Ontology-Based Modelling of Sensor and Data Processing Ressources Using OWL

A Proof of Concept

Denis Smirnov and Peter Stütz
Institute of Flight Systems
University of the Bundeswehr Munich
Neubiberg, Germany
e-mail: denis.smirnov@unibw.de, peter.stuetz@unibw.de

***Abstract*—In this paper, we describe an ontology-based system to inventory and model installed sensor and respective data processing resources on-board airborne surveillance aircrafts. The algorithms are packaged and described in form of discrete processing modules, each representing a low level image processing step. While the implementation of the algorithms is kept detached in a separate library, the description of modules and its parameters are stored in an ontology, representing a knowledge database using Web Ontology Language as a knowledge representation language. Based on the module description stored in the knowledge database, it is possible to identify and manage processing chains capable of solving complex image processing tasks.**

Keywords-Ontology; Web Ontology Language (OWL); Image Processing Management; Knowledge Management; sensor and data ressources.

I. INTRODUCTION

Presently we witness an increasing demand for highly automated deployment of heterogeneous sensors on-board unmanned aircraft, either to yield better environmental awareness in the context of collision free flight or, as in the given case, to conduct typical Intelligence, Surveillance & Reconnaissance (ISR) missions in a more automated fashion, thereby relying mostly on imaging sensors operating in various spectral regions. However in aviation space, power and processing resources are limited. Therefore it is necessary to work economically with resources and manage them in an intelligent way. Furthermore, the sensor data processing and evaluation on-board a flying platform takes place under changing circumstances for example resulting from changing position and orientation of the Unmanned Aerial Vehicle (UAV), varying lighting conditions and different surface backgrounds (e.g., rural, urban, maritime). To cope with this situation it is meaningful to have a wide set of different sensors and associated data processing algorithms, since there is no algorithm that performs in an adequate way in every situation. For a complex image processing task (e.g., vehicle- and person detection) there are several equal processing steps, which have to be executed for each task, e.g., preprocessing steps or region-of-interest (ROI) selection. When executing multiple tasks one can save resources and computational time reusing the processing steps, which are required repeatedly instead of starting the same algorithm multiple times. Thus, there is a need for a system that manages sensor resources and image processing capabilities in a meaningful way. The Institute of Flight Systems published several papers ([1]–[5]) on the topic of airborne sensor- and perception management. In this paper we now focus on the ontology based knowledge extension of the so called *Sensor and Perception Management System (S&PMS)*, first introduced in [4].

The S&PMS is best described as a three layer architecture to inventory relevant resources (e.g., sensors and image processing algorithms) and manage their usage as shown in Figure 1.

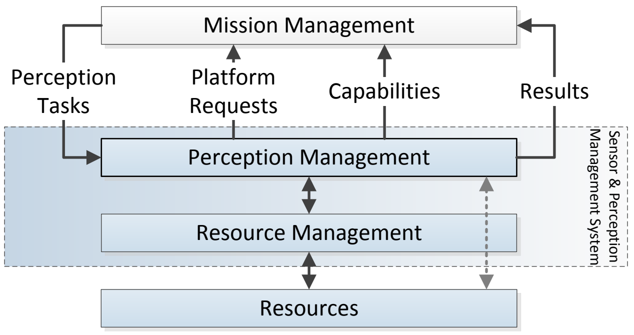


Figure 1. Three layer architecture of the Sensor and Perception Management System.

One of the S&PMS's key aspects is the module based approach to package image processing algorithms into perception modules. Each module fulfils a special low level image processing requirement, e.g., noise reduction or ROI. Modules are designed to be standalone or to be combined with different modules to solve a higher level image processing task (*perception task*), like vehicle detection within a given street segment. The combination of at least two low level modules or a sensor-module combination creates a *perception chain*. Eventually a considerable variety of different perceptions chains (redundant chains) results, which potentially solve the same perception task, based on the (sensor) configuration of the UAV and the available perception modules. This entirety of resources (modules, sensors, etc.) and possible combinations is called *perception graph*. Such graph can be used to visualize all possible chain combinations for different perception tasks.

