THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

2014 NASA Annual IV&V Workshop

Fairmont, WV, USA

Capturing Autonomy Features for Unmanned Spacecraft with ARE, the Autonomy Requirements Engineering Approach

Emil Vassev and Mike Hinchey
Lero @ University of Limerick

September 11, 2014

Outline

1. Introduction:
    - Problem Statement
    - Automation vs Autonomy
    - Autonomy Levels for Unmanned Missions
2. Autonomy Requirements Engineering
3. Case Study: BepiColombo
    - Requirements Elicitation
    - Requirements Specification
4. Summary & Future Work

Autonomy Requirements Engineering
Problem Statement

Space Industry gradually adds autonomy to flight and ground systems to:

- increase the amount of science and reduce mission costs.

Promotion of autonomy in space missions - an extremely challenging task:

× NASA and ESA are currently approaching the AC paradigm:

- develop **autonomous components** for spacecraft (e.g., BepiColombo);
- use traditional development approaches;
    - inappropriate – pays scant attention to the autonomic features;

× automatic is not autonomic;

✓ First challenge - *elicitation and expression of autonomy requirements*:

- determine what *autonomic features are to be developed for a particular space mission;*
- *generate autonomy requirements supporting those features.*

Autonomy Requirements Engineering
Automation vs Autonomy

Processes may be executed without human intervention.

- *Automated processes* :
  - replace routine manual processes with software/hardware ones that follow a step-by-step sequence.
- *Autonomous processes*:
  - have the more ambitious goal of emulating human processes rather than simply replacing them.
- Complete autonomy may not be desirable or possible:
  - *adjustable autonomy* - the level of autonomy of the system (e.g., spacecraft) can vary depending on the circumstances or the needed interaction and control.
  - autonomy can be adjusted to be either *complete*, *partial* or *no autonomy*.

Autonomy Requirements Engineering
Autonomy Levels for Unmanned Missions

Autonomy Level	Description	Functions
E1	1) Mission execution under ground control; 2) Limited onboard capability for safety issues.	1) Real-time control from ground for nominal operations. 2) Execution of time-tagged commands for safety issues.
E2	Execution of pre-planned, ground-defined, mission operations onboard.	Capability to store time-based commands in an onboard scheduler.
E3	Execution of adaptive mission operations onboard.	Event-based autonomous operations. Execution of onboard operations control procedures.
E4	Execution of goal-oriented mission operations onboard.	Goal-oriented mission re-planning.

Autonomy Requirements Engineering
Starting Point

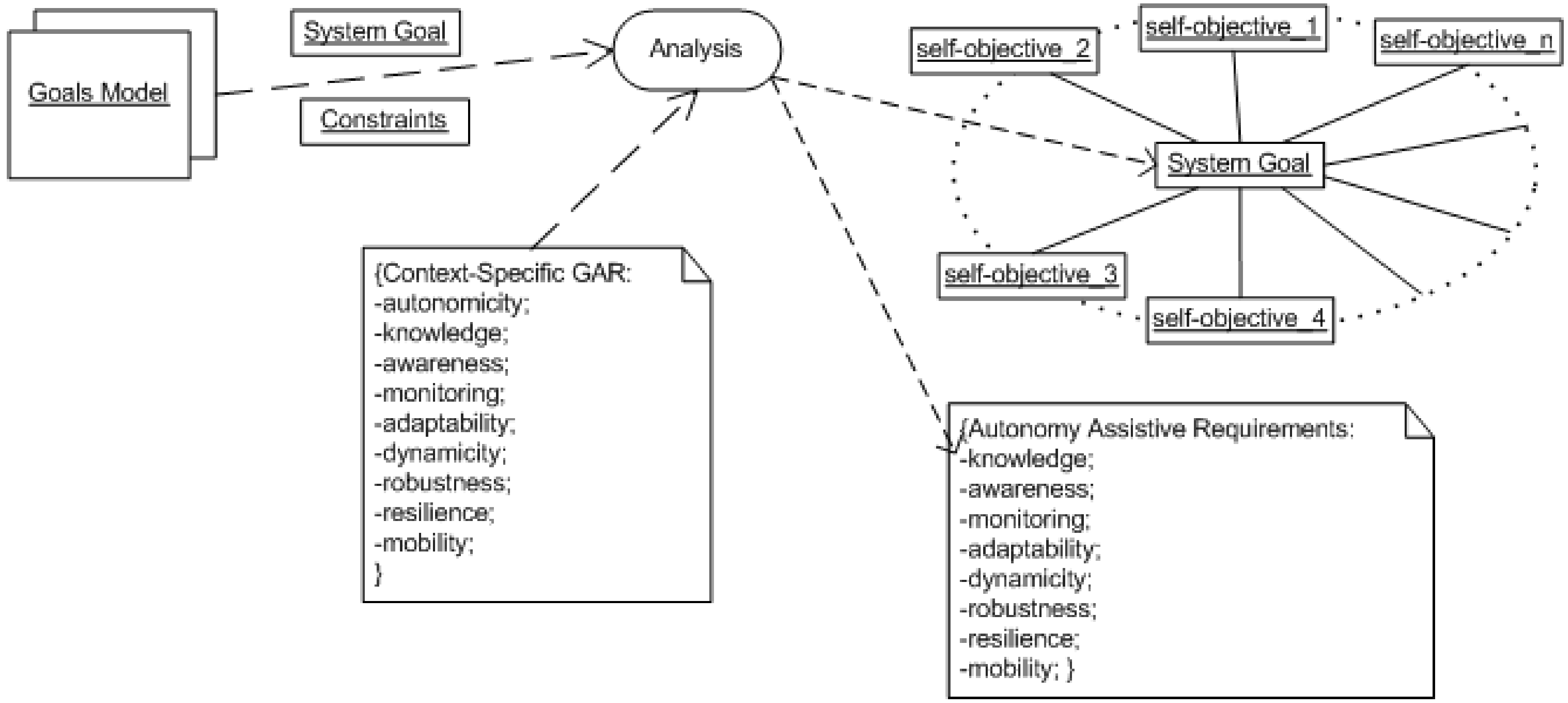
- Where to start from?
  - system's autonomic and self-adaptive nature:
    - able to autonomously pursue goals;
    - monitor environment and subsystem;
    - eventually modify its behavior and/or structure according to changes in the operational environment or goals.

