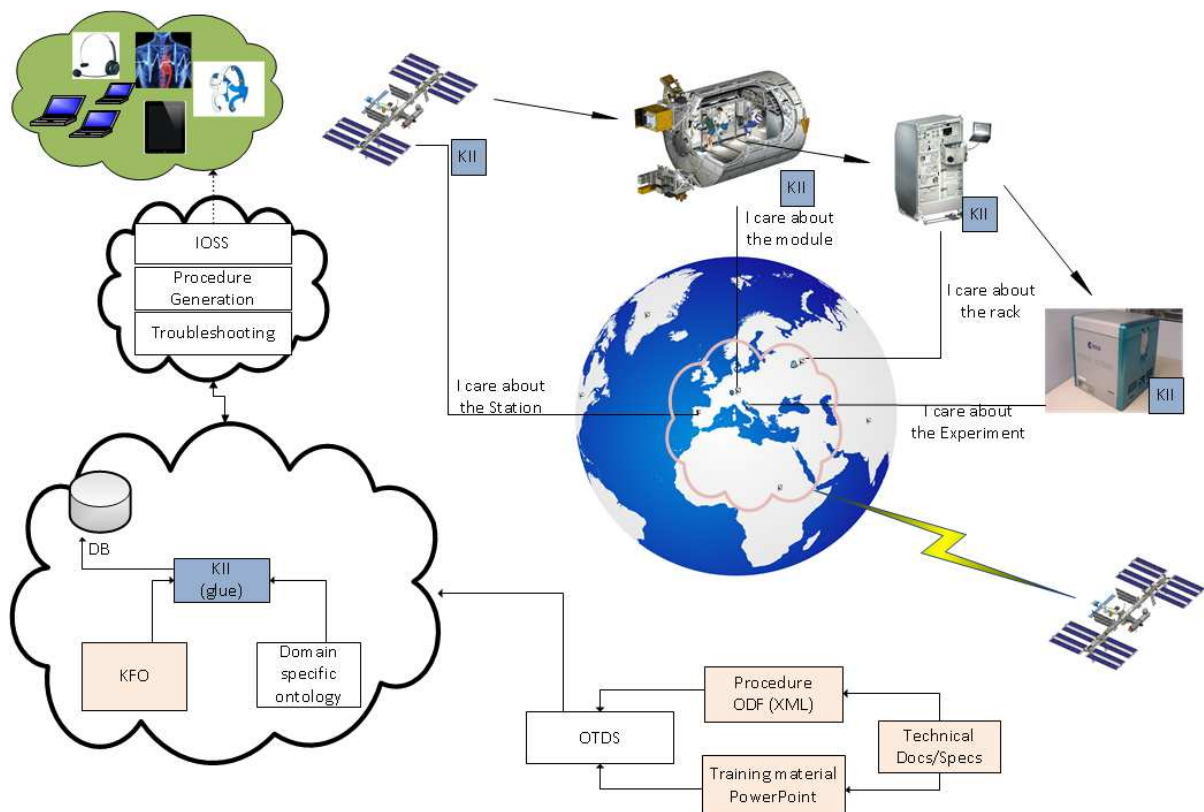


# Mission Execution Crew Assistant (MECA) lite - Knowledge modelling to support the crew / machine interaction

Daniel Esser  
Daniel.Esser@email.de



Utrecht University

Master Thesis for 30 ECTS

Cognitive Artificial Intelligence

Student: 3807894

Under the supervision of Prof. Dr. John-Jules Meyer

ESA ESTEC  
Keplerlaan 1  
2201 AZ Noordwijk



## **Acknowledgements**

I would like to express my gratitude to Alberto Tobias and Joachim Fuchs and Prof. Dr. John-Jules Meyer for giving me the opportunity to write my thesis at ESA-ESTEC.

I furthermore would like to thank Mikael Wolff and David Martinez Oliveira for supervising me during this task and their constant help, in keeping me on track as well as their professional assistance and their patience.

I would like to especially thank Yan Tang and Serge Valera for supporting me with their professional expertise in modelling as well as their domain knowledge through which they were able to point me to the right documentation.

Generally I want to thank all those numerous interlocutors not being mentioned here explicitly for their help, their advice and professionalism.

## Abstract

Artificial Intelligence is one of the most demanding new research areas and just “the first small step” has been taken whilst the “giant leap for mankind” is still outstanding.

In the meantime enormous efforts have been undertaken to try to find out in which area and how far supporting and self-learning elements could be defined and should be introduced in order to finally copy the status or imitate -more or less- a human being, its behaviour, reactions, feelings and especially its ability to continuously broaden one’s own horizon of knowledge independently of the actual environment and situation.

In this context space and the human research curiosity about deep space exploration resulting in plans for long duration human space missions indispensable requires adequate intelligent supporting tools.

These tools may be simple MMI instruction, simple expandable technical tool boxes in sense of a technical data base, personal electronic partners, complex ontology data bases with medical and behavioural data and of all kinds more.

Space operations are of high complexity due to the numerous interfaces and difficult to manage, even in the nominal cases, let alone in off-nominal situations (see current situation on the next page and cover page image). We can expect that for future human (deep) space missions this complexity will only increase. And the greater the distance between Earth and the spacecraft will be, simply due to the technical effect of the runtime delay of the radio signal (about 14<sup>1</sup> minutes in one direction to MARS) effective assistance in all different areas of human aspects will be of relevance.

It thus will be crucial to have the support of a Knowledge Interoperability Infrastructure (KII) that will be able to provide the right information at the right time to the right person. The context for this is a multi-user environment, where it is expected that geographically (and perhaps even temporally) distributed teams, consisting of various combinations of humans (crew, flight control, scientific experts, etc.) and software (reasoning agents, smart systems and instruments, robots, crew *ePartners*, etc.) work together on common goals and objectives.

The focus my thesis, the MECA-lite project, which is based on the KII (section 3.2) and MECA (section 3.1) project, therefore is on investigating the considerations and requirements to identify relevant use cases and scenarios and to propose a suitable formalism for Knowledge Interoperability and Representation. This formalism is then evaluated in practice by building an example knowledge base that uses it, and showing how the information present in this knowledge base can be used to add value to the human-machine teams working on solving common problems.

---

<sup>1</sup> The minimum delay is 4 minutes and the maximum 24 minutes. From now on we assume a mid-way of 14 minutes as was the case during the Curiosity EDL (Entry, Descent, Landing)

## Current situation

The image on the cover page, illustrates the state of the art handling of missions as well as intended extensions. On the one side (upper right side) we have the interest and responsibilities of different MCC (Mission Control Centers) and/or USOCs (User Support Operation Centers), where each Control Center (CC) has a different focus and therefore a different set of knowledge (this can be mission or hardware related). On the other side this knowledge (lower left side) has to be communicated and provided to the ISS (International Space Station) crew. This is currently done by refining technical documents and specifications for hardware (created by the industry) into astronaut training material in form of PowerPoint presentations and nominal as well as off-nominal work procedural operation charts ODF (Operation Data File). So there is no standardized OTDS (Operation and Technical Documentation and Specification) combining this material. In addition, if however there is no communication between the ISS and any CC knowledge is 'lost'. To prevent this loss in future missions, the current 'knowledge support framework' should be extended with IOSSs (Intelligent Operation Support Systems) to distribute location specific knowledge through a KII (Knowledge Interoperability Infrastructure) using KFO (KII Foundation Ontology) and other domain specific ontology's, understandable by man and machine, by using personalized (Laptop, Headset etc.) user interfaces to support mission related procedures and their troubleshooting aspects in unexpected circumstances where no communication is available.

In this thesis the focus was to further investigate this knowledge support framework by reviewing an existing KFO prototype, propose improvements and the development of a domain specific ontology. During the evolution of the thesis it turned out, as a recommendation that a general OTDS should be incorporated in the knowledge framework. This would improve inter-human communication as well as an easier access existing and new knowledge. Furthermore it is demonstrated that this knowledge framework is beneficial for IOSS in procedural generation as well as troubleshooting situations. All components that do not exist so far and can improve current efficiency and are a necessity for interplanetary missions.

## Contents

Contents .....	5
1 Introduction .....	8
1.1 General descriptions .....	10
1.1.1 ISS - International Space Station.....	10
1.1.2 Columbus - European Module of ISS.....	10
1.1.3 KUBIK.....	10
2 Methodology (thesis) .....	10
3 Related work and background research.....	12
3.1 MECA.....	12
3.2 KII .....	12
3.3 CRUISE .....	13
3.4 iPVTouch .....	13
3.5 SICRA.....	14
4 Background knowledge.....	14
4.1 Basics .....	14
4.1.1 Data, Information and Knowledge? .....	15
4.1.2 Space System Databases.....	16
4.1.3 Object-Role Modelling.....	17
4.1.4 Possible Dialogue systems .....	18
4.2 Domain related.....	19
4.2.1 What to model?.....	19
4.2.2 Procedure in ODF format .....	21
4.2.3 KUBIK procedure overview .....	22
4.2.4 Formalism .....	23
4.2.5 OWL vs ORM .....	27
4.3 Description of ORM modelling.....	28
5 Challenges .....	31
6 Goal.....	32
6.1 Approach.....	33

7	Requirements .....	34
8	Scenarios .....	35
8.1	Scenario 1 .....	35
8.2	Scenario 2 .....	35
8.3	Scenario 3 .....	36
8.4	Scenario 4 .....	36
9	KUBIK- an elaborated troubleshooting Scenario .....	37
9.1	Scenario Description .....	37
9.2	Kubik Functional Model.....	37
9.3	Troubleshooting Process .....	38
9.4	Extended troubleshooting scenario.....	38
9.5	Operations Functional Model .....	39
9.6	Troubleshooting scenario (Continuation) .....	39
9.7	Troubleshooting scenario knowledge as model.....	39
9.8	Scenario analysis.....	42
9.9	Final notes.....	42
10	Model structure.....	43
10.1	Ontology structure analysis .....	43
10.2	Resulting general structure .....	45
10.3	Core knowledge example .....	47
11	Conceptual Model (payload) .....	50
11.1.1	Model of the Generic information .....	51
11.1.2	Model of the Activity .....	53
11.1.3	Model of the Equipment.....	54
11.1.4	Model of the Special precaution.....	55
11.1.5	Model of the Hardware Setup.....	56
11.1.6	Model of the Meta model (dynamic view) .....	57
11.1.7	Model of the Unit types.....	57
12	The expert system / IOSS.....	58
13	Evaluation of achievements with respect to goals and requirements.....	61
13.1	Achievement of Goals .....	61
13.2	Requirements .....	62
14	Conclusion and future work.....	64

15	Acronyms .....	66
16	Glossary .....	67
17	Bibliography .....	69
18	Appendix.....	72
18.1	ORM Software.....	72
18.2	Possible scenario dialogues.....	73
18.3	Scenario 3 example relationships Verbalized.....	75

## 1 Introduction

Science fiction films, as a representation of a vision of a possible future, are always giving the impression to the audience that artificial solutions of man- machine interfaces, self-learning machines and every other kind of cooperating systems are nearly about to be realized.

In reality the development of such systems and tools are far from getting reality, but are subject to a step wise technological development and approach.

There are still more open questions than solutions. Thus the challenge is to develop these 'requested systems'. Developments in the space area are always of long duration, vaguely technological formulated, influenced by international collaboration goals that tackle/ deal with incalculable challenges to answer scientific questions using allegedly well defined missions. Right now a huge amount of planning is done in long distance (e.g. Mars) manned missions for which the used technology is still in development using the ISS as a first suitable test bed to tackle the problems encountered in a not yet fully known environment. Furthermore the problematic fact of communication delays as further we distance (e.g. from Mars to earth ~14min delay one direction) ourselves from earth as well as LOS<sup>2</sup> (Loss of Signal) situations require controllable self-learning and self-reacting systems to assist the mission objective.

The current problem with a project like MECA (Mission Execution Crew Assistant, also see section 3.1) and all the documentation that have been produced over the years is indeed its hundreds of thousands of excellent ideas. However all of them are on the conceptual level. Not many have been actually realized into a working system. So it is very challenging to build a complete system like MECA. Up to now, we have just seen a couple of prototypes, like CRUISE (section 3.3) or iPVTouch (section 3.4), but they were all special single aspect solutions and only a small part of the whole system.

Early research of KII (S&T, KII - TN1: Scenarios and Use-cases, 2012) (S&T, KII - TN2: Formats and Formalisms, 2012) (S&T, KII - TN3: Design Considerations, 2012)(section 3.2) and MECA (Neerincx, Smets, Breebaart, Soler, Plassmeier, & van Diggelen, 2013) concluded that to make optimal use of the vast amount of digitally stored information we need a system that delivers the right message at the right moment in the right modality. It should have the following properties:

- **Ontology based** - based on one or a set of extendable OWL ontologies which specify the formats in which information can be represented and shared
- **Information push and pull** – can send information to a user (pro-active), but can also answer requests of information
- **User adaptation** – The system adapts its information supply to its users (e.g. information fit user/task)

---

<sup>2</sup> We can't prevent those situations since we cannot guarantee that our 'radio' signals can reach a target. E.g. we cannot communicate with objects behind the Moon without setting up a satellite to relay the signal.



- **Multi device** – Supports multi devices such as visual display, stationary computers etc. Meaning must support multiple modalities and choosing of best interface.
- **Policy management** – It should be ensured that the astronauts follow security and equipment related policies. E.g. if an astronaut would try EVA (Extra Vehicular Activity) without a suit he should not be allowed to open the door.

The property “multi device” will however not be considered in depth in this research, since it is mainly a question of the most suitable GUI. The research will only include a small amount of recommendations for suitable interfaces in this area. It will be assumed that there exists a system capable of having dialogues with the human using a GUI or verbal conversation, which can be seen as being equipollent (van Dam, 2006).

The main objective of this paper is to develop a conceptual 'MECA-Lite' version with the hope that a full extended version could be developed in the next 2-3 years. Furthermore I will show that within the thesis FBM (Fact Base Modelling) is a viable methodology to simplify complex intermediate steps and also will reduce in general the workload.

Furthermore the currently used ESA (European Space Agency) ODF (Operation Data File) standard (NASA, 2013) to describe standardized presentation of crew tasks for space infrastructures is planned to be renewed. The hope is that results of this thesis could support such new standards or become part of the standard in 5-10 years.

Another aspect to be emphasized is, what I will call 'multifunctional model optimizing' (MMO). Meaning the developed (or used) model should be capable to replace/incorporate different existing models (ER, UML, OWL etc.). This way it would be possible to transfer the generated information of the new model into these older (scientifically approved) existing models, while keeping all the knowledge gathered in one model instead of several. As a result in addition the remodelling for different knowledge sources (astronauts, operation engineers, researcher, contractors...) will be avoided.

In section 2 the methodology used to tackle the problem at hand will be elaborated on as well as changes and adaptation to the original planning. Section 3 then will provide the reader with knowledge about projects that are directly related to or had influence on the development of this thesis. The theoretical background knowledge required will be illustrated in section 4. This section is divided into basic knowledge, elaborating on data information knowledge in modelling and major (domain related) objectives to be reached within this paper as well as decisions taken to illustrate chosen design approaches. In section 5 the challenges related to space domain as well as to the system will be described. Followed by a summary of the intelligent modelling goal in the frame of space oriented applications, describing the system to be created and why in section 6. Section 6.1 details the working order chosen for the solution of the approach of the thesis. Section 7 will elaborate on the requirements identified for this project. The Scenarios which were used to guide the development are illustrated in section 8 while section 9 consists of an elaborated walkthrough of one scenario which was chosen as the main source of examination. Section 10 concludes on findings of section 9 and illustrates the final model structure. In section 11 the model of the created MECA-lite system to capture procedural and component knowledge as well as a design decision will be described. Section 12 focuses on the possible solutions for the IOSS and proposes a suitable expert system approach before in section 13 the project is validated against the goal, challenges and requirements, followed by the conclusions in section 0.

## 1.1 General descriptions

### 1.1.1 ISS - International Space Station

The International Space Station (ISS) (Lyndon, 1998) is a low earth orbit (LEO, Perigee-413km, Apogee-418km) space station. It is the ninth space station to be inhabited by crews and has a modular structure whose first component was launched in 1998 and is currently the largest artificial body in earth orbit. It is a joined project of NASA (US – National Aeronautics and Space Administration), Roskosmos (Russian Federal Space Agency), ESA (European Space Agency), CSA (Canadian Space Agency) and JAXA (Japan Aerospace Exploration Agency). It serves as a microgravity and space environment research laboratory in which crew members conduct experiments in biology, human physiology, physics, astronomy, meteorology and other research areas. It is also suited for the testing of spacecraft systems and equipment required for missions to e.g. Moon or Mars.

### 1.1.2 Columbus - European Module of ISS

The Columbus module, created by ESA, is a science laboratory that is part of the ISS. The activities in the lab are controlled by the Col-CC (Columbus Control Center) and USOCs (User Support Operation Centers) throughout Europe. It can accommodate ten active ISPR (International Standard Payload Racks) for scientific payloads.

### 1.1.3 KUBIK

KUBIK (ESA, KUBIK Incubator inside the European Drawer Rack) is a small controlled-temperature incubator or cooler used to study biological samples in a microgravity environment located within the Columbus module. It is equipped with removable inserts designed for self-contained, automatic experiments using seeds, cells, and small animals.

There are no data or command communication possibilities between the experiments and Kubik. Therefore, the experiment hardware design must include automatic operation. Alternatively it is possible to use manually operated experiment hardware, which the crew removes from the incubator for operations. The design allows for easy mounting and dismounting by the crewmembers. Kubik operates as a standalone facility in Columbus or in the Kubik Interface Drawer (KID) in the European Drawer Rack (EDR) in Columbus.

## 2 Methodology (thesis)

*Think? Why think! We have computers to do that for us  
– Jean Rostand*

At the beginning, time was spent on getting familiar with the technical dependencies related to the topic. This included the reading of several ESA internal technical notes and manuals related to space domain as well as scientific publications related to this work. After the close relation between the KII and MECA project was identified, it was decided to focus on results that could be used for the MECA project in the future and to validate as well as extend initial results from the KII project. The utmost goal that was decided upon was to create a closed loop prototype of the whole MECA project in small scale, focusing on knowledge assisting properties of the IOSS for procedural tasks. That would have included:

1. Create a Domain specific ontology restricted in its knowledge, containing only knowledge related to installation / repair / maintenance of the hardware
2. Make a link between this and the existing ontology, i.e. link the created ontologies (from step1) together with the already existing (not complete) ontology created as a prototype for the KII project (KII glue) and if necessary to a database as well.
3. Create an ePartner for troubleshooting i.e. an expert system (ePartner / IOSS) that uses the knowledge, created in step 1 and 2. in order to assist the astronaut (user) in the settled scenarios.

3.1 Simulate (or Wizard of Oz) an astronaut`s taskload, i.e. including the cognitive taskload model used in MECA in a limited version using for example commercial available EEG devices

4. Let the ePartner create/update procedure documentation since the purpose of the ePartner is also to create new knowledge and/or update existing knowledge. So if it finds a possible solution and it works out, it should not only update the existing documentation but do it in a way that it is 'easily' understandable for the (future) user.

The focus of steps 1-4 was intended to be the infrastructure of the whole system, before the creation of the steps itself. That includes interface definitions, decision on the model and expert system to use as well as evaluation possibilities and thoughts about how the process can be logged (to increase the gaining of new knowledge).

After three months, the original planning had to be adjusted. Experience due to personal investigation, many discussions at ESTEC and the ESA-Astronaut Trainers at EAC (European Astronaut Centre at Cologne/Germany) indicated that step 3 could not be created in time (in a practical manner) and with this also step 4 (since it depends on step 3). But the investigations also lead to a solution that would make step 4 obsolete and even simplify the intended solution. Therefore it was decided that these two steps would be tackled on a theoretical basis, but still demonstrating a conceptual solution to be further analysed and realized in a separate thesis or industrial contract. This also led to the decision to drop step 2, since the existing KFO prototype contains (mainly) physical location information which for a conceptual IOSS are completely unnecessary. Also the investigation showed that a linkage would theoretically be incorporated very easy through a shared knowledge root. Another reason for this decision was the re-use requirement imposed on the ontology which could not be met in case of the initial planning.

During the course of the thesis adjustment to the initial or changed planning were several times required because of missing knowledge related to the space domain which was not anticipated in the beginning as well as some tool related issues (see conclusion). The working structure chosen at the end, is described in section 6.1.

### 3 Related work and background research

This section describes other projects that are related in general to the issue and / or gave additional knowledge to the subject matter. If not documented differently, all described projects were created (and contracted) by ESA in cooperation with TNO, S&T, OK-Systems and EADS-Astrium.

#### 3.1 MECA

The Mission Execution Crew Assistant (MECA) (Neerincx, Smets, Breebaart, Soler, Plassmeier, & van Diggelen, 2013) project comprises crew support acting in a ubiquitous computing environment as an “electronic partner” (ePartner), helping the crew to take care of their mental and social conditions, to train and schedule tasks during nominal and off-nominal situations, and to enhance the shared situation awareness, sense making, and problem solving processes during operations.

The current objective is to prepare an Experiment Science Requirements document for a Columbus on-board experiment based on the situated Cognitive Engineering (sCE) methodology (Neerincx M. , 2011), focusing on a ‘simple’ prototype that contains the first feasible set of MECA core functions.

The conceptual prototype created in this document (see sections 10, 11, 12) could be considered as a lite version of a first fully functional MECA prototype. In the lite version the mental and social conditions will only take a minor role, while the focus will be on problem solving aspects during operations.

#### 3.2 KII

In the ESA research project KII (Knowledge Infrastructure for Interoperability in mixed Human-Machine teams) conducted by S&T (with GTI6) an investigation (S&T, KII - TN1: Scenarios and Use-cases, 2012) (S&T, KII - TN2: Formats and Formalisms, 2012) (S&T, KII - TN3: Design Considerations, 2012) was made into the subjects of the Knowledge Representation and Knowledge management in the context of space operations.

The intention of the project was to investigate and evaluate the possibilities and needs of knowledge management in a multi-actor environment. Space operations have a high complexity and area difficult to manage, even in nominal cases, let alone off-nominal situations due to the distributed nature of the knowledge. Some information is stored in one of the many ODF procedure documentations in the space craft, others can just be gained through communication with the MCC (Mission Control Center) and then there are also the situations were in an unexpected off-nominal case the MCC has to contact the contractor to retrieve the required knowledge. We can expect that for future manned (deep) space missions this complexity will only increase, leading to an increased requirement for autonomy for the onboard crew.

It is therefore crucial to have the support of a KII that will enable tools to provide the right information at the right time to the right person.

