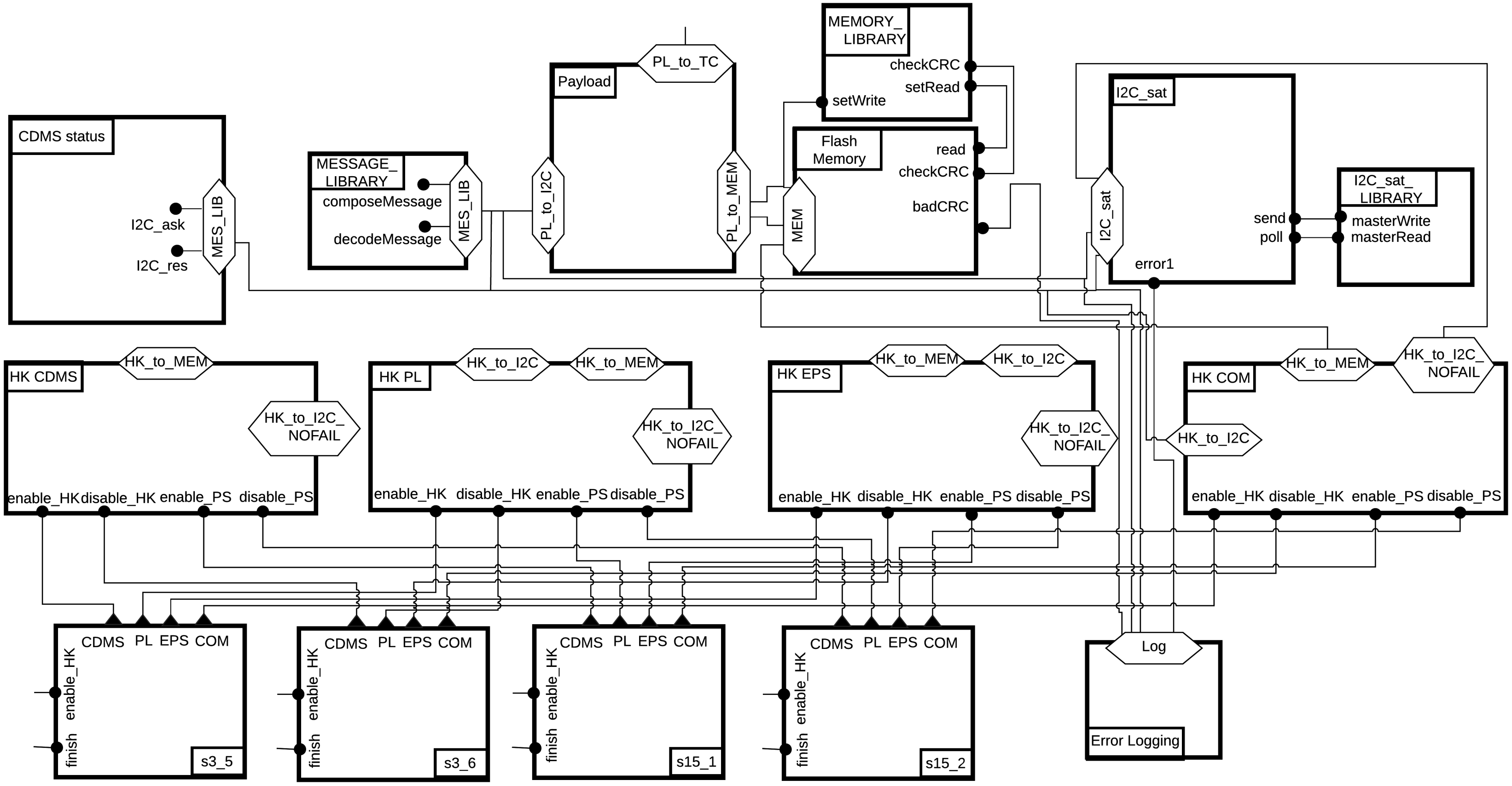
- From the CubETH satellite On-Board software:

  A. Mavridou, E. Stachtiari, S. Bliudze, A. Ivanov, P. Katsaros, J. Sifakis. *Architecture-based Design: A Satellite On-board Software Case Study*. 13th Int. Conf. on Formal Aspects of Component Software (FACS 2016), Besançon, France, 2016

  - The CDMS shall have a Housekeeping activity dedicated to each subsystem (HK-001).
  - When line-of-sight communication is possible, housekeeping information shall be transmitted through the COM subsystem (HK-003).
  - When line-of-sight communication is not possible, housekeeping information shall be written to the non-volatile flash memory (HK-004).
  - A Housekeeping subsystem shall have the following states: NOMINAL, ANOMALY, and CRITICAL_FAILURE (HK-005).
  - If a process failure occurs or if the engineering data are not correct, the subsystem shall enter the ANOMALY state (HK-007).
  - After MAX seconds in ANOMALY, the subsystem shall enter the CRITICAL_ FAILURE state (HK-008).

Example (contd): design model in BIP

- Design model for the CubETH satellite On-Board software (only the interactions are shown):



Example (contd): requirements & properties

- Requirements & formal properties in Computational Tree temporal Logic (CTL) from the CubETH satellite On-Board software:
  - When line-of-sight communication is possible, housekeeping information shall be transmitted through the COM subsystem (HK-003).

    `AG (HKPL_ask_I2C_TTC → HKPL_PSModeMngment_inState(TTC))`
  - When line-of-sight communication is not possible, housekeeping information shall be written to the non-volatile flash memory (HK-004).

    `AG (HKPL_mem_write_req → HKPL_PSModeMngment_inState(MEMORY))`

Whether line-of-sight communication is (not) possible depends on the satellite's visibility status from the ground.

- TTC mode: line of sight communication is possible
- MEMORY mode: line of sight communication is not possible

Verifiability & abstraction levels

- Design correctness:
  - enforcement of formal properties by construction by applying known solutions to the design model (architectural patterns)
  - a posteriori formal verification of the design model for properties that cannot be enforced
- For the properties to be verifiable, they can refer only to model elements representing valid entities for the current development phase.
- Different abstraction levels of design along the development lifecycle (e.g., system, avionics, software).
  - Properties at a particular level will have to be consistent.
  - Established properties must be preserved at lower abstraction levels.

Catalogue of System & Software Properties (CSSP)

Catalogue of Requirement
Categories
per abstraction level

Catalogue of System & Software Properties (CSSP)

Catalogue of Requirement Categories

per abstraction level

categories comprise

Boilerplates:
requirement patterns with placeholders for concepts

	<event>		<system>		<action>		<quantity>	<unit>
if	robot moves into an occupied zone		robot control	shall	stop robot	within	10	ms

Catalogue of System & Software Properties (CSSP)

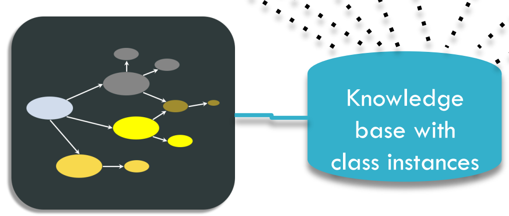
Catalogue of Requirement Categories
per abstraction level

categories comprise

Boilerplates:
requirement patterns with placeholders for concepts

	<event>	<system>		<action>		<quantity>	<unit>
if	robot moves into an occupied zone	robot control	shall	stop robot	within	10	ms

concepts mapped to



CSSP Ontology: explicit specification of a shared conceptual model of the domain

Catalogue of System & Software Properties (CSSP)

Catalogue of Requirement Categories
per abstraction level

categories comprise

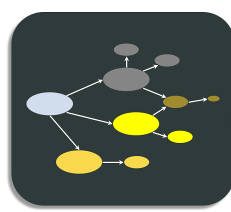
Boilerplates:
requirement patterns with placeholders for concepts

<event> <system> <action> <quantity> <unit>

if robot moves into an occupied zone | robot control | shall | stop robot | within | 10 | ms

Requirements

concepts mapped to



Knowledge base with class instances

CSSP Ontology: explicit specification of a shared conceptual model of the domain

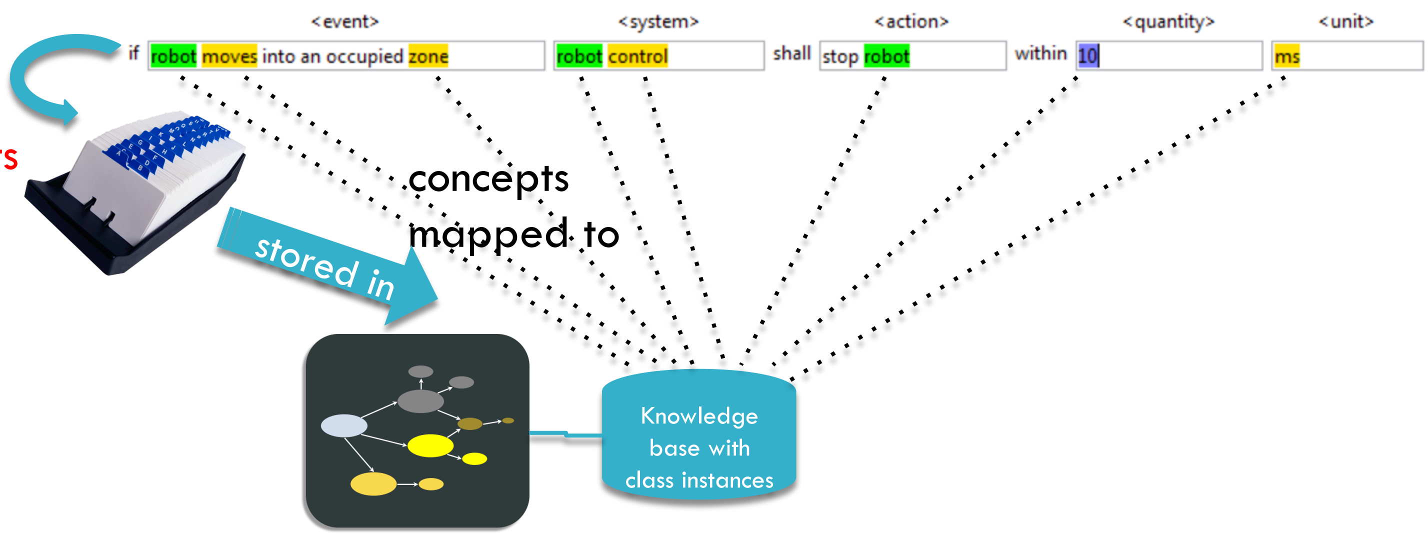
Catalogue of System & Software Properties (CSSP)

Catalogue of Requirement Categories
per abstraction level

categories comprise

Boilerplates:
requirement patterns with placeholders for concepts

Requirements



CSSP Ontology: explicit specification of a shared conceptual model of the domain

Catalogue of System & Software Properties (CSSP)

Catalogue of Requirement Categories
per abstraction level

categories comprise

Boilerplates:
requirement patterns with placeholders for concepts

<event> <system> <action> <quantity> <unit>

if robot moves into an occupied zone | robot control | shall | stop robot | within | 10 | ms

Requirements

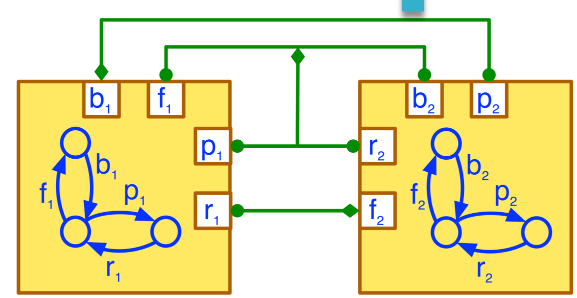
stored in

concepts mapped to

Knowledge base with class instances

elementary components found by querying

CSSP Ontology: explicit specification of a shared conceptual model of the domain



BIP design model

Catalogue of System & Software Properties (CSSP)