Autonomy Requirements Engineering
Approach

- relies on GORE (Goal Oriented Requirements Engineering) to elicit and define the system goals;
- uses GAR (Generic Autonomy Requirements) interpreted in the specific system's domain to <u>derive</u> and <u>define</u> assistive and often alternative goals (**self-* objectives**) the system may pursue in the presence of factors threatening the achievement of the initial system goals;
- merges GORE with GAR to produce goals models where system goals are supported by self-* objectives promoting autonomicity in system behavior;
- relies on formal languages complying with GAR (e.g., KnowLang) to specify the autonomy requirements;

Autonomy Requirements Engineering
Approach

GORE + GAR = self-* objectives



{Context-Specific GAR:
-autonomicity;
-knowledge;
-awareness;
-monitoring;
-adaptability;
-dynamicity;
-robustness;
-resilience;
-mobility;
}

{Autonomy Assistive Requirements:
-knowledge;
-awareness;
-monitoring;
-adaptability;
-dynamicity;
-robustness;
-resilience;
-mobility; }

Autonomy Requirements Engineering
Mission-Specific GAR

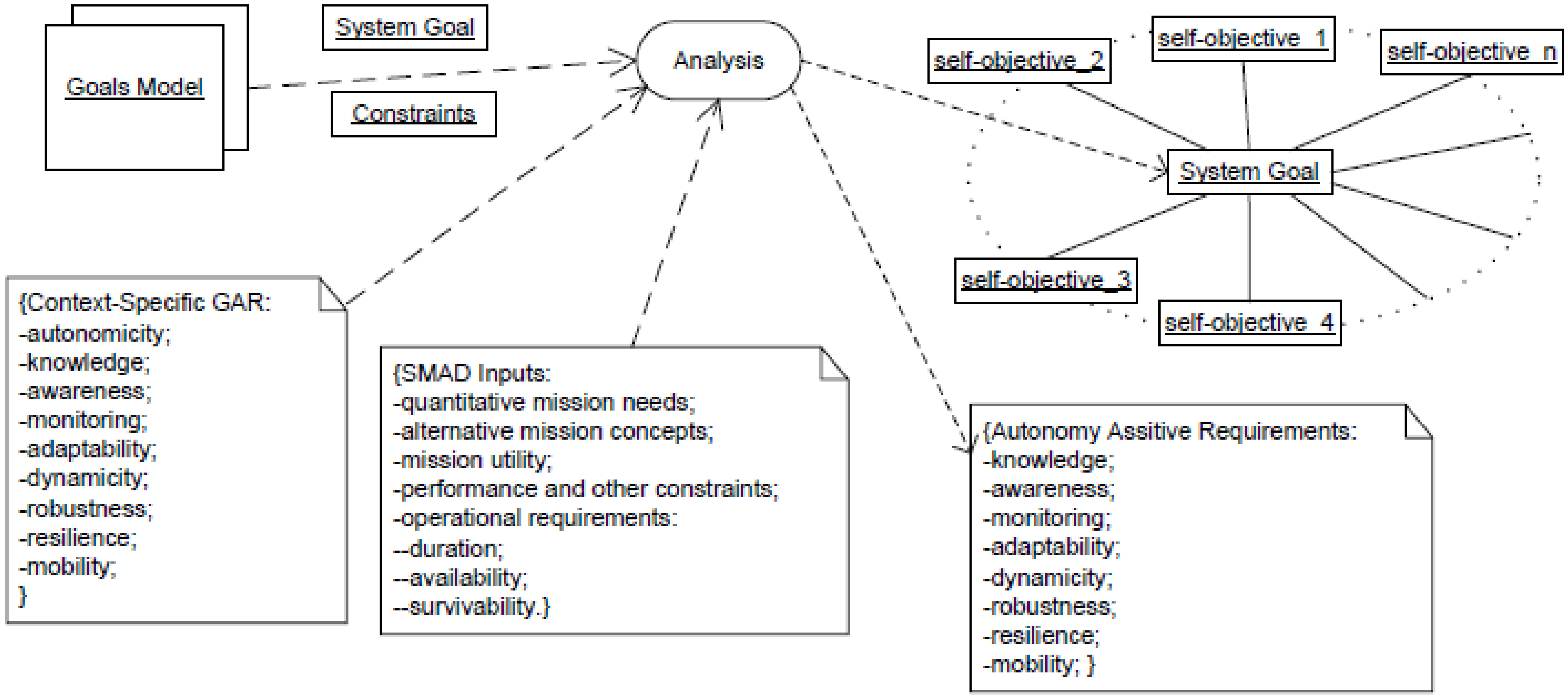
Domain-specific GAR defined for:

- Earth-Orbiting Missions:
  - Polar Low Earth Orbit (LEO)/Remote-Sensing Satellite Missions;
  - Satellite Constellation Missions;
  - Geostationary Earth Orbit (GEO) Missions;
  - Highly Elliptic Orbit Missions:
    - » Space-borne Observatories;
    - » Communication Spacecraft.
- Interplanetary Missions:
  - Small Object Missions – "To Orbit" and "To Land" Missions;
  - Missions using Low-Thrust Trajectories;
  - Planetary Atmospheric Entry and Aeromaneuvering Missions.

P. Fortescue, G. Swinerd, J. Stark (eds.), "Spacecraft Systems Engineering", 4th Edition, Wiley, 2011.

Autonomy Requirements Engineering
Space Mission Analysis and Self-* Objectives

GORE + GAR + SMAD = self-* objectives



{Context-Specific GAR:
-autonomicity;
-knowledge;
-awareness;
-monitoring;
-adaptability;
-dynamicity;
-robustness;
-resilience;
-mobility;
}

{SMAD Inputs:
-quantitative mission needs;
-alternative mission concepts;
-mission utility;
-performance and other constraints;
-operational requirements:
--duration;
--availability;
--survivability.}

{Autonomy Assitive Requirements:
-knowledge;
-awareness;
-monitoring;
-adaptability;
-dynamicity;
-robustness;
-resilience;
-mobility; }

Autonomy Requirements Engineering

Safety-Related Autonomy Requirements

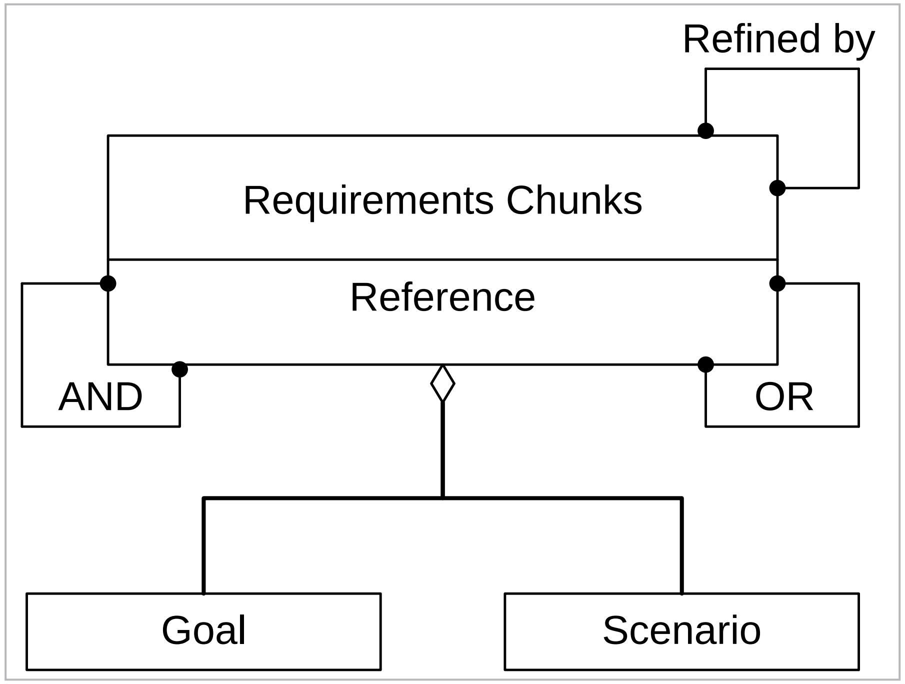
Can be derived by a four-stage process:

- 1. *Hazard identification* – a hazard might be regarded as a condition that may lead to an accident.
- 2. *Hazard analysis* – possible causes of the system's hazards are explored and recorded.
- 3. *Identifying Safety Capabilities* – a key step is to identify the capabilities the system needs to have in order to perform its goals and remain safe.
- 4. *Requirements derivation* – safety requirements to either prevent the hazards occurring or mitigate the resulting accidents via self-* objectives.