The focus of the KII project was on investigating the considerations and requirements imposed by team-based space operations problem-solving context, on identifying relevant use cases and scenarios, and on proposing a suitable formalism (section 4.2.4) for Knowledge Interoperability and Representation. This formalism was then applied in practice to build an example knowledge base (KFO), showing how the information present in this

knowledge base could be used to add value to a human-machine team working on solving common problems.

One of the outcomes of the KII project was a strong recommendation for a high-level, open-world 'modelling view' of knowledge representation, with the example Knowledge Base successfully being built using OWL ontologies as modelling representation formalism.

Another outcome was the recommendation for a loosely coupled architecture highly dependent on well-described interfaces to make it possible to interoperate with as wide a variety of secondary data sources and external applications as possible.

The KII project can be seen as another practical validation of the design choice in MECA to require a cognitive architecture supported by an ontology-based Knowledge Base.

### **3.3 CRUISE**

The CRUISE (Crew User Interface System Enhancements) project (CRUISE, 2013) had the objective of investigating advanced concepts for executing crew procedures in the Columbus module of the ISS. This was investigated using two experiments:

Cruise PD (Procedural Displays) – A procedure executor with integrated command and control capabilities, developed as an extension to the current operational ESA Columbus system laptop application software.

Cruise vaPV (Voice Activated Procedure Viewer) – A procedure viewer with support for voice controlled procedure execution (using a headset as interface).

A key lesson learned was that onboard experiments require detailed planning as well as intense coordination with ESA and NASA that exceeded initial expectations (e.g. crew time is a very scarce resource). Also while crew members appreciated the use of Cruise PD saying the keyboard input is suitable for the microgravity environment and leaving just a few improvement suggestions, the use of Cruise vaPV (imagine it to be a headset forwarding input to a voice recognition entity) was unexpectedly troublesome (however still considered a success). Small side communication with ground control would throw the system off as well as the (on earth) standard case of using a headset became troublesome on the ISS because a floating headset cord was often found to be distracting.

Other lessons learned were that crew medical data are sensitive information. Questionnaires about emotional state are considered medical data. Also if at all possible, the use of hardware not already on board should be avoided since the environment will probably pose challenges that went undetected during experiment preparation and crew training on ground. Last but not least the crew training is important, in particular for new user interface concepts or unfamiliar ways of working, and should be complemented with on-board refresher training.

### **3.4 iPVTouch**

The project iPVTouch was a small study conducted by OK-Systems in 2011 (Neerincx, Smets, Breebaart, Soler, Plassmeier, & van Diggelen, 2013) for ESA/ESTEC with the goal to gain knowledge on how to use and validate ODF procedures on multi-touch tablets.

The results of the iPVTouch study showed that multi-touch devices (smartphones and tablets) are feasible and useful platforms for on-board crew support tools, and that with relatively small development effort new applications could be developed and deployed in

distributed environments where the ePartner could run in stand-alone mode or connected to other devices.

The results go hand in hand with astronauts feedback concerning Cruise PD usability as well as controllability.

### 3.5 SICRA

OK-Systems and S&T contributed to the SICRA (Intelligent Avalanche Collaborative Rescue System) project (Neerinx, Smets, Breebaart, Soler, Plassmeier, & van Diggelen, 2013), a procedure ontology which was intended to be integrated in an intelligent assistance for rescue teams aiming to help in rescue tasks of avalanche victims. The assistant was designed to be installed in smartphones interfacing a specialized SICRA device combining GPS receiver with sensors which should locate victims sharing the positions of victims and rescuers.

## 4 Background knowledge

*The single biggest problem in communication is the illusion that it has taken place*  
– George Bernard Shaw

A good example why we should model even ‘common knowledge’ in a proper model is the Mars Climate Orbiter which burned up in the Martian atmosphere 1999. The problem was not an error in the hardware or software itself but rather a miscommunication between two teams. One team was working with USA customary units and the other using the metric system. Since however this fact was not known to the teams no conversion for the numbers was included in the software, turning the 125 million Dollar equipment into ashes. This is quite evident; someone having the weight of 180 in the metric system is definitely overweight with his 180kg. However 180 pounds (~81.6 kg) would not shock anyone. What was missing for both teams was a ‘common’ information base, the meaning or semantics of the used data since a computer will not judge if the value seems to be unfitting!

*To err is human, but to really foul things up you need a computer*  
– Paul R. Ehrlich

### 4.1 Basics

The reader shall get acquainted with general definitions of data, information and knowledge (section 4.1.1). Understand space system databases (section 4.1.2) with the focus on different modelling levels when being confronted with development of a database. The implication of object-role modelling (section 4.1.3) is being described, added by the different approaches of possible dialogue systems (section 4.1.4).

#### 4.1.1 Data, Information and Knowledge?

'Data' on its own carries no meaning. An example would be:

*"23", "14653", "Alphasat"*

In order for 'data' to become 'information'<sup>3</sup>, it must be interpretable, that means extended with semantics (meaning the conceptual data model) e.g. "what the nouns refer to and what the verbs mean". To continue the above example:

*"Alphasat" refers to a communication satellite  
"23" is referring to the application process identification (APID)  
"14653" is referring to the sequence number of the APID*

Knowledge is when we know everything about the validation associated with the information.

*Alphasat is a communication satellite that is running application. The application with the identification 23 sends Telemetry packets. The packets are sent in a sequence, where the packet with the sequence number 14653 has a wrong checksum*

Only when we understand (have the knowledge) of what kind of raw data we are talking about, have identified the information relevant to these data and therefore also their meaning we can see/identify the context and application to form our knowledge. This knowledge then can be transferred/interpreted as 'wisdom' in the context of the domain.

The described process of identifying the knowledge and deriving at wisdom is often referred to as the DIKW (Data-Information-Knowledge-Wisdom) hierarchy or knowledge pyramid (Bernstein, 2009) as illustrated in Figure 1. While this hierarchy is receiving critiques for several aspects like e.g. that it is levelled and less wisdom is present (due to its pyramidal form) at the end than data in the beginning it serves here as a good illustration of the problem at hand. The astronaut just needs the wisdom (the actual action to perform) while there is a huge amount of data, information and knowledge present at all times!

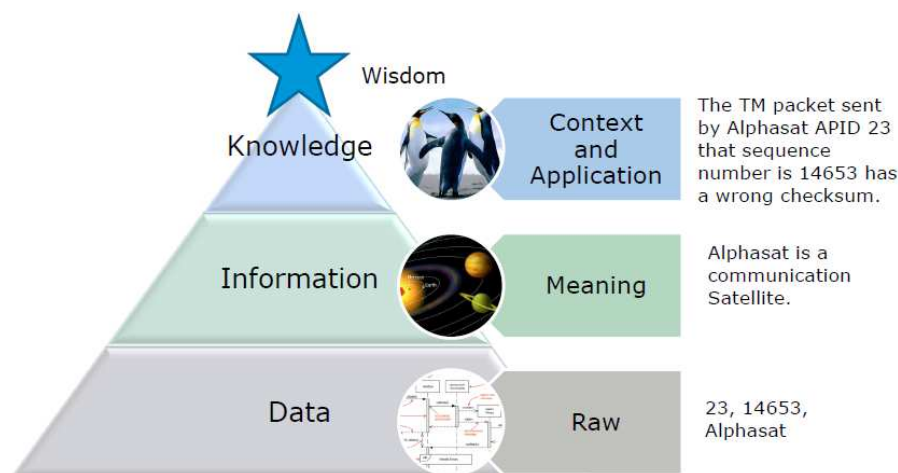


Figure 1: From collection of information to knowledge TM (Telemetry) APID (Application process Identification)

<sup>3</sup> There is no relationship to the 'Information theory' here. Information should be understood literally (no statistics involved).

#### 4.1.2 Space System Databases

*The most important thing in communication is to hear what isn't being said*  
– Peter Drucker

One of the big mistakes of databases (DB) is the aspect of making assumptions. Any assumption (by definition “not yet validated”) has a recovery cost that might increase “exponentially” with the time taken to acknowledge that it was a wrong assumption.

What is a database? In short, it is a software system that includes a repository (i.e. the physical layer) where data is structurally organised according to a technology dependent data model (i.e. the DB logical layer). Being conceptual, a model can be shared among different software agents that use different logical schemata. Hence, “interoperability” is achieved. Data models (Abiteboul, Hull, & V., 1995) are compliant software functions that implement means for manipulating data including man machine interfaces (e.g. editors, viewers, report generators), external interfaces (e.g. import/export), quality analysers (e.g. consistency checking) and version control.

Data modelling is context dependent. From a database software development viewpoint:

- What:
  - Modelling domain at Conceptual level – describing the semantics “the What” of the domain to be handled by the database → Requirements
- How:
  - Modelling data at Logical level – transforming the semantics into a particular data manipulation technology (relational, hierarchical, object oriented, network, etc.) → Design
  - Modelling data at Physical level – describing the physical means by which data are stored → Coding
  - From a database population viewpoint modelling data for suiting user needs → Utilization

One advantage of the conceptual level is that it is the most stable of all levels. It is unaffected by changes in user interfaces or storage and access techniques. One disadvantage however is, if you make a mistake and you are required to change it, the impact is massive.

So when creating DB user specification we should focus on the conceptual data model to create a full specification of all data requirements. That means identifying 100%<sup>4</sup> (ISOTR9007, 1987) of the ‘What’ and eliminating, as much as possible, the ‘How’. How data is grouped into a structure (e.g. attribute-based entity types, classes, XML schemas) is an implementation issue that is irrelevant to the capturing of essential “business” semantics. Avoiding ‘How’ enhances semantic stability, as well as facilitating natural verbalization and more productive communication with all stakeholders.

---

<sup>4</sup> But only for the given domain problem. Usually we don't need to identify 100% of everything, since the nature of modelling is to simplify reality in a convenient way and therefore avoiding the need to identify everything. However for the space domain these simplifications can lead to major issues!



The conceptual model is the basis for the various implementations required to develop the data repository but also the man machine interfaces, the import/ export interfaces etc. Tailoring for each stakeholder can also be derived from conceptual models.

Popular conceptual modelling languages include Entity Relationship Model (ER) (Chen, 1976), Conceptual Graph (CG)(Sowa, 1984), Unified Modelling Language (UML<sup>5</sup>), SysML<sup>6</sup>, Fact Based Modelling language (FBM<sup>7</sup>), Web Ontology Language (OWL<sup>8</sup>).

Most languages have been (or are being) standardized. In particular, FBM is an ongoing international standard that has been initialized by ESA. It is a methodology for modelling the semantics of a subject area (i.e. the universe of discourse (UoD), the domain of interest). The roots of FBM are traced to research and application in business of semantic modelling for information systems during the 1970s. Subsequently, several developments have taken place in parallel, resulting in several FBM dialects (including NIAM, ORM2, CogNIAM, DOGMA, and FCO-IM).

Although these dialects are developed for different fields using various techniques, they comply with a few fundamental FBM principles, such as, using controlled natural languages to verbalize models in order to enhance the communication between domain experts and modellers. As such, if you are not a data modeller expert, specifying your data requirements usually requires the use of natural language (e.g. plain English) in a “controlled way” (e.g. Simplified Technical English). Other important FBM principles include the 100% principle (as discussed earlier), fact based (not assumption-based) and support of graphical notations.

Developing Ontology Grounded Methods and Applications (DOGMA) (Meersman, 1999) (Spyns, Tang, & Meersman, 2008) is an FBM dialect for ontology engineering. It uses the graphical notations from ORM, the background will be discussed in the following section and section 4.3, for modelling domain ontologies.

#### 4.1.3 Object-Role Modelling

Object-Role Modelling (ORM) (Halpin, Object-Role Modelling, 1998) (Halpin & Morgan, Information modeling and Relational Databases: From Conceptual Analysis to Logical Design, 2008) is an information modelling technique that views the world as simple facts and is primary used for modelling and querying an information system at the conceptual level. Simple facts describe the objects and their corresponding roles and are one of many ‘dialects’ of Fact Base Modelling (FBM). A simple example would be you, who is right now performing the role of reading, while this thesis is playing the role of being read. It is used to model the universe of discourse of a business<sup>9</sup> domain.

---

<sup>5</sup> <http://www.uml.org/>

<sup>6</sup> <http://www.sysml.org/>


<sup>7</sup> <http://www.factbasedmodeling.org>

<sup>8</sup> <http://www.w3.org/TR/owl-ref/>

<sup>9</sup> I will refer to a business domain which is as well suitable to the space domain, however keep in mind that the space domain has some unique ‘business’ restrictions/ differences which however are not of importance at this point for the theory.

To illustrate the differences between ORM and Entity Relationship(ER) or other Object Oriented (OO) approaches like e.g. UML. ER and OO make explicit use of attributes. Let us consider the example of the database entry *Person* which has an attribute *countryBorn*. With ORM we would not use it as an attribute but model the relationship 'Person was born in Country' making it more stable since attributes may evolve into entities or relationships. In the presented database example we would not be able to record the population of a country without reformulating it as a relationship.

Common used 'versions' of FBM e.g. are the Semantic of Business Vocabulary and Business Rules (SBVR), developed in 2007 by the Object Management Group.

The used version of FBM in this thesis is a fact-oriented description known as second generation of the, in 1970 introduced Object-Role Modelling (ORM 2), and is based on extensions to NIAM<sup>10</sup> (Natural-language Information Analysis). Unless indicated otherwise, in this thesis the term ORM is understood to mean ORM 2. The major difference between version one and two is the change of graphical notations to decouple the modelling approach from the English language and make it more natural for non-English speakers (Halpin, ORM 2 Graphical notation, 2005). While in version one constraints would be illustrated with characters the second version uses standardized symbols (i.e. instead of 'ir', which would stand for 'irreflexive' the symbol  is used.). Also minor changes were done to the graphical representation to make it not as space consuming as version one, hence improving the overview as well as comprehensibility.

Unlike Entity-Relationship (ER) diagrams and Unified Modelling Language (UML) class diagrams ORM treats all elementary facts as relationships, thus regarding decisions for grouping facts into structures as implementation concerns irrelevant to business semantics.

An advantage of ER and UML is that they are good for compact summaries and that their model is closer to a final implementation. However an advantage of FBM it the availability to generate those models from the ORM model (Valera & et.al., Fact Based Modelling: Exchanging Conceptual Data Models, 2011). Hence supporting the advantages of ORM like e.g. the easy verbalization as well as population for validation with domain experts or the fact that those models are more stable under changes to the business domain as well as the higher number of business rules captured in diagram form. While not losing any of the ER and UML (closer to implementation) advantages. The only things that cannot be derived are model specific processes like e.g. the UML sequence diagrams.

Note that from this point onwards FBM and ORM may be used interchangeable, always referring to the FBM methodology using ORM for data modelling.

#### 4.1.4 Possible Dialogue systems

In general we can distinguish between three different approaches to a dialog system that is used to communicate between human and machine (Vergunst, 2011)

- Keyword based: It is very easy to create but very limited in the sense of applying knowledge. It consists of a set of rules that triggers, depending on words used within the communication.

---

<sup>10</sup> ORM is in Europe often addressed under the NIAM name

- State based: As the Keyword based system, it also is triggered by certain key words but has multiple predefined states it can enter. From a state you can only go to pre-specified other states, limiting the keywords in each state. This approach is good when the system should follow predefined patterns.
- Agent based: It is very difficult to model, but goal oriented like a user. Giving it the benefits of the State based system, while increasing the flexibility. It also allows the conversation to be pro-active since the approaching goal removes the passive waiting for keywords to trigger actions/states.

A state based system could handle most of the requirements for MECA light, except the pro-active requirement. For simple procedural handling this is not a problem but requires the use of keywords and reduces the flexibility of the IOSS. With an Agent based approach however it would be possible to program it with general task-oriented plans that can be used in multiple domains. Another downside for the State based system would be off-nominal cases in which we may not be able to generate or predefine states to transition to.

## 4.2 Domain related

In this section we will take a closer look at the objective of this thesis. I will start with describing the working field that has to be modelled (section 4.2.1) followed by a description of different procedural formats existing (section 4.2.2 , 4.2.3). Followed by an evaluation of formalisms examined in the KII project (section 4.2.4). Then I will illustrate problems and drawbacks of the current approach leading to my choice of ORM as suitable formalism approach (4.2.5).

### 4.2.1 What to model?

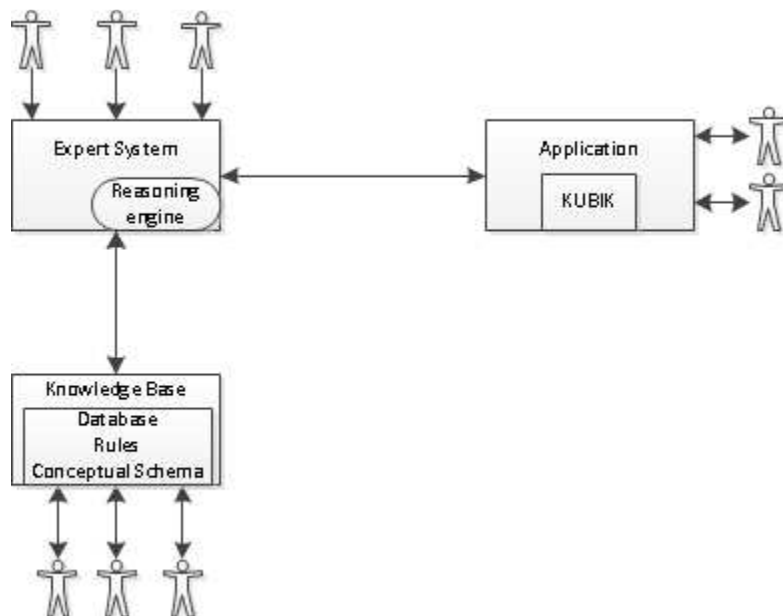


Figure 2: Illustration of the different working fields and their relations

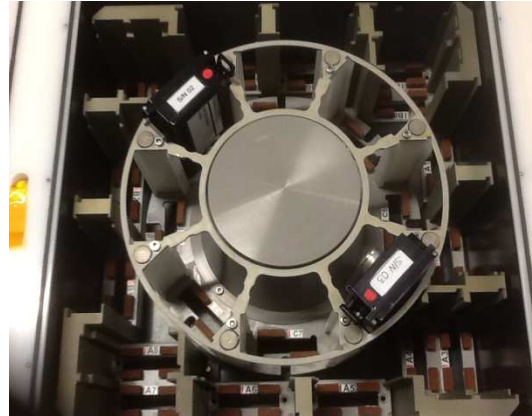
The goal in Figure 2 is to have an expert system (upper left) that is connected to a knowledge base (lower left), containing databases, conceptual schema, rules etc. Inside the expert system you have some reasoning engine which is interlinked with the dedicated experiment called KUBIK (ESA, KUBIK Incubator inside the European Drawer Rack), in general called 'application field (upper right)'. The application field in this thesis is KUBIK but

the knowledge base should be generic enough to also work with other application fields since from an architectural point (knowledge management) of view it is always ideal not to assume to have just one application.

In the end there will be the KUBIK end users. There will also be domain experts to feed the database system, conceptual scheme, knowledge base. The last group of 'users' will design the expert system. That means even here we have already three different communities using and 'maintaining' the knowledge.



**Figure 3: Different kinds of experiment containers. The lower right side illustrates an experiment that has to be 'activated'.**



**Figure 4: The Centrifuge Insert of Kubik with two experiment containers inserted (Note: the containers are both different)**

What is intended to be modelled (in my model) is procedural knowledge (4.2.2) for the KUBIK module. i.e. how to set up the temperature or start an experiment. This includes the whole set-up of an experiment from beginning to the end as it is defined for the International Space Station (ISS). Starting with taking the experiments (Figure 3), if necessary activate<sup>11</sup> them and put them into the centrifuge (Figure 4). Note that the Centrifuge Insert (CI) is not the only available configuration to run experiments for KUBIK. There are three more configurations: Passive Insert (PI), Rack Insert (RI) and special oversized experiments. A special experiment refers here to an experiment that is too big to put in any of the three mentioned inserts (CI, PI, RI) and has to replace those using the Kubik Interface Plate (KIP) to run within the box. In general KUBIK is very low on the diversity (quite operational) of possible actions and has except the mentioned difference in inserts a straight forward configuration and usability. However all of the experiment setups have to follow well defined rules (e.g. all experiments in a centrifuge have to be balanced). From this point onward though when we refer to KUBIK we assume a KUBIK setup using the CI and 'normal' experiments (not to be activated) if not indicated differently.

The focus of this paper will be on the knowledge base, keeping in mind that it has to be used by an expert system and humans, therefore providing conceptual consideration/illustrations. The interface between machine and human will be kept untouched (in a practical manner). It will be assumed that the interface will look in a specific way (e.g. Cruise PD or Cruise vaPV)

<sup>11</sup> Some experiments are not delivered in a 'ready' state. Figure 3 illustrated experiments that are ready (left side of the image) and an example experiment that has to be activated (lower right side). Activated here refers to that the experiment has to be put in a specific moulding tool before the astronaut has to apply pressure allowing the experiment internal substances to react with each other.

which could be a generated button interface using the FAMOUS (Valera & et.al., FAMOUS Statement of Work: Fact based Modelling Unifying System: Toward implementing solution for ECSS-E-TM-10-23A, 2012) tool or something else, since the interface or the expert system cannot be used in a proper way unless it is provided with the knowledge that will be necessary and represented.

#### **4.2.2 Procedure in ODF format**

In any space mission the astronauts can only be informed with a certain amount of knowledge related to the spacecraft and its equipment. It cannot be expected of him/her to know how each component works and how each configuration step (and its related actions) has to be performed by heart. Currently astronauts are therefore given procedural checklists describing step by step each action they have to perform to achieve the targeted result. Sometimes astronauts even have no knowledge about (handling) new equipment that is delivered to them other than those procedures.

The procedure documents are created by domain experts before the equipment is launched and/or delivered to the astronauts and is created by translations of contractors delivered operation manuals into Operation Data Files (ODF) as well as training material<sup>12</sup>. While the training material is usually just a small PowerPoint presentation with a few slides describing the new equipment following no standards, the ODF procedures follow a certain standard. This standard includes fixed symbols for certain actions, overview of required equipment, number of crewmembers required for the procedure, expected time needed, a vast amount of acronyms and much more. Those procedures are so precise and elaborate (e.g. using illustrations to support the description) that they can be performed by anyone who is familiar with the standard without having any pre-knowledge about the equipment, actions or result of said procedure.