Catalogue of Requirement Categories per abstraction level

categories comprise

Boilerplates: requirement patterns with placeholders for concepts

	<event>		<system>		<action>		<quantity>	<unit>
if	robot moves into an occupied zone		robot control	shall	stop robot	within	10	ms

Requirements

stored in

concepts mapped to

Property patterns with implicit CTL representation

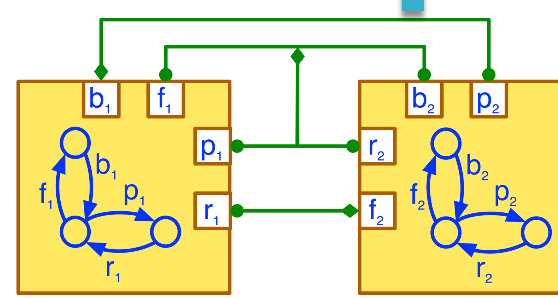
```
whenever request occurs
response occurs
during [0ms,10ms]
```

entity states & events

Knowledge base with class instances

elementary components found by querying

CSSP Ontology: explicit specification of a shared conceptual model of the domain



BIP **design model**

Catalogue of System & Software Properties (CSSP)

Catalogue of Requirement Categories per abstraction level

categories comprise

Boilerplates: requirement patterns with placeholders for concepts

<event>		<system>		<action>		<quantity>	<unit>
if	robot moves into an occupied zone	robot control	shall	stop robot	within	10	ms

Requirements

stored in

concepts mapped to

elementary components found by querying

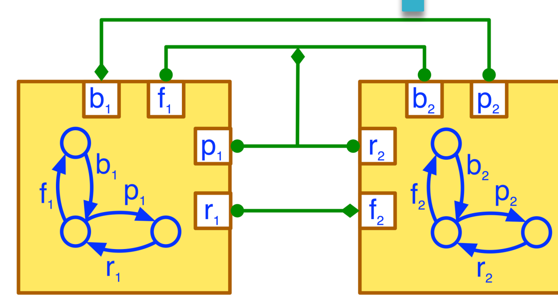
Property patterns with implicit CTL representation

```
whenever request occurs
response occurs
during [0ms,10ms]
```

entity states & events

Knowledge base with class instances

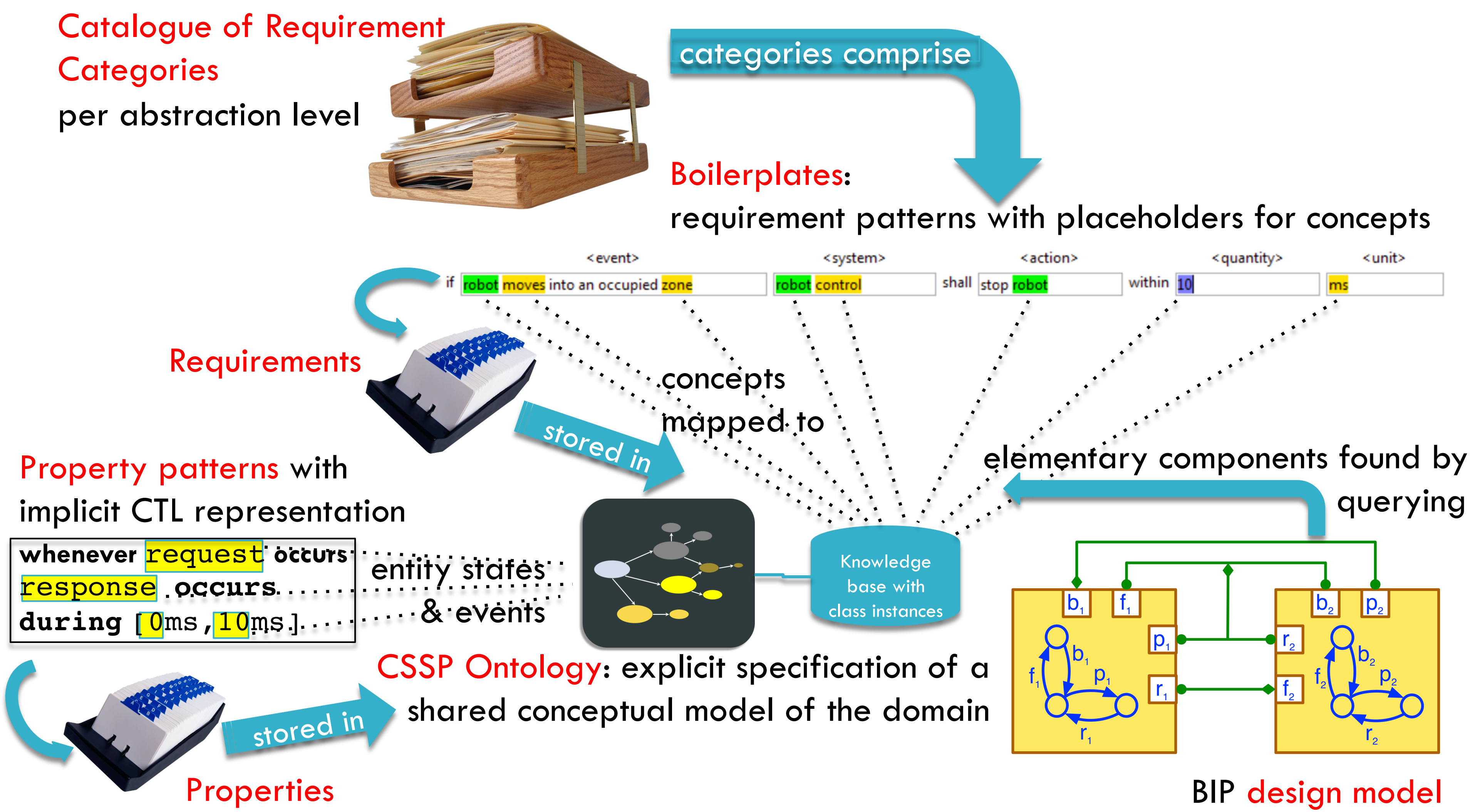
CSSP Ontology: explicit specification of a shared conceptual model of the domain



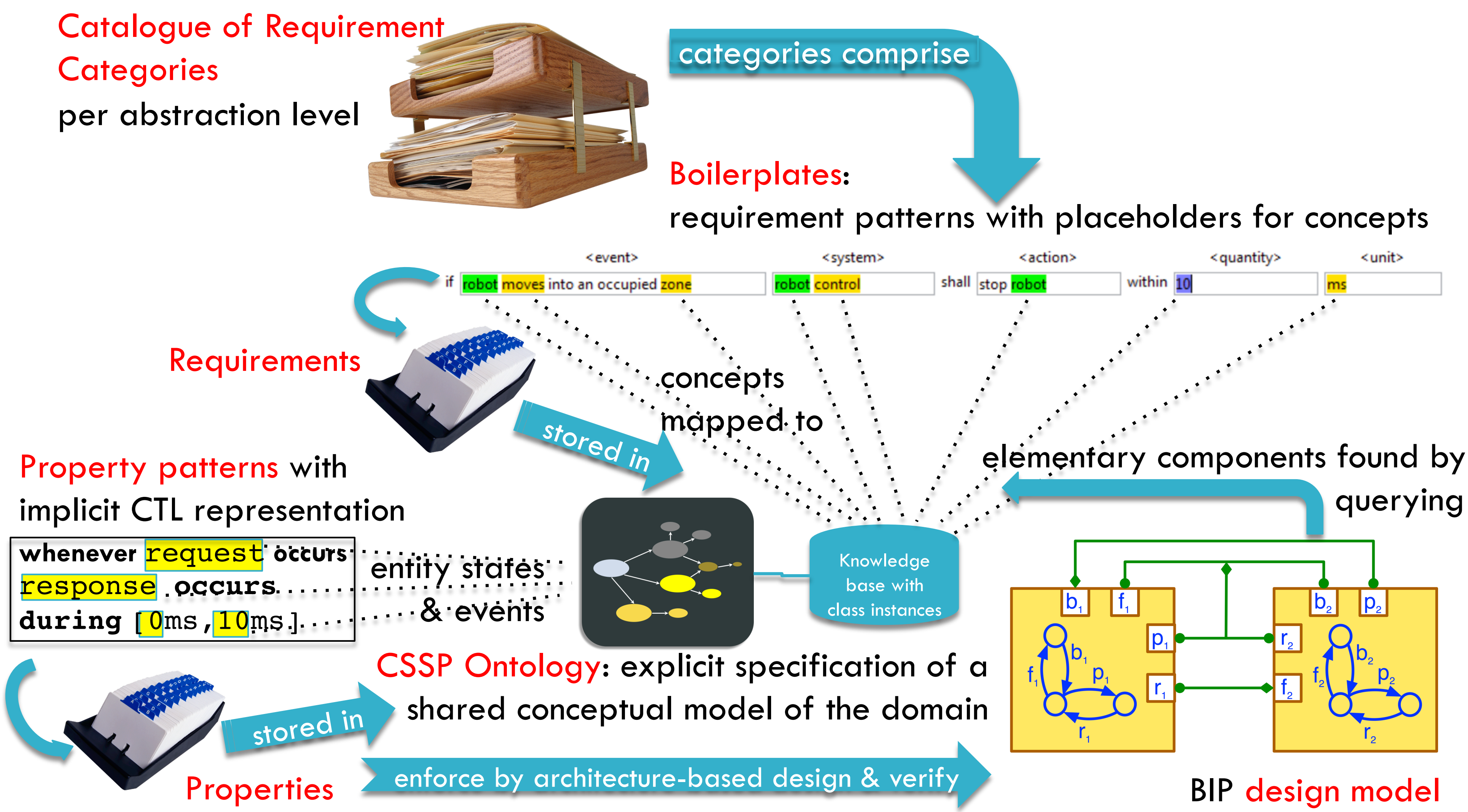
BIP design model

Properties

Catalogue of System & Software Properties (CSSP)



Catalogue of System & Software Properties (CSSP)



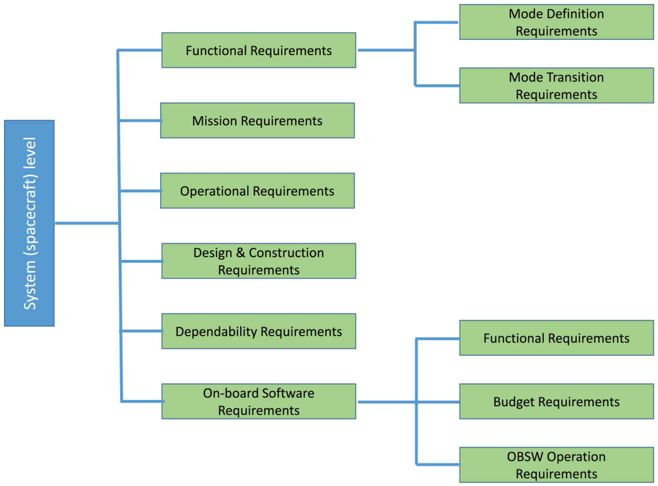
Catalogue of Requirement Categories

- Guides the engineers throughout the requirements specification per abstraction level.