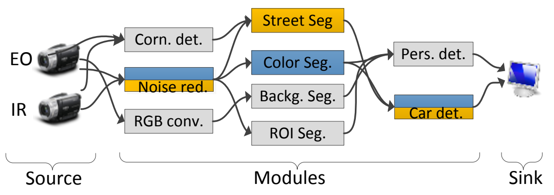


Figure 2. Perception graph. Interaction between sources, perception modules and possible chaining.

Figure 2. illustrates the relationship between sensors, the perception- modules, chains and graph. The graph is represented by the infrared (IR) / electro-optical (EO) sensor, all rectangles, each one providing a different low level capability, and their connections. Furthermore, there are two different perception chains in the figure to detect, e.g., a car from on-board the aircraft, sharing the first and last module (blue and orange rectangles). Both chains can use either an electro-optical sensor or an infrared sensor.

Packaging image processing algorithms into perception modules allow the interpretation of algorithms as capabilities. It also supports reusability and therefore limits the power consumption and processing power onboard UAVs. Figure 2. shows several approaches to achieve the same goal (e.g., detecting a vehicle). A goal oriented usage of perception chains is enabled through detailed descriptions of the modules containing:

- What is the output of a module (*Capabilities*)
- How can a module be combined in a meaningful way (*Requirements*)
- Under which circumstances can a module be used (*Constraints*)

Furthermore, there is a need for a managing entity that loads and interprets the module descriptions to create relational knowledge. Such knowledge is used to identify the availability of low level and high level capabilities depending on given circumstances. The statements also provide information about possible module combinations to create perception chains that provide high level capabilities (HLCs). These high level capabilities in turn can be used to achieve different perception tasks.

In this paper we therefore propose an ontology based approach, using the knowledge representation language OWL, that has been first introduced in [1], to create a knowledge database. This knowledgebase can be inferred by a reasoning mechanism to create statements, describing which resources are available and how they can be used.

This paper is structured as follows: The structure and concept of the presented ontology is being explained in section 2. In section 2.A the class taxonomy is being presented. Next, in section 2.B we will describe the differences between persistent and dynamic individuals. In section 2.C we get into detail with the *resource-capability-resource* concept. The data and object properties are discussed in section 2.D. In the last part of section 2 we describe the rules of the Semantic Web Rules Language (SWRL). In section 3 the experimental evaluation and results are being presented. First, in section 3.A we show how we realize the identification of available high level capabilities. Next, in section 3.B methods to provide valid perception chains are being exposed. In section 3.C we discuss the results for a proof-of-concept ontology. In section 4 there is a conclusion and an outlook into future work.

II. Structure and Concept of the Ontology

The ontology has been created using the Web Ontology Language OWL [6]. Three versions of OWL are available:

- OWL Lite, very inexpressive and mostly used just to create taxonomies
- OWL DL (description logic), suitable for practical applications
- OWL Full, too expressive creating situations, where the inference mechanism will loop infinitely (see [6] for details).

Since OWL DL is widespread and has an advanced tool and library support it is used to model the presented ontology. OWL contains three main concepts to model information: classes (concepts), individuals (instances) and properties (roles).

A. *Class Taxonomy*

The OWL classes serve as group container for different types of individuals (In OWL individuals are instances of (real) objects that belong to special class, e.g., “Sony CBR” is an individual of the class “Sensor”). There are two ways how an individual can be assigned to a class:

1. When an individual is loaded into the ontology it gets its main class (e.g., an electro optical sensor would belong to the class “EOSensor” (Figure 3.)).
2. The individual gets additional class assignments by the inference mechanism.

Our ontology has six top level classes:

Concept: contains the definition of the high and low level capabilities. Low level capabilities (LLC) are split into more detailed groups, for example sensor-, image processing- and platform capabilities. LLCs are provided by resources like perception modules or sensors. Simultaneously, each module needs a predefined set of LLCs as input so it can work as intended. The input LLC required by a module though is different to the LLC that is provided by this module. High level capabilities are used by perception tasks, which can be commanded from a third party system. The more detailed subdivision is based on the taxonomy presented in [7].