Currently each action an astronaut performs on the ISS is based on following a procedure or instruction from the MCC. The delivered operation manuals usually also contain a certain foreseeable amount of off-nominal cases for which they already deliver troubleshooting instructions. These foreseeable cases however are not the only ones that can arise during the missions. If therefore a case appears that was not delivered to the astronaut (not translated or included in any procedure) he has to stop working until MCC figures out what is the problem and presents him a solution step by step.

Another problem with the procedures is sometimes that the operation manual that is delivered to the domain expert is not in a final version or the equipment was modified and the operations manual was not updated (or the procedures were not updated).

---

<sup>12</sup> Training material is used for astronauts that know they have to work with a certain equipment and are not yet in space (e.g. stationed at ISS) and/or serve as a small overview to get to know the components general purpose and functioning.

### 4.2.3 KUBIK procedure overview

The KUBIK operations manual (Aerospace, 2007) is somewhat an exception of all investigated manuals in the context of this thesis. Its structure already closely resembles a ODF procedural step by step description for achieving objectives while many other manuals are more complex because side conditions have to be checked and more in depth configurations (hard- and software) have to be performed.

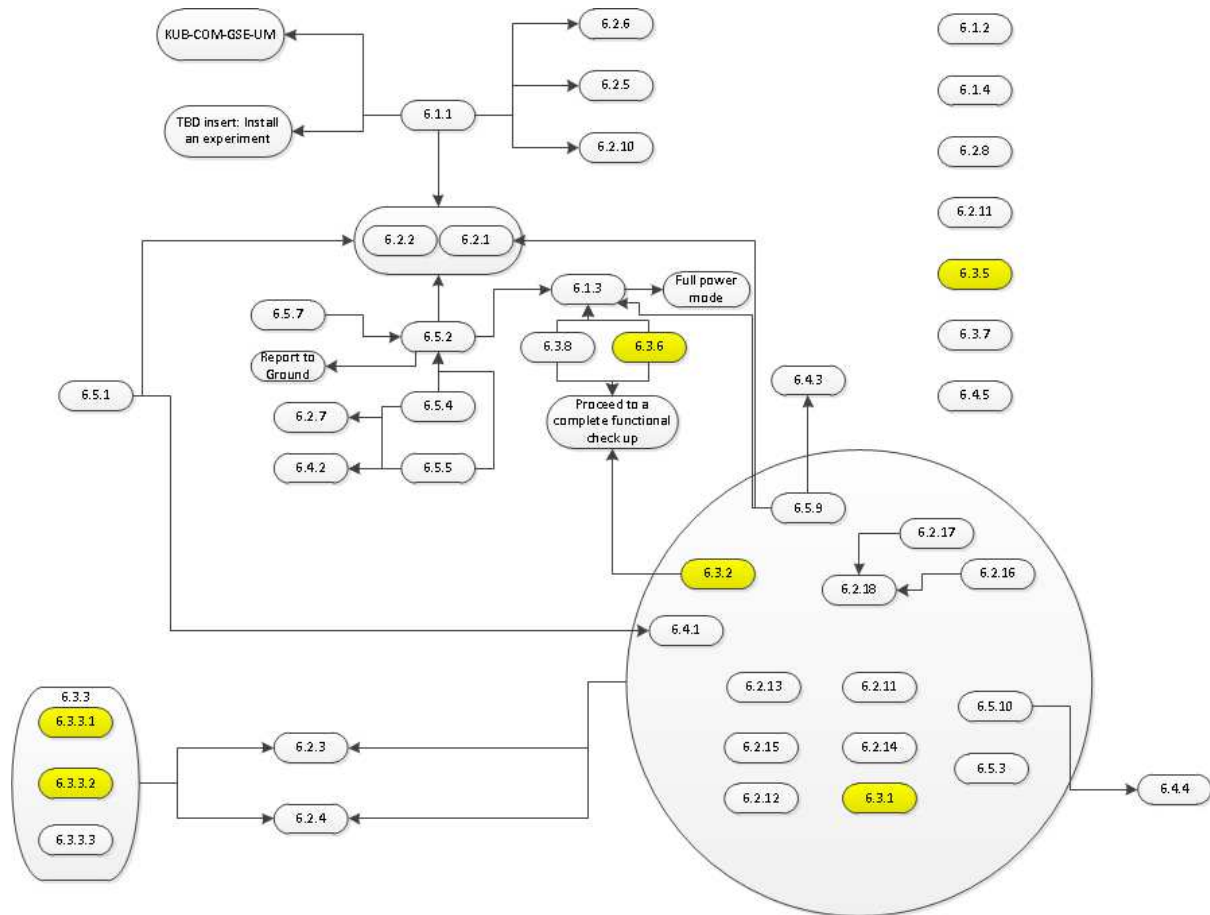


Figure 5: Overview of the in the KUBIK operation manual described procedures and their dependencies with each other

Figure 5 illustrates all procedures<sup>13</sup>, described in the KUBIK operation manual. Except seven (upper right side), all have a linkage to at least one other procedure. 16 are just being linked to but don't link themselves to another procedure. While some have a linkage that is only circumstantial (if condition true follow x.x.x else x.y.x) others have fixed linkages. The most used and elaborated example would be the procedures 6.2.3 and 6.2.4 which correspond to opening and closing the KUBIK lid and is referenced by most procedures. As an example how this is finally translated to ODF procedures take a closer look at the yellow marked rounded rectangles in Figure 5. All marked instances together form the ODF procedure 1.101 (KUBIK Setup).

The procedures generated in ODF that are handed out to the astronauts have their own visualisation and representation standards but correspond/are equal to actions within the

<sup>13</sup> Note that I refer here to the procedure of the KUBIK manual which doesn't follow the ODF standard meaning that they cannot be used before modifying/ adding additional knowledge.

operation manual and are often just rewritings of these illustrated linkages to fit them in one procedure (one document) as a whole. Note that the course of actions for a few procedures is not defined like e.g. 'Proceed to a complete functional check' which can change depending on the current configuration and is translated in multiple ODF procedures.

Also note that the operation manual procedures only relate to the equipment and don't include any environmental actions. A good example would be that the manual just states what kind of power input is required and where the cable has to be connected to, while the ODF procedure includes a specific rack and power connection in the Columbus module.

However since the operation manual has less duplications than the collection of ODF procedures related for KUBIK, it was decided that the ontology concerning the equipment will be created mainly using the operation manual as a source of knowledge input. Not losing the property of ODF generation since the KUBIK operation manual is provided in ODF similar form.

#### 4.2.4 Formalism

The target, as already mentioned several times by now is to create a knowledge base that contains all mission relevant information which are currently provided using ODF procedures with focusing especially on off-nominal (troubleshooting) cases (including not documented cases). Typically knowledge bases can be divided into three groups:

- Machine-readable knowledge bases store the knowledge in a 'pure' computer-readable form where the data are stored in form of rules which describe the knowledge in a logical manner.
- Human-readable knowledge bases like the described operation manuals or ODF procedures. Usually supported by a search function to allow fast access to the required knowledge.
- Text-based knowledge bases like e.g. Wikipedia<sup>14</sup>. These knowledge bases allow using hyperlinks for cross-references and granting several user accesses at the same time.

Note that none of the above groups fits the requirement of using the knowledge base by both human and machine. The KII project (Section 3.2) came to the conclusion that the best solution would be an ontology to represent knowledge as indicated in the introduction. The rest of this section will be used to illustrate a short summary of these results and review on made decisions. (see finalised evaluation according to Table 1)

In the KII project four existing formalism (F1-F4) (S&T, KII - TN2: Formats and Formalisms, 2012) were examined in depth using the following parameters for validation: Expressivity, Coverage, Understandable by human, Consistency, Efficiency, Easiness for modifying and updating, Support the intelligent activity which uses the knowledge base.

---

<sup>14</sup> [www.wikipedia.org](http://www.wikipedia.org)

### **F1 Production system**

Productions consist of two parts: a precondition (IF) followed by action (THEN). These rules are contained within a database, sometimes also referred to as working memory, which contain data about the current state / knowledge and serve as a rule interpreter. It works using either forward (generate new knowledge or react to environment) or backward (support decision making) chaining. The effectiveness of those systems depends on the pattern-matching algorithm used to collect production rules with matched conditions (e.g. a basic approach is to store all rules in a simple list, accessing them in storage order).

### **F2 Semantic network**

A semantic network represents semantic relations between concepts. It is a directed or undirected graph consisting of vertices, which represent concepts, and edges, which represent relations (e.g. 'is a' and 'is an' as in 'Cat is a Mammal is an Animal'). The most known representation of semantic network is LISP.

### **F3 Neural network**

A neural network<sup>15</sup> intends to imitate to some level a biological neural network. They are composed of small processing blocks (artificial neurons) which have input and outputs. These blocks are connected to each other forming several layers (usually an input layer and an output layer with optional hidden layers in between). This results in having the output of one layer being the input of another layer (for hidden layers and the output layer). The input layer works as the neural input and the output layer as the networks output. Once a neural network is created it has to be trained (learn) using supervised (we know the result and can use this for validation of the network) or unsupervised (we only know a cost function, which is used to calculate the distance of the output to the 'ideal' result) learning. Once the network has learned it is very robust in tasks like pattern finding and non-linear decision making.

### **F4 Ontology**

An ontology is used to represent knowledge as a set of concepts within a domain and the relationships between those concepts. It is used to describe entities within a domain allowing to reason about those entities or as a description of the domain. They form a structural framework for organizing information about the world or some parts of it. In an ontology domain objects are represented as attributes, and classes individuals and relationships are represented as relations, restrictions and rules. Even though the graphical representation of an ontology looks like a semantic network they are not the same. Semantic networks don't describe attributes, have no hierarchical structure (usually missing of root elements and having more points of concentration) and don't define the type of relation between objects (without having an actual instance of it). Since ontologies tend to cover a wide domain they can be differentiated in so-called domain and upper ontologies. A domain ontology represents a specific part of the world and upper ontology ontologies represent common objects which are applicable across a wide range of domain ontologies.

---

<sup>15</sup> Artificial Neural Network



Note, that some commonly used formalism (e.g. HTML and XML) were not included since they lack specific features<sup>16</sup> to store knowledge specifically and missing interfaces to reasoning systems.

The results of the in depth examination of the four formalisms were then checked against three groups of requirements to the KII foundation KB:

- Static content requirements which define real world objects that do not change (except for smaller updates) during the lifetime of the KB. KUBIK e.g. would be one of those 'static' elements since its functionality would not change significantly. Maybe a new insert could be introduced at some point, but the functionality of the device would not be affected.
- Dynamic content requirements on the contrary could change quickly. A good example would be the location property for physical objects like e.g. satellites which changes very quickly (within the respective orbit).
- Reasoning support requirements define the ability of the KB to assist reasoning. This could e.g. be a comparison of values provided by several sources, while this task can be handled by either the KB (internally) or an external application which uses the KB.

The results of the investigation are summarized in the following table (direct copy from(S&T, KII - TN2: Formats and Formalisms, 2012))

Requirement groups	Productions	Semantic networks	Neural networks	Ontology
<b>Static content</b>	Can be presented as a combination of values in left side of productions	Good representation of concepts relationships, poor representation of attributes	Each neural network can present only a small domain area	Natively supported
<b>Dynamic content</b>	Needs specific extensions to reflect dynamic content	Very poor	Available in form of learning operations, though it is a balance between reactivity and accuracy	Supported, but adding to many individuals leads to loss of effectiveness
<b>Reasoning support</b>	Natively supported	No straightforward support	Reactions of neural network can be treated as fuzzy logic production	Supported

Table 1: KII formalism summary

Semantic and Neural networks both seem not to be suitable as formalism while neural networks however may find some use as implementations for fuzzy<sup>17</sup> logic reasoner for reasoning support.

<sup>16</sup> We are e.g. required to first write an interpreter to translate the knowledge contained in those formalism.

Interesting is however that the production system and ontologies, seem to complement each other. This leads to the choice of using a combination of both for the KII foundation KB. The ontology was used to represent static content. This ontology would be extended with a database in which multiple instances of the same class would be stored and attributes of permanently present object(s) which change frequently. It would also be extended with a production base, which covers needs in deductive and/or inductive reasoning.

Note that the choices made were about the knowledge representation and not the data structure or an exact format. The final results of the project were:

- OWL language is used for the ontology description
- Relational DB is used as the correspondent DB
- RuleML<sup>18</sup> (RuleML, 2013) language is used for productions

I however decided that OWL was not the best choice (see 4.2.5), agreeing however that a DB would be required for storing dynamic content and certain special case reasoning support requirements (see section 12 - Case-based reasoner) while using the KB for most of the reasoning requirements content as well.

Also after investigation of the created KII ontology from a generic knowledge engineer point of view a few 'errors' were found. Below find suggestions for improvement:

- Sometimes instances are mistreated as classes. E.g. "EADS Astrium" is modelled as a subclass of "Contractor" while it is more likely an instance instead of a classes.
- There exist extra semantics that may not be needed. As an example PressurizedMechanical is defined as class and as subclass of [axiom 1] *providesInterfaceFor some Mobility*, [axiom 2] *providesInterfaceFor some Habitation*, and [axiom 3] *providesInterfaceFor only (Habitation or Mobility)*.

The semantics from axiom 1 and axiom 2 are however stronger than axiom 3 and axiom 3 can even be derived from them. Therefore axiom 3 could theoretically be removed without the loss of semantics.

It is okay to provide extra information, also to have axioms containing duplicated semantics but it should be avoided for computational tasks to improve the efficiency

- The naming conventions request to use singular for nouns. E.g. ""Person" not "Persons"
- Granularity of model e.g. "unmanned lander, cargo capacity less than 1 metric ton" is treated as an annotation (informal text) of "LightLanderCargo". It would however be more useful to model it formally. So it could be used to check consistency.

---

<sup>17</sup> Note that 'fuzzy logic' is a completely different thing than a neural network. But used as in the Table 1 literally.

<sup>18</sup> Rule Mark-up Language is a family of XML languages, which are serialization representation of productions

#### 4.2.5 OWL vs ORM

For the KII foundation KB it was decided to use the ISWG (Interface Standards Working Group) ontology (ESA, Support to ESA-NASA collaboration activities in preparation of Space Exploration Programme, 2008) and extend it with new hierarchies for the required knowledge while leaving most of the ISWG ontology content unchanged using the OWL 2.0 language (W3C, 2009) to describe the ontology and Protégé-OWL 4.1.0 (Protege-OWL ontology editor, 2013) as the ontology editor.

Both the tool and language are reasonable choices for handling the task at hand, but after experimenting with different ontology languages the FBM (Fact Base Modeling) in form of ORM (Object Role Modeling) using the NORMA tool seemed to be a better fit. ORM delivers the same capabilities as OWL 2.0 and is even capable (see section 4.1.3) of being translated into OWL 2.0. However it comes with a few differences which lead to the decision to switch the ontology language (while some aspects maybe just depending on the use of the tool). For the comparison we will just use OWL to refer to OWL 2.0 and Protégé and ORM to refer to ORM 2 and NORMA (NORMA for Visual Studio, 2013) :

- In ORM the visualisation of knowledge is more structured than in OWL. This makes it easier to communication between knowledge engineer and domain expert (of the knowledge to be modelled).
- ORM comes with a feature called 'verbalization' which allows to verbalize relationships between classes and individuals within the knowledge base. These verbalisation can either be just for the relationship/rule itself or for instantiations of the knowledge stored within the KB and may support the expert systems to communicate about the content of the KB with any user.
- ORM allows for easier adding of logical rules to the model and clearer visualisation of classes having rules.
- While OWL allows for good graphical overviews of the whole KB, ORM models are 'split' into different diagrams (still allowing for a complete overview) which makes it easier to separate facts and keep each part of the KB reasonable small, understandable and digestible using 'hyperlinks' between the different facts to navigate through the KB

Note that neither of the options may be the ultimate solution for the problem at hand. The proposed change is due to personal experience and conversation with various knowledge engineering experts about the problem at hand.

The ISWG ontology was also not included / linked to the ontology of this thesis since the included information were too general to be of direct use in the question if an ontology could be used to usefully store procedures. But since, as already mentioned, a transfer from ORM to OWL is possible no generated knowledge in ORM would be lost if it would be decided to not use it in the final delivery.



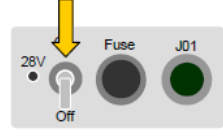
### 4.3 Description of ORM modelling

Generally modelling depends on a clear coverage of all requirements. This includes requirements of the environment, be it hardware or software. To highlight a few concepts and relationships of such a model, let us first take a look at the KUBIK operation manual procedure ‘6.1.3 – Transfer from Soyuz<sup>19</sup> to ISS-RS (Figure 6)’ and continue with a description of the iterative verification and validation process being applied to ORM modelling.

6.1.3 KUBIK: Transfer from Soyuz to ISS-RS

Requirement	Budget	Comment
Duration	20 mn	
Manpower	1 to 2 Cosmonauts	
Equipment	Configuration	Comment
FM Hardware	KUBIK plugged and on	At the beginning, KUBIK is still functioning in SOYUZ.
Additional Flight hardware	M10 Allen key ISS/SOYUZ adapter	Parts of the KUBIK Toolkit
Ground hardware	-	
Special precautions	1) Do not shock KUBIK during transport 2) Do not use the safety belt for handling KUBIK 3) Respect the air clearances when installing	

Step	Type	Action
1		Check technical condition and integrity of equipment
2		Pull then switch off KUBIK main power switch 

●  
●  
●

●  
●  
●

●  
●  
●

Figure 6: Small cut out from one of the procedures illustrated in the operation manual. Note that not all steps of the 3 page procedure are illustrated here. But the top table (Requirements, Equipment etc.) is the same for all procedures (except the included content like e.g. different ‘Special precautions’)

Let us leave aside for now the lower table (the one including steps) and focus on the upper table which is given for each procedure. Duration, Manpower and the other requirements listed there are all concepts (which are also used in ODF) so we are required to model them. We will use the FBM convention. Inside FBM we have entity type, relationships and factor type. Let us take a small example of a satellite.

A satellite is a concept, on the other side we are required to have some kind of operation system (OS) to keep the satellite running. So what kind of relationship do these two entities have? We can say a satellite has an operation system. The other way around we can say the operation system is of (one) satellite. In FBM these concepts are also called entity types. The relationship between both entity types specifies the role these two concepts play/have to each other. But that is not enough to model the whole knowledge, we also have to add constraints, meaning we have to specify what kind of population/rules we are looking for. If we e.g. want to say that each satellite has exactly one and at most one OS, we put a dot

<sup>19</sup> One of the most reliable Russian spacecraft’s launched by the Soyuz rocket. It is used to transport cosmonauts from and to the ISS.

(meaning at most one). This leads to a formal rule which says: 'Each satellite has exactly one OS' (exactly here combines 'at least one' and 'at most one').

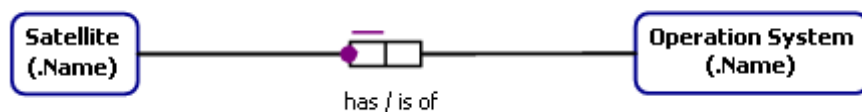


Figure 7: Illustration of the Satellite – Operation System relation. The left and right soft rectangle (rectangle with rounded corners) are entity types and the center box represents the role/constraints.

If we now populate (i.e. fill the KB with knowledge/data) this we have exactly one fact saying Satellite1 has OS1. The attempt to add another fact like Satellite1 has OS2, would let the reasoning engine tell us that this is not possible since each Satellite can just have exactly one OS and is therefore rejecting the attempt to add another OS.

Let us come back to the example procedure (Figure 6). We have a requirement which is called Duration. Consequently we add an entity type Duration and have our first concept. We can also see that the Duration has something given by 20 mn. This Duration has a value 'DurationValue' which is given in the example by 20 minutes. This duration is related to a task (6.1.3) since every concept has to be properly defined. In this case, associate it with the scenarios/ procedures from KUBIK. And this scenario has as well a name 'Transfer from Soyus to ISS-RS'.

So if we have these instances we can lift them up to a model like:

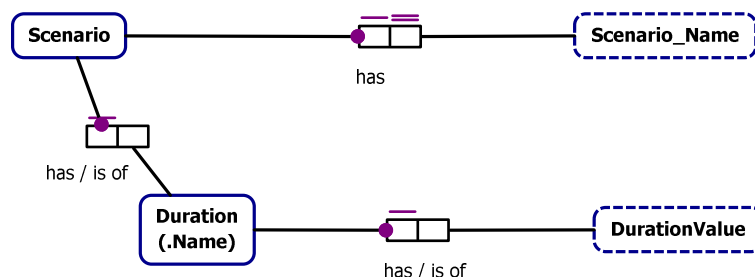


Figure 8: Each Scenario has exactly one Scenario\_Name. For each Scenario\_Name, at most one Scenario has that Scenario\_Name. This association with Scenario\_Name provides the preferred identification scheme for Scenario. Each Scenario has exactly one Duration. It is possible that some Duration is of more than one Scenario. Each Duration has exactly one DurationValue. It is possible that some DurationValue is of more than one Duration. (Complete description was created using NORMA verbalisation)

Since, so far we are only working at the constraint level, we do not yet consider that duration value may have different values (e.g. hours, week etc.). First we focus on the factor type level. Also before we can say that scenario has different durations, we first have to check if that is true. Naturally it is obvious that time is also represented using hours, days etc., but it could be possible that in the present domain all duration values are given by minutes.

Once we are done with creating the model on the constraint level we have to start populating it. Populating refers here to entering actual information as in the earlier example with the

satellite: Satellite1 has OS1. When we are able to add all relevant information without having the reasoning engine informing us that this would violate a defined rule or realizing that some concepts are not needed we have successfully created a 'functional' knowledge base representing and storing our knowledge. If we however are unable to enter the whole population we have to go back to the constraint level and modify it to fit the new constraints before we start populating the model again. This iterative process is repeated until we are able to enter all the knowledge we want to store. All relevant steps that are required are illustrated in Figure 9. Even though not all steps were described in detail, the triangle should be able to illustrate how many (and which) steps are required for a full cycle.

Once we have a model that seems to fit all our requirements, we now have to validate it with domain experts, making sure that we did not use rules that are incorrect. If, in this last step, no problems occur we can be fairly certain that our ontology is correct (see 11 Conceptual Model).