Autonomy Requirements Engineering
Requirements Chunks

ARE Requirements Chunks: associate each goal with scenarios

- the goal-scenario pairs can be assembled together through composition, alternative and refinement relationships;
- AND and OR structures of requirements chunks + hierarchy of chunks of different granularity.

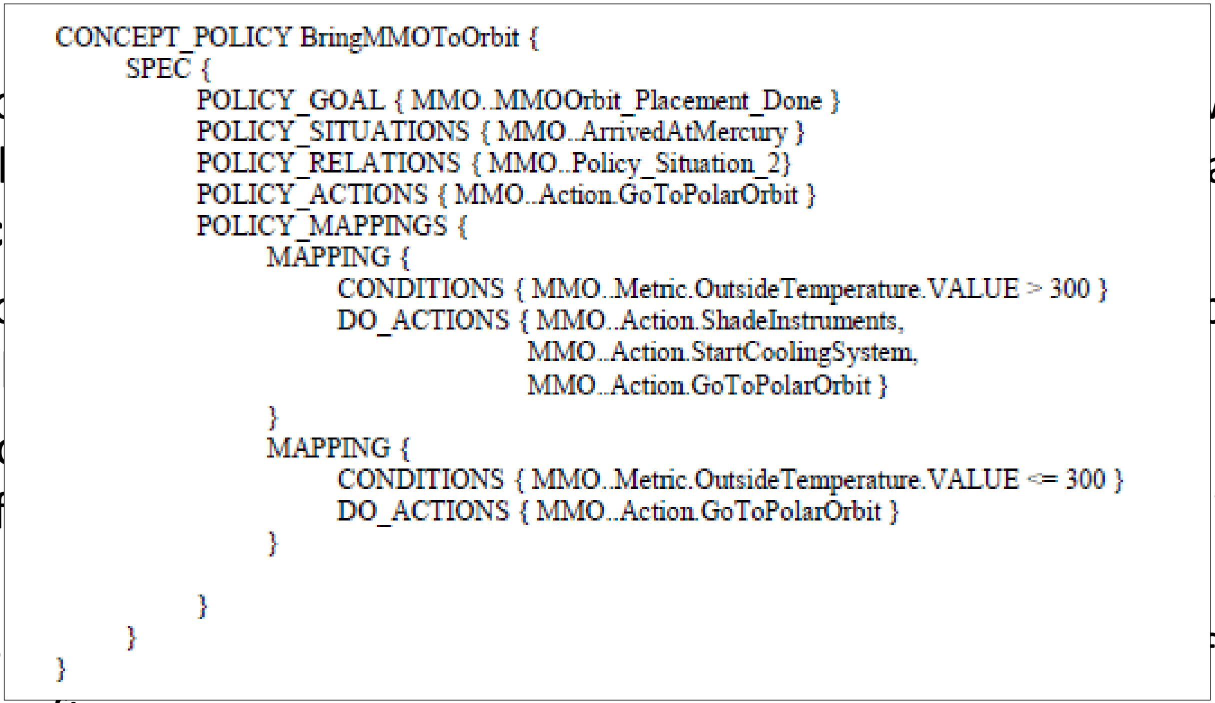


Autonomy Requirements Engineering

Formal Specification

ARE relies on **KnowLang** for the formal specification of the elicited autonomy requirements :

- self-* c with **goals**, special abilities), **metric**
- self-* c bstract level what t
- situati met in order f ing a system goal;
- policie f the spacecraft.

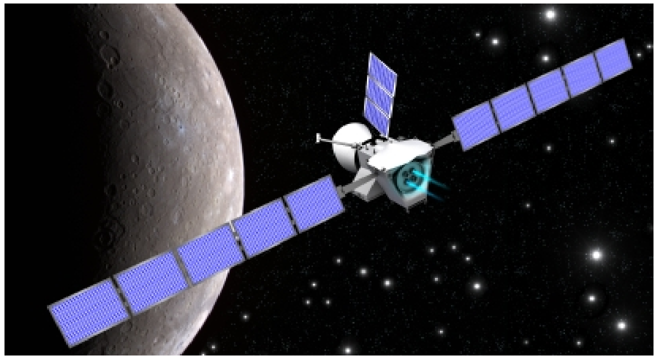


```
CONCEPT_POLICY BringMMOToOrbit {
        SPEC {
                POLICY_GOAL { MMO..MMOOrbit_Placement_Done }
                POLICY_SITUATIONS { MMO..ArrivedAtMercury }
                POLICY_RELATIONS { MMO..Policy_Situation_2}
                POLICY_ACTIONS { MMO..Action.GoToPolarOrbit }
                POLICY_MAPPINGS {
                        MAPPING {
                                CONDITIONS { MMO..Metric.OutsideTemperature.VALUE > 300 }
                                DO_ACTIONS { MMO..Action.ShadeInstruments,
                                                 MMO..Action.StartCoolingSystem,
                                                 MMO..Action.GoToPolarOrbit }
                        }
                        MAPPING {
                                CONDITIONS { MMO..Metric.OutsideTemperature.VALUE <= 300 }
                                DO_ACTIONS { MMO..Action.GoToPolarOrbit }
                        }
                }
        }
}
```