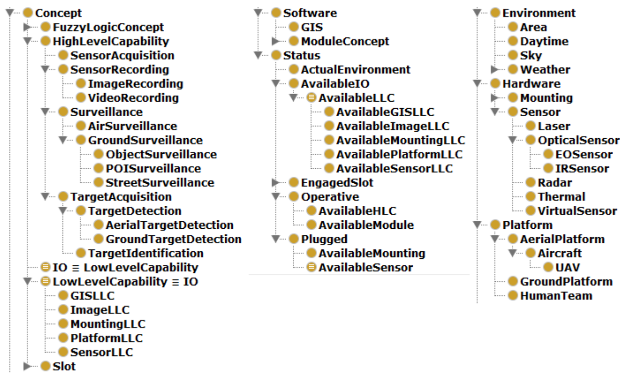
Environment: covers all individuals to describe the composition of the ground, daytime, weather and lighting conditions (e.g., sky formations).

Hardware: describes the sensors and sensor mountings that are attached to the Unmanned Aerial Vehicle or another platform. Subdivision categories of the sensor class are radar, thermal, optical, laser and virtual sensor.

Platform: includes the classes that describe the user of the S&PMS. The subcategories are aerial platform, ground platform and human team.

Software: implies the image processing algorithms represented as perception modules and other services (e.g., a geo information service (GIS Service)).

Status: This class is empty when the S&PMS is started. It states if any individual is available and can be used or not. The inference mechanism assigns each individual to its status class. E.g., if a module can provide a certain low level capability the module is assigned to the "operative module" class and the capability that is provided by this module to the "available low level capability" class.



```
Concept
  FuzzyLogicConcept
  HighLevelCapability
    SensorAcquisition
    SensorRecording
      ImageRecording
      VideoRecording
    Surveillance
      AirSurveillance
      GroundSurveillance
        ObjectSurveillance
        POISurveillance
        StreetSurveillance
    TargetAcquisition
      TargetDetection
        AerialTargetDetection
        GroundTargetDetection
      TargetIdentification
  IO ≡ LowLevelCapability
  LowLevelCapability ≡ IO
    GISLLC
    ImageLLC
    MountingLLC
    PlatformLLC
    SensorLLC
  Slot

Software
  GIS
  ModuleConcept
Status
  ActualEnvironment
  AvailableIO
    AvailableLLC
      AvailableGISLLC
      AvailableImageLLC
      AvailableMountingLLC
      AvailablePlatformLLC
      AvailableSensorLLC
  EngagedSlot
  Operative
    AvailableHLC
    AvailableModule
  Plugged
    AvailableMounting
    AvailableSensor

Environment
  Area
  Daytime
  Sky
  Weather
Hardware
  Mounting
  Sensor
    Laser
    OpticalSensor
      EOSensor
      IRSensor
    Radar
    Thermal
    VirtualSensor
Platform
  AerialPlatform
    Aircraft
      UAV
  GroundPlatform
  HumanTeam
```

Figure 3. Taxonomy of the ontology.

Figure 3. illustrates the taxonomy of the ontology. In total there are 123 classes.

B. *Persistent and dynamic individuals*

The concept of the ontology takes two different behaviors of individuals into account: the *persistent individuals* are always a part of the ontology and *dynamic loaded individuals*, based on the connected systems. The ontology itself contains only high and low level capabilities and their assigned SWRL Rules (see section E.). These capabilities are modelled by an expert and remain persistent in the database. When the S&PMS is loaded or services and sensors are connected to the S&PMS, the represented individuals (e.g. *Sony Sensor1*) are loaded into the ontology. If a sensor stops working or a perception module crashes, the representative individual is removed from the ontology. As soon as the system notices a change in the ontology the "reasoner" gets invoked to update the overall status of the ontology and notify the S&PMS about system changes. Depending on services and sensors connected, and with respect to requirements and constraints of perception modules that are modelled using SWRL Rules, different low and high level capabilities, sensors and modules are unlocked. The reasoner infers, using SWRL rules and OWL axioms, which individuals can be assigned to the status classes mentioned before in section A.

C. *The resource-capability-resource concept*

Since there are individuals that are added and removed from the ontology in a frequent way during runtime, it is not possible to make a statement about available individuals. For this reason you can never tell, which resources (e.g., modules, sensors, etc.) are currently available hence it is not possible to connect two individuals directly. Therefore we introduced the resource-capability-resource concept with permanent capability individuals that are always a part of the ontology and therefore can be used as a reference to create rules for individuals that are dynamically added to the ontology. These permanent low level capability individuals can be seen as input and output configurations for image processing modules or other resources.