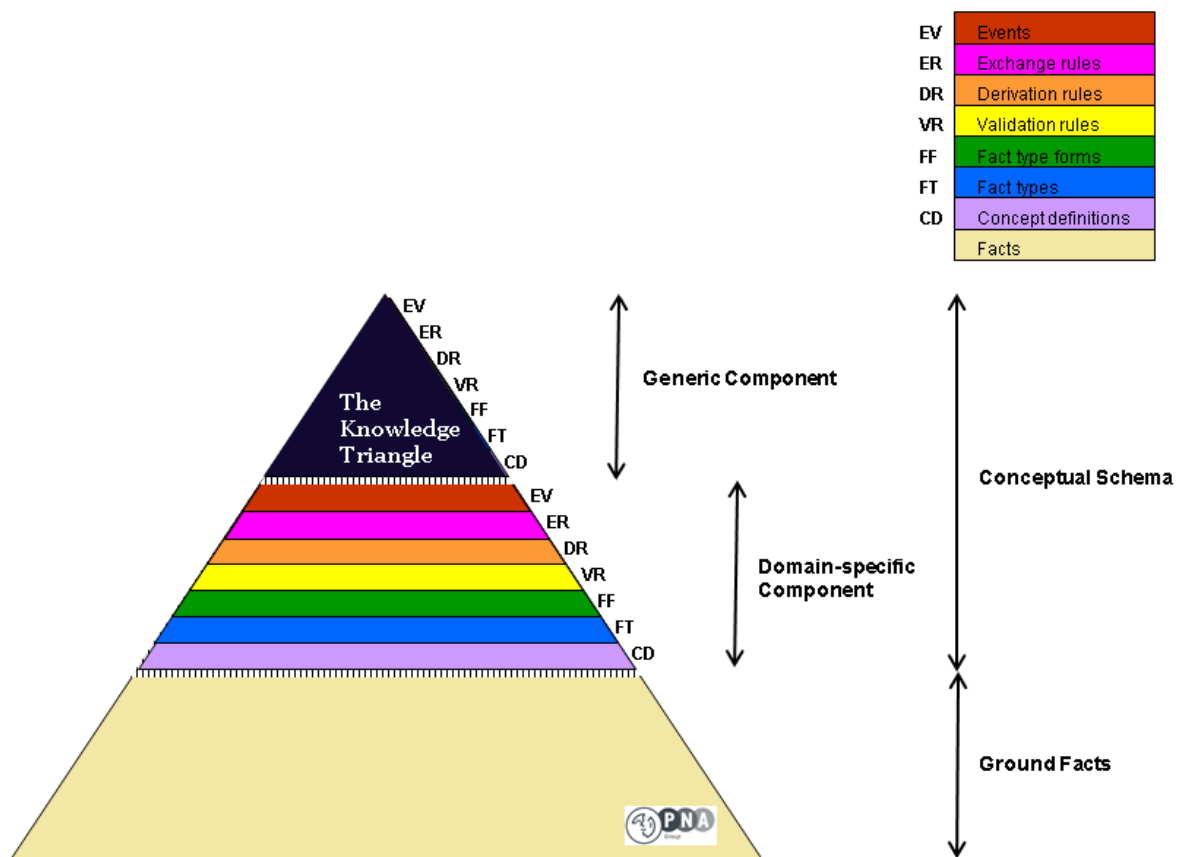
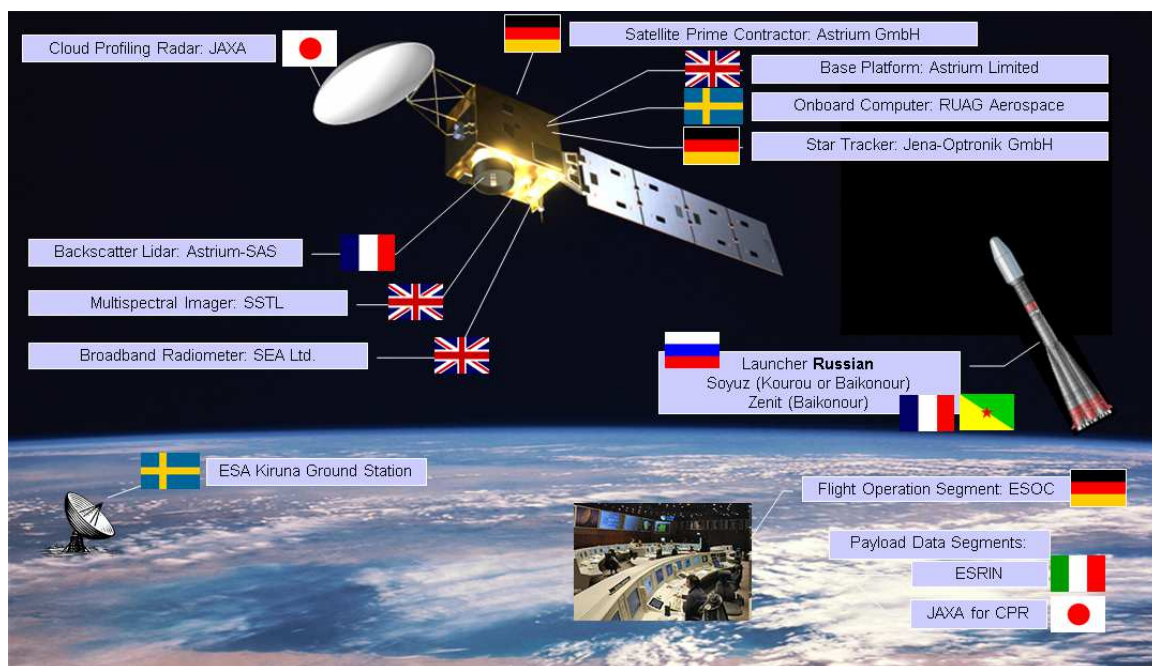


Figure 9: Knowledge Triangle associated with FBM modelling.

## 5 Challenges

*If the only tool you have is a hammer, you tend to see every problem as a nail  
– Abraham Harold Maslow*

The challenge is to create what I already called 'multifunctional model optimizing' in the introduction. Meaning to develop (or use) a model which could replace different existing models (ER, UML, OWL etc.) but still maintain the capability to deduce the generated information of the new model into these older (scientifically approved) existing models. As a result the remodelling for different knowledge sources (astronauts, operation engineers, researcher, contractors...) of different nations (see Figure 10) will be avoided by creating a generic knowledge base understandable by all participants (Challenge number 1).



**Figure 10:** This image copied from a project presentation (of the Open Group Conference in Amsterdam 2010) shows the participants of the ESA Project - EarthCARE and which components are developed/used by each nation to illustrate the international aspects of a typical project.

In the exemplified case of astronauts, the difference in knowledge levels (experience of astronaut) could be accounted for by adjusting existing procedures to fit to the experience of the actor (Challenges number 2). A beginner (e.g. does it the first time) will need more elaborated explanation and a more detailed stepped approach. An expert will only need a small summary.

However there is nothing like an "OTDS" (Operation and Training Doc and Spec) which would summarize (or be more elaborate) for any of the documentation used. Currently procedure ODF follows in the XML form a printing specific tagging. This cannot be used to easily extract information. In addition to that, the training material (usually PowerPoint) is also not standardized making the information hard to extract.

This leaves us with the need to first extract the semantics from the data to introduce it to the system which then can be forwarded to the domain specific ontology.

Note, all this has to be done in an open world assumption i.e. not explicitly asserted knowledge has an unknown truth value. While we are not forced to include this assumption in our knowledge model, this assumption could be forced upon the IOSS.

Taking into account the cognitive state of mind of the user, to identify for example how busy a user is, this can be done using a task load model (described in MECA). Realization can be done using different strategies e.g. the simple and straight forward one would be an EEG<sup>20</sup> (Electro Encephalogram) device to measure how 'busy' a user is.

Based on this, some tasks should be done automatically but never without the knowledge of another participant (knowledge source) (Challenge number 3), the operator. When the operator would be busy, the ePartner instead does give precise instructions/help. Or alternatively the latter performs side tasks on its own (assuming an ePartner that can physically interact with its environment). However, in our case, never without agreement of the astronaut, since just because it can take decisions and give advice this does not mean it should do it every time.

The final system should be closed loop. So if it can find a new solution, it should be able to create the documentation for this as well. Thus expanding its knowledge base like a 'self-learning mechanism' by removing the requirement for operators or other experts to update the newly gathered knowledge (Challenge number 4). This is however a design aspect that is not allowed/ requested for all those updating task, so it should always be specifiable if the knowledge should make a local update (just for the agent) or a global update which may (both) have to be counter evaluated by a human.

The aspect of LOS (Loss of Signal) will not be focused at, due to the fact that it is included in the scenarios as a possible technical problem.

## 6 Goal

The goal is to develop an IOSS (Intelligent Operation Support System) for the onboard crew of the ISS (which could be adapted for deep space missions) that assists them in their routine work. This includes e.g. assistance during the execution of a procedure, while assistance here refers not to physical assistance but more to the theoretical aspects. Meaning that the IOSS does not take workload from the crew member but helps the crew member, to have all relevant information for the work at hand, with the availability of trouble shooting mechanisms in case of problems related to the procedural execution. This IOSS is intended as a combination of conceptual and practical proof of concept of a MECA ePartner and should therefore also have self-adapting mechanisms in order to learn in small steps how to do adjust the instructions in space.

Nowadays this assistance, during the execution of procedures, is done through permanent communication between crew members and MCC as well as onboard ODF documents. In case of ISS operations this concept works well. Since the availability for successful connection establishments is high as well as the communication delay is small. But even in this low earth orbit (LEO) scenario communication gaps can in case of occurring problems lead to serious delays in the crewmembers time schedule. Since it is a very distributed system and there is knowledge between the earth and many persons, which are not always available neither do they all share the same knowledge. But in case of problems the knowledge of all may be required. Therefore the IOSS is an attempt to systemize, categorize the knowledge, making sure it is available when it is required (in small scale).

---

<sup>20</sup> Such a device could be e.g. <http://www.heise.de/newsticker/meldung/EEG-im-Ohr-1945567.html>



Some other challenges that make the work in this domain interesting, is the fact that the current supply chain is very distributed and non-homogeneous. That results in various kinds of unique internal processes and process documentation. The results of this project could be used to create templates how to structure the data to receive (in the future) something that is much more cost effective for long term relations. And also in the control room already today the whole program is under cost pressure. But if IOSS can enhance the performance of the people (in space and on earth) costs could be cut down (IOSS is not intended to replace people!).

The created proof of concept KFO (KII foundation ontology) from the KII project indicated that this non-standardized semantics from multiple sources can be represented by creating an ontology, readable by humans and machines. This process could be made easier by defining standards for the creation of an 'OTDS document' which later on could be used to create a standardized way (which could be forwarded to the contractors) of updating knowledge to the ontology which right now still has to be done by hand.

The creation of project related domain specific symbolic knowledge (in this project the knowledge will be information about procedural work plans as well as troubleshooting aspects from the KUBIK experiment) will allow further insight into the definition of such standards for OTDS as well.

During the course of the project a main focus was the creation of a Knowledge Infrastructure for knowledge usage as well as acquisition and the probability aspects of possible solution proposals.

## **6.1 Approach**

Based on the general described challenges in section 5, the thesis goals in section 6 in line with the deduced requirements in section 7 (deduced from the background research in section 3, 4) led to the definition of the four scenarios in section 8.

Scenario three (section 8.3), based on an existing troubleshooting case from the KUBIK operations manual, was chosen for further investigation (section 9); due to the fact that all relevant knowledge required to solve the problem was available. This investigation was necessary in order to check any kind of information related to KUBIK and troubleshooting and to find out and define how a possible functional dialogue between such an expert system and an end user could look like. Furthermore it had to be analyzed how such knowledge could be modelled and if further requirements would arise.

From these, conclusions concerning the structure and evaluation of knowledge were drawn (section 10). This was necessary to avoid continuous repetition of knowledge definitions (generating of new ontologies) and to establish a general valid structure in which knowledge, generated in this thesis as well as previous generated knowledge (KFO), could be linked together.

After a suitable general structure was identified the conceptual model (section 11) related to the KUBIK payload containing knowledge previously described in section 9 was created. Following (section 12) an in depth analysis of a possible suitable expert system.

## 7 Requirements

The presented requirements are necessary to be fulfilled to establish a functional version of an IOSS. Not all listed requirements will be able to be met in the course of this thesis. The numbering does not refer to the importance of the requirement.

The resulting system...

1. ... should be generic, meaning that it should be applicable to several application areas
2. ... should be used by Human and Machine
3. ... has to have the capability to update the knowledge
4. ... has to have the capability to forward knowledge to other systems (computer, ePartner) and humans to ensure availability of knowledge (necessary data), at the right time and point for the right person.
5. ... should be capable of self-learning, meaning it should update the knowledge also without being told what to update (e.g. just by observing). However this knowledge should be updated locally and will have to be verified by a human (controlled update).
6. ... should follow domain specific rules.
7. ... should be capable of solving problems (i.e. troubleshooting).
8. ... should include safety aspects, in order to be able to be controllable (e.g. once developed a neuronal network is nearly impossible to be understood by humans, which should not be the case in this system).
9. ... should distribute biunique (i.e. clearly identifiable) knowledge.
10. ... should guarantee reliability (i.e. it should not be able to solve just one domain problem).
11. ... should be reusable!
12. ... should produce the results in a reasonable computation time (i.e. even if the answer is perfect, it will not help if it takes 5 hours to compute it and we have an imminent issue at hand).
13. ... should allow for dynamic knowledge forwarding to other systems (not of the same domain) e. g. transfer to UML
14. ... should be easily understandable by the user (e.g. good GUI).

Not all requirements listed can be fulfilled in this thesis, but enumerative they should be valid as evaluation criteria in presented results as well as future developments enhancing results presented in this thesis.

Note that the number of requirement for the full MECA is 164. Some are summarized, others are left out in the requirements listed above.

## 8 Scenarios

*To err is human ; and to blame it on a computer even more so  
– Robert Orben*

The following scenarios serve the purpose of illustrating the working area of the troubleshooting assistance system (ePartner / IOSS) as well as serving as a guideline during the development. Since it serves the purpose for scenario 1 and 2 the procedure 6.3.3 (Insert: Specificities for mounting and demounting) from the KUBIK operation manual (Aerospace, 2007) was chosen. It includes several screws and hardware parts that need to be handled in a specific order and includes also primitive/ generic actions (assemble, screw, mount, turn, etc.).

KUBIK was chosen as base for all scenarios since from all investigated operation manuals (Aerospace, 2007) (Lepore, 2011) (Asberto, 2012) (ПКК "ЭНЕРГИЯ", 2009) it was the closest one to ODF procedures (NASA, 2013) in design, as well as turned out to be the most basic in functionality as well as troubleshooting aspects. Scenario 3, inspired by a known off-nominal case, was chosen to be used as an in depth examination in the upcoming section 9.

Scenarios created for the KII project (S&T, KII - TN1: Scenarios and Use-cases, 2012) were not of use since they focused on areas that were including more communication or no ODF procedure handling like e.g. EVA Spacesuit Failure. It was therefore decided that scenarios closer to the KUBIK manual would serve as a good test bed.

### 8.1 Scenario 1

Astronaut Billy is performing (operation manual) procedure 6.3.3 as part of an ODF procedure and realises that one of the screws is damaged (the screw thread is damaged). He asks his ePartner for help, which checks the specification of the screw and 'forwards' the specification request on locations for spare parts or other parts having the same specification and proposing the result to Billy. Since unfortunately no spare parts are found, but a different experiment which is only scheduled a week from now has a screw with the same specification, Billy decides to use it and is able to finish its work. His ePartner adds the fact of the missing screw for the other experiment as starting step (Special precaution) to the other experiments procedures. His or one of the other ePartners however will therefore remind the astronaut performing the experiment next week, if no new spare parts will arrive by then, to take the screw back before starting new experiments.

### 8.2 Scenario 2

Billy is performing procedure 6.3.3 for the tenth time and realises that step 4 and 5 (fictive) can and should be switched, so he updates the knowledge after consulting with MCC. A day later, Lily has to do the same procedure but has never done it before. The ePartner will propose both possibilities ('step4 then step5' or 'step5 then step4') concerning the steps (Insert: Specificities for mounting and demounting).

### **8.3 Scenario 3**

(Illustrated more elaborate in section 9) The KUBIK payload has been set up by Billy following a wrong procedure or not following a procedure at all. When the setup is finished, KUBIK does not work and a troubleshooting activity is started to fix the problem. The external evidence of the problem is the power led (28V) on KUBIK which is off. Billy is requested to check all relevant physical connections step by step by the IOSS. After checking all possibilities, it turns out that the ebox seems to be damaged, which is not supposed to be fixed by the astronauts. So Billy instructs the IOSS to set a new KUBIK on the list of required parts for the next Automated Transfer Vehicle (ATV) delivery.

### **8.4 Scenario 4**

The base scenario is the same as described in scenario 3, however this time we are not able to deliver a new ebox or other spare parts since we are on a long term mission to Mars. So we have to figure out what is the most likely technical issue with the component.

## 9 KUBIK- an elaborated troubleshooting Scenario

A presentation of a troubleshooting scenario for scientific payloads on the Columbus module is given. This scenario is based on a troubleshooting procedure that today is included in the KUBIK operation manual (procedure 6.5.1). The analyses of the functional model with the existing data and information will lead to the build-up of one possible dialogue between man and machine. Following elaboration on modelling the dialogue required knowledge. In the end the evaluation of our modelling will be analysed to define possible missing requirements and draw attention to possible deficits.

For illustration of possible dialogues for the other scenarios see appendix 18.2.

### 9.1 Scenario Description

The Kubik payload has been set up, following a wrong procedure or not following a procedure at all. When the setup is finished, Kubik does not work and a troubleshooting activity is started to fix the problem.

The external evidence of the problem is the power LED (28V) on Kubik which is off.

### 9.2 Kubik Functional Model

In order to present the troubleshooting scenario, a simple functional model for the turn on process of Kubik is required. The functional model is based on the following design information (inferred from training material).

The indication LED will be lit whenever the following conditions are met:

- $PWR_{in} \rightarrow \text{True}$ .  
There is a power connected to the device
- $SW1 \rightarrow \text{True}$ .  
The switch is in the “on” position
- $LED_{ok} \rightarrow \text{True}$ .  
The LED is not broken

These three conditions can be summarised in the following rule:

$PWR_{in} \wedge SW1 \wedge LED_{ok} \rightarrow LED_{it}$

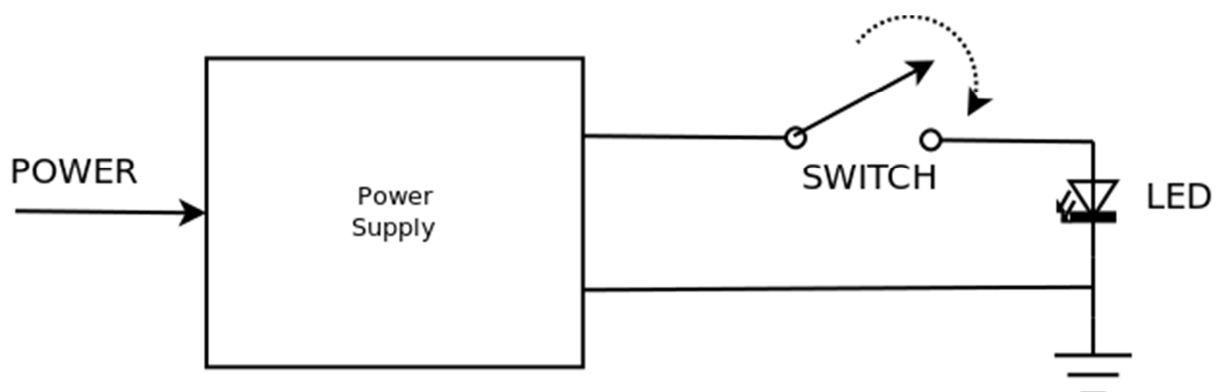


Figure 11: Simplified KUBIK Electric Circuit

### 9.3 Troubleshooting Process

The troubleshooting process will start and evolve depending on how the reasoning engine is used to work. In this section we will just assume a certain, arbitrary order on the questions presented to the user by the system.

Therefore, in our specific example (without loss of generality), the following dialogue could happen:

- COMPUTER → Is the LED ON?
- USER → No
- SYSTEM : LEDlit → False

Supposing the system will work using backwards propagation, the functional rule introduced above will be re-write as:

$$\neg\text{LEDlit} \Rightarrow \neg\text{PWRin} \vee \neg\text{SW1} \vee \neg\text{LEDok}$$

In other words:

*If the LED is off, either power is not applied, the switch is not in the ON position or the LED is not ok*

In order to assess which one of the three potential causes of the problem is the one causing the problem, the system shall ask, sequentially, about each one until the answer is found.

- COMPUTER → Is the Power Connected?
- USER → YES
- SYSTEM : PWRin → TRUE
- COMPUTER → Is the Switch in the On Position?
- USER → YES
- SYSTEM : SW1 → TRUE

At this point the reasoning engine should be able to assess that the problem is a defective LED as this is the only remaining condition. Alternatively, the system may allow the user to answer "Do not know" and set the assertion as a free variable resolving the logic expression. In any case the answer for the system will be that the LED is broken and the corrective action will be to change the ebox<sup>21</sup>.

### 9.4 Extended troubleshooting scenario

The user changes the ebox on the Kubik payload, but the problem still persists. The problem is that the functional model of the payload is not complete. In this example, the functional model presented above is exclusively based on the design information provided by the payload developer. However, the payload developer, in general, does not have the knowledge about the deployment infrastructures and therefore the functional model is incomplete (for operations). The ODF procedure author or operations personal shall update the original functional model to take into consideration the infrastructure were the payload will be deployed. In this example within the Columbus module and here within the EDR (European Drawer Rack). Based on this information a new functional model is developed.

---

<sup>21</sup> KUBIK has two control boxes attached to it. One if for controlling temperature and centrifuge handling the other one (ebox) is to power the system.

## 9.5 Operations Functional Model

The simple Kubik functional model is extended on the following way:

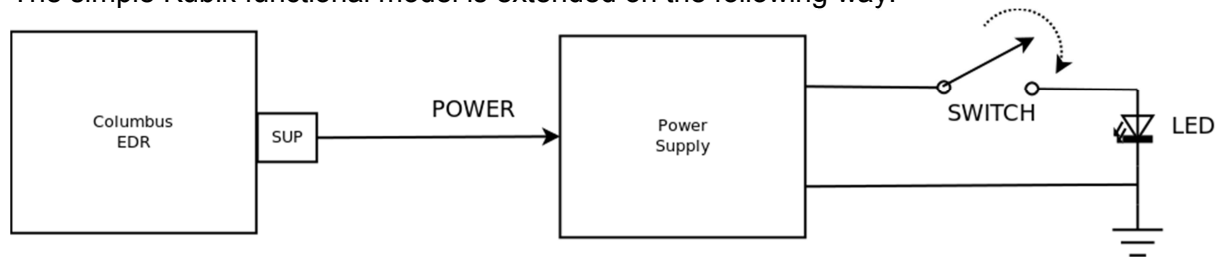


Figure 12: Simplified KUBIK Electric Circuit. Operational environment

The model has been extended with the concept of EDR SUP (power socket on Columbus Rack) and its operational constraints.