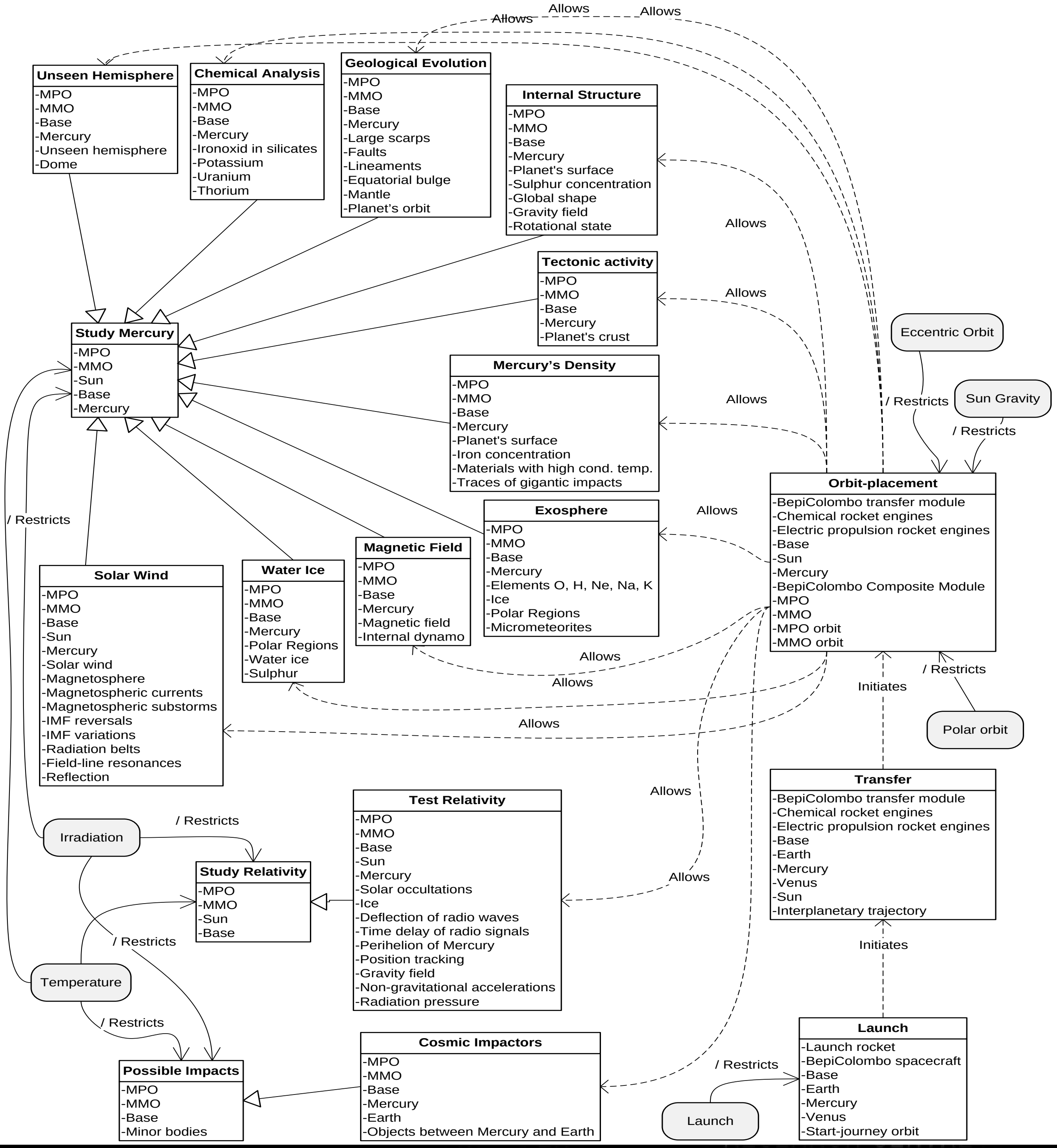
Autonomy Requirements Engineering

Case Study: BepiColombo

- ESA mission to Mercury scheduled for launching in 2015;
  - will perform a series of scientific experiments, tests and measures.
  - the space segment of the BepiColombo Mission consists of two orbiters:
    - a *Mercury Planetary Orbiter (MPO)*
    - a *Mercury Magnetospheric Orbiter (MMO)*.
  - Initially, the two orbiters will be packed together into a special *composite module* used to bring both orbiters into their proper orbits.



By applying GORE, we build a *goals model* that can help us consecutively derive and organize the *autonomy requirements* for BepiColombo.



Autonomy Requirements Engineering
Case Study: GAR for BepiColombo

- BepiColombo falls in the category of *Interplanetary Missions*
  - inherits GAR for such missions;

- associate GAR with each level of objectives, i.e., autonomy requirements (including self-* objectives) associated with:
  - the *Transfer Objective*;
  - the *Orbit-placement Objective;*
  - the *Scientific Objectives*, grouping all the middle-level objectives;

Orbit-placement Objective

- **self**-* **requirements (autonomicity) (partial)**:
  - *self-jettison*: the Transfer Module shall automatically release its SEPM when the right jettison attitude is reached; the Composite Module shall automatically release MMO when the polar orbit is reached.
  - *self-capture*: the Composite Module shall autonomously determine a steering law and use low thrust to achieve capture around Mercury.
  - *self-escape*: the Composite Module shall autonomously acquire the escape procedure and use it to leave Mercury if necessary;
  - *self-thermal-control*: both MMO and MPO shall maintain the onboard equipment and the spacecraft structure in proper temperature range.

Autonomy Requirements Engineering
Case Study: GAR for BepiColombo

Orbit-placement Objective

- **knowledge:** *central force field physics; steering law model for weak stability boundary capture; MMO orbit; MPO orbit; maximum rate of change of orbital energy for MMO and MPO; maximum rate of change of orbital inclination for MMO and MPO; instruments onboard together with their characteristics* (acceptable levels of radiation)*; Base on Earth*; *propulsion system* (chemical propulsion rockets)*; communication links, data transmission format, communication mechanisms onboard*; *gravitational forces* (Sun gravity and Mercury gravity);