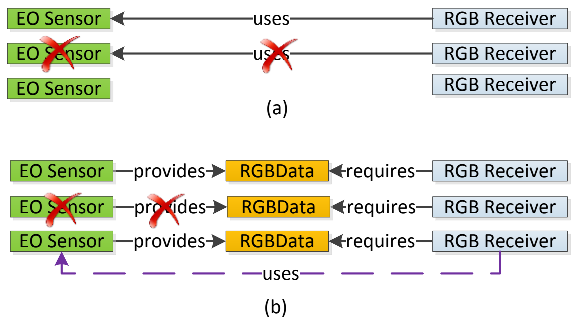


Figure 4. Resource-capability-resource vs. resource-resource connection.

Figure 4. illustrates the difference between the direct connection of resources and the usage of capability individuals between two resources. In (a) the sensor and module are connected directly. Removing the targeted sensor (*EO Sensor*) implies removing the rule from the source module (*RGB Receiver*), because there is no more reference to the target. Re-adding the targeted sensor does however not imply adding the rule to the source module since the source module does not get notified about the existence of the targeted sensor. Another reason against solution (a) is the fact that during modelling time of e.g., *RGB Receiver* there is no knowledge about other resources like sensors or modules. So it would not be possible to create a rule that connects both resources since there is no way to get the information of the existence of e.g., *EO Sensor*.

In (b) each resource holds rules connected to a capability-individual. In this case, when *EO Sensor* gets deleted, only rules included in *EO Sensor* get removed and no other resource is "touched". When *EO Sensor* gets added again, its rules get added too. Since the individual *RGBData* is a permanent individual that is always a part of the ontology, rules that are used by *RGB Receiver* can reference it. Using the inference mechanism a "*uses*" relationship between the sensor and the module can be established.

D. *Data and Object properties*

In OWL we see two different property types, the object and the data properties. Data properties are used to connect individuals with their data represented as parameters. These parameters can be of different built-in types, e.g., string, integer, byte, date or bool. In our ontology, data properties are used to describe the parameters of an image processing algorithm and other numeric information.

Object properties are used to describe relationships between individuals. The most common property is the "is-a" relation between classes (e.g., UAV *is-a* Aircraft). Each object property has several characteristics that can be assigned to it to affect its functional role (e.g., functional, transitive, reflexive, etc.). Additional information can be found in [8]. Within the framework there are four main object properties and their inverses (TABLE I.).

TABLE I. OBJECT PROPERTIES: LEFT: PROPERTY, RIGHT: INVERSE PROPERTY.

object properties	
providesCapability	capabilityProvidedBy
requiresCapability	capabilityRequiredBy
uses	usedBy
subCapabilityOf	superCapabilityOf

The first two describe the relationship between a module and its capabilities. The third and fourth describe the relationship between modules and the relationship within capabilities. The "*is-a"*-property to connect individuals with its classes or classes with subclasses is not listed, because it is not a custom property but a basic property that is available in every ontology.

Figure 5. illustrates a usage of the different object properties to describe the relationship between the individuals and their classes. For clarity the inverse properties are omitted.

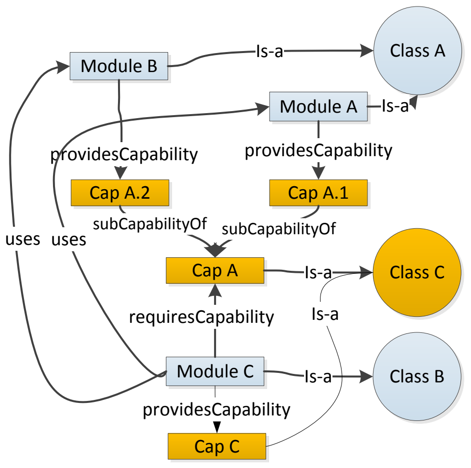


Figure 5. Example for the usage of the different object properties.