The following additional conditions are required:

- $SUP_{conf} \rightarrow True$   
The SUP on EDR is configured properly
- $SUP_{conn} \rightarrow True$   
The SUP is connected to the payload. In reality this will be more complex and involve specification of connectors to be used, etc...

Using this new conditions we can rewrite the rule for the Kubik LED as:

$PWR_{in} \wedge SW1 \wedge LED_{ok} \rightarrow LED_{lit}$

where

$PWR_{in} \Leftrightarrow SUP_{conf} \wedge SUP_{conn}$

leading to the new rule:

$SUP_{conf} \wedge SUP_{conn} \wedge SW1 \wedge LED_{ok} \rightarrow LED_{lit}$

## 9.6 Troubleshooting scenario (Continuation)

Using the new functional model the troubleshooting scenario can be extended with additional questions to the user, which eventually will find the problem. Continuing the dialogue presented above:

- COMPUTER  $\rightarrow$  Is the Power Connected to SUP on EDR Rack?
- USER  $\rightarrow$  Yes
- SYSTEM:  $SUP_{conn} \rightarrow TRUE$
- COMPUTER  $\rightarrow$  Is the EDR SUP configured to provide 28V?
- USER  $\rightarrow$  No
- SYSTEM: Oops. **That's it.** Please run procedure...X.XXX
- COMPUTER  $\rightarrow$  Is the LED on?
- USER  $\rightarrow$  Yes
- SYSTEM: Problem Solved!

## 9.7 Troubleshooting scenario knowledge as model

For further considerations let us assume the example scenario of the component LED again which is supposed to light up when the switch is activated, indicating that the component is turned on.

There are multiple tools being used today at ESA. One being FAMOUS (Valera & et.al., FAMOUS Statement of Work: Fact based Modelling Unifying System: Toward implementing solution for ECSS-E-TM-10-23A, 2012), in development by ESA. Another one is NORMA (NORMA for Visual Studio, 2013), a public available ORM tool that was used during the development in the course of this thesis. They both specify completely the information we

want to work with. Once developed FAMOUS will includes all NORMA features plus manifold additional functional aspects.

The main challenge in modelling is to catch nominal and off nominal or failure cases in such a way that your model at the end still leads to valid information. The difficulty is connected to the fact that in space we are modelling complex software and hardware (equipment) on different hierarchical levels being influenced by men and/or machines. Any modelling is facing the request to be generally valid and should result in a fact type population that shall always be true, whether it is asserted or derived.

To illustrate these circumstances let us take a closer look at our scenario and try to find out the necessary solution steps, classify specific level aspects to be addressed to finally find the right solution.

In our example Power (PWR) is needed. The first question is how do we model it? Is it a system element? Is it just a component, an electrical unit or even a lower level equipment part like a switch? For simplicity we will go with the simple relation that we have a component which needs/has power and is turned on/off with a switch.

The component can be turned/ electrically controlled by a switch. The switch can have two values/ switch status. The switch status is 'On' or 'Off'. We can derive that the component is 'On' and can make a derivation 'Is the switch On?'. We have formally to write 'If the component is electrically controlled by a switch that is 'On' → 'Then the component is said to be 'On'; forming our first derivation rule for the scenario.

Logically we can derive that the component can be 'Off' as well. To derive this we have two options, either the component switch is 'Off' or 'Is not On'(-On). Let assume decision has been taken to model it as 'Is not On'. We can directly derive an implicit constraint that 'On' and 'Off' cannot both be true at the same time, to complete the model.

An error case or off-nominal case could be defined if the switch would be on, but the component would not be on. This could happen if the component would be damaged or not connected the right way to another component (EDR Columbus), thus having no electricity or alternatively the switch could be damaged. This off-nominal case also has to be modelled. (see Figure 13)

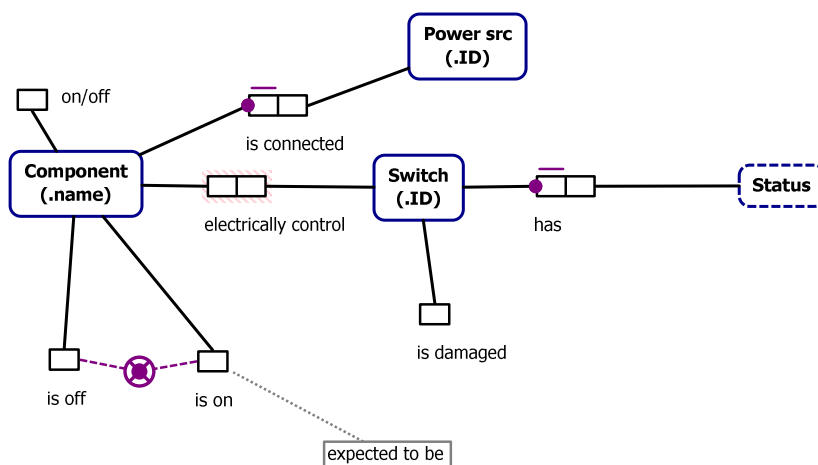


Figure 13: Conceptual model of the problem case of Scenario 3



So how can an expert system help the user to figure out things that were not thought of in the beginning? How to realize and evaluate differences in expected results, observed realities and the modelled expertise?

As illustrated, we can derive a certain level of error cases, but may have missed some. As mentioned in section 4.2.2 the developer of the component may not have the full knowledge of the 'final' environment. For example he is missing knowledge to be modelled due to a recent event that happened on the ISS, where a cooling system stopped working. Here we enter the formal modal logic. An expert system serves and supports a number of things /activities and determines certain behaviour. The expert system (Halpin, Fact-Oriented Meets Agent-Oriented, 2005) is modelled at the requirement level.

If we have knowledge included that we are not sure about, we add 'expected to be'. 'Expected to be' standing for an expert system to my best knowledge. Either I determine that the LED is 'On' or I determine that it is expected to be 'On'. More or less we realize what is observed and accept it as the only existing reality.

In order to verify the status and whether it is adequate we have to instantiate the system with a number of measurement points. In space-crafts this could be sensors within the system or an LED which is perceivable by the human user to receive exactly the right status information.

The previous described process can be improved and covered via additional expert systems, namely state of the art diagnosis tools (e.g. Model-based reasoner (Davis & Hamscher, 1988)) used to help designing systems. It is however required for such a diagnosis system to have a basic model and measurement points which it can use to simulate and reason different possible cases.

The initial set of measurement points by default should come out of the component design (see FMECA (Failure Mode, Effects, and Criticality Analysis) (Borgovini, Pemberton, & Rossi, 1993)). The idea is, if someone makes a (complete) design, quality management should back him by informing him about possible equipment failures under certain conditions. For each (basic) element we will know, what is the nominal behaviour and hopefully, thanks to the FMECA analysis and possibly diagnosis tools, you know how it can behave 'wrongly'. In case of the space domain this is true for all the equipment. So we have 'a' reality from the perspective of a measurement point.

## 9.8 Scenario analysis

From the discussion above the following can be concluded:

- A functional model of the payload considered shall be provided in order to be able to start a troubleshooting activity
  - The functional model should be derivable from the payload engineering documentation
  - Ideally the payload developer should provide such a functional model whenever a standardised specification to do that is on place
  - The functional model should cover the items and functions exposed to the final user. In our example, for our current level of abstraction, it is not expected that the final user will open the Temperature ebox and change the LED in case the troubleshooting process concludes that the LED is broken
- The functional model should be extended with additional rules to take into consideration the infrastructure. In our case Columbus:
  - The payload developer will probably get just a minimal set of interface requirements. For instance, in our example, the payload developer would probably have requirements on the input voltage and the kind of connector to be provided but it won't get information on how a SUP is configured to provide power to a given payload
  - Currently this knowledge is managed by procedure authors and operations experts. For our example, KUBIK, the ODF procedures and the Operation procedures provided by the payload developers are not exactly the same.
  - Additional requirements (safety, security) specific to the operational environment will also be added at this point.
- Ideally, the system should have the following components:
  - A knowledge base for the "infrastructure". Columbus in our case
  - A knowledge base provided by the payload developed together with a functional model.
  - A glue knowledge to connect the previous elements.

## 9.9 Final notes

- The scenario is heavily simplified.
- Temporal constraints (sequence/order of actions has not been considered).
- It is envisioned that payload model might be easily derived using generic libraries (e.g. electrical payload). This will also enforce standard development and interfaces from the beginning.

## 10 Model structure

*Connection, not collection: That's the essence of knowledge management.*  
– Tom Stewart

In section 10 the reader will get acquainted with the fact that one dedicated ontology will not lead to acceptable solutions, due to the manifold different sources of information (section 10.1). It is shown that instead a splitted ontology structure, on different parallel levels, following a hierarchical tree structure would be the best solution to integrate core knowledge, ensuring sustainability and maintainability (section 10.2). Considering core knowledge an example will detail the necessary knowledge required for man machine communication (section 10.3).

### 10.1 Ontology structure analysis

While it is in theory possible to add all relevant information into the same ontology it is not viable. The resulting model of such an approach would be more troublesome for the users than helpful. It would just exchange the current “thick” folder containing all procedures to perform with an electronic ‘super’ model that would probably even be harder to browse through (until the user would be used to it).

This drawback could be removed or decreased by using a well design GUI to access the knowledge but the maintainability would still be a major issue. To check such a huge model for inconsistencies would take a long time as well as it would make it more troublesome for adding new knowledge for an IOSS.

The trouble of doing so already arises when modelling just one component (e.g. Kubik), since e.g. functional and mechanical information are not necessarily required at the same time. While they are strongly linked considering completeness of a component the user as well as the IOSS may be lost in the amount of information to check. For humans the model would grow to a size where it does not serve anymore as a helping and supporting overview. For the computer however it increases the amount of logical dependencies to check and with this the amount of computations. A good example for this would be the combinational explosion problem<sup>22</sup>, the more variables are introduced the combinational considerations grow rapidly. Translated to the current domain it would mean we would have to check more and more ‘possible’ dependencies to a related problem<sup>23</sup>. Imagine the problem from section 9 where the model would not just include the small amount of mechanical information but also all functional information. The illustrated logical rule would continue getting bigger and bigger up to the point where the IOSS would have to ask an unreasonable number of questions to solve the issue. Involving questions that maybe not related to actual problem as well as involving questions that the human shall not even deal with (e.g. imagine the mechanical level includes cable wiring and starts asking about cable connection within the ebox). While, as mentioned, this knowledge should be present for long term missions it is not viable for each troubleshooting situation. The human would most likely still be able to differ between the importance of information or at least be able to order them reasonably. The IOSS would not even understand the difference.

---

<sup>22</sup> [http://en.wikipedia.org/wiki/Combinatorial\\_explosion](http://en.wikipedia.org/wiki/Combinatorial_explosion)

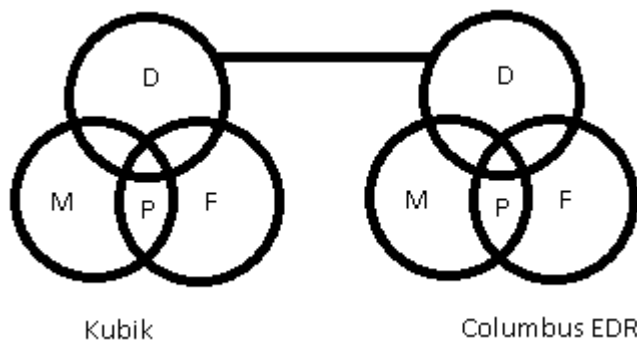
<sup>23</sup> Note that this search could however be improved using the RETE algorithm  
[http://en.wikipedia.org/wiki/Rete\\_algorithm](http://en.wikipedia.org/wiki/Rete_algorithm)

Therefore to prevent creating an overly complicated KB the knowledge should not just be 'split' to be specifically adapted to a given module but also within the module. In case of KUBIK (and other modules) a logical split would be, to differentiate between mechanical and functional knowledge. A complete split can however not be done since both domains require some physical information as well. The physical aspects in the mechanical domain would include the cables, ebox connection etc. While they would include e.g. air fans distance (because this is at the same time mechanical knowledge as well), switches etc. in the functional domain. The connection between the KB's then could be created using a third ontology here called the dependency ontology.

That would leave us with the requirement to model three different 'levels':

- Mechanical/Physical
- Functional/Physical
- Dependencies

As illustrated below in Figure 14.



**Figure 14: (M=Mechanical, P=Physical, F=Functional, D=Dependencies). On the left side the knowledge of Kubik is gathered. On the right side the knowledge of the Columbus EDR is gathered. In both KB knowledge is not included in the same model and just certain related aspects are connected through the dependency ontologies.**

Note however, that there will be a dependency ontology for each component, that i.e. links the mechanical with the functional knowledge as well as relates it to other *relevant* modules/components. The dependencies also will not be bi-directional. It would be unreasonable to add to the Columbus EDR the KUBIK module as a permanent dependency, maybe there is an error in the EDR and the IOSS tries to find the error. It will propose to check something on KUBIK (e.g. to switch it off to reduce energy consumption) while KUBIK may not even be connected at that time.

With a knowledge design like this the IOSS could first start checking the knowledge related the closest to the issue at hand (in the given scenario the functional knowledge) before checking mechanical or move to check dependencies (extended scenario, Chapter 8.4).

A drawback of this design however is the requirement of creating a new (or updating / modifying) dependency ontology every time the component is used in a new environment<sup>24</sup>.

<sup>24</sup> This however would only include knowledge to connected or dependent components and not internal component knowledge since that would not even change in a new environment as long as the component was not modified.

The biggest advantage would though be, that the IOSS using the knowledge, has all relevant information related to the component it is responsible for while not being forced to check unrelated knowledge. This knowledge could then be forwarded (or extracted) to create semantic decision tables (Tang, Semantic Decision Tables - A new, Promising and Practical Way to Organizing Your Business Semantics with Existing Decision making Tools, 2010) which can be used to create the information retrieval step as well as include the logical order and dependencies without having to search for dependencies 'which are not there' (wasted computation time).

## 10.2 Resulting general structure

The knowledge will be stored using FBM. However it will not be all stored in one 'interconnected' ontology, but instead be split to several smaller domains and upper ontologies which together will form the overall interconnected ontology. It will have local knowledge hierarchies. By using an intelligent 'level' design it will be possible to exchange or extend knowledge parts without influencing the overall structure too much. Note that the parts that are exchanged however, have the need of supporting the same fundamental knowledge information, so that dependencies from lower levels will not be lost.

But even in this levelled approach all has to be embedded into a higher structure (upper ontology) including general knowledge like locations as well as 'common' technical information/properties shared between all components of the structure. This way component specific information is disconnected from domain specific information.

The knowledge engineer also has the advantage that he can modify certain aspects without needing to worry about the influence on the complete system. This however is not true for the core knowledge (most general upper ontology) e.g. location (see section 10.3) which has to follow a certain set of regulations when being modified. Luckily we can use the design aspects of FBM to take this limitation into account. The reason for this is that each aspect of the model is held in a separate 'map'. This way we can include or change things while we just have to consider certain maps. Note, this also means that the core knowledge is not just one single core but a collection. Quite clearly understandable, as we cannot afford to link domain knowledge not belonging to a component.

This results in a somewhat unintuitive ontology. We have some kind of component knowledge embedded in a tree hierarchy (KUBIK->EDR->Columbus->ISS) which is embedded in a parallel intersecting hierarchy (e.g. Mechanical, Functional, Physical) which in turn is embedded in a level hierarchy (e.g. lvl1 Core knowledge, lvl2 dependency knowledge, lvlX Component knowledge). If we represent this graphically it would look like a cone architecture as illustrated in Figure 15 (ECSS, Space engineering: Space system data repository (ECSS-E-TM-10-23A), 2011). Note however that the parallel intersecting 'aspects' are illustrated in Figure 16 demonstrating this for the three domain views system-, operations- and thermal-engineering!

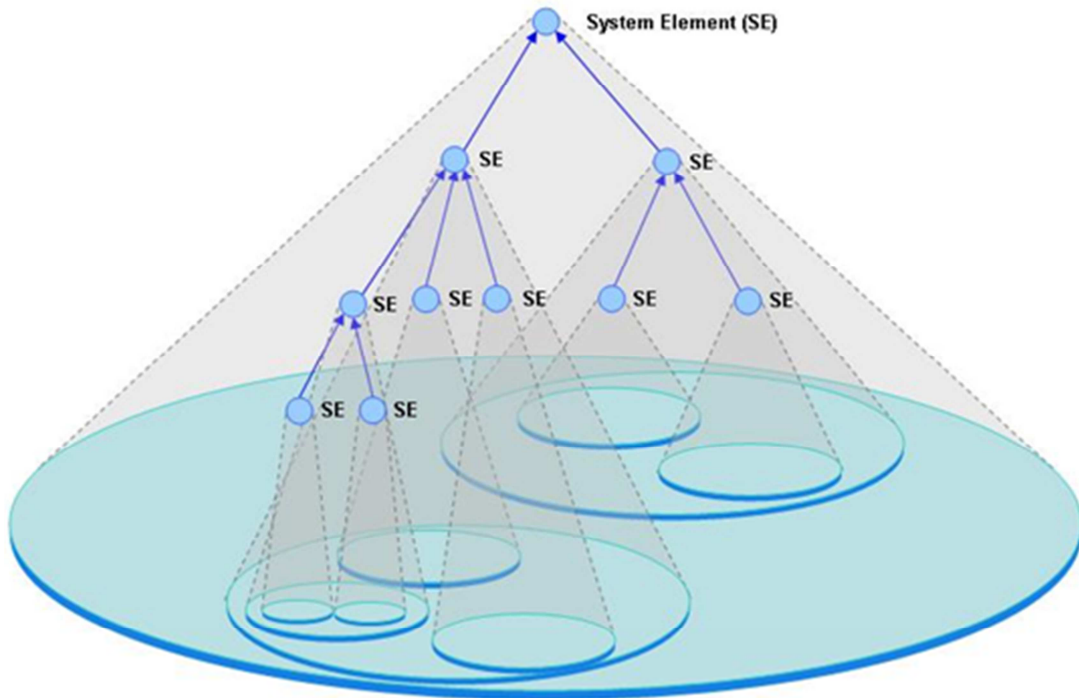


Figure 15: Graphical representation of how the KB structure would look like.

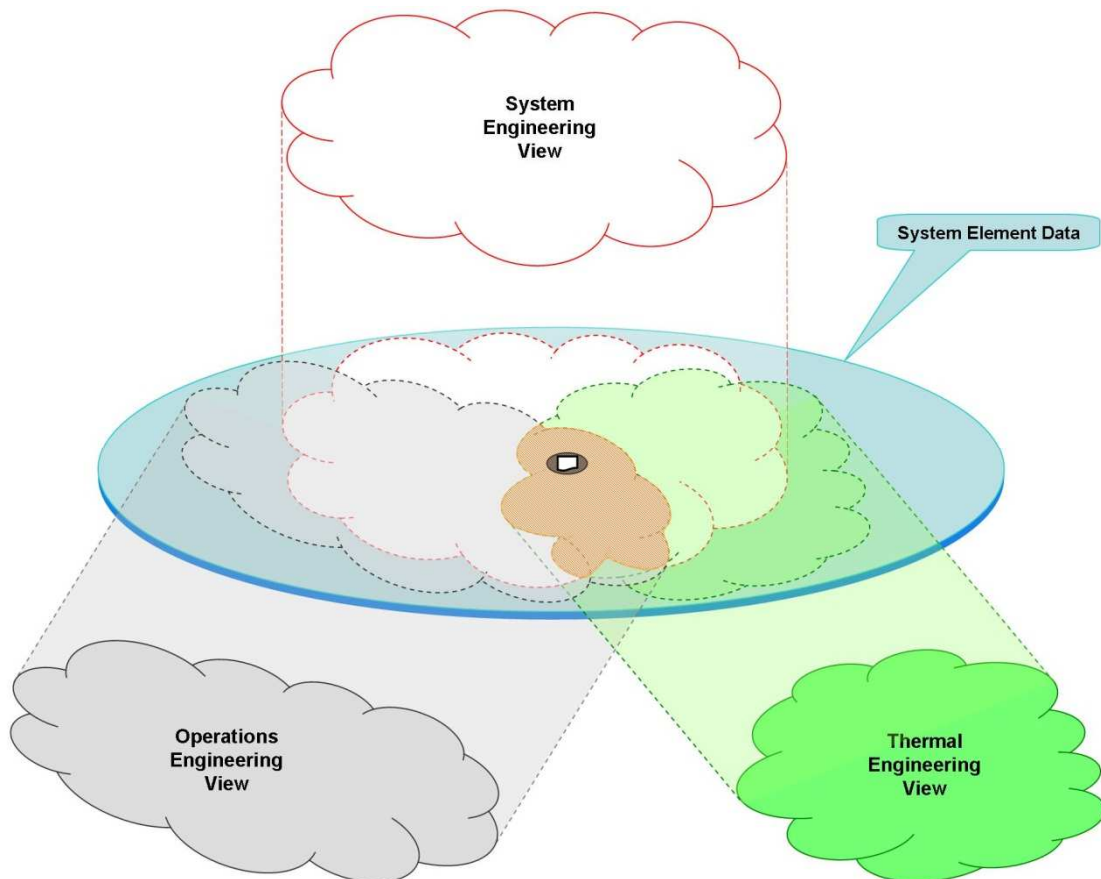


Figure 16: Illustration of intersecting domain knowledge

But why should we use such a complex representation for the knowledge?

Using this, we can take full benefit of the three important features of ORM/FBM. We can use the Boolean types as well as the derivation rules to explain all nominal and off-nominal cases that we know about and can foresee (for human and IOSS). While allowing at the same time the agent to iteratively increase the problem complexity in off-nominal cases, from the lowest to the highest 'level'. We do not need any more to create ODF procedures but can just generate step by step instructions for any (nominal and to a certain degree off-nominal) arbitrary task required to be performed, while still being able to generate complete procedure lists. In addition to that we are able to communicate each of the logical dependencies of all knowledge items to the agent as well as to the human user through the use of verbalisation. Though for the human user the currently used ORM verbalisation should be extended to be closer to ODF procedure descriptions (a representation way he/she is already trained/used to understand).

Finally the whole KB is connected to a DB storing dynamic and reasoning support information related to equipment and components as well as a history of occurred off-nominal cases accessed by the agent.