Autonomy Requirements Engineering
Case Study: GAR for BepiColombo

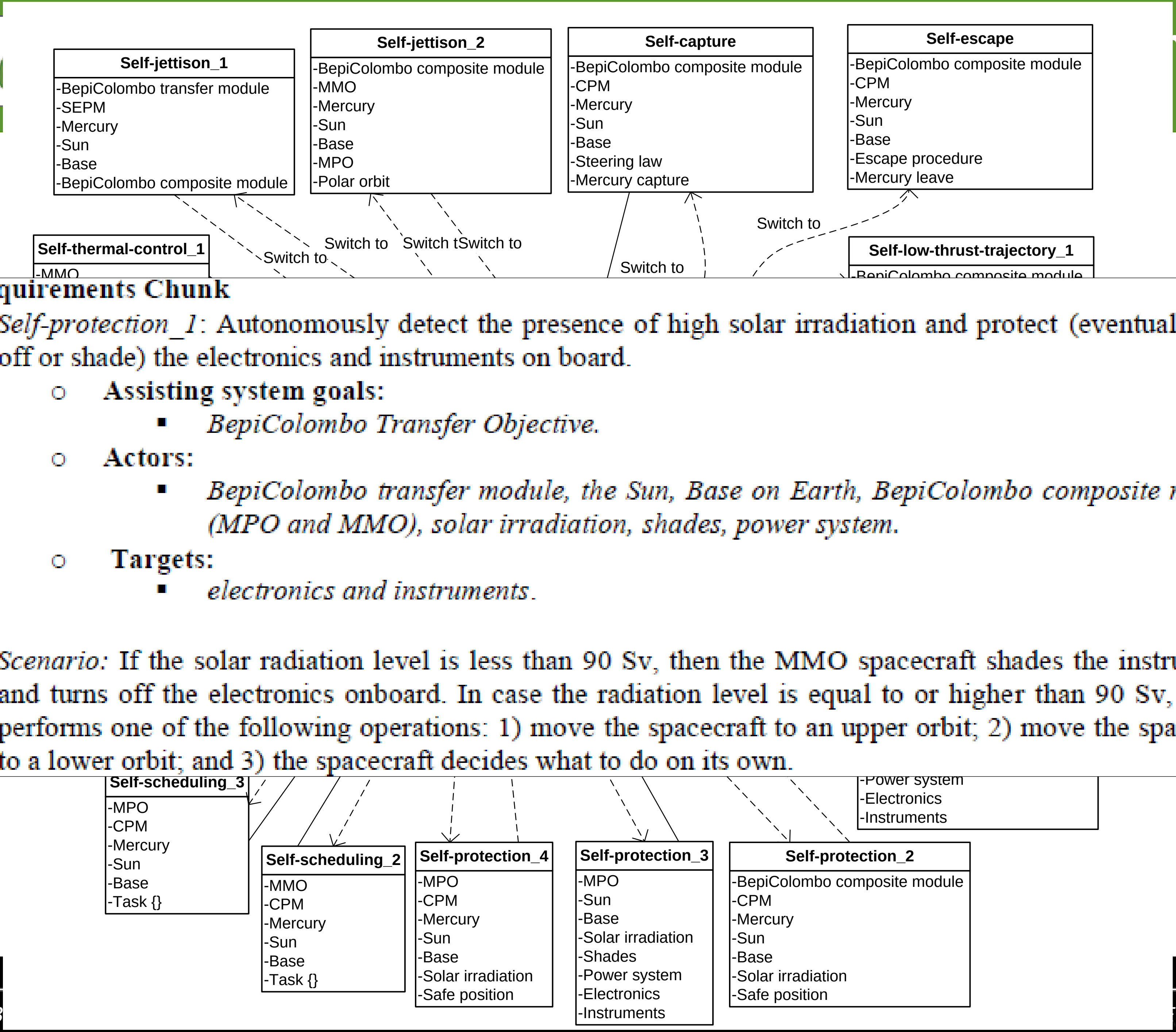
Orbit-placement Objective

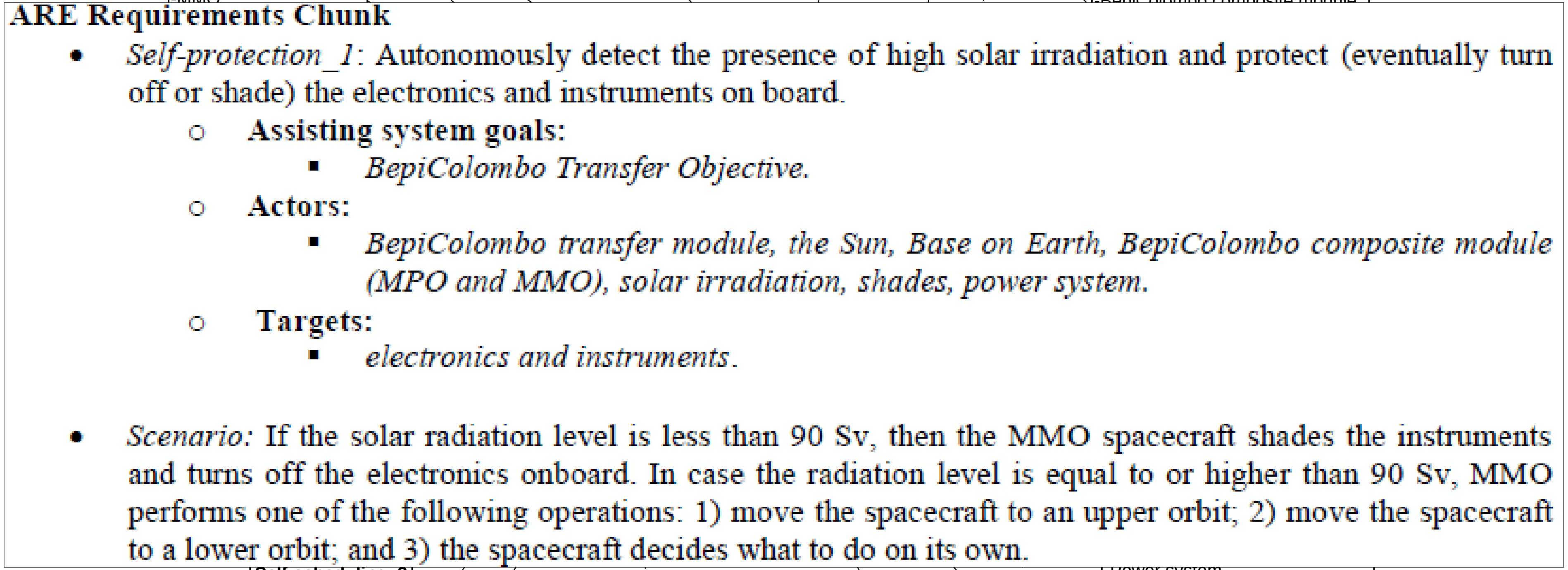
- **awareness** (for both the Composite Module and MPO)**:** *Mercury capture awareness; Mercury escape awareness; trajectory velocity awareness; Mercury's magnetic field awareness; Mercury's gravitational force awareness; Sun's gravitational force awareness; awareness of the spacecraft's position on the projected trajectory perturbations; radiation awareness; instrument awareness; sensitive to thermal stimuli; data-transfer awareness; speed awareness; communication awareness*.
- **monitoring** (for both the Composite Module and MPO)**:** *the environment around Mercury* (e.g., radiation level, Mercury, the Sun); *planned operations* (status, progress, feasibility, etc.).

Autonomy Requirements Engineering
Case Study: GAR for BepiColombo

Orbit-placement Objective

- **adaptability** (for both the Composite Module and MPO)**:** *adapt the low thrust trajectories to orbit and/or altitude perturbations*.
- **dynamicity** (for both the Composite Module and MPO)**:** *dynamic near-body environment*; *dynamic trajectory following procedure*; *dynamic communication links*.
- **robustness** (for both the Composite Module and MPO)**:** *robust to solar irradiation*; *robust to temperature changes* (high temperature amplitude)*; robust to orbit-placement trajectory perturbations; robust to communication losses.*
- **resilience** (for both the Composite Module and MPO)**:** *resilient to magnetic field changes*.
- **mobility** (for both the Composite Module and MPO)**:** *trajectory maneuvers for avoiding orbit and/or altitude perturbations*.