E. SWRL Rules

To grant OWL more flexibility and expressive strength it is possible to use rule based languages in combination with the rule markup language (RuleML). Semantic Web Rule Language (SWRL) [9] can be seen as a combination of OWL DL and RuleML. A rule in SWRL is defined as follows:

$$a_1 \wedge a_2 \wedge \dots \wedge a_n \rightarrow b_1 \wedge b_2 \wedge \dots \wedge b_n \tag{1}$$

Variables are called atoms. While a_i describes a precondition (body), b_i describes a post condition (head). Atoms can be class expressions ($C(x)$) or property expressions ($P(x,y)$), in other words relationships between two individuals. There are built in expressions that can be used to model a rule. Some of them are $sameAs(x,y)$, $differentFrom(x,y)$ and $builtin(r, z_1 \dots z_n)$. Built in expressions contain but are not limited to: date, mathematical, string operation. A rule can be read as:

"If precondition X is true, then the post condition is also true."

An empty precondition is always true, an empty post condition always false. All rules that can be accomplish with the OWL axioms can also be modelled with SWRL rules, on the other side there are SWRL rules that cannot be accomplish with OWL axioms.

Each individual that is added dynamically by a service into the ontology contains its own SWRL rule set. Since the capabilities-individuals of the ontology are permanently stored in the database immutable, the SWRL Rules can refer to the individual's names of the capabilities but not to the names of other individuals like sensors or modules.

The SWRL rule belonging to Module C of Figure 5. can be written as follows:

AvailableLLC(Cap A) ^ capabilityProvidedBy(Cap A, ?a)
→ AvailableModule(Module C) ^ (2)
uses(Module C, ?a) ^ providesCapability(Module C, Cap C)

In SWRL "?x" is being used to declare variables. The rule reads as:

"If the individual Cap A belongs to the class AvailableLLC and the individual Cap A has the object property capProvidedBy, referencing to any other individual ?a then assign Module C to the class AvailableModule and assign the object property uses referencing to any other individual ?a to the Module C and the object property providesCap Cap C."

Since there are two other modules that provide Cap A, Module C will belong to the classes "Module" and "AvailableModule" and will have the object property "uses Module B" and "uses Module A" and also have the object property "providesCap Cap C". If another module or a perception task intends to use "Cap C", there are two perception chains that can be used:

- Module B → Module C
- Module A → Module C

III. EXPERIMENTAL EVALUATION

The evaluation of the ontology comprises two categories. It is necessary to know (see section III.A) if at least one perception chain exists, that can solve a given perception task or respectively can provide a high level capability. Next it is necessary to (see section III.B) investigate if all available perception chains for a given HLC are valid and if all possible solutions have been found. Therefore some proof of concept experiments have been done to validate that the identification of HLCs and the perception chains work as expected.

A. *Identifying available high level capabilities*

Testing to check if the ontology identifies available HLCs correctly, can be accomplished directly in the widely used ontology editor Protegé [10]. As mentioned in Figure 3. there is a special class category "Status", more accurate "AvailableHLC" where an HLC individual gets assigned by the reasoner when there is a perception chain that provides this individual. This evaluation includes several SWRL rules for chain components that have to be tested. Each chain component should work self-contained and in combination with other components. Each component requires a *positive* and a *negative* test. The positive test describes a situation where the configuration of the ontology provides individuals that should allow the reasoner to assign a given HLC to the "AvailableHLC" class. For the negative test, the ontology gets changed in a way, that there is no valid path anymore to assign the HLC to the "AvailableHLC" class.

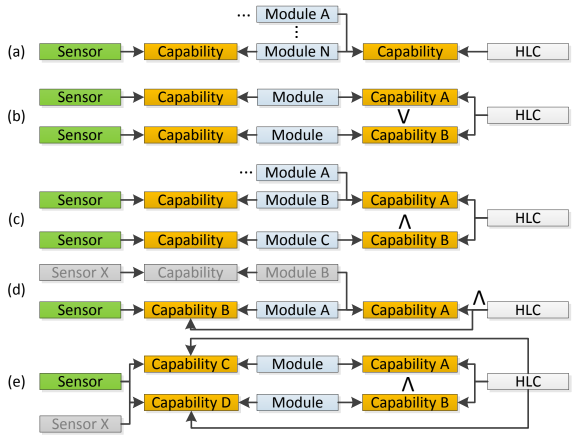


Figure 6. Five chain components that can appear during modelling perception chains.