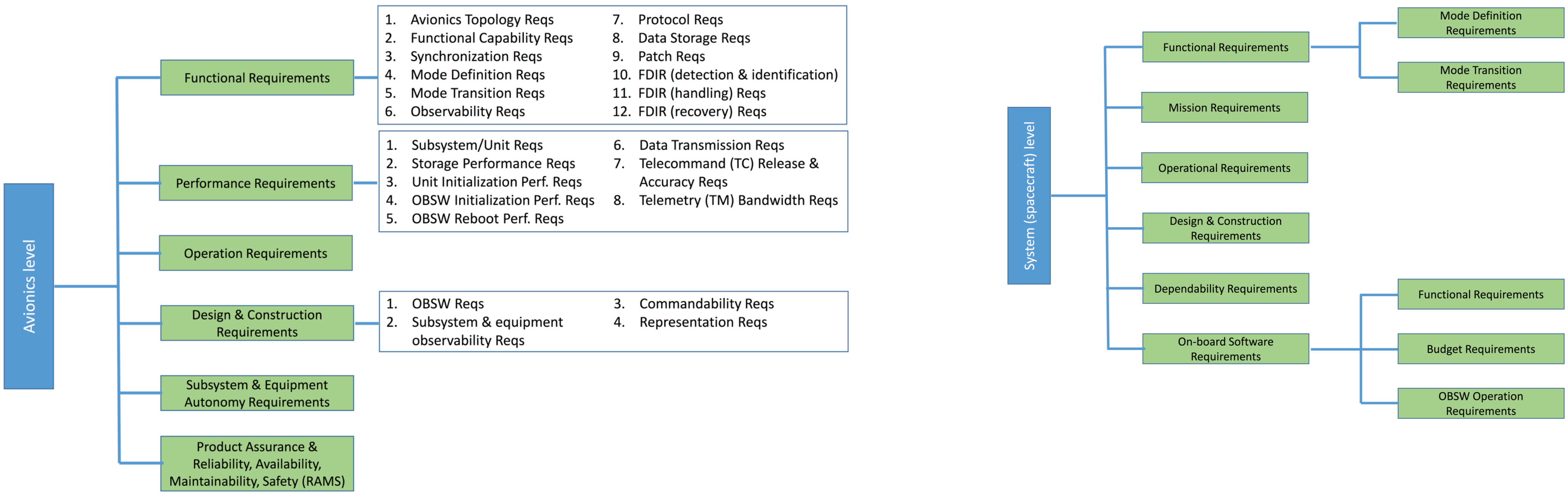
- Proposed by Thales Alenia Space France based on:
  - the ECSS standard on Technical Requirements Specification
  - empirical evidence from reference satellite projects
    - Sentinel 3, which is a LEO Earth Observation satellite
    - Exomars-TGO, a planetary orbiter with specificities regarding autonomous behaviour and fail-operational modes

- Partial classification: only requirements that plausibly influence at lower level those that are relevant to the OBSW.

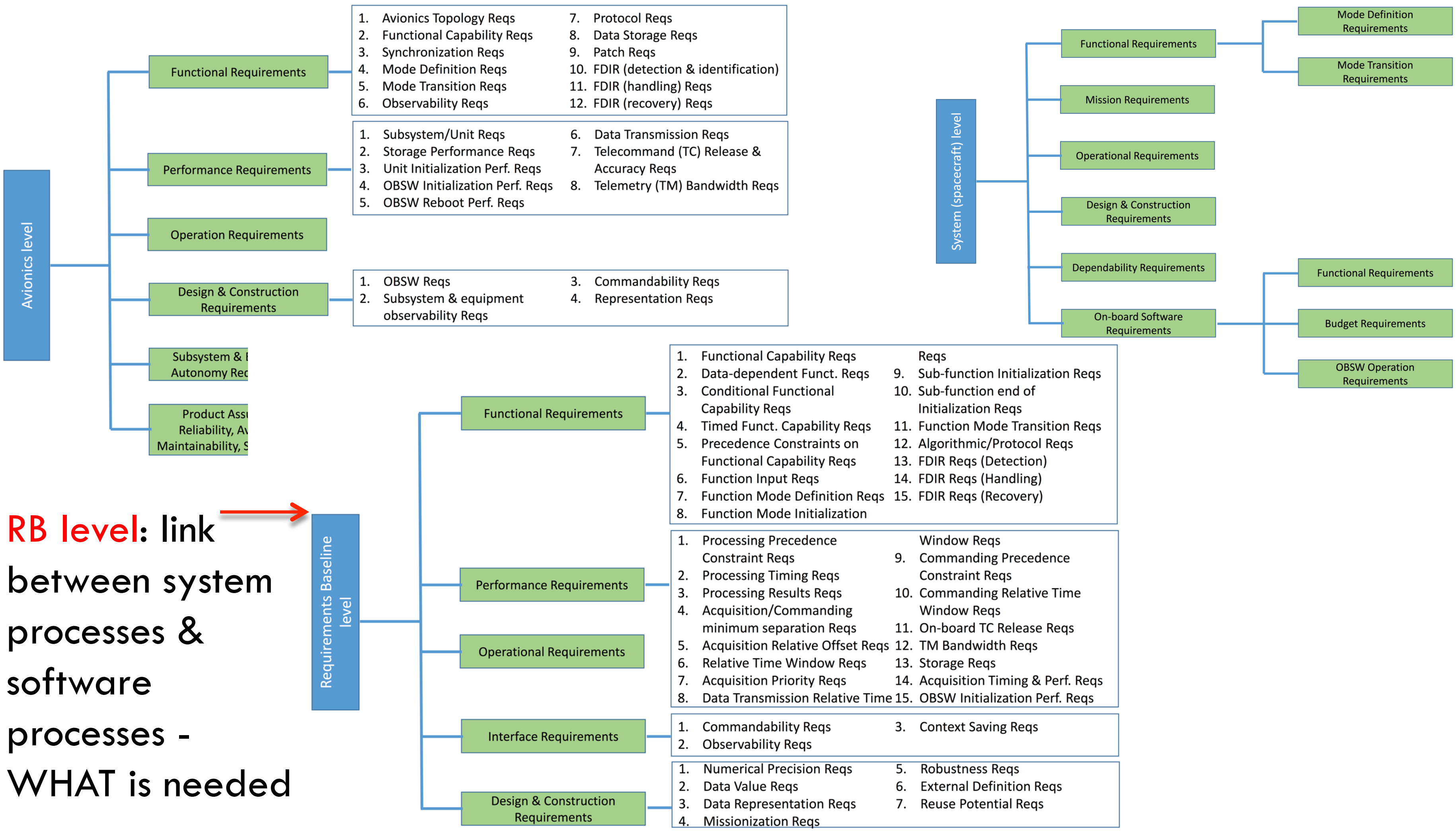
Categories at four abstraction levels



Categories at four abstraction levels

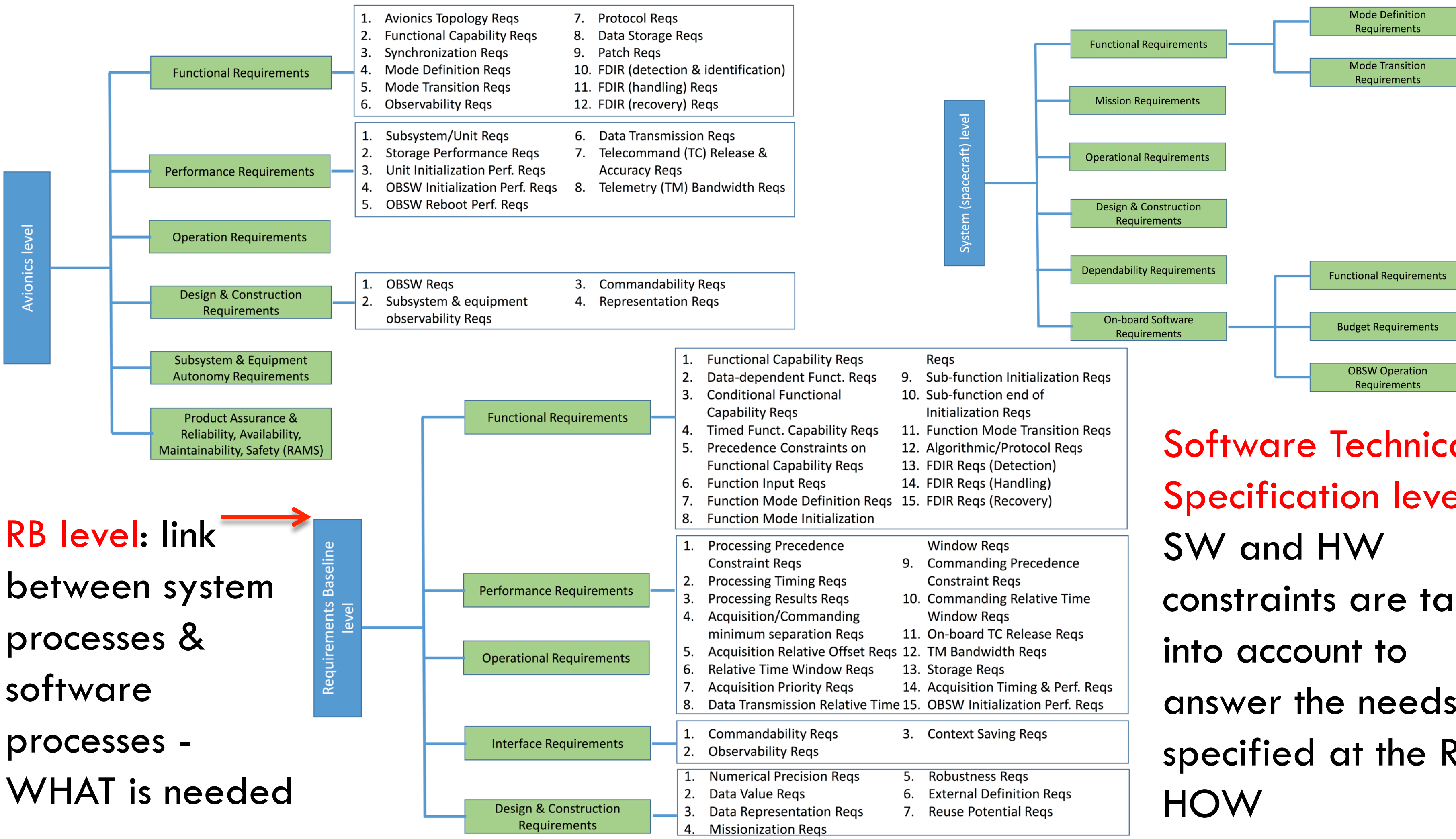


Categories at four abstraction levels



RB level: link between system processes & software processes - WHAT is needed

Categories at four abstraction levels



RB level: link between system processes & software processes - WHAT is needed

Software Technical Specification level: SW and HW constraints are taken into account to answer the needs specified at the RB – HOW

Boilerplates language I

- Boilerplates are requirement patterns with placeholders.
  - instantiated into requirements by replacing placeholders with entities appropriate for the particular system context of the mission under design.

- They are used in order to:
  - eliminate the fuzzy syntax of natural language specifications
  - assign concrete meaning to language constructs in order to avoid diverse interpretations

    (e.g. connective words used to determine time, order/ sequence, consequence, comparison, contrast and various types of conjunctions)

Boilerplates language II

$$\langle \mathit{boilerplate} \rangle ::= \langle \mathit{prefix} \rangle\ \langle \mathit{main} \rangle\ \langle \mathit{suffix} \rangle$$

Boilerplates language II

$\langle boilerplate \rangle ::= \langle prefix \rangle \langle main \rangle \langle suffix \rangle$

ID	Template
P1	If/unless $\langle event \rangle$,...
P2	If/unless/while $\langle state \rangle$,...
P3	If/unless/while $\langle action \rangle$,...