ARE Requirements Chunk

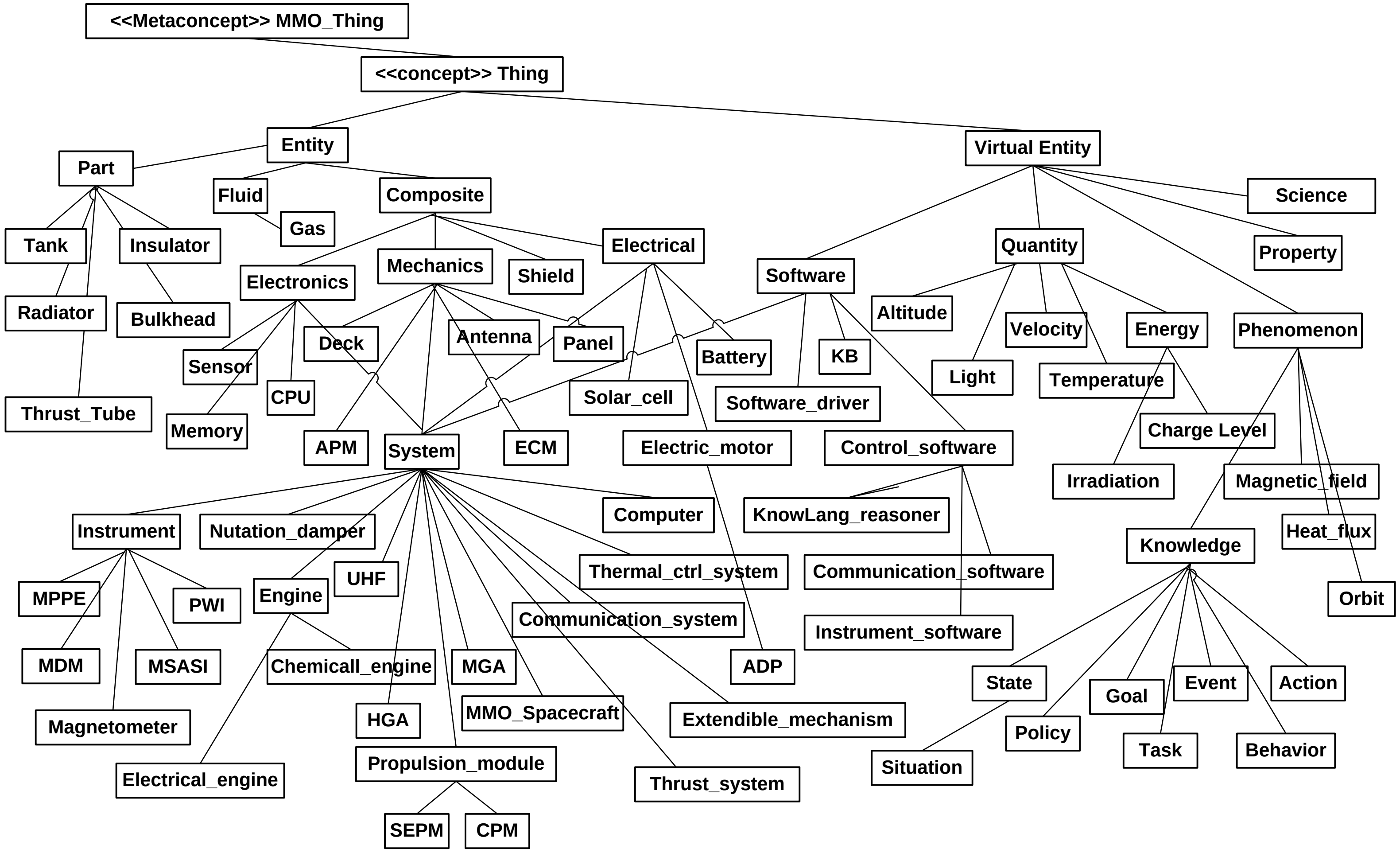
- *Self-protection_1*: Autonomously detect the presence of high solar irradiation and protect (eventually turn off or shade) the electronics and instruments on board.
  - **Assisting system goals:**
    - *BepiColombo Transfer Objective.*
  - **Actors:**
    - *BepiColombo transfer module, the Sun, Base on Earth, BepiColombo composite module (MPO and MMO), solar irradiation, shades, power system.*
  - **Targets:**
    - *electronics and instruments.*

- *Scenario:* If the solar radiation level is less than 90 Sv, then the MMO spacecraft shades the instruments and turns off the electronics onboard. In case the radiation level is equal to or higher than 90 Sv, MMO performs one of the following operations: 1) move the spacecraft to an upper orbit; 2) move the spacecraft to a lower orbit; and 3) the spacecraft decides what to do on its own.

Autonomy Requirements Engineering
Case Study: Requirements Specification

KnowLang used to specify:

- o O,
  B nd
  B
- d gh
  re
- e
- s

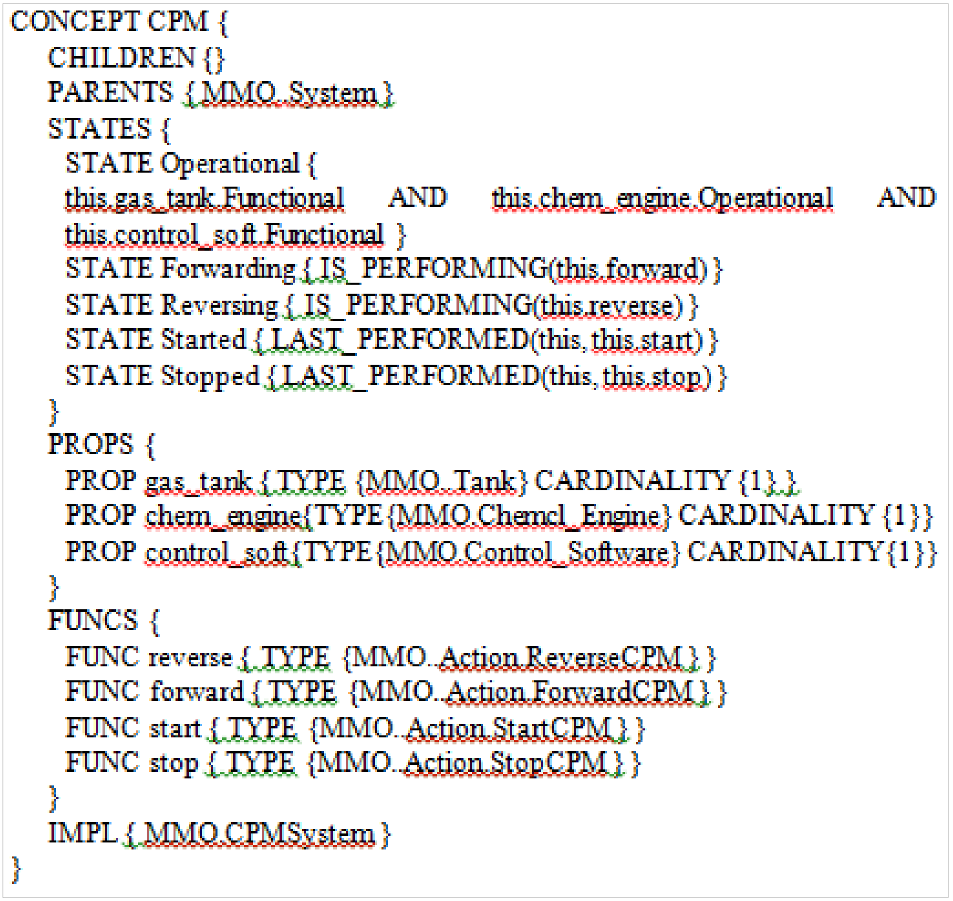


THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

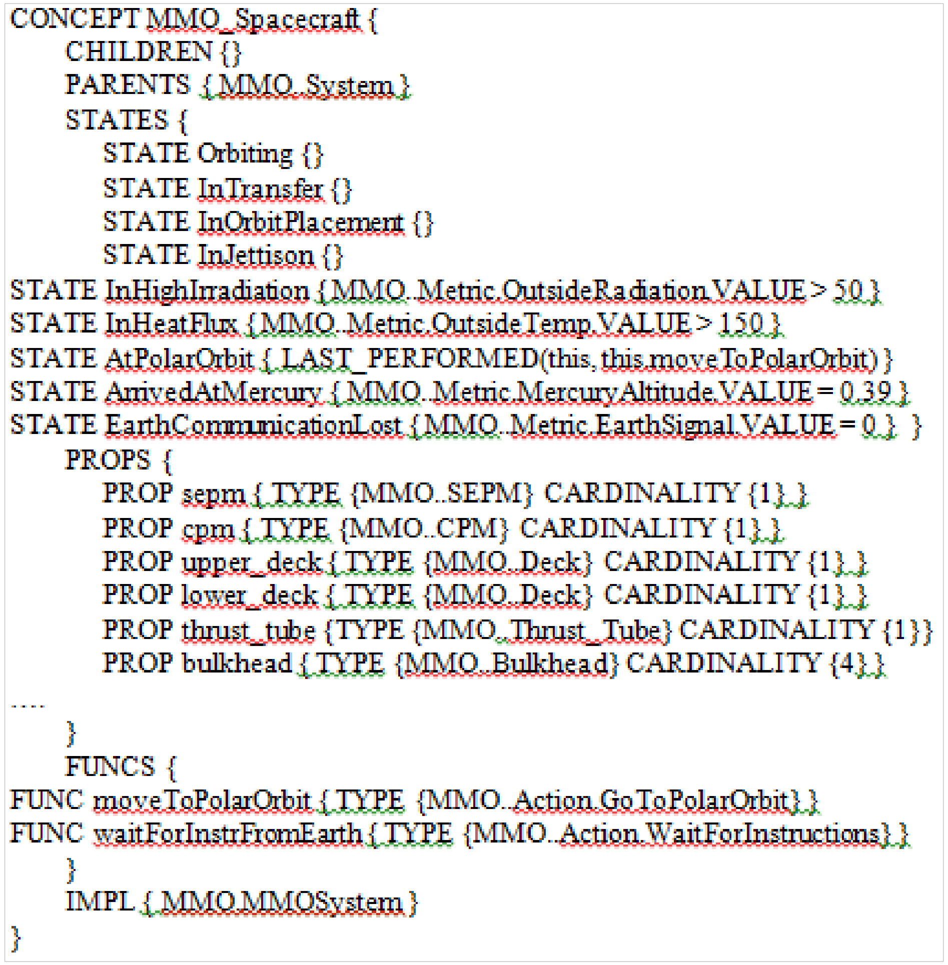
Autonomy Requirements Engineering

Case Study: Requirements Specification

Specified concepts (with states):



```
CONCEPT CPM {
   CHILDREN {}
   PARENTS { MMO.System }
   STATES {
     STATE Operational {
     this.gas_tank.Functional    AND    this.chem_engine.Operational    AND
     this.control_soft.Functional  }
     STATE Forwarding { IS_PERFORMING(this.forward) }
     STATE Reversing { IS_PERFORMING(this.reverse) }
     STATE Started { LAST_PERFORMED(this, this.start) }
     STATE Stopped { LAST_PERFORMED(this, this.stop) }
   }
   PROPS {
     PROP gas_tank { TYPE {MMO..Tank} CARDINALITY {1} }
     PROP chem_engine{TYPE {MMO.Chemcl_Engine} CARDINALITY {1}}
     PROP control_soft{TYPE {MMO.Control_Software} CARDINALITY {1}}
   }
   FUNCS {
     FUNC reverse { TYPE {MMO..Action.ReverseCPM } }
     FUNC forward { TYPE {MMO..Action.ForwardCPM } }
     FUNC start { TYPE {MMO..Action.StartCPM } }
     FUNC stop { TYPE {MMO..Action.StopCPM } }
   }
   IMPL { MMO.CPMSystem }
}
```



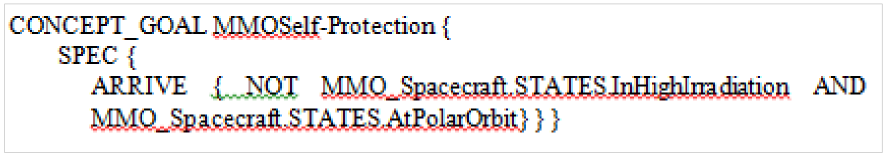
```
CONCEPT MMO_Spacecraft {
     CHILDREN {}
     PARENTS { MMO..System }
     STATES {
        STATE Orbiting {}
        STATE InTransfer {}
        STATE InOrbitPlacement {}
        STATE InJettison {}
STATE InHighIrradiation { MMO..Metric.OutsideRadiation.VALUE > 50 }
STATE InHeatFlux { MMO..Metric.OutsideTemp.VALUE > 150 }
STATE AtPolarOrbit { LAST_PERFORMED(this, this.moveToPolarOrbit) }
STATE ArrivedAtMercury { MMO..Metric.MercuryAltitude.VALUE = 0.39 }
STATE EarthCommunicationLost { MMO..Metric.EarthSignal.VALUE = 0 }  }
     PROPS {
        PROP sepm { TYPE {MMO..SEPM} CARDINALITY {1} }
        PROP cpm { TYPE {MMO..CPM} CARDINALITY {1} }
        PROP upper_deck { TYPE {MMO..Deck} CARDINALITY {1} }
        PROP lower_deck { TYPE {MMO..Deck} CARDINALITY {1} }
        PROP thrust_tube {TYPE {MMO..Thrust_Tube} CARDINALITY {1}}
        PROP bulkhead { TYPE {MMO..Bulkhead} CARDINALITY {4} }
....
     }
     FUNCS {
FUNC moveToPolarOrbit { TYPE {MMO..Action.GoToPolarOrbit} }
FUNC waitForInstrFromEarth { TYPE {MMO..Action.WaitForInstructions} }
     }
     IMPL { MMO.MMOSystem }
}
```