Figure 6. illustrates five different chain components that can appear in the modelling phase of HLCs. For each component there are different SWRL rules to achieve a desired behavior. The illustration shows a very simple setup starting with the sensor, using one layer of perception modules and connecting it with the HLCs.

(a) is the most simple component where the HLC does need only one capability. The HLC does not care, which or how many modules there are, providing this capability, as long as there is at least one module available. The rule looks like:

AvailableLLC(Capability) ^ capProvidedBy(Capability, ?a) → AvailableModule(HLC) ^ uses(HLC, ?a) (3)

(b) illustrates a component where the HLC can be activated either by "Capability A" or "Capability B". The rule is similar to listing (3) but in this case there is the need for two SWRL Rules, one for "CapabilityA" and the other for "CapabilityB":

AvailableLLC(Capability A) ^ capProvidedBy(Capability A, ?a) → AvailableModule(HLC) ^ uses(HLC, ?a)

(4)

AvailableLLC(Capability B) ^ capProvidedBy(Capability B, ?a) → AvailableModule(HLC) ^ uses(HLC, ?a)

In (c) there is an "and"-relationship between "CapabilityA" and "CapabilityB". The HLC does need both capabilities to get classified as "AvailableHLC". In Figure 6. there are two possible configurations: (Module A ∧ Module C) and (Module B ∧ Module C). The corresponding rule is:

AvailableLLC(Capability A) ^ AvailableLLC(Capability B) ^ capProvidedBy(Capability A,?a) ^ capProvidedBy (Capability B, ?b) → AvailableModule(HLC) ^ uses(HLC, ?a) ^ uses(HLC, ?b) (5)

(d) shows a more restrictive component. Here it is not enough that there is a perception module that provides "Capability A"; there is also the restriction that the module that provides "Capability A" should also use "Capability B". This guarantees that only the combination (Sensor ∧ Module A) but not the combination (Sensor X ∧ Module B) is a valid chain.

AvailableLLC(Capability A) ^ AvailableLLC(Capability B) ^ capProvidedBy(Capability A,?a) ^ capProvidedBy (Capability B, ?b)^ uses(?a,?b) → AvailableModule(HLC) ^ uses(HLC, ?a) (6)

The rule in listing (6) looks similar to listing (5), expect that in (6) there is a "*uses(?a,?b)*" in the body that determines that individual "?a" that also provides "Capability A" has to use individual "?b", which also provides "Capability B".

(e) illustrates a exception where the HLC does need both capabilities "Capability A" and "Capability B". But in this case it must be guaranteed that data, which is used by the modules providing both capabilities, must be from the same sensor.

AvailableLLC(Capability A) ^ AvailableLLC(Capability B) ^ AvailableLLC(Capability C) ^ AvailableLLC(Capability D) ^ capProvidedBy(Capability A, ?a) ^ capProvidedBy(Capability B, ?b) ^ capProvidedBy(Capability C, ?d) ^ capProvidedBy(Capability D, ?e) ^ uses(?a,?c) ^ uses(?b,?c) → AvailableModule(HLC) ^ uses(HLC, ?a) ^ uses(HLC, ?b) (7)

To realize the behavior shown in (e) it is necessary to introduce another variable "?c" representing an individual. This individual has to be used by both modules, the ones that provide "Capability A" and the others that provide "Capability B". If "Capability C" would be provided only by "Sensor" and "Capability D" would be provided only by "Sensor X" all capability and perception module individuals would be available but the HLC would still be not available because the rule "*uses(?a,?c) ^ uses(?b,?c)*" from listing (7) would be false.