Boilerplates language II

ID	Template
M1	⟨*subject*⟩ shall [*not*] ⟨*action*⟩
M2	⟨*subject*⟩ shall [*not*] be ⟨*state*⟩
M3	⟨*subject*⟩ shall [*not*] *allow* ⟨*entity*⟩ [of ⟨*quantity*⟩] [to be ⟨*state*⟩]
M4	⟨*subject*⟩ shall [not] *allow* [**number**] ⟨*entity*⟩ [to be ⟨*state*⟩]
M5	⟨*subject*⟩ shall [not] have [**number**] ⟨*entity*⟩
M6	⟨*subject*⟩ shall [*not*] be able to ⟨action⟩
M7	⟨*subject*⟩ shall [*not*] support ⟨action⟩
M8	⟨*subject*⟩ shall [*not*] execute ⟨*action-block*⟩
M9	⟨*subject*⟩ shall [*not*] handle ⟨entity⟩
M10	⟨*subject*⟩ shall [*not*] make available ⟨*entity*⟩
M11	⟨*entity*⟩ shall [*not*] be defined in ⟨*entity*⟩
M12	⟨*entity*⟩ shall [*not*] allow ⟨*entity*⟩ to ⟨action⟩

⟨*boilerplate*⟩ ::= ⟨*prefix*⟩ ⟨*main*⟩ ⟨*suffix*⟩

ID	Template
P1	If/unless ⟨*event*⟩,...
P2	If/unless/while ⟨*state*⟩,...
P3	If/unless/while ⟨*action*⟩,...

Boilerplates language II

⟨*boilerplate*⟩ ::= ⟨*prefix*⟩ ⟨*main*⟩ ⟨*suffix*⟩

ID	Template
M1	⟨*subject*⟩ shall [*not*] ⟨*action*⟩
M2	⟨*subject*⟩ shall [*not*] be ⟨*state*⟩
M3	⟨*subject*⟩ shall [*not*] *allow* ⟨*entity*⟩ [of ⟨*quantity*⟩] [to be ⟨*state*⟩]
M4	⟨*subject*⟩ shall [not] *allow* [**number**] ⟨*entity*⟩ [to be ⟨*state*⟩]
M5	⟨*subject*⟩ shall [not] have [**number**] ⟨*entity*⟩
M6	⟨*subject*⟩ shall [*not*] be able to ⟨action⟩
M7	⟨*subject*⟩ shall [*not*] support ⟨action⟩
M8	⟨*subject*⟩ shall [*not*] execute ⟨*action-block*⟩
M9	⟨*subject*⟩ shall [*not*] handle ⟨entity⟩
M10	⟨*subject*⟩ shall [*not*] make available ⟨*entity*⟩
M11	⟨*entity*⟩ shall [*not*] be defined in ⟨*entity*⟩
M12	⟨*entity*⟩ shall [*not*] allow ⟨*entity*⟩ to ⟨action⟩

ID	Template
P1	If/unless ⟨*event*⟩,...
P2	If/unless/while ⟨*state*⟩,...
P3	If/unless/while ⟨*action*⟩,...

ID	Template
S1	[⟨*quantifier*⟩] **number unit** [per **time-unit**]
S2	after/before ⟨*event*⟩
S3	[*quantifier*] [every/for a period of/within/for at least] **number time-unit** [from ⟨*event*⟩]
S4	without [affecting] (⟨*action*⟩ ∣ ⟨*entity*⟩)
S5	other than (⟨*action*⟩ ∣ ⟨*entity*⟩)
S6	in the order (: ⟨*entity-list*⟩ ∣ ⟨*entity*⟩)
S7	at even intervals
S8	using ⟨*entity*⟩

Requirement categories & boilerplates

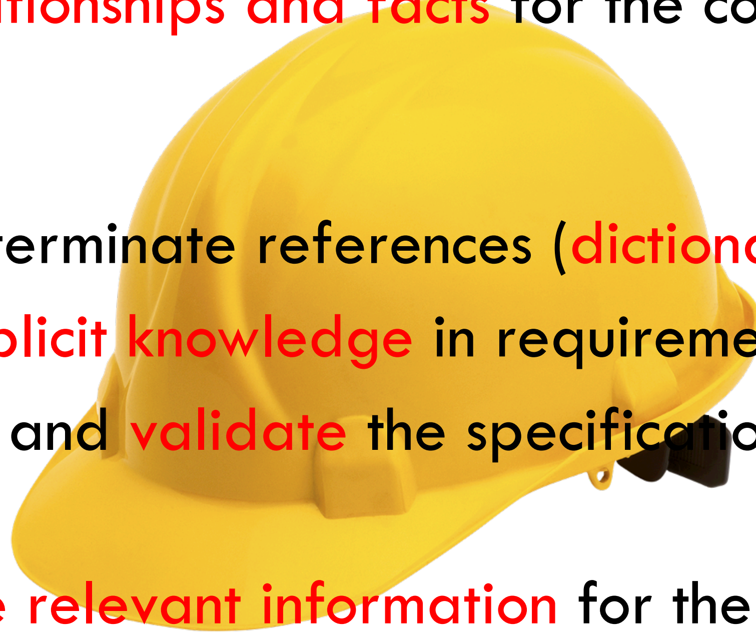
- Empirical knowledge from the RB-level of Sentinel 3

Abstraction Level	Category of Requirements	Used Boilerplates
RB	Functional > Functional Capability Reqs	main: M9 main: M1
RB	Functional > Conditional Functional Capability Reqs	prefix: P1, main: M1 prefix: P2, main: M1 prefix: P2, main: M1, suffix: S5 prefix: P2, main: M1, suffix: S11
RB	Functional > Timed Functional Capability Reqs	prefix: P1, main: M1
RB	Functional > Precedence Constraints on Functional Capability Reqs	main: M1, suffix: S10 main: M1, suffix: S6
RB	Functional > Function Mode Definition Reqs	main: M4
RB	Functional > Function Mode Initialization Reqs	prefix: P1, main: M1
RB	Functional > Function Mode Transition Reqs	prefix: P2 and P1, main: M1 prefix: P1, main: M1
RB	Functional > Algorithmic/Protocol Reqs	main: M1, suffix: S8 and S2 main: M1, suffix: S8 prefix: P2, main: M1, suffix: S8 prefix: P1, main: M1, suffix: S9
RB	Functional > FDIR (Detection) Reqs	prefix: P1, main: M1 prefix: P1 and P2, main: M1
RB	Functional > FDIR (Handling) Reqs	prefix: P1, main: M1 prefix: P2, main: M1 prefix: P1 and P2, main: M1
RB	Performance Reqs	main: M6, suffix: S1
RB	Interface > Commandability Reqs	main: M4, suffix: S8 main: M10, suffix: S8 and S8 and S3

CSSP Ontology I

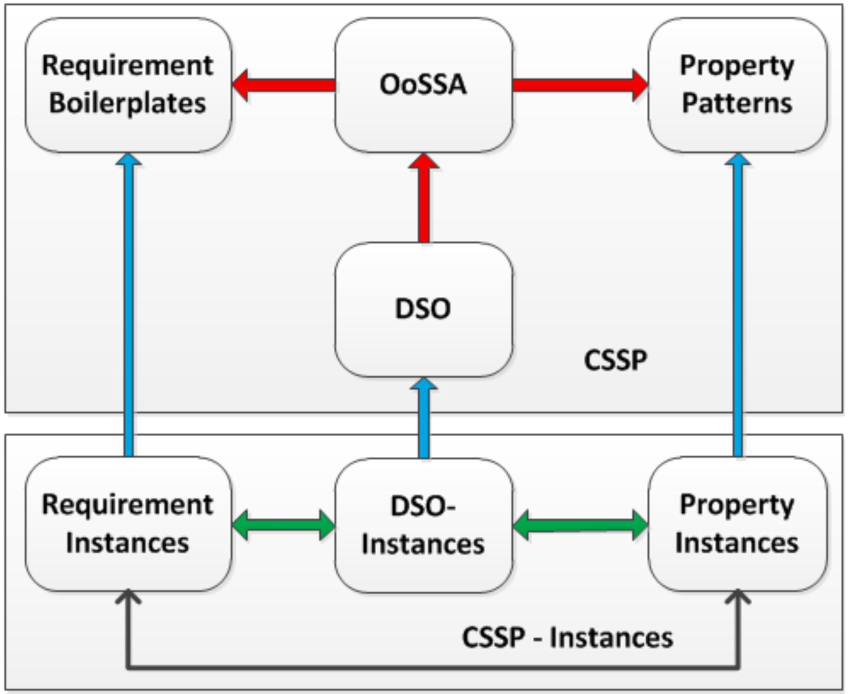
- Encodes the conceptual model of the system's domain and the specification language.
  - logical relationships and facts for the concepts

- It is used to:
  - avoid indeterminate references (dictionary)
  - capture implicit knowledge in requirement specifications
  - search into and validate the specifications by ontology-based reasoning
  - retrieve the relevant information for the subsequent modelling activities



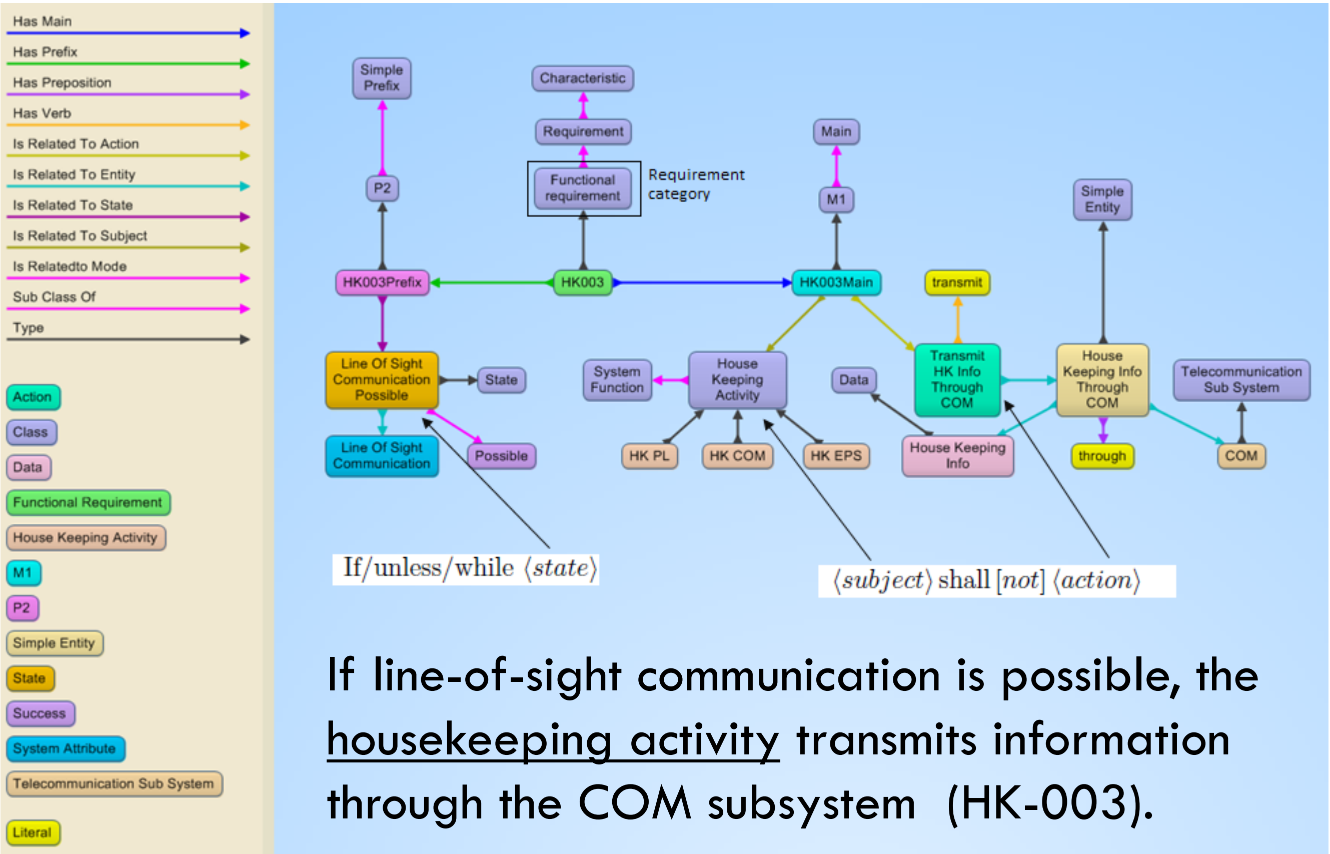
CSSP Ontology II

- Concepts organized into sub-ontologies with well-defined scope: Ontology Engineers know where to apply the needed changes.
  - Ontology of System & Software Attributes (OoSSA)
  - Domain Specific Ontology (DSO)
  - Requirement Boilerplates (RBLP)
  - Property Patterns (PRP)
- Logical (rule-based) reasoning can infer implicit relationships between system/software entities and requirements.