### 10.3 Core knowledge example

A good example of one of the several times mentioned core knowledge would be the translation of visual illustrations (pictures) included in all operation manuals and ODF procedures as illustrated in Figure 17.

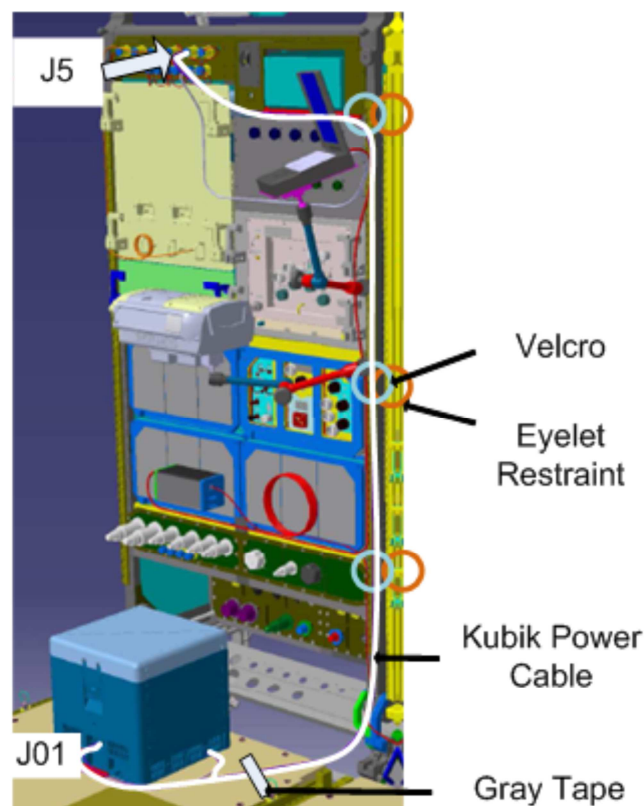


Figure 17: This image is used within KUBIK ODF procedure 2.110 to illustrate positional information concerning a sequence of actions

While for the human this representation is enough to understand to which positions a cable has to be fixed to, the machine (agent) cannot derive this information. Even though it is possible to store such an image in the knowledge base and use a camera integrated into the agent to be able to make a picture - image analysis (pattern recognition) and comparison to determine where the cable in the 'taken' picture is missing, it would not be a suitable solution. Under the assumption that it however would be able to tell the exact difference, it would still lack the capability to communicate its findings in any other way than providing the user with an image that illustrates where the two images diverge. What we therefore need is the knowledge to be able to give 'precise' instructions by the agent or where the human has to check its environment. This knowledge does not have to be 100% accurate, meaning "move your hand 5 centimetres to the left, then 2 cm forwards..." but has to be capable to understand human directional input as well as communicate those.

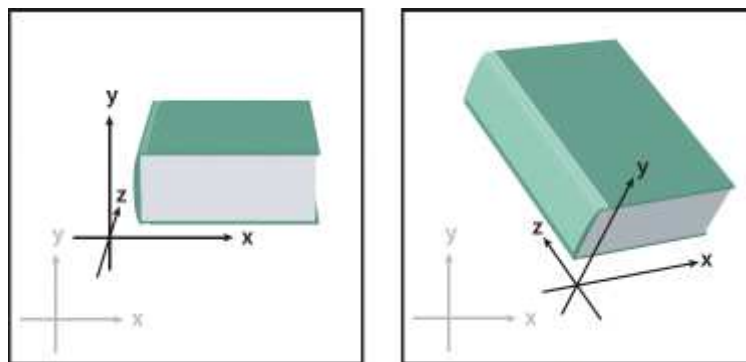


Figure 18: Three dimensional axis from the adobe creation of 3D objects ([http://help.adobe.com/de\\_DE/illustrator/cs/using/images/rs\\_34.png](http://help.adobe.com/de_DE/illustrator/cs/using/images/rs_34.png))

Figure 18 shows a standard three dimensional axis which can be used to communicate all relevant positions within one room. If we were to teach this knowledge to the agent we would only need to translate the x,y and z axis into natural language. The easiest way would be to define something along the line of:

$Z < Z_0$ = behind the	$Y < Y_0$ = above	$X < X_0$ = to the right
$Z > Z_0$ = in front of	$Y > Y_0$ = below	$X > X_0$ = to the left
$Y = Y_0 \wedge X = X_0$ = in the middle (of)		

Note, though this definition is arbitrary, it serves only to illustrate how an agent would be able to communicate directional / positional information to the human and could also understand questions from the human (even if not illustrated, the addition of rotational information should be straight forward). ESA is currently in the process of (testing) visualizing many of their equipment into 3D models that could be used as training material or for other dedicated purposes. These models, once created for an equipment, could be used to feed the agent with the required information how to direct action for the human. Without the need of a knowledge engineer, who would have to decide on a position and orientation of the axis within the component description. Without such a model, a knowledge engineer would be required for each image (or the whole component, depending on what illustrations are included in the procedures) to define the starting point depending on and adapted to the future perspective of the user.



It could be captured using a model (the presented model is an example and is not validated for a complete global domain) like illustrated in Figure 19 below.

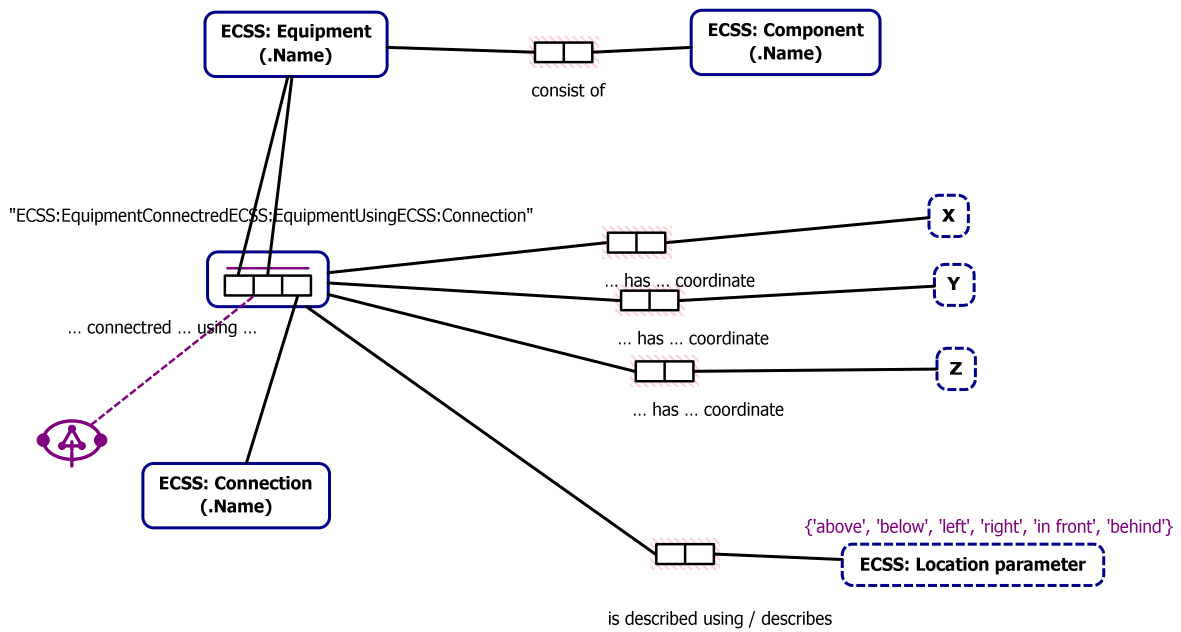


Figure 19: Example model of core knowledge related to positioning

## 11 Conceptual Model (payload)

*To effectively communicate, we must realize that we are all different in the way we perceive the world and use this understanding as a guide to our communication with others*  
– Tony Robbins

As addressed already in the abstract within section 11 an example knowledge base will be presented. The generic model description demonstrates on the basis of a splitted KB and FBM standards the interwoven connectivity of model aspects like equipment, activities, hardware setups etc.

The resulted conceptual model would already enable knowledge management of similar components to KUBIK and fulfill procedural (ODF) aspects.

### Model description

In this section the most important aspects of the created model to fit the existing information of the KUBIK operation manual as well as aspects required for ODF procedures are illustrated. The description does not include derivation rules. The described models focus on presenting the core for most of the examined operation. However very mechanical aspects as e.g. illustrated in Figure 20 are only included in the form of visualisations for the human, assisted by the step to step action description that is delivered with these. Knowledge like this is usually unique to certain components and since the focus was to create and investigate a reusable core that is not just usable for KUBIK it was decided not to be included. This knowledge would need to be linked to the described core for each specific equipment / component and would otherwise lead to having knowledge in the core that is not used for many components.

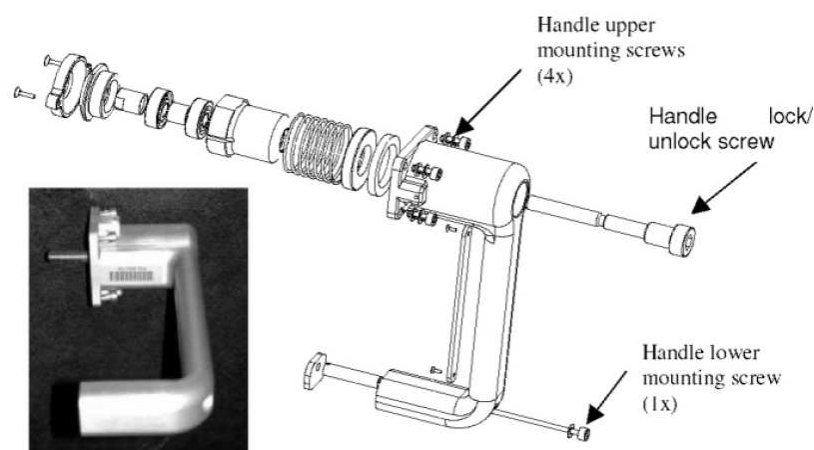


Figure 1. Facility Core Element Handle

Figure 20: Illustration of the FSL Drawer Rack handle included in the FSL manual and procedures

Note: Each of the wordings chosen/ used has to be clearly specified. So all possible wordings were chosen after consulting ECSS standard documents (ECSS, ECSS system: Glossary of terms, 2012)!

Note: The presented models are read from left to right and from top to bottom. Except, when there is an arrow pointing in the opposite direction of this rule.

It is also important to mention that not all relationships included in the model are necessarily described since this should not serve as a FBM technique (Halpin & Morgan, Information modeling and Relational Databases: From Conceptual Analysis to Logical Design, 2008) description but rather as illustration of a possible KB for procedural information. The first

section (generic requirements) will though be very elaborated, considering how to design and store knowledge, while the other section will more focus on the usefulness for reasoning considering an agent.

The first part of the model is managing the general requirement involved in each procedure

6.1.3 KUBIK: Transfer from Soyuz to ISS-RS

Requirement	Budget	Comment
Duration	20 mn	
Manpower	1 to 2 Cosmonauts	

Figure 21: General requirements each procedure has.

### 11.1.1 Model of the Generic information

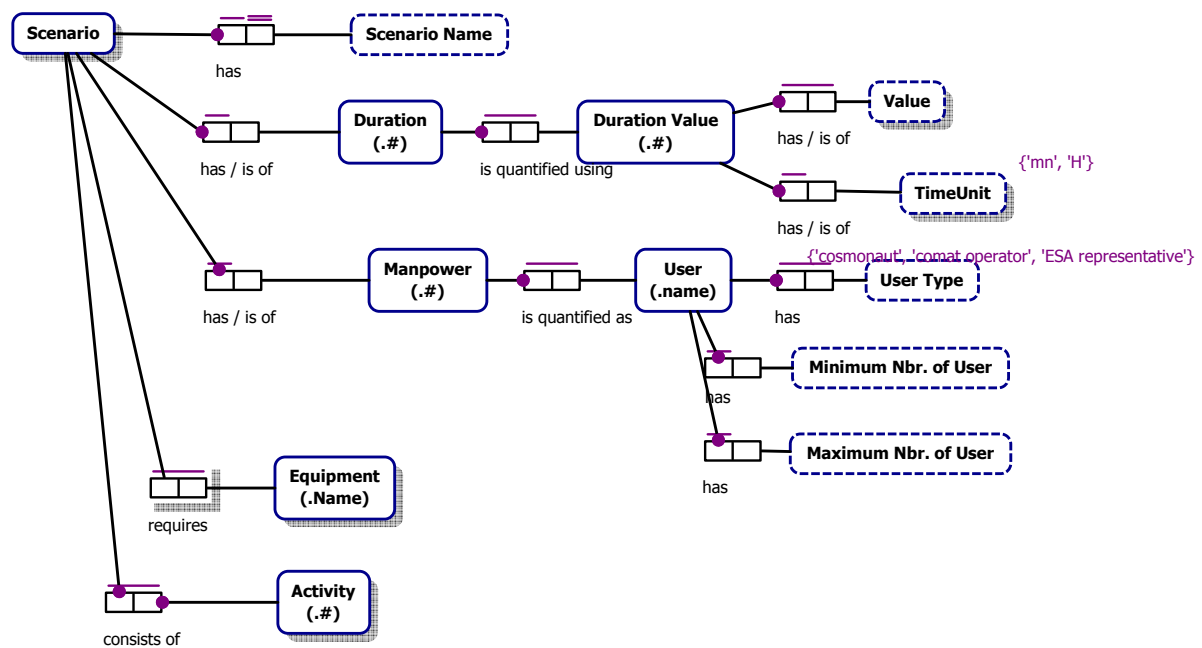


Figure 22: Illustration of the general requirements model

As already elaborated in the theory section 4.3 we group the different procedures<sup>25</sup> into scenarios. We have a requirement which is called “Duration”. So we add an entity type “Duration” and have our first concept. We can also see that to the “Duration” something like 20 mn has been assigned. So this “Duration” has a value 'DurationValue' which is given in the example by 20 minutes. This duration is related to a task (6.1.3), since every concept has to be properly defined, in this case we associate it with the scenarios/ procedures from KUBIK. And this scenario has as well a name 'Transfer from Soyuz to ISS-RS' (which is the preferred identifier of a scenario).

Each “Duration Value” has exactly one “TimeUnit” and that it is possible that some

<sup>25</sup> Procedure and scenario will be used interchangeably in the description but both always refer to one complete course of actions, described in the operation manual

“TimeUnit” is of more than one “Duration Value”. While it is possible that some “Duration Value” has more than one value and that some value is of more than one “Duration Value”. However in each population of “Duration Value”, each Duration Value, Value combination occurs at most once. This association with “Duration Value”, Value provides the preferred identification scheme for “DurationValueHasValue” so that each “Duration Value” has some Value.

The next task to model is the “Manpower” (Figure 21). The manpower can vary across the scenarios and is except in special cases (as illustrated or when the ground is involved) usually one astronaut/cosmonaut. But since we can have multiple people, it is possible that some “Manpower” is quantified for more than one “User” and that for some “User”, more than one “Manpower” is quantified as that “User”. This requires to also specify a minimum and maximum number of users, where each “User” has exactly one Minimum Nbr. of User (Maximum Nbr. of Users) and it is possible that more than one “User” has the same Minimum Nbr. of User(Maximum Nbr. of Users). Since not all users are astronauts/cosmonauts, but all cosmonauts also have the specialities (e.g. Mechanics, Mathematics etc.) it is possible that some “User” has more than one “User Type” and that for some “User Type”, more than one “User” has that “User Type”. However in each population of “User” has “User Type”, each “User”, “User Type” combination occurs at most once.

Since the scenario is associated also with equipment (e.g. a hardware specific screw driver) we add this in form of a rule, but start the equipment model on a separate sheet to maintain a good overview and structure of the model. Same applies to the activities of which the scenario consists of.

The knowledge contained within this part of the model however has no major use in assisting an IOSS by solving problems. It does not contain information about a component and only is helpful in specific working processes, where an agent could use the information to plan tasks to be performed. Furthermore does it serve as a root linkage for each respective procedure. Under the assumption that each actor involved in the mission (MCC, astronaut etc.) has its own agent, it also can be used to get everyone available when required (since the manpower and type is defined).

This model is extended by the activity and equipment model which are presented next and focus on their adequate knowledge.

### 11.1.2 Model of the Activity

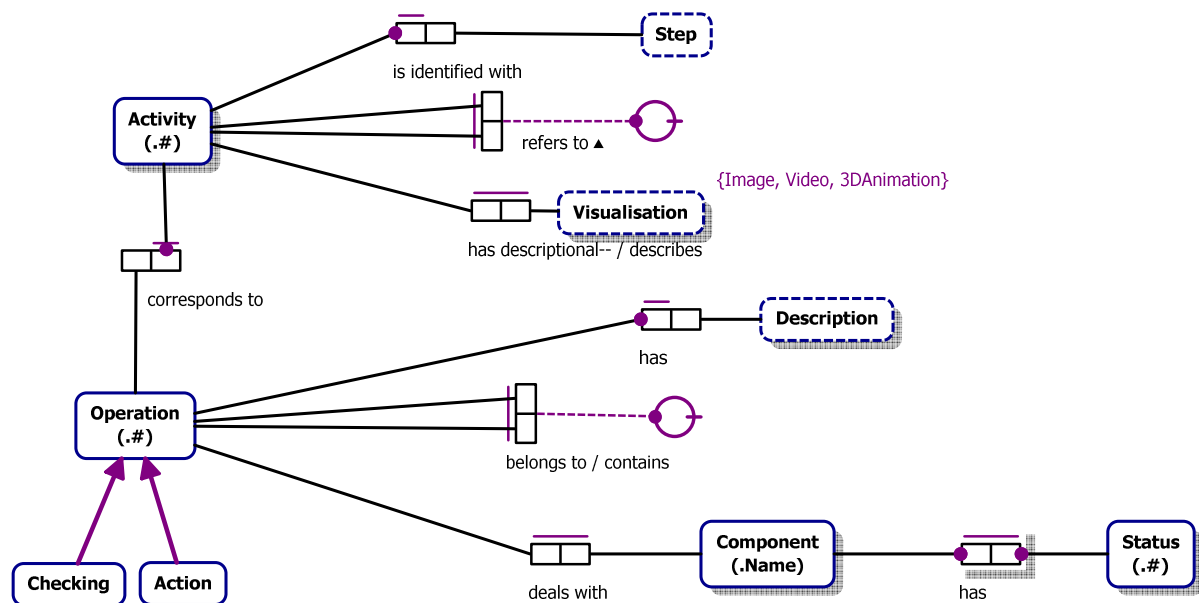


Figure 23: Activity related information, necessary for a step by step procedure

The activities of a scenario/ procedure form the main body and extend the generic scenario. They are a step by step description of what actions the user has to perform and in which order. These activities can refer to other activities (not the own ones) and are usually further elaborated with a visualisation (picture, video, 3D animation, etc.) and would be required to update/ connect in order to content that the agent would be able to understand (see section 10.3). These activities can be (for KUBIK) split into two different operations. On the one side checking activities like e.g. ‘check that the lid is closed properly’ and on the other side action activities like e.g. ‘close the lid’. The Kubik operation manual also includes other operations like e.g. “End of procedure”, however those have no need to be modelled since they are of a purely informal domain and do not add any new knowledge. By not taking them into account no knowledge is lost although this fact can be communicated to an agent by defining the code in such a way that, once there are no following steps, the procedure is finished! Both types of operations are provided with a description of what has to be done and can as well as activities refer to other operations. All operations deal with (are related) the component (e.g. Kubik) which can be ‘found’ in different states.

### 11.1.3 Model of the Equipment

Equipment	Configuration	Comment
FM Hardware	KUBIK plugged and on	At the beginning, KUBIK is still functioning in SOYUZ.
Additional Flight hardware	M10 Allen key ISS/SOYUZ adapter	Parts of the KUBIK Toolkit
Ground hardware	-	
Special precautions	1) Do not shock KUBIK during transport 2) Do not use the safety belt for handling KUBIK 3) Respect the air clearances when installing	

Figure 24: Equipment requirements for the scenarios

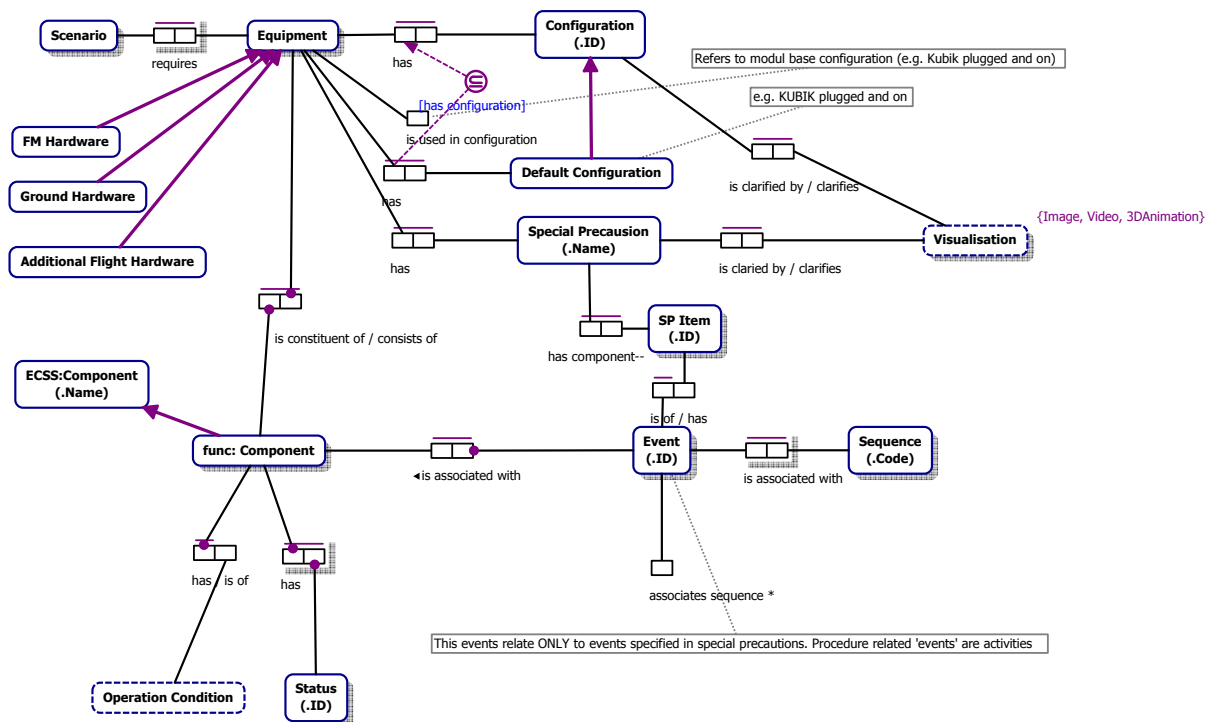


Figure 25: Model of the equipment requirements

The equipment, as extension of the generic model, can have different types of ‘Special Precautions’, like something must happen before or while as well as during a certain process. That means a special precaution also has an event. This relationship to events is required for the machine to be able to reason about possible mistakes done/occurring during the actual activities. Without this requirement we would not be required to link it to an event, since the human could just “read” the precaution but would not use the precautions for troubleshooting cases.