B. Provide valid perception chains

After collecting information about available high level capabilities it is necessary to verify that the provided combinations of perception modules (in form of perception chains) result in the correct outcome. The list of existing perception chains for a specific HLC does not allow chains that cannot handle the perception task. Therefore the chain count as well as the chain composition is tested against an expert model. All valid chains must be represented. The concatenation and validation of modules into perception chains is done outside the ontology in a special application that can read, write and parse the ontology. The algorithm checks the dependency from one individual to another, starting with the HLC individuals. Recursively each dependency is put into a list (the perception chain list). If an individual has more than one dependency, the chain gets split. The process ends, when an individual has no more dependencies to other individuals. Individual can have multiple SWRL rules, resulting in equal chains. During the concatenation process chains can arise that are formally correct but not valid for the specific HLC since not all low level capabilities can be satisfied within the chain. After the recursive process terminates, duplicates and invalid chains are filtered. There are some cases where the algorithm cannot filter all invalid chains due to rule complexity. In these cases, special data properties are parsed after the initial validation. Whichever parameters are set, special filtering mechanisms are being triggered inside the application to erase the remaining invalid chains.

To guarantee that all valid chains have been found smaller ontologies can be manually matched against an expert design result. For bigger ontologies the complexity rises with each individual added. Above a certain ontology size it gets very difficult for an expert to observe all possible outcomes. It may also be the case that the inference mechanism discovers perception chains, which the expert did not intend to create. This outcome must also be checked against an expert's design results manually. This can be an advantage since solution can arise that are more intelligent or less resource intensive. But it can also be a disadvantage due to the difficult way to evaluate the systems correct way of working.

C. Proof of concept

Based on the founding functions in A) and B) a more general proof of concept was conducted using the example depicted in Figure 6. . The perception graph obtained from the ontology is illustrated in Figure 7.

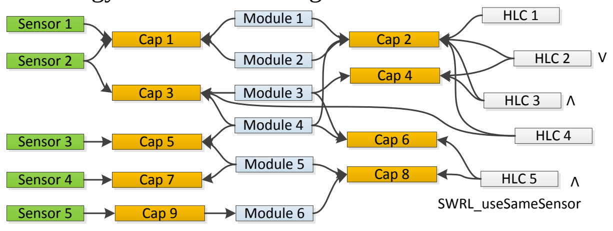


Figure 7. Perception graph generated by example ontology

The system assumes all resources (sensors and modules) to be available and operational. From this point all available HLCs and their perception chains are calculated. Starting the reasoner, we can observe that all HLCs are available like expected (see TABLE II. (a)).

Next each sensor is deactivated until no HLC is available. Each step the perception chains are recalculated by the inference mechanism. The results can be observed in TABLE II. One can see that the modeled rules are working as anticipated: the available HLCs and chain count decreases.

TABLE II. EVALUATING THE DEACTIVATION OF SENSORS

Sensor	Module	HLC	Chains
Sensor 1	Module 1	HLC 1	6
Sensor 2	Module 2	HLC 2	7
Sensor 3	Module 3	HLC 3	6
Sensor 4	Module 4	HLC 4	3
Sensor5	Module 5	HLC 5	1
	Module 6		

(a)

Sensor	Module	HLC	Chains
Sensor 1	Module 1	HLC 1	4
Sensor 2	Module 2	HLC 2	5
Sensor 3	Module 3	HLC 3	4
Sensor 4	Module 4	HLC 4	3
Sensor5	Module 5	HLC 5	1
	Module 6		

(b)

Sensor	Module	HLC	Chains
Sensor 1	Module 1	HLC 1	1
Sensor 2	Module 2	HLC 2	1
Sensor 3	Module 3	HLC 3	0
Sensor 4	Module 4	HLC 4	0
Sensor5	Module 5	HLC 5	1
	Module 6		

(c)

Sensor	Module	HLC	Chains
Sensor 1	Module 1	HLC 1	0
Sensor 2	Module 2	HLC 2	0
Sensor 3	Module 3	HLC 3	0
Sensor 4	Module 4	HLC 4	0
Sensor5	Module 5	HLC 5	0
	Module 6		

(d)

In TABLE II (a) all sensors are available hence all modules and HLCs are available with a different amount of perception chains. In TABLE II (b) "Sensor 1" is deactivated. Since "Sensor 1" and "Sensor 2" provide the same capability no module is being affected but the chain count for "HLC1"-"HLC3" decreases.