Example (contd): requirement formalization

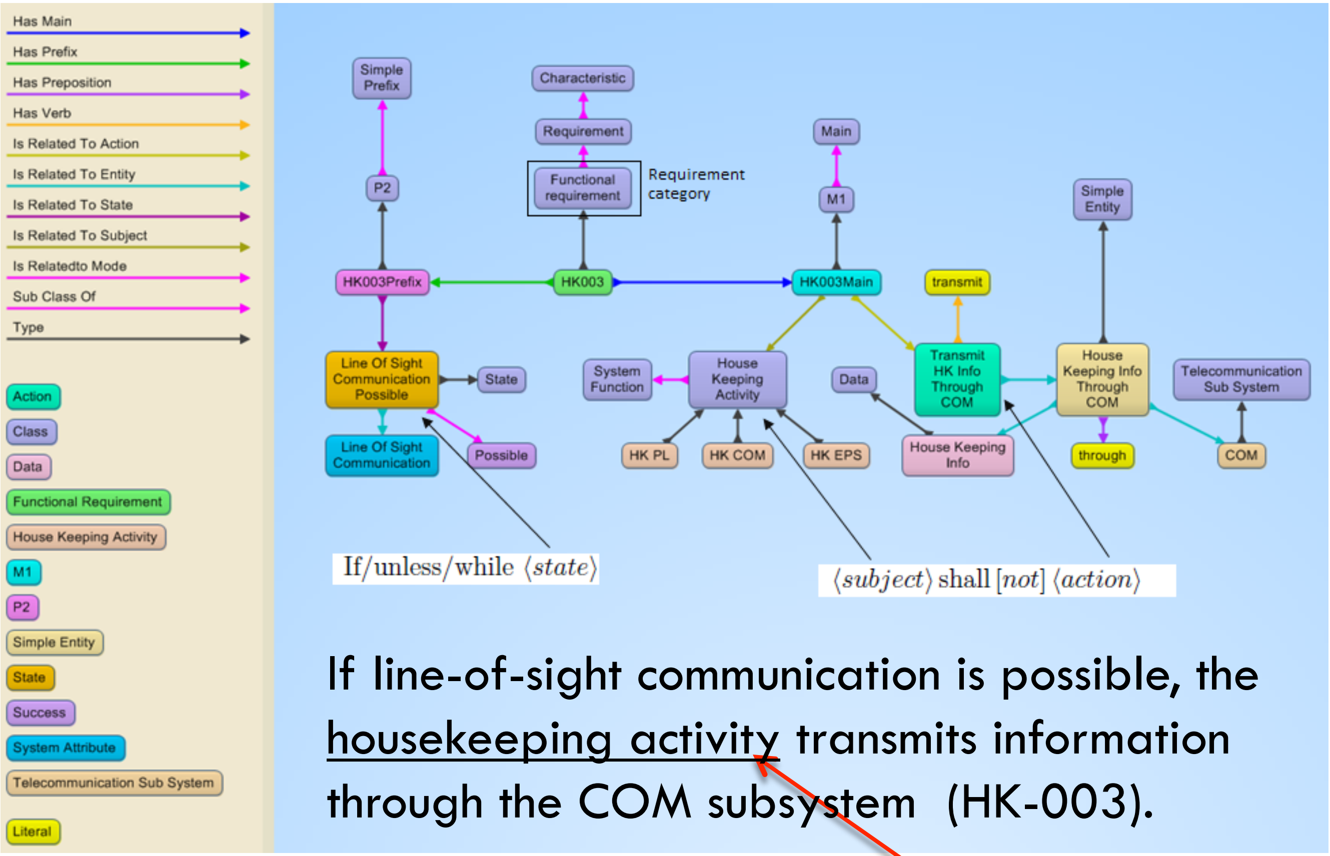
Boilerplate-based representation in the CSSP ontology:



- An abstract requirement refers to an abstract entity (class) and implies that the requirement should be fulfilled for all instances of this abstract entity.

Example (contd): requirement formalization

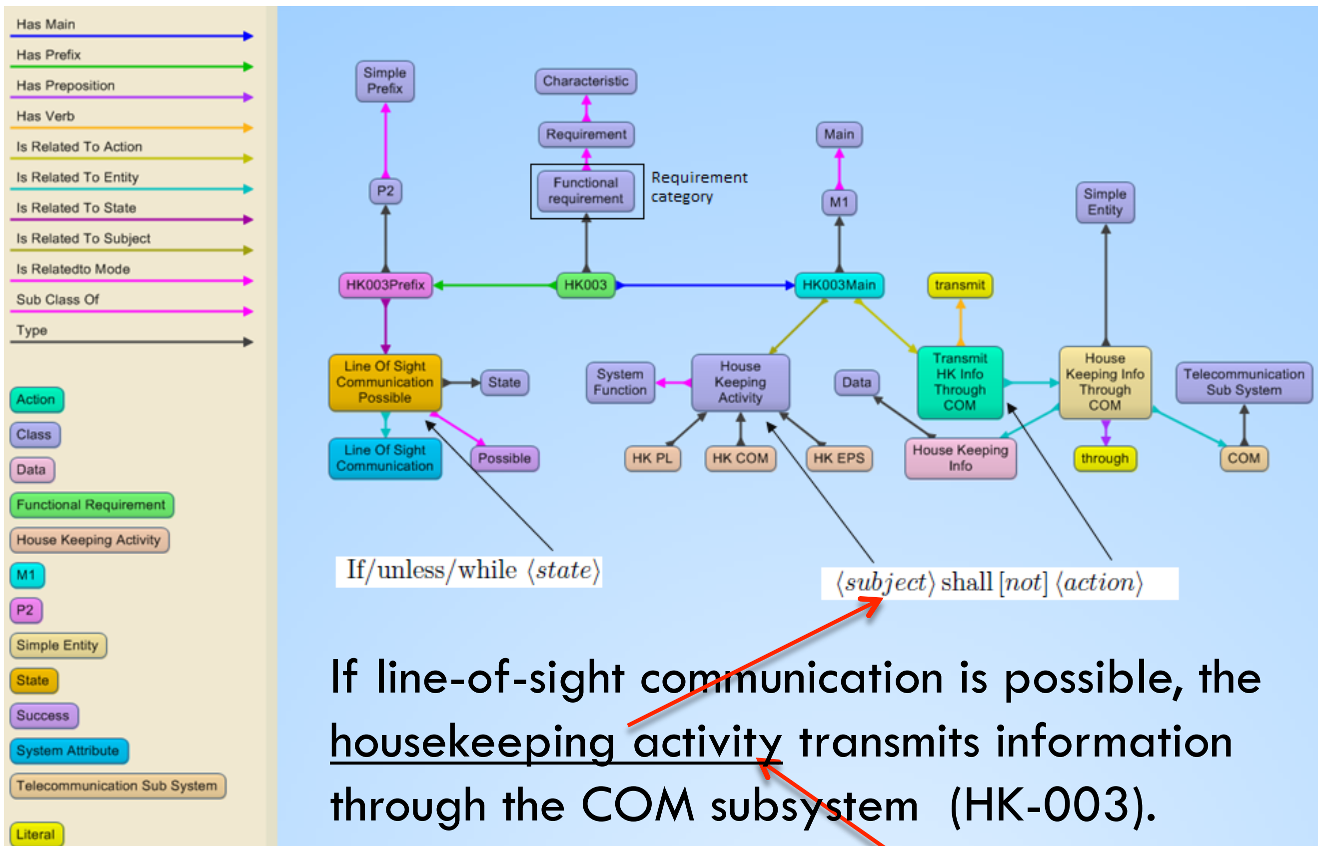
Boilerplate-based representation in the CSSP ontology:



- An abstract requirement refers to an abstract entity (class) and implies that the requirement should be fulfilled for all instances of this abstract entity.

Example (contd): requirement formalization

Boilerplate-based representation in the CSSP ontology:



- An abstract requirement refers to an abstract entity (class) and implies that the requirement should be fulfilled for all instances of this abstract entity.

Ontology-based Validation

- Implemented SPARQL queries:
  - Check that there are no missing concrete requirements.
  - Find entities for which no requirements have been specified.
  - Find entities that do not appear as subject in the specified requirements.
  - Check for inconsistent requirement specifications with contradictory parts.

Property patterns language I

- Property patterns: formalism-independent specification abstractions, an input mechanism to capture properties:
  - in terms of events and state variables of some design model
  - associated with implicit formal representations in a logic language

- Property patterns for properties:
  - that can be enforced by design, i.e. there is available design solution
  - that should be specified in a verifiable form (CTL specification amenable to model checking)

Property patterns language II

- Patterns for property specifications in a verifiable form:

$$(\text{(whenever } \underbrace{\textit{behavior}}_{\textit{context}}\text{, then)} \mid \text{always}) \underbrace{\textit{behavior}}_{\textit{result}}$$

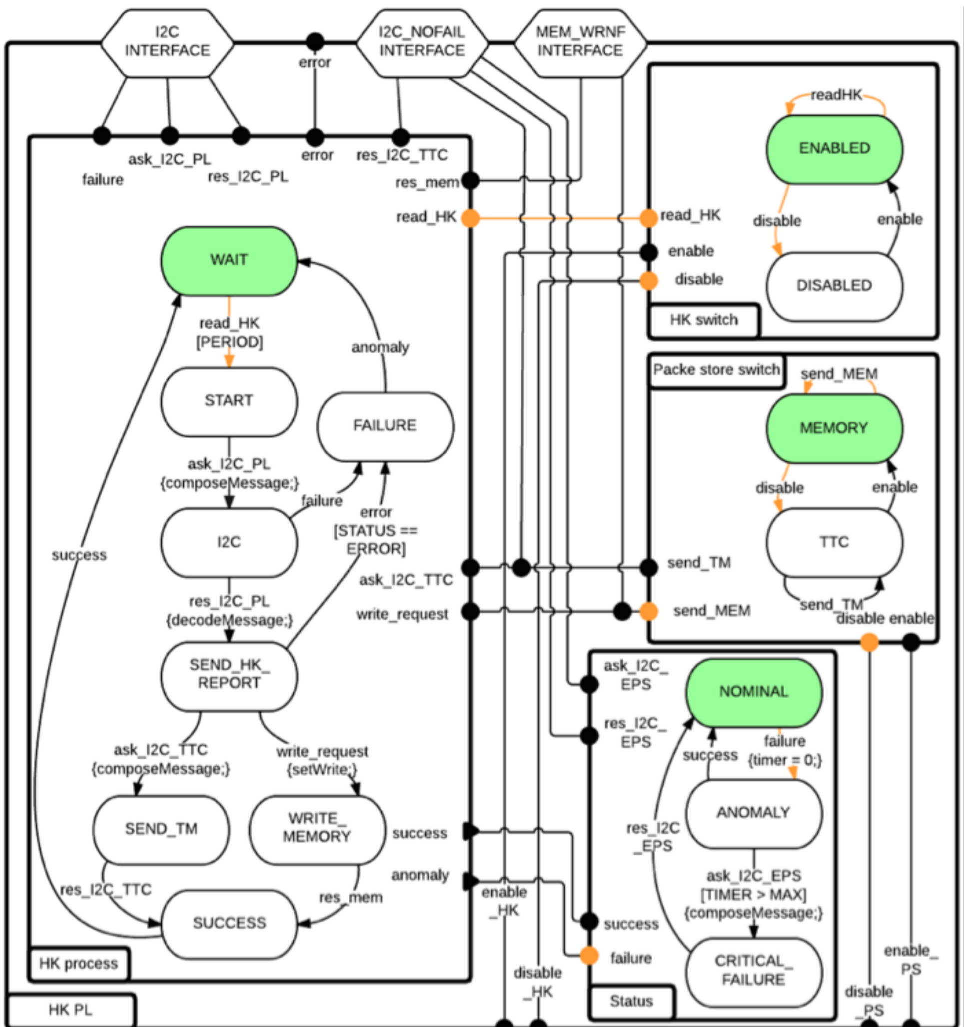
e.g. Whenever telecommand acquisition fails, then a telemetry anomaly report shall be generated.