Autonomy Requirements Engineering

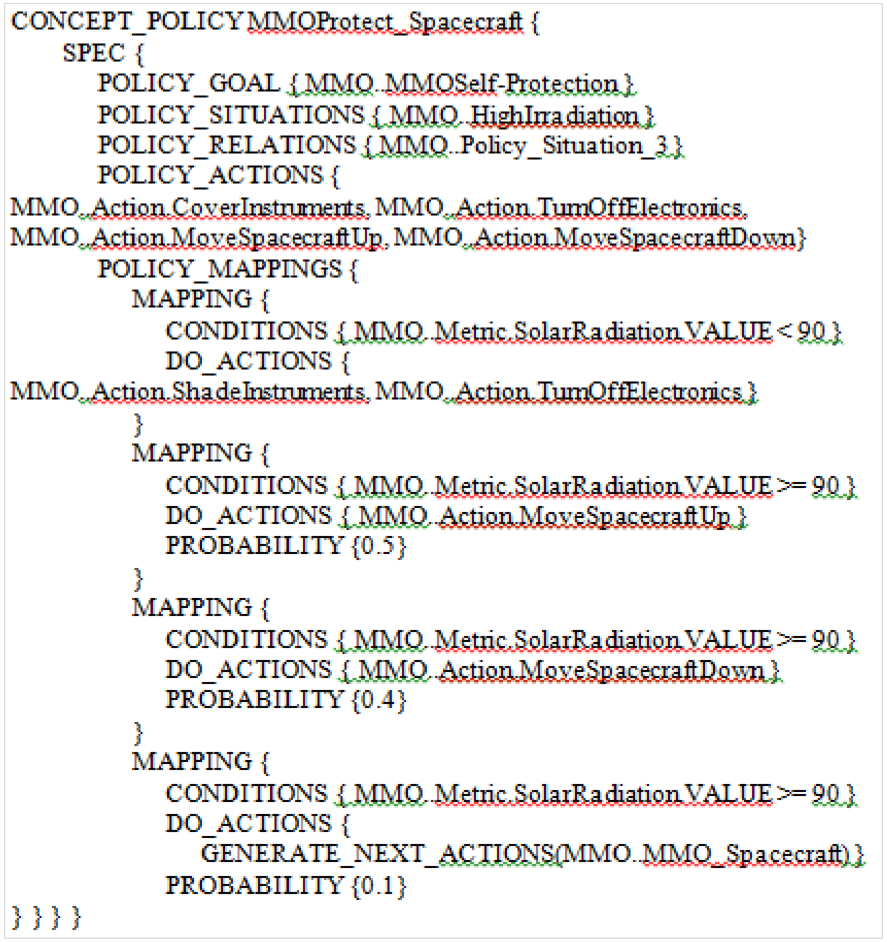
Case Study: Requirements Specification

GENERATE_NEXT_ACTIONS – an operator increasing the goal-oriented autonomy:

- automatically generate the most appropriate actions;
- the action generation is based on the computations performed by a special *reward function* implemented by the KnowLang Reasoner.
- *KnowLang Reward Function* (KLRF) observes the outcome of the actions to compute the possible successor states of every possible action execution and grants the actions with special *reward number* considering the current system state (or states, if the current state is a composite state) and goals.



```
CONCEPT_GOAL MMOSelf-Protection {
    SPEC {
        ARRIVE { NOT MMO_Spacecraft.STATES.InHighIrradiation AND
        MMO_Spacecraft.STATES.AtPolarOrbit} } }
```



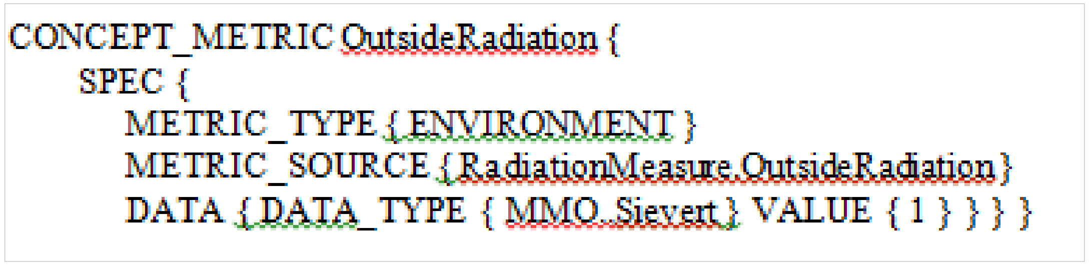
```
CONCEPT_POLICY MMOProtect_Spacecraft {
    SPEC {
        POLICY_GOAL { MMO..MMOSelf-Protection }
        POLICY_SITUATIONS { MMO..HighIrradiation }
        POLICY_RELATIONS { MMO..Policy_Situation_3 }
        POLICY_ACTIONS {
MMO..Action.CoverInstruments, MMO..Action.TurnOffElectronics,
MMO..Action.MoveSpacecraftUp, MMO..Action.MoveSpacecraftDown}
        POLICY_MAPPINGS {
            MAPPING {
                CONDITIONS { MMO..Metric.SolarRadiation.VALUE < 90 }
                DO_ACTIONS {
MMO..Action.ShadeInstruments, MMO..Action.TurnOffElectronics }
            }
            MAPPING {
                CONDITIONS { MMO..Metric.SolarRadiation.VALUE >= 90 }
                DO_ACTIONS { MMO..Action.MoveSpacecraftUp }
                PROBABILITY {0.5}
            }
            MAPPING {
                CONDITIONS { MMO..Metric.SolarRadiation.VALUE >= 90 }
                DO_ACTIONS { MMO..Action.MoveSpacecraftDown }
                PROBABILITY {0.4}
            }
            MAPPING {
                CONDITIONS { MMO..Metric.SolarRadiation.VALUE >= 90 }
                DO_ACTIONS {
                    GENERATE_NEXT_ACTIONS(MMO..MMO_Spacecraft) }
                PROBABILITY {0.1}
} } } }
```

Autonomy Requirements Engineering

Case Study: Requirements Specification

Monitoring

- handled via the explicit *Metric concept*;
- system's sensors generate raw data that represent the physical characteristics of the world;
- MMO's sensors are controlled by a software driver (e.g., implemented in C++) where appropriate methods are used to control a sensor and read data from it;
- a *Metric concept* introduces a class of sensors to the KB, and by specifying instances of that class, we represent the real sensors.



```
CONCEPT_METRIC OutsideRadiation {
     SPEC {
        METRIC_TYPE { ENVIRONMENT }
        METRIC_SOURCE { RadiationMeasure.OutsideRadiation }
        DATA { DATA_TYPE { MMO.Sievert } VALUE { 1 } } } }
```

Autonomy Requirements Engineering

Case Study: Requirements Specification

Awareness

- handled by the KnowLang Reasoner via specified concepts:
  - metrics that support both *self*- and *environment monitoring*;
  - states where metrics are used we introduce awareness capabilities for *self-awareness* and *context-awareness*;
  - *situations* introduce the basis for *situational awareness*.

Resilience, **Robustness**, **Mobility**, **Dynamicity** and **Adaptability**:

- handled by *soft goals* - unlike regular goals, soft-goals can seldom be accomplished or satisfied;
  - degree of satisfaction (using probabilities and/or policy conditions);
- *mobility*, *dynamicity* and *adaptability* - soft-goals with *relatively high degree of satisfaction.*

Autonomy Requirements Engineering
Summary

Space Industry gradually adds autonomy to flight and ground systems:

- increase the amount of science and reduce mission costs;
- extremely challenging task.

ARE meets the challenge of capturing autonomy requirements:

- merges GORE with GAR to produce goals models where system goals are supported by self-* objectives promoting autonomicity in system behavior;
- relies on formal languages complying with GAR (e.g., KnowLang) to specify the autonomy requirements.

BepiColombo – a proof of concept case study.

Autonomy Requirements Engineering
Future Work

Efficient tools supporting ARE

- autonomy requirements validation;
- test bed: runtime knowledge representation and reasoning shall be provided along with monitoring mechanisms to test the autonomy behavior and awareness capabilities of a system;
- an intelligent GAR framework using adaptation patterns;
- integration of tools handling SMAD;
- smooth transition from requirements to implementation:
  - test case generation;
  - code generation.

Lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

Thank You!

Questions?