It is possible that some equipment’s have a default configuration and that for some default configuration, more than one equipment has that default configuration. For example the case that ‘KUBIK plugged on GSE (Ground Support Equipment) (Lepore, 2011) with status on could be the example of one of those default configurations (i.e. we may also have another ‘component plugged on GSE and on’).

Special precautions as well as default configurations are clarified using visualisations (images, videos etc.). For the agent we need more information as already described in section 10.3. Aspects of the described general structure in section 10.2 can be seen by taking a closer look at the Component entity. There exists a functional (func) component and a component with the ECSS prefix. ECSS should illustrate here the standardized upper ontology, including component knowledge that is relevant for all components that are not necessarily linked to a specific procedure (e.g. not all off-nominal situation (or nominal) will be linked or included in a specific procedure).

Events related to special precautions have to be performed in certain sequences which can be controlled checking if the user performs the right order.

#### 11.1.4 Model of the Special precaution

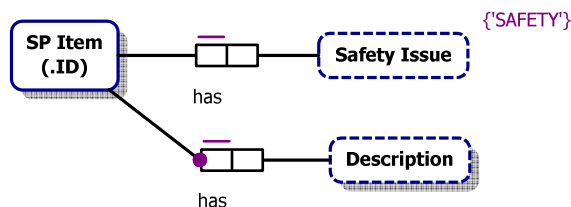


Figure 26: Special precaution

Each procedure can have special precautions, as part of their equipment configuration. A special precaution could be e.g. that the lid of KUBIK has to be closed (if it is not closed the centrifuge will not work). While some precautions are of a safety nature not to damage the hardware and are also necessary for the IOSS to reason for troubleshooting aspects, most of the precautions are of a general domain just if interest for the human. One of these general precautions e.g. would be that the lid should be closed as soon as possible, in order to preserve the internal temperature and limit the internal condensation. An agent could communicate this to the human but is unable to check it or assist in such cases without disturbing or introducing pressure to the user.

### 11.1.5 Model of the Hardware Setup

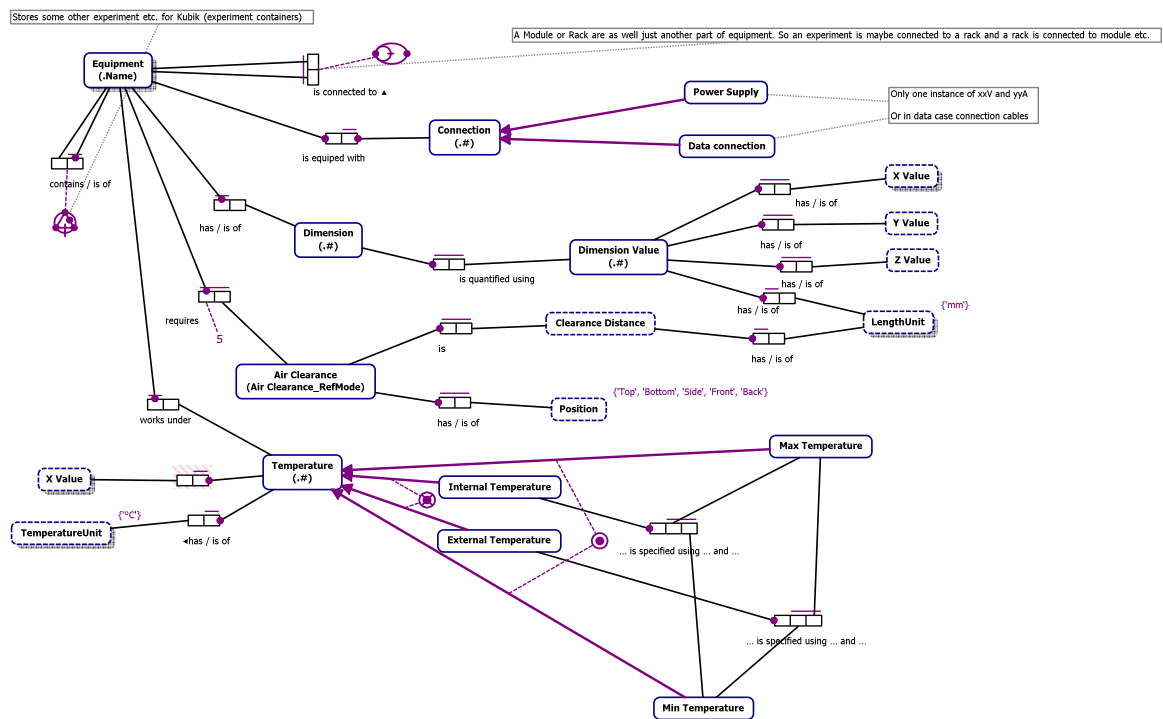


Figure 27: Hardware Setup

The hardware setup demonstrates the difference between equipment and hardware modelling. This is necessary to understand that equipment is described via functional procedural aspects and mechanical hardware aspects as indicated in section 10.1.

This hardware setup is a small scale representation of KUBIK specific knowledge. This part therefore is not viable to be used as part of the core knowledge. A good explanation why would be the temperature description example. While for KUBIK it is important to maintain proper internal (relevant for the experiment) temperature and as well as external temperatures (for the device to work) this may not be true for all components. Dimensional aspects however will be required for all components related to missions as well as connections like power as well as data connections.

This part of the model can be used by an agent to assist in setting up a component like KUBIK as well as might be used if the component fails to work. Knowledge modelling here is not directly relevant for the procedures but required for off-nominal cases.



### 11.1.6 Model of the Meta model (dynamic view)

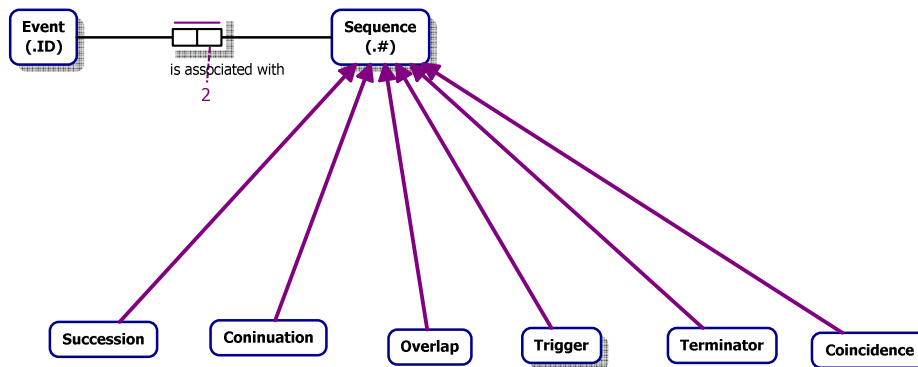


Figure 28: Meta model for sequence and the possible ‘side conditions’

As mentioned before some procedures do require events that are carried out during the whole course of the procedure (see special precautions). Others require feedback (go or no go) from MCC. Also reaching certain conditions can trigger or terminate events while the finishing of all procedural steps (without problems or errors) will lead to the succession of that scenario.

This knowledge is required for the user as well as for the agent to evaluate if an error might occur due to ignoring an e.g. overlapping configuration or a malfunction of a trigger.

### 11.1.7 Model of the Unit types

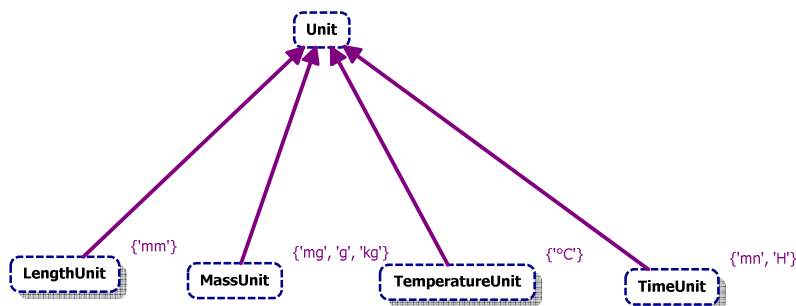


Figure 29: Illustration of the different unit types used in the model

Even though each unit, may it be a length or time unit, can be stored in one and the same unit type (data type) it was decided to further split the unit types involved in the scenarios following the proposed QUDV standard (Halpin, WP2500). In practice, many domain models include entity types that have the same name, but not the same semantics, as for example a root quantity kind. Assume a model that describes a movie. Each movie has a length (minutes: Duration). Now assume a model that describes a Person, which has a family name. That family name also has a length (characters: StringLength).

Both use “Length” to name an entity type very differently from the standard quantity kind name “Length”. This knowledge again is most likely not of direct use for the agent in troubleshooting situations, but is necessary to prevent problems like illustrated in the beginning of section 4 (Mars Climate Orbiter).

## 12 The expert system / IOSS

The interface between human and machine as indicated in the introduction was not considered in this investigation. It was assumed that the interface would be a dialogue which could also be performed using a graphical user interface (GUI) (van Dam, 2006). Such a GUI could e.g. also automatically be generated if FAMOUS (Valera & et.al., FAMOUS Statement of Work: Fact based Modelling Unifying System: Toward implementing solution for ECSS-E-TM-10-23A, 2012) would be used since the automatic generation of GUI to 'browse' the knowledge is part of the development.

After the investigation of different agent designs it appeared to be most suitable to have a combination between two 'independent' agent based systems for handling the assistance of nominal and off-nominal cases. Due to the high safety standards and requirements in space environment, in both cases the agent is not allowed (even being able to do so), and also not supposed to solve the problem alone, but only in cooperation with the human (dialogue). This cooperation can be understood from the dialogues illustrated in section 9 including proposed alternatives in case of a probabilistic reasoner.

Note that the dialogue does not have to necessarily be performed using natural language but could and maybe should<sup>26</sup> also be restricted to a smaller portion. This could be achieved using Simplified Technical English<sup>27</sup>, which is already today used in numerous aerospace related documentation.

In nominal cases and foreseeable<sup>28</sup> off-nominal cases a Belief-Desire-Intention (BDI) agent (Rao & Georgeff, 1995) assists the human. Important to note is that for nominal cases no real intelligent aspects are needed. The only required capabilities of an IOSS would be to identify the right component, understanding the task to be performed and then introduce the actions step by step to the human as in a 'simple' state based (see 4.1.4) dialogue system. Were each step of a procedure would refer to a new state. A problem with such a state based reasoner would be, that the conversation could not be pro-active (which is a requirement imposed on the system) because the IOSS would not try to reach a goal (itself).

The basic approach of troubleshooting should be a dialogue between the machine and the human as already illustrated in section 9.3. The design of such an agent in this thesis was heavily influenced by (Vergunst, 2011)<sup>29</sup>. In this publication the focus of the agent is to assist the human during the process of cooking, following a recipe. Transferred to the thesis domain this 'recipe approach' can be translated straight forward to the handling of actions included in a (e.g. ODF) procedure.

---

<sup>26</sup> For an user which has English as his mother language it might be easier to communicate its intentions than for someone who doesn't use English that much even though in the space domain all participants are well educated in English.

<sup>27</sup> [http://en.wikipedia.org/wiki/Simplified\\_Technical\\_English](http://en.wikipedia.org/wiki/Simplified_Technical_English)

<sup>28</sup> Refers here to situations that can be derived using the logic rules included in the KB not just to off-nominal cases as included in operation manuals.

<sup>29</sup> Note: The model used in 'BDI-based Generation of Robust Task-Oriented Dialogues' is too simple (the ontology). It considers ontology just as a vocabulary which is not enough (not rich enough).

Due to space specific safety aspects, the Vergunst BDI cannot be applied without changes as it even would try to change safety driven procedural steps, as part of its problem solving routine. This is absolutely not acceptable for the space domain. Therefore in case of off-nominal situation that cannot be resolved by the mentioned Vergunst BDI agent, a case-based agent (Murdock & Goel, 2001) (Valera, Case Based Reasoning Support System for Satellite AIT/AIV Diagnosis, 1992) or model based reasoner (Davis & Hamscher, 1988) suggested promising results, especially for the space domain. Case based agents have three outstanding properties which increase their performance:

- Case-based reasoners tend to become increasingly flexible over time and could prove ideal for future human interplanetary missions to come.
- This is due to the fact that over time the 'library' of cases is further refined as well as increased with new cases. Here the space domain is of a major advantage since every<sup>30</sup> off-nominal situation is logged in detail, leaving us with an already existing library that will increase in size allowing access to more and more possible domain comprehensive solutions.
- Case-based reasoners are able to 'redesign' themselves. E.g. if our case library only includes disassembly cases it is possible to also handle assembly cases by inverting the actions to be performed (under the assumption that the component and steps are not completely different).

This could result in a similar algorithm as proposed in (Murdock & Goel, 2001):

```

Input: main-task: A task to be executed
       initial-condition: The input knowledge for the main-task
IF [main-task does not have a known implementation] THEN
  IF [there is a known-task which is similar to main-task] THEN
    [adapt the methods of that known-task to perform main-task]
  ELSE
    [build a method for main-task from scratch]
REPEAT
  [execute main-task under initial-condition]
  IF [execution has failed] THEN
    [modify the implementation of main-task]
UNTIL [execution has succeeded]
SAVE [add result as new known implementation]

```

(In short Retrieve->Reuse->Revise->Retain)

For a BDI agent to reason with case-based reasoning is straight forward (Corchado & Laza, 2002). The definition of an BDI agent mental attitudes is: Belief <state> Desire <final state> Intention <action>

Note that an intention is an ordered set of actions, where the agent 'remembers' each action it has performed and the state is was in during that action as well as the result of this action.

On the contrary a case in a case-based reasoning system is: Problem <initial state> Solution <action, (intermediate state)> Result <final state>

---

<sup>30</sup> While on earth this is not always the case. As a good example each support service dealing with computer problems usually starts with the question 'Did you already try a reboot' before even investigating the problem at hand.

Note that the solution *action* may be repeated multiple times while intermediate states are also optional.

Looking at above relationship between BDI and case-based reasoner it is obvious that both could be converted into each other. Thus supporting the idea of creating a case-based reasoner with BDI goal oriented properties.

Important however would be that the a database with case memory would be created which would include a 3-tuple  $\langle B, D, I \rangle$  representation of past belief, desire and intentions where the belief would be a 2-tuple  $\langle E, Eval \rangle$ . E describing the environment, and Eval describing each value on certain instances. Though saving this information not just the library for off-nominal cases is updated but new 'foreseeable' solutions can be added and included for the nominal BDI agent. A basic example of such a troubleshooting is illustrated in section 18.2, Scenario 4.

Also if the knowledge in the functional model is thorough enough to provide large pieces of the solution being able to narrowing down the problem, other system (with less computational cost) may be used to solve the problem allowing the case-based reasoner also to function as some kind of 'stepping stone'. Remember (section 4.2.4) and the hints to a neural network that could for example be used as a fuzzy reasoner. As long as the result would fall into its reasoning domain it could be used as a possible third reasoner (more in future research).

Decisions which could not be resolved by the model or forwarded to another reasoner are addressed through reinforcement learning (as a simple example image Q-Learning) during this execution process (Murdock & Goel, 2001) (see the REPEAT aspect of the illustrated algorithm), or result in the human deciding to drop the troubleshooting process and forwarding the problem to the MCC.

Model-based reasoners (Davis & Hamscher, 1988) generally perform some sort of diagnosis. First a hypothesis (create model) is generated, followed by a testing (simulation) of this hypotheses and comparison with the real observation (reality). The result of this process is the discrimination of 'impossible' and 'possible' cases. Like in the illustrated scenario in section 9 this could be a narrowing down of possible error cases. With an elaborated KB this could continue until the reasoner would run out of possible hypothesis to test.

## 13 Evaluation of achievements with respect to goals and requirements

### 13.1 Achievement of Goals

The main Purpose of this thesis has been to develop a MECA Lite version. The challenge has been to create a 'Multifunctional Model Optimizing'. The possibility of developing a model which could replace different existing models (ER, UML, OWL etc.) but still maintains the capability to deduce the generated information of the new model into these older, already scientifically approved and existing models had to be proved. A model that would help to ease the usage in the context of man machine interfaces.

	<b>Goals</b>	<b>Results</b>
1.	Knowledge Interoperability Infrastructure (KII) with following properties <ul style="list-style-type: none"> <li>– Ontology based</li> <li>– Information push and pull</li> <li>– User adaptation</li> <li>– Multi device</li> <li>– Policy management</li> </ul>	Qualified successful <ul style="list-style-type: none"> <li>-Ontology fulfilled</li> <li>-Push &amp; Pull fulfilled</li> <li>-User Adap. limited fulfilled</li> <li>-Multi Dev. Only referenced</li> <li>-Policy Mgmt. fulfilled</li> </ul>
2.	Considerations and requirements to identify relevant use cases and scenarios	Successful – see section 7, 8
3.	Propose a suitable formalism for Knowledge Interoperability and Representation	Successful – see section 4.2.4, 4.2.5, 9.7, 10
4.	Evaluation of formalism in practice by building an example knowledge base	Successful – see section 10, 11
5.	Demonstration of KB (knowledge base)usage (MM I/F)	Successful – see section 9
6.	Introduction of a Multifunctional Model Optimization (MMO)	Successful – see section 4.1.3, 4.2.5, 10

## 13.2 Requirements

Not all requirements listed concerning the “resulting system” could be fulfilled in this thesis, but enumerative they should be valid as evaluation criteria in presented results as well as future developments enhancing results presented in this thesis.

	<b>Requirements</b>	<b>Results</b>
	<b>The resulting system:</b>	
1.	... should be generic for several application areas	Solution – see section 4.1.3
2.	... should be used by Human and Machine	Solution – see section 9
3.	... should have the capability to update knowledge	Solution – see section 12
4.	... has to have the capability to forward knowledge to other systems (computer, ePartner) and humans to ensure availability of knowledge (necessary data), at the right time and point for the right person	No infrastructure solution from agent to agent communication integrated; Man machine communication via dialogue integrated.
5.	... should be capable of self-learning, updating knowledge without being told to. However updated locally and verified by a human (controlled update).	Solution – see section 12. Remark: Due to space environment human involvement for critical solutions indispensable.
6.	... should follow domain specific rules.	Not applied within thesis
7.	... should be capable of solving problems (i.e. troubleshooting).	Solution – see section 9, 12
8.	... should include safety aspects, in order to be able to be controllable	Safety aspects can be included in the model or agent design but were not investigated in depth.
9.	... should distribute biunique (i.e. clearly identifiable) knowledge.	Solution – see section 9.7
10.	... should guarantee reliability	Could not be tested
11.	... should be reusable!	Solution – see section 10
12.	... should produce the results in a reasonable computation time	Not investigated since no prototype was developed
13.	... should allow for dynamic knowledge forwarding to other systems, e. g. transfer to UML	Solution – see section 4.1.3
14.	. ... should be easily understandable by the user (e.g. good GUI)	Not investigate in depth

**Further remarks:**

ECSS standards, from a developer's point of view, are a double edged sword. On one side they define a European standard and are thus giving common rules to all engineers. Engineers from different countries, with different languages and different cultural and educational backgrounds. Thus leading to a harmonized approach for general solving problems in demanding space projects, they deed form themselves a KB on their own, especially for the space community (but also for other domains). This positive aspect may put on the other side additional cost to the development and definitely 'cut the wings' of developers as there is no degree of freedom remaining because they are forced to apply them. It would be a future demanding request to define the ECSS KB with a self-learning and adapting mechanism in order to facilitate development.

The currently available tools (NORMA, DOGMA, CogNIAM etc.) using ORM are all having (at least) one property that is required or beneficial to the ontology. Might it be for inter human interaction or for the agent to communicate with the human. FAMOUS hopefully will finally combine all the necessary characteristics into one application.

The boundary conditions of work (e.g. ECSS, security concerns, etc.) in the space domain were heavily underestimated in initial schedule as well as during the course of the project, resulting rather in a scale down than a scale up of the planned prototype system.

## 14 Conclusion and future work

Taking into account the above described achievements with respect to the original ideas and intentions of this thesis it has to be said that, even if this thesis is only covering a simplified KB, the results are paving the way and forming a major improvement for future developments and analysis. That is the reason the project was named *MECA-lite*.

Even though the research presented in this paper focus on technical knowledge related to malfunctions or assistance in man machine interactions, the system is extendable to capture the whole space domain. Space domain, here also referring to capture the current health state of an astronaut including also his cognitive task level. This actually means that we can integrate into the ontology next to the technical information also different persons or status of different persons, state of minds as well as physical locations of mentioned entities and any kind equipment.

My proposed MMO concept has demonstrated that it is of utmost importance for communication interchangeability to offer an overall system instead of continuing with a bulk of isolated applications or distributed self-standing solutions

The key benefits of this thesis can be summarized as follows:

- Procedures can be captured using an ontology
- We can create a proper IOSS that supports
  - o Nominal procedural handling
  - o Assists in off-nominal situations
  - o Remembers solutions of off-nominal or improvements for nominal situations
- A more detailed defined ontology can generate ODF standards documents. Due to the limited frame of the thesis the presented ontology however only includes a small part of the spacecraft knowledge.
- A more detailed defined ontology can exchange operation manuals or provide optimized templates to possible user in order to facilitate their inputs to be translated faster into the ontology, improve and enhance it. Thus the user does not need any more exact knowledge about modeling.
- An Ontology provides improved understanding of specification between the different participants in a space activity (departments,; user; industrial contractor etc.)
- Depth of instructions can be adjusted via a cognitive task model (Neerincx, Smets, Breebaart, Soler, Plassmeier, & van Diggelen, 2013)

Furthermore it could be shown, that the thesis could be a first step for creating a new standard system. In order to realize such a system a core KB including all knowledge related to ODF and procedures in general has to be build-up. For procedure related knowledge the presented KB could be reused as a baseline. Next, an expert system as communication tool and help for the astronaut, (interface still needs to be defined) has to be realized as well as making the KB accessible to this expert system. From this point onwards an assistance in nominal and known off-nominal situations is already given to the user. For better troubleshooting however the expert system would need to be extended with a case-based reasoner (CBR) or model based reasoner approach. Followed by the creation of an off-nominal situation DB, storing the knowledge for a CBR system. From this point onwards only the extension of the KB with more components is required as well as the creation of reasonable templates for contractors to allow for easier input/transfer of knowledge. But there are still a lot of tasks open and it will take some time and further huge efforts to end up



with a system that would not only be able to cover the defined requirements and needs for space applications but will also be a self-learning, self-adapting and self-regulating overall system.