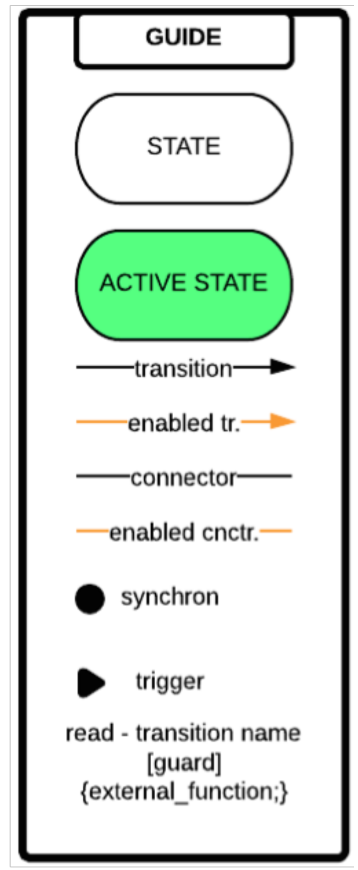
- Patterns for mode management properties (enforced by design):

Pattern MD1:	⟨*component*⟩ has exclusive modes ⟨*mode*⟩+
This pattern is used to enumerates the set of exclusive modes. The set of modes shall be defined so that (a) the system is in one of these modes at an time, otherwise the set is underspecified. (b) not all modes allow the same set of interactions, otherwise the set is overspecified. **Natural Language Requirement**: TC processing can be either Nominal or Restricted. **Property**: The following exclusive modes exist: ⟨*mode* : nominal⟩ , ⟨*mode* : restricted⟩	

Pattern MD4:	Whenever ⟨*port*⟩ occurs, mode is set to ⟨*mode*⟩ [, except if ⟨*proposition*⟩ *holds*.]
This pattern specifies interactions, upon which the mode shall be changed. The change of mode is not effective if an exceptional case holds. **Natural Language Requirement**: At the end of CSW initialization, the TC processing mode shall be set to Nominal. **Property**: Whenever ⟨*interaction* : CSW.initEnds⟩ occurs, mode is set to ⟨*mode* : nominal⟩	

Behaviour-Interaction-Priority (BIP)

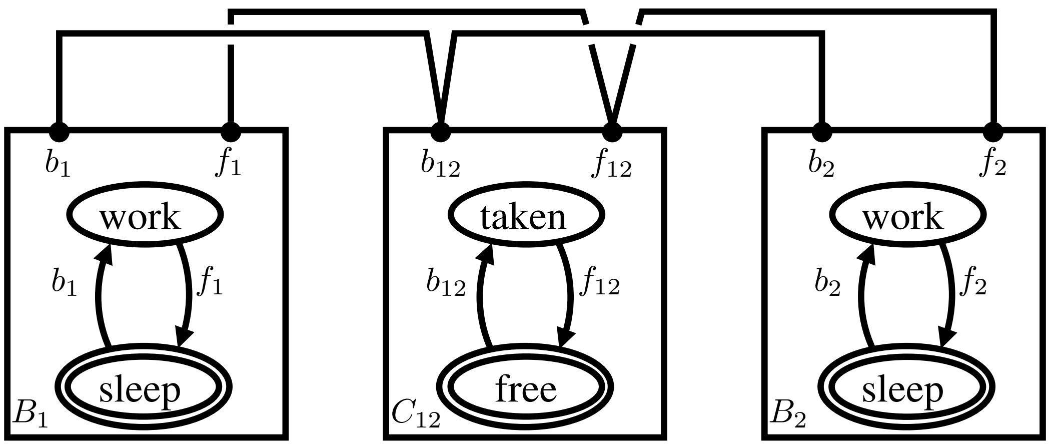




slide courtesy of
Marco Pagnamenta

- Components
  - state machines
- Coordination
  - synchronisation
  - defined by connectors
- Language
- Analysis
- Code generation

Architecture-based design

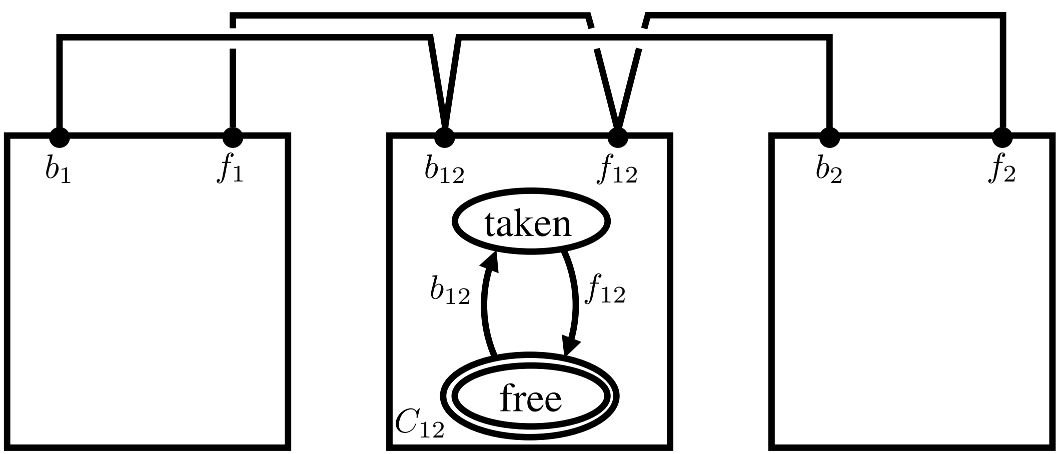


- Enforced property
  - Mutual exclusion

$$\mathtt{AG}\,\neg(cs_1 \wedge cs_2)$$

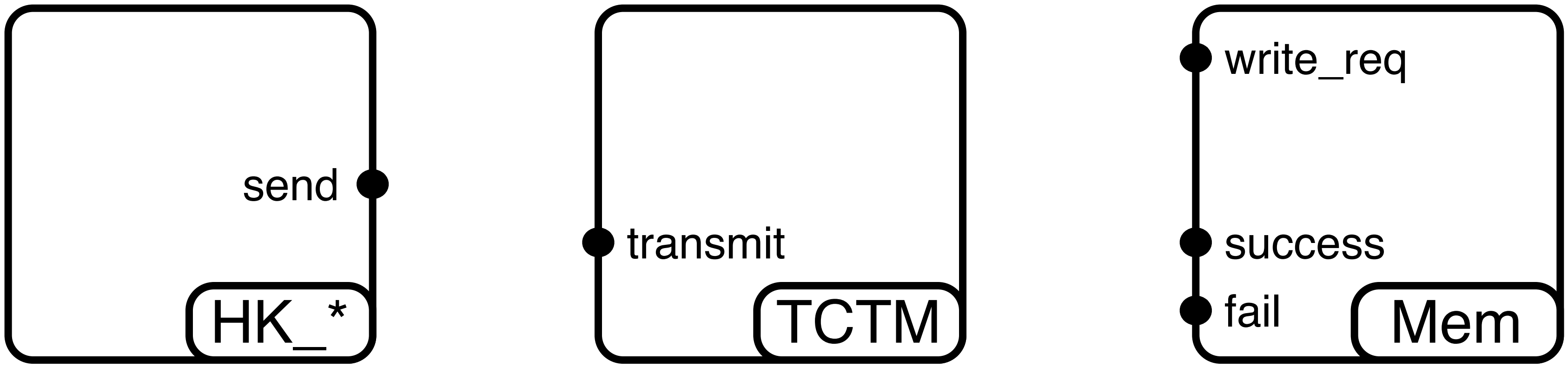
- Assumed property
  - Not in the critical section after *finish*

$$\mathtt{AG}\big(f_i \to \mathtt{A}[\neg cs_i \;\mathtt{W}\; b_i]\big)$$



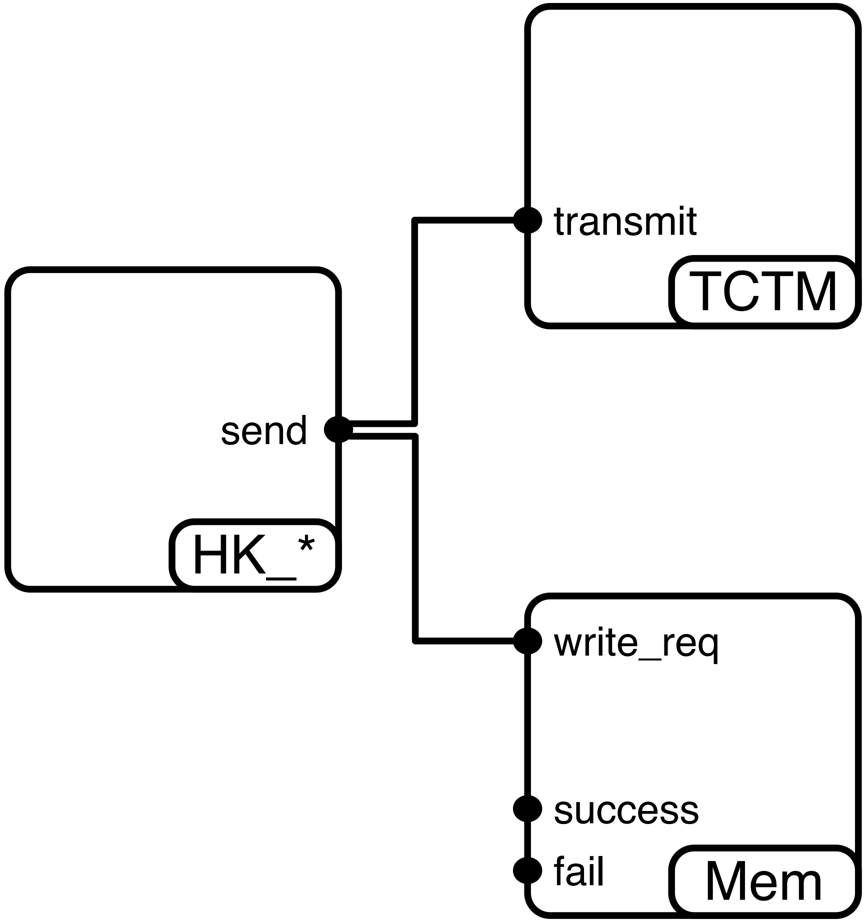
Example (contd): Building blocks

- From the CubETH satellite On-Board software:
  - The CDMS shall have a Housekeeping activity dedicated to each subsystem (HK-001).
  - When line-of-sight communication is possible, housekeeping information shall be transmitted through the COM subsystem (HK-003).
  - When line-of-sight communication is not possible, housekeeping information shall be written to the non-volatile flash memory (HK-004).



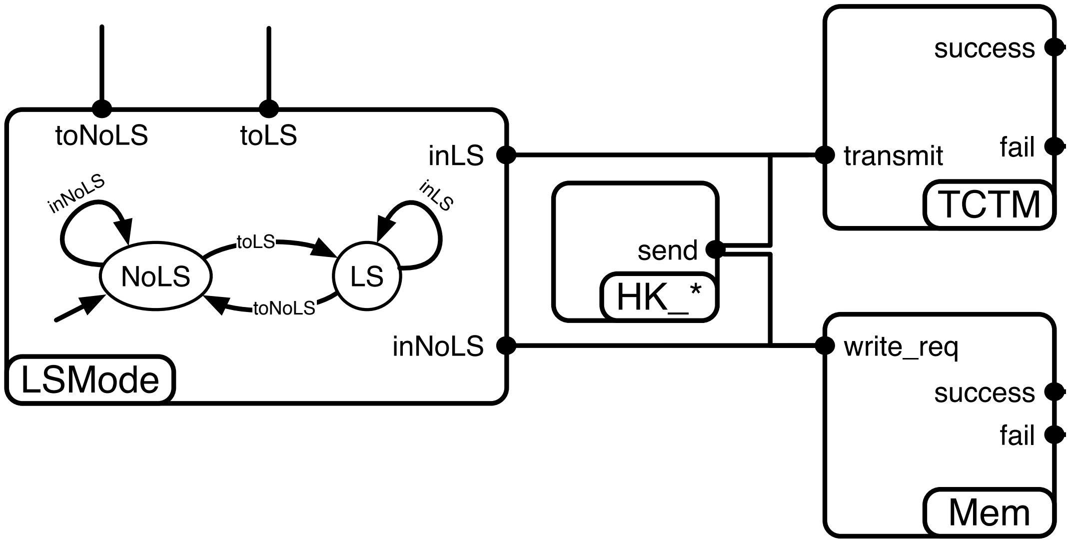
Example (contd): Client-server

- From the CubETH satellite On-Board software:
  - When line-of-sight communication is possible, housekeeping information shall be transmitted through the COM subsystem (HK-003).
  - When line-of-sight communication is not possible, housekeeping information shall be written to the non-volatile flash memory (HK-004).



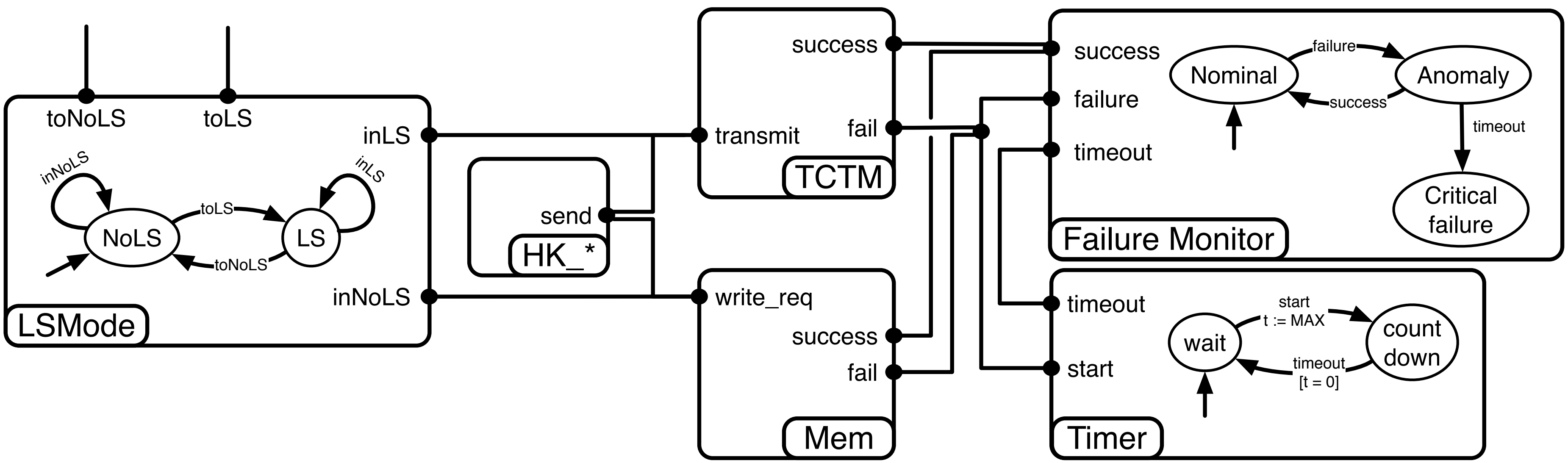
Example (contd): Mode management

- From the CubETH satellite On-Board software:
  - When line-of-sight communication is possible, housekeeping information shall be transmitted through the COM subsystem (HK-003).
  - When line-of-sight communication is not possible, housekeeping information shall be written to the non-volatile flash memory (HK-004).



Example (contd): Failure management

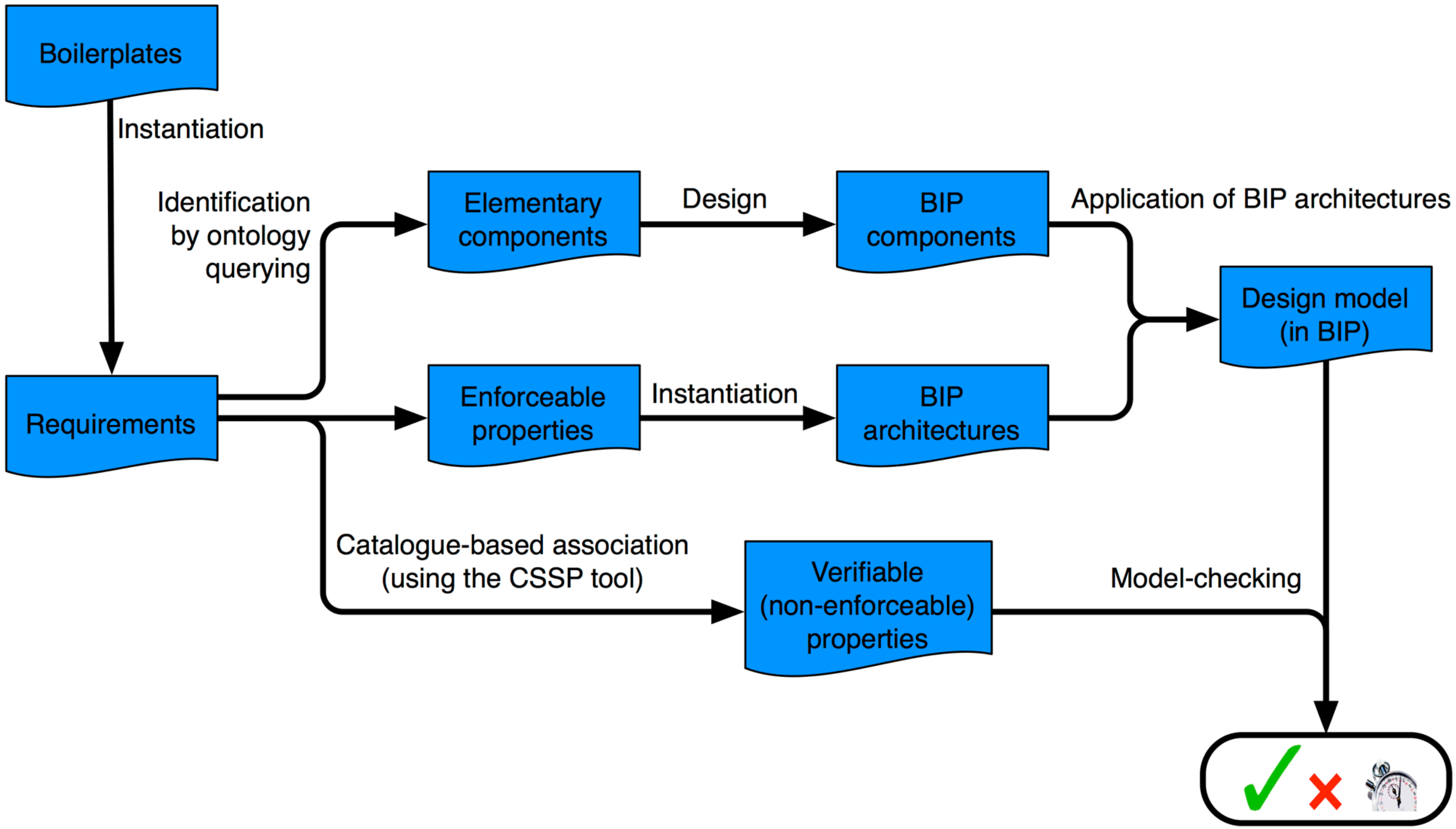
- From the CubETH satellite On-Board software:
  - A Housekeeping subsystem shall have the following states: NOMINAL, ANOMALY, and CRITICAL_FAILURE (HK-005).
  - If a process failure occurs or if the engineering data are not correct, the subsystem shall enter the ANOMALY state (HK-007).
  - After MAX seconds in ANOMALY, the subsystem shall enter the CRITICAL_ FAILURE state (HK-008).



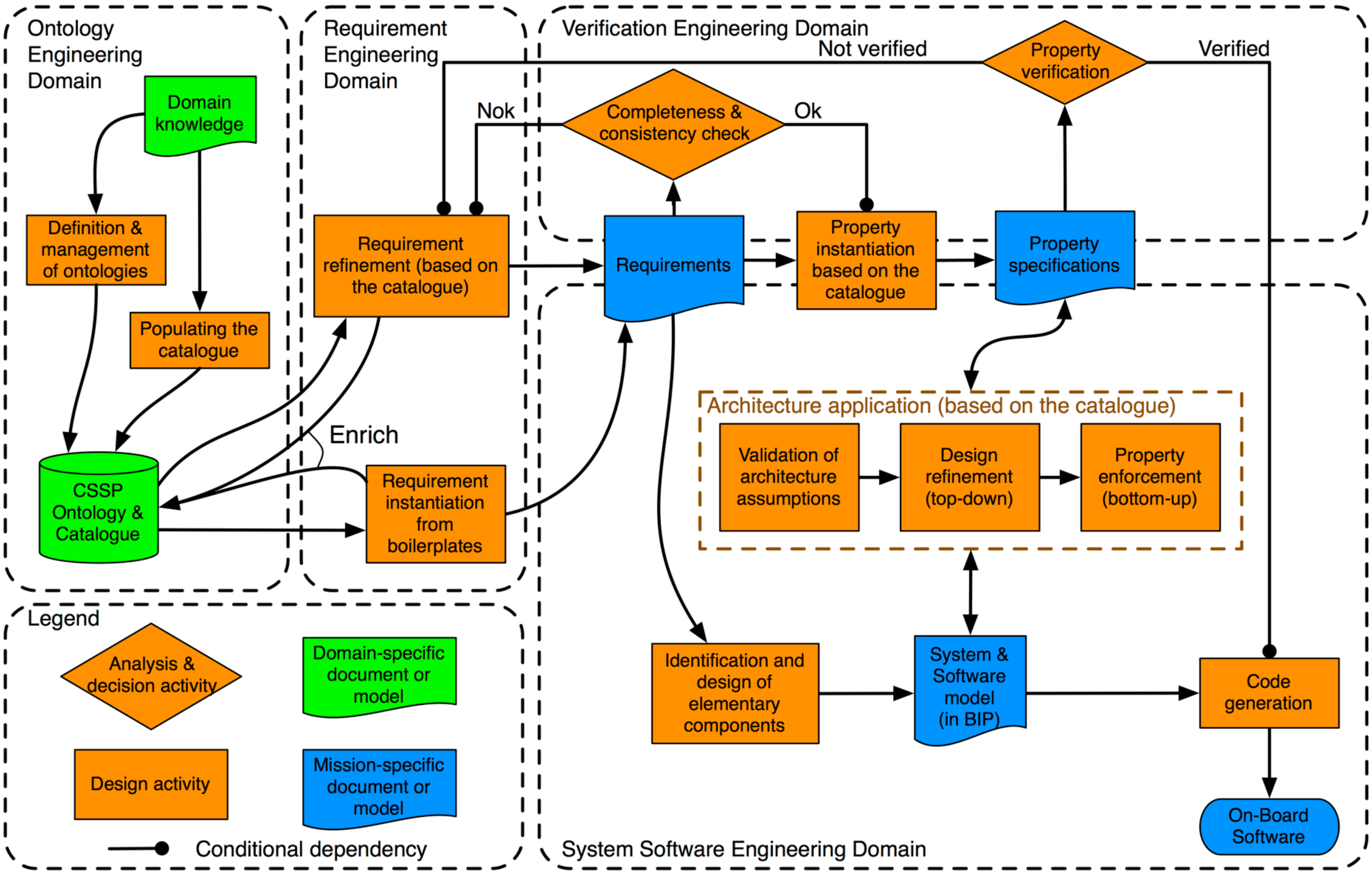
Taxonomy of architecture styles

- 9 architectures identified through case studies
  - Mutual exclusion
  - Client-server
  - Action flow
  - Action flow with abort
  - Failure monitoring
  - Mode management
  - Buffer management
  - Event monitoring
  - Priority management