Future work for the near and mid-term will/should be:

- Correct the errors found / include the suggestions (end of section 4.2.4) for the KFO if it is still intended to be reused
- Link KFO and MECA lite ontology after finding proper 'root'
- Create DB with previous off-nominal cases readable by CBR. Note however that this would include the remodeling of the existing off-nominal case database used by ESA to make it machine readable, which is right now not the case. Further to provide the CBR with knowledge for new components an in depth research of those components could be performed using model-based reasoners as an addition to the current FMECA analyses.
- Improve verbalization to also include a closer to natural language version like Standard technical English(STE). This could improve the man-machine communication since the agent would be capable to translate the natural language of the user into logical rules as well as would be capable to communicate in a closer to natural language form, not limiting the verbalization anymore to pure logic.
- Identification/Creation of core knowledge
- Standardize operation manuals so they can be automatically transferred into the model. Two solutions seem promising. The first would be to already create the manual close to ODF standards (like the KUBIK manual). Or write them clear enough that so that e.g. the Flemish OntoBasis project (appendix 18.1) could be used to generate the model with close to 100% of the content. In both cases a review by a knowledge engineer will however be required.
- In the frame of creation of a suitable interface between human and machine. I would propose a combination of headset and touchscreen to cover all required aspects of a dialogue. In cases where the user needs both hands to fulfill a task it will be helpful if he can request information verbally. And for situations that appear more often, like e.g. the navigation to select which procedure to execute next, the input via touchscreen should be faster and less time consuming. The touchscreen also serves as a backup in case that the voice recognition does not perform as required.
- Due to lack of the necessary environment (no centralized KB existing) the aspect of self-learning and self-adapting expert systems could not be analyzed within the frame of this thesis but will be the future challenging tasks, especially for long term deep space missions. This could be done in a stepped approach by building up dedicated knowledge branches (e.g. mechanical knowledge, physical knowledge, etc).

Reviewing the results of this thesis it is important to note, that even with the currently available tools like e.g. NORMA a full functional prototype could be created as a further practical proof of concept, however missing certain modalities that would improve man-machine interaction.

Today it is of utmost importance to have experienced knowledge engineers to find proper solutions for complex challenges as analyzed within the thesis. But once a future complete solution will be found, "model experience" will only be limited necessary.

## 15 Acronyms

Acronym	Explanation
AI	Artificial Intelligence
APID	Application Process Identification
ATV	Automatic Transfer Vehicle
BDI	Belief / Desire / Intention
CC	Control Center
CTL	Cognitive Task Load
CRUISE	CRew User Interface System Enhancements
ECG	Electro CardioGram
ECSS	European Cooperation for Space Standardization
EDL	Entry, Descent, Landing
EDR	European Drawer Rack
EVA	Extra Vehicular Activity
ER	Entity Relationship
ESA	European Space Agency
ETECA	Expert Tool to Support Crew Autonomous Operations in Complex Human Spacecraft
FBM	Fact Base Modelling
ePartner	Electronic Partner
GOAL	Goal-Oriented Agent Language
GPS	Global Positioning System
GUI	Graphical User Interface
IOSS	Intelligent Op Support System
ISS	International Space Station
KB	Knowledge Base
KII	Knowledge Interoperability Infrastructure
KFO	KII Foundation Ontology
LOS	Loss of Signal
MCC	Mission Control Center
MECA	Mission Execution Crew Assistant
MMI	Man-Machine Interface
MMO	Multifunctional Model Optimization
NASA	National Aeronautics and Space Administration
OO	Object Oriented
ORM	Object-Role Modelling
OTDS	Operation and Training Docs and Specs
ODF	Operation Data File
OS	Operation System
OWL	Web Ontology Language
RuleML	Rule Mark-up Language
sCE	Situated Cognitive Engineering methodology
SICRA	Intelligent Avalanche Collaborative Rescue System
TM	Telemetry
TN	Technical Note
UML	Unified Modelling Language
UoD	Universe of Discourse
XML	eXtensible Markup Language

## 16 Glossary

The glossary defines the terms as I will be using them in my description of the project. Some of the definitions given in this glossary are adapted for the KII project from definitions used in projects such as MECA.

Term	Description
<i>Actor / Agent</i>	An entity that (1.) has Knowledge and that (2.) is capable of communicating this Knowledge to another entity. Examples of Actors are crew members, smart rovers, and personal IOSS/ePartners software agents. The term 'Actor' will sometimes be preferred over 'agent', to indicate that the entity under discussion is not just software but can also be human. Mostly, it should be clear from context whether the use of 'agent' in this document refers to software agents only, or is used as a synonym for Actor.
<i>Domain</i>	Domain here does not refer to the computer networking aspects but to a field or scope of knowledge referring to a set of all possible values (knowledge) that an independent variable (actor/agent) can take in a specific function.
<i>Object</i>	Entities that are not Actors. Objects (also known as Resources) can be observed, used, assigned certain roles ('tool', 'food') or even diagnosed – but they cannot autonomously engage in communication with an Actor. Sensors, torches, rocks, inventory items, but also external entities such as the atmosphere or sand dunes can be Objects. It is possible for an Actor to function as an Object in certain scenarios. In that sense, the 'Actor-ness' or 'Object-ness' of an entity is itself more a role than an intrinsic property. A human crew member can be an Actor in one scenario (e.g. when performing a payload experiment), and an Object in another (e.g. when being operated upon by a doctor).
<i>Knowledge</i>	The collection of statements and beliefs that defines the context in which an Actor operates, i.e. its concept of the current state of the System under Consideration. Raw sensor data, interpreted sensor data, models, operational procedures, schedules, and justifications are all examples of Knowledge. Note that knowledge does not have to be true – it is entirely possible for an Actor to know (and to communicate)

	falsehoods or undecidable proposition.
<i>Knowledge infrastructure</i>	A collection of resources and processes that support organising, accessing and manipulating Knowledge in order to facilitate learning, collaboration , and action towards a desired collective future (state)
<i>Knowledge base</i>	A computer database for storing Knowledge (using a form of Knowledge Representation)
<i>Knowledge representation</i>	The formalism in which Knowledge (symbols describing statements about the world) is represented in a Knowledge Base, so that software agents can accurately and effectively reason about the world. Knowledge representation should be computer-interpretable and system-independent
<i>Goal</i>	The goal of greatest interest, i.e. the intention an Actor or Team in a Use case wishes to achieve. The goal is typically modelled as a desired end state for the System under Consideration.
<i>Use case</i>	A Goal-oriented set of interactions between a Team of Actors and the system under consideration. Use cases and Goals can be extremely high level (“Establish a human habitat on Mars”), or very low level (“Execute operational procedure X on payload P”). Typically, a Use case will lead to a collection of Scenarios that cover that particular Goal.
<i>Scenario</i>	A complete sequence of actions and interactions that illustrate on particular path or occurrence in a Use case. A Scenario can be seen as a kind of instantiation of (part of) a Use case.
<i>Plan</i>	A sequence of step-by-step actions that an Actor can execute in order to implement (or as part of implementing) an Intention.

## 17 Bibliography

- Abiteboul, S., Hull, R., & V., V. (1995). *Foundation of Databases*. Addison-Wesley.
- Aerospace, C. (2007). *KUBIK Operations Manual*. Comat Aerospace.
- Asberto, A. (2012). *Integrated EDR/FASTER Operations Manual*. Internal Thales Alenia Space.
- Bernstein, J. H. (2009). *The Data-Information-Knowledge-Wisdom Hierachy and its Antithesis*. Proceedings North American Symposium on Knowledge Prganization, Vo l2.
- Borgovini, R., Pemberton, S., & Rossi, M. (1993). *FMECA: Failure Mode, Effects, and Criticality Analysis*. Rome: Reliability Analysis Center.
- Chen, P. (1976). Entity-relationship model: towards an unified view of data. *ACM Trans. Database Syst.*, 9-36.
- Corchado, J., & Laza, R. (2002). Construction of BDI Agents from CBR systems. *1st German Workshoop on Experience Management*. Berlin.
- CRUISE, T. (2013). *CRUISE Final Report*. Astrium / ESA.
- Davis, R., & Hamscher, W. C. (1988). *Model-based Reasoning: Troubleshooting*. Massachusetts: Insitute of Technology: Artificial Intelligence Laboratory.
- ECSS. (2011). *Space engineering: Space system data repository (ECSS-E-TM-10-23A)*. Noordwijk: ESA-ESTEC.
- ECSS. (2012). *ECSS system: Glossary of terms*. Noordwijk: ESA-ESTEC.
- ESA. (2008, June 23). Support to ESA-NASA collaboration activities in preparation of Space Exploration Programme. *Technical note part 3,4*.
- ESA. (n.d.). *KUBIK Incubator inside the European Drawer Rack*. ESA.
- Halpin, T. (1998). Object-Role Modelling. In K. Mertins, & G.Schmidt, *Handbook on Architectures of Information Systems* (p. Ch.4). Springer Verlag.
- Halpin, T. (2005). *Fact-Orientation Meets Agent-Orientation*. Springer Verlag.
- Halpin, T. (2005). *ORM 2 Graphical notation*. Neumont University: Springer Verlag.
- Halpin, T. (n.d.). *WP2500*.
- Halpin, T., & Morgan, T. (2008). *Information modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann.
- ISOTR9007. (1987). *Information processing systems - Concepts and terminology for the conceptual schema and the information base*. ISO.
- Lee, M. (2000). *Model-based reasoning: a principle approach for software engineering*. University of Wales: Centre for Intelligent Systems, Department of Computer Science.

- Lepore, P. (2011). *EDR and GSE Operations Manual*. Internal Thales Alenia Space.
- Lyndon, B. (1998). *International Space Station Familiarization: Mission Operations, Directorate, Space Flight Training Division*. Johnson Space Center: National Aeronautics and Space Administration.
- Meersman, R. (1999). Semantic Ontology Tools in IS Design. *ISMIS 99 Conference* (pp. 30-45). London: Springer-Verlag.
- Murdock, J. W., & Goel, A. K. (2001). *Meta-case-Based Reasoning: Using Functional models to Adapt Case-Based Agents*. Springer Verlag.
- NASA. (2013). *Operations Data File Standards: International Space Station Program*. Johnson Space Center: National Aeronautics and Space Administration.
- Neerinx, M. (2011). Situated Cognitive Engineering for Crew Support in Space. In *Personal and Ubiquitous Computing Volume 15 Issue 5* (pp. 445-456).
- Neerinx, M., Smets, N., Breebaart, I., Soler, A. O., Plassmeier, F., & van Diggelen, J. (2013). *MECA: Testing ePartners for Missions beyond Low-Earth in the International Space Station*.
- NORMA for Visual Studio*. (2013). Retrieved November 2013, from <http://www.ormfoundation.org/>
- Protege-OWL ontology editor*. (2013). Retrieved November 2013, from <http://protege.stanford.edu/>
- Rao, A. S., & Georgeff, M. P. (1995). *BDI Agents: From Theory to Practice*. Australia: Artificial Intelligence Institute.
- RuleML. (2013). *The Rule Markup Initiative*. Retrieved November 2013, from <http://ruleml.org/>
- S&T. (2012). *KII - TN1: Scenarios and Use-cases*. S&T.
- S&T. (2012). *KII - TN2: Formats and Formalisms*. S&T.
- S&T. (2012). *KII - TN3: Design Considerations*. S&T.
- Sowa, J. (1984). *Conceptual structures - Information processing in Mind and Machine*. London: Addison-Welsey.
- Spyns, P., Tang, Y., & Meersman, R. (2008). An Ontology Engineering Methodology for DOGMA. *Journal of Applied Ontology*, 13-39.
- Tang, Y. (2010). *Semantic Decision Tables - A new, Promising and Practical Way to Organizing Your Buisness Semantics with Exisiting Decision making Tools*. Saarbruecken: LAP LAMBERT Academic Publishing AG & Co. KG.
- Tang, Y., & Debruyne, C. (2013). *SDRule-L: Managing Semantically Rich Buisness Decision Processes*. Semantics Technology and Applications Research Laboratory.

- Tang, Y., Spyns, P., & Meersman, R. (2007). *Towards semantically grounded decision rules using ORM+*. Semantics Technology and Applications Research Laboratory.
- Valera, S. (1992). *Case Based Reasoning Support System for Satellite AIT/AIV Diagnosis*. Noordwijk: ESA.
- Valera, S., & et.al. (2011). *Fact Based Modelling: Exchanging Conceptual Data Models*. ESA.
- Valera, S., & et.al. (2012). *FAMOUS Statement of Work: Fact based Modelling Unifying System: Toward implementing solution for ECSS-E-TM-10-23A*. ESA.
- Valera, S., Kueke, R., & Smith-Meyer, H. (1993). ESA EXACT-Expert Assistance in Conduction Tests. *ESA's Technology Programme Quarterly Vol.3 No.4*, 18,19.
- van Dam, H. (2006). *Dialogue Acts in GUIs*. TU Eindhoven: Eindhoven.
- Vergunst, N. L. (2011). *BDI-based Generation of Robust Task-Oriented Dialogues*. University Utrecht.
- W3C. (2009). *OWL 2 Web ontology Language Document Overview*. Retrieved October 2009, from W3C Recommendation: <http://www.w3.org/TR/owl2-overview>
- РКК "ЭНЕРГИЯ", S. K. (2009). *Service Module: Long Duration Mission*.

## 18 Appendix

### 18.1 ORM Software

#### **NORMA**

NORMA is an extension for Microsoft Visual Studio, developed in C#, that allows to create ORMs. The 'commercial' version has some restrictions concerning e.g. the specification and formulation of global rules for the model, while however there are no restrictions for the actual ORM model. NORMA comes also with an ORM Verbalization Browser which allows as the name indicates for verbalisation of 'local' constraints and rules. This verbalization is one of the most powerful aspects of the tool since it allows a standardized (except domain expert relationship vocabulary) formulation of the model.

#### **Flemish OntoBasis project**

Allows to generate from a verbalization an ORM model. It also is able to generate a model from a 'common' text which is not restricted to a standardized vocabulary. In such cases about 80% of the written text can be identified and translated into a model including the relations between the described entities.

#### **FAMOUS**

Is a tool that is developed by ESA. Except from the features already described in other tools it will also be able to provide auto generated interfaces. The target for these generated interfaces is to provide the potential user with a web page like button browsing application that could be visualized on a smartphone or tablet device.



## 18.2 Possible scenario dialogues

### Scenario 1)

COMPUTER → Screw screw2 to fix the insert  
USER → Not possible! Does not seem to work  
COMPUTER → Are you turning it clockwise?  
USER → Yes  
COMPUTER → Is the screw damaged?  
USER: Examines the screw and sees it is in fact damaged  
USER → Yes  
SYSTEM: Checks what kind of screw is required.  
SYSTEM: Checks if there are still spare parts.  
COMPUTER → to my knowledge there are no spare screw but please confirm it by checking positions XY  
USER → No spares left that is right!  
COMPUTER → Can you order spares or do you wish to continue  
USER → I need to finish this so please continue  
SYSTEM: Checks where those screws are used for other components  
COMPUTER → I compiled a list of possible places to take a spare please verify.  
SYSTEM: Prints list  
USER → Can you give me a timeline when the components will be used again  
SYSTEM: Checks planned actions with other components and adds it to the illustration  
USER → I will take a spare from ZZ  
COMPUTER → Which screw are you taking from ZZ  
USER → The screw on the top lid in the front left  
SYSTEM: Add which screw is missing to the components procedures  
COMPUTER → I added the missing screw to the procedure, shall I request spares for the next delivery?  
USER → Yes, please do so  
SYSTEM: Adds spare request.  
USER: Finishes procedure with acquired screw.

### Scenario 4)

COMPUTER → Is the LED ON?  
USER → No  
SYSTEM: LEDlit → False  
COMPUTER → Is the Power Connected?  
USER → YES  
SYSTEM: PWRin → TRUE  
COMPUTER → Is the Switch in the On Position?  
USER → YES  
SYSTEM: SW1 → TRUE  
COMPUTER → Is the Power Connected to SUP on EDR Rack?  
USER → Yes  
SYSTEM: SUPconn → TRUE  
COMPUTER → Is the EDR SUP configured to provide 28V?  
USER → Yes  
COMPUTER → Please start the 'in depth' troubleshooting agent!  
SYSTEM: Checks cases related to this problem

COMPUTER → I have found 2 cases related to this problem. 1. Related to Satellite Alphasat  
2. Related to Drawer Rack COL-4. where 1. Does not require spare parts 2. Requires spare parts. Would you like to choose one?

USER → Just start with the first case I have no Idea how this component works either way and am not sure if I have the spare parts.

COMPUTER → Please put Switch in Off Position

USER → Done

COMPUTER → Please disconnect KUBIK from Power

USER → Done

SYSTEM: Start procedure of removing ebox

USER → I can't remove the ebox!

COMPUTER → Have you unscrewed the ebox?

USER → YES

COMPUTER → Please make sure you remove the complete ebox in a straight manner. It may be necessary to use some force for this task

USER → Ah got it, was just a bit stuck.

COMPUTER → I do not understand? Have you succeeded in removing the ebox? Or did you find a new problem?

USER → YES succeeded and NO, NO new Problem!

COMPUTER → Please clean the connection for the ebox and KUBIK

USER → Done

SYSTEM: Start procedure attaching ebox

USER → Done

COMPUTER → Please connect the Power

USER → Done

COMPUTER → Please put switch in the On position.

USER → Done

COMPUTER → Is the problem resolved?

USER → Interestingly yes...

COMPUTER → The problem was a dirty/lose connection as for Alphasat, case will be saved.

SYSTEM: Saves result.

### 18.3 Scenario 3 example relationships Verbalized

If we translate this functional scenario into FBM (Note that this is a very limited model and a different then used in 8.7) the resulting model would be Figure 30.

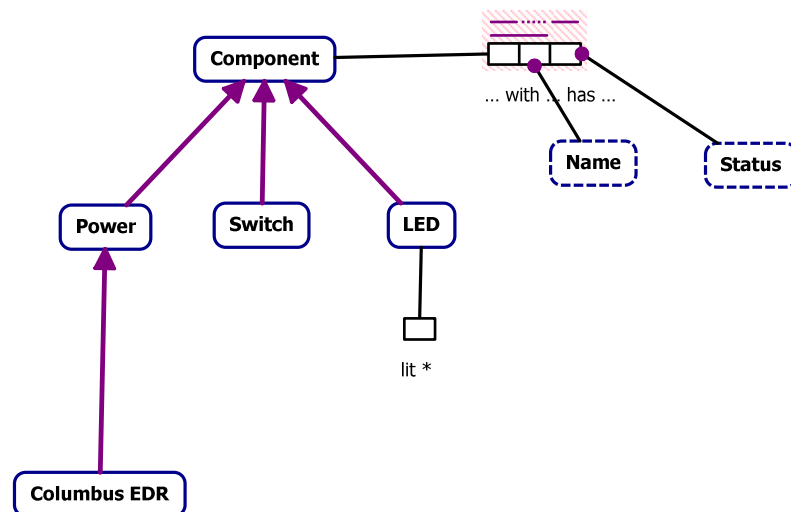


Figure 30: Possible (simplified) model of scenario 3

After entering the functional rules (Note that we will only focus on  $PWR_{in} \wedge SW1 \wedge LED_{ok} \rightarrow LED_{lit}$  here) we can create the following illustration/verbalization<sup>31</sup> of those.

\*LED<sub>1</sub> lit if and only if

that LED<sub>1</sub> is some Component<sub>1</sub> that with some Name<sub>1</sub> has some Status <true or false is unknown> where the possible value of that Name<sub>1</sub> is 'LEDlid'

if {for some Power, that Power is some Component<sub>2</sub> that with some Name<sub>2</sub> has some Status<sub>1</sub> where the possible value of that Name<sub>2</sub> is 'PWRin' where the possible value of that Status<sub>1</sub> is 'True' and for some Switch, that Switch is some Component<sub>3</sub> that with some Name<sub>3</sub> has some Status<sub>2</sub> where the possible value of that Name<sub>3</sub> is 'SW1' where the possible value of that Status<sub>2</sub> is 'True' and for some LED<sub>2</sub>, that LED<sub>2</sub> is some Component<sub>4</sub> that with some Name<sub>4</sub> has some Status<sub>3</sub> where the possible value of that Name<sub>4</sub> is 'LEDok' where the possible value of that Status<sub>3</sub> is 'True'.}

a better one would be

LED<sub>1</sub> is some Component<sub>1</sub> that with some Name<sub>1</sub> has some Status where the possible value of that Name<sub>1</sub> is 'LEDlid' and the possible value of that Status<sub>1</sub> is 'True'

if {for some Power, that Power is some Component<sub>2</sub> that with some Name<sub>2</sub> has some Status<sub>1</sub> where the possible value of that Name<sub>2</sub> is 'PWRin' and the possible value of that Status<sub>1</sub> is 'True' and for some Switch, that Switch is some Component<sub>3</sub> that with some Name<sub>3</sub> has some Status<sub>2</sub> where the possible value of that Name<sub>3</sub> is 'SW1' and the possible value of that Status<sub>2</sub> is 'True' and for some LED<sub>2</sub>, that LED<sub>2</sub> is some Component<sub>4</sub> that with some Name<sub>4</sub> has some Status<sub>3</sub> where the possible value of that Name<sub>4</sub> is 'LEDok' and the possible value of that Status<sub>3</sub> is 'True'.}

In the ESA developed tool FAMOUS, it is designed to be

LED<sub>1</sub> is a Component<sub>1</sub> with Name <LEDlid> that has Status <True>  
if {  
there exists some Power, that Power is a Component with Name <PWRin> that has Status <True>  
and

<sup>31</sup> Note that for the current verbalization the tool NORMA was used, not inherently not meaning that the final version should look like this as well.

```
there exists some Switch, that Switch is a Component with Name <SW1> that has Status <True>
and
there exists some LED2, that LED2 is a Component with Name<LEDok> that has Status <True>.
}
```

The FAMOUS design is the most useful since it is not mentioning too many redundant and unimportant information. This printout of the rules could be directly given to the human to evaluate it step by step or it could, as described above, be forwarded to a reasoning engine to start the described dialogue.

Note however that the EDR Columbus is not presented here. It is more viable to create a model for the Columbus EDR itself and link it to the physical model of the KUBIK module.