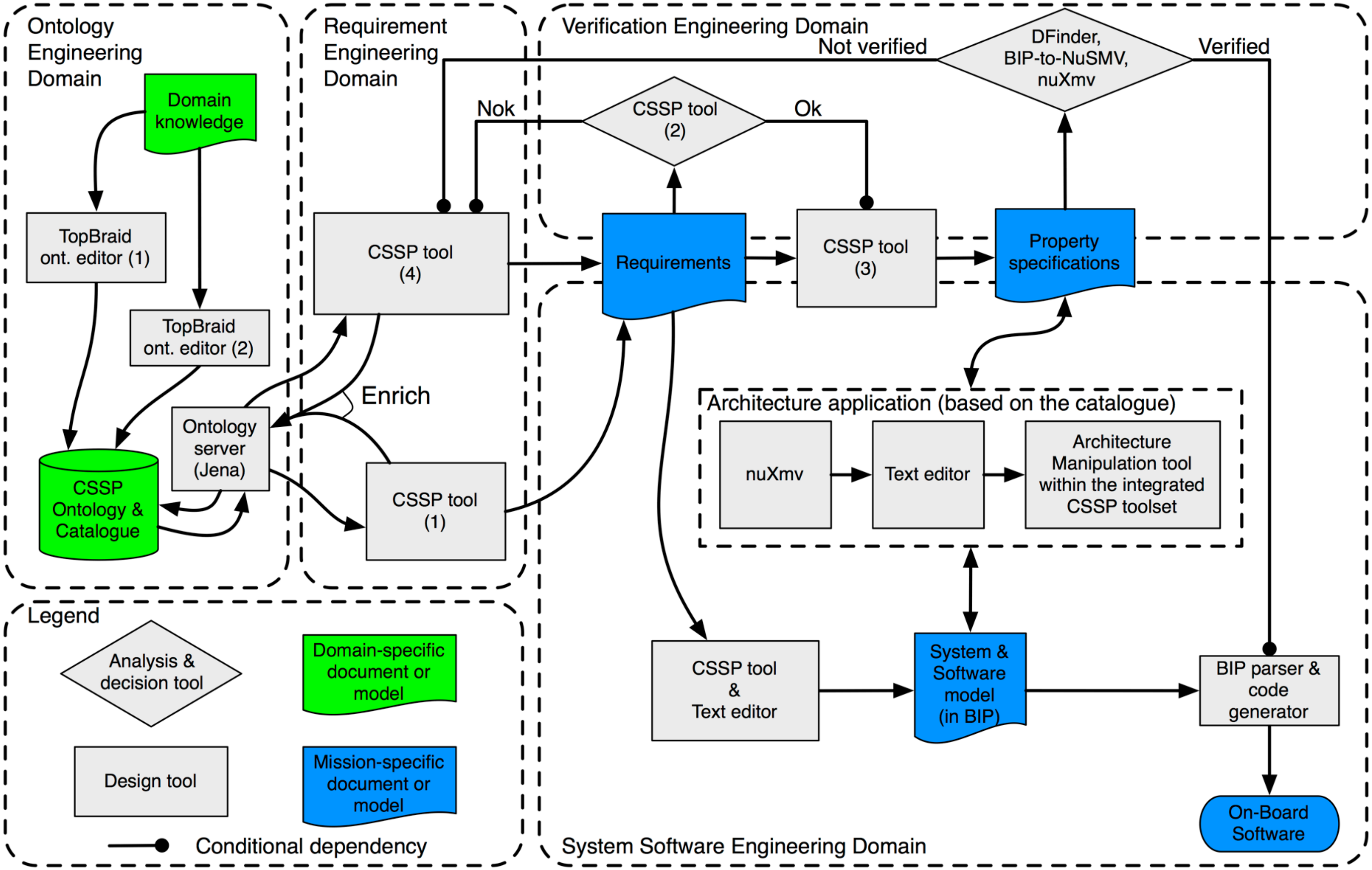
CSSP Process: High-level view



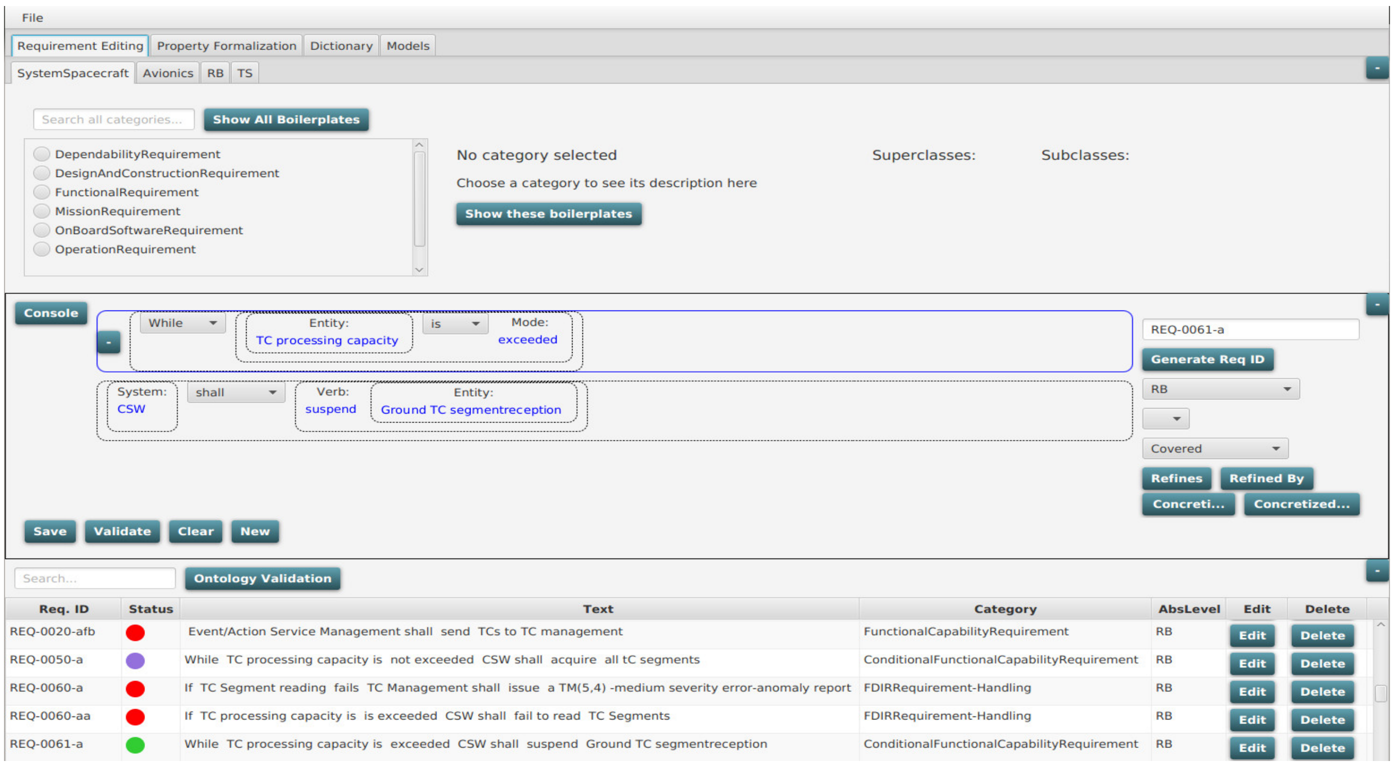
CSSP Process: Detailed view



CSSP Properties Specification & Verification framework



CSSP tool



- Catalogue-driven specification guidance
- Boilerplate-based specification of requirements
- Aid to avoid concepts that are not mapped to the CSSP Ontology
- Semantic search & validation of requirement/property specifications
- Guidance for specification of (i) enforceable and (ii) verifiable properties
- Pattern-based specification of properties
- Correctness-by-construction through BIP model transformations that enforce specified properties

Case studies

- CubETH satellite On-Board software (internal consortium study)
  - 36 requirements were covered by 38 enforceable properties
  - To increase the confidence in the architecture-based design approach, additional verification was conducted using the nuXmv tool.

- Sentinel 3 Telecommand Management function (provided by Thales Alenia Space)
  - Aim: to validate the CSSP process and framework of tools
  - 27 RB-level requirements covered by 34 properties (2 verifiable properties)
    - State explosion for the complete model
    - Properties were shown to hold for the subsystems
    - They also hold for the complete model (sound abstraction)
  - The burden of verification is shifted from the final design model to architectures, which are considerably smaller in size and can be reused.

Conclusions

- The design model:
  - means to ensure design correctness
  - requirements that cannot be enforced & verified have to be refined
    - inconsistencies due to specification errors or due to an overly weak assumption for the environment of the involved entities
  - baseline for formal design refinement to introduce new requirements (and properties) at a lower abstraction level
    - two more properties + action refinement are formally checked to ensure consistency

- Software properties (TS-level) are allocated on a BIP model of the software component architecture (OSRA) – specifies the software components behaviour.

Future work

- Defining domain models within the DSO to enable effective ontology-based validation.
  - system and software engineers experienced in diverse types of missions (currently working for the AOCS)
  - need to encode various types of implicit assumptions:
    - general, e.g. mass cannot be negative
    - mission specific, e.g. the temperature within the orbiting range cannot rise above N degrees
- Further develop the existing BIP model of the software component architecture to enable model-based code generation:
  - static architecture (OSRA)
  - dynamic architecture (Ravenscar + semaphore-based task synchronization)
- Improve the tool support to achieve a higher TRL.

Aristotle University Of Thessaloniki - Greece

- Panagiotis Katsaros, Prof.
- Nick Bassiliades, Prof.
- Ioannis Vlahavas, Prof.
- Ioannis Stamelos, Prof.
- Emmanouela Stachtiari, PhD student
- Manolis Rigas, PhD student
- Alexandros Papageorgiou, Student

École Polytechnique Fédérale de Lausanne - Switzerland

- Joseph Sifakis, Prof.
- Simon Bliudze, Researcher
- Anastasia Mavridou, Researcher

Thales Alenia Space France

- Marco Panunzio, On-board software R&D Engineer

Contact: katsaros@csd.auth.gr

http://www.researchgate.net/project/Catalogue-of-System-and-Software-Properties

Catalogue Of System & Software Properties
ESA/ESTEC - Dec. 7, 2016