TABLE III. CHAIN COMPOSITIONS FOR (A) FROM TABLE II

HLC 1	6
Sensor 1	Module 1
Sensor 1	Module 2
Sensor 2	Module 1
Sensor 2	Module 2
Sensor 2	Module 4
Sensor 3	Module 4

HLC 2	7
Sensor 1	Module 1
Sensor 1	Module 2
Sensor 2	Module 1
Sensor 2	Module 2
Sensor 2	Module 4
Sensor 3	Module 4
Sensor 2	Module 3

HLC 3	6	
Sensor 1	Module 1	Module 3
Sensor 1	Module 2	Module 3
Sensor 2	Module 1	Module 3
Sensor 2	Module 2	Module 3
Sensor 2	Module 4	Module 3
Sensor 3	Module 4	Module 3

HLC 4	3
Sensor 2	Module 1
Sensor 2	Module 2
Sensor 2	Module 4

HLC 5	1	
Sensor 3	Module 4	Module 5

TABLE IV. CHAIN COMPOSITIONS FOR (B) FROM TABLE II

HLC 1	4
Sensor 2	Module 1
Sensor 2	Module 2
Sensor 2	Module 4
Sensor 3	Module 4

HLC 2	5
Sensor 2	Module 1
Sensor 2	Module 2
Sensor 2	Module 4
Sensor 3	Module 4
Sensor 2	Module 3

HLC 3	4	
Sensor 2	Module 1	Module 3
Sensor 2	Module 2	Module 3
Sensor 2	Module 4	Module 3
Sensor 3	Module 4	Module 3

HLC 4	3
Sensor 2	Module 1
Sensor 2	Module 2
Sensor 2	Module 4

HLC 5	1	
Sensor 3	Module 4	Module 5

TABLE V. CHAIN COMPOSITIONS FOR (C) FROM TABLE II

HLC 1	1
Sensor 3	Module 4

HLC 2	1
Sensor 3	Module 4

HLC 3	0	

HLC 4	0

HLC 5	1	
Sensor 3	Module 4	Module 5

When deactivating “Sensor2” (cf. (c)) three modules are not operative any more since there is no sensor that can provide the required data respectively capabilities. As a hoped consequence “HLC3” and “HLC 4” is being deactivated and the chain count for the operative HLCs drops drastically. When “Sensor 3” is also being deactivated we can observer that in (d) “Module 4” stops working and there are no more available high level capabilities.

TABLE III. TABLE IV. and TABLE V. list the possible module compositions for the results illustrated in TABLE II. (a), (b) and (c). In TABLE IV. one can see that no more chain compositions for *Sensor 1* are available anymore. In TABLE V. only chain compositions using *Sensor 3* are available since *Sensor 1* and *Sensor 2* are deactivated and for the other two sensors there are no perception chains. Overall the experiments show a supposed behavior of the inferred results taking the modeled relationship and SWRL rules into account. The results prove a suitable usage of the ontology to model sensor and data processing resources using OWL.

IV. CONCLUSION AND FUTURE WORK

We presented an approach to manage sensor and data resources with an ontology based knowledge management system. It was shown how the knowledge representing language OWL can be used respectively. The presented solution proposes to model image processing algorithms as perception modules providing different low level capabilities, which in turn can be combined to high level capabilities, representing various perception tasks e.g., vehicle detection. For each task different perception chains are calculated, dependent on the current environmental situation and platform setup respectively resource configuration (sensors, algorithms, etc.).

An important next step is to develop a decision-making system that takes available perception chains for a given perception task in account and determines, based on different parameters and meta-information, which chain is most suitable to solve the given task.

The system shall be further tested in a multi UAV scenario where each UAV has a different sensor and perception module configuration. The aim here is to combine different capabilities onboard UAVs and let the UAVs collaborate to solve a complex perception task as a team.

Eventually investigations are planned in human-machine scenario, where a helicopter operator can fall back to S&PMS functions that assist him during his mission and therefore reduce the operator’s workload. The operator can choose between different automation levels so that the S&PMS can process full perception tasks or only parts of it [11]. In this scenario, the human capabilities are a part of the knowledge base and are modeled into the ontology. The inference mechanism takes the human capabilities into account when generating perception chains for different perception tasks. For example, when there is no algorithmic way for a processing step, the S&PMS can make use of human capability to still find an adequate perception chain.

REFERENCES

[1] D. Smirnov and P. Stuetz, “Knowledge elicitation and representation for module based perceptual capabilities onboard UAVs,” in *AIAA SciTech 2014*, 2014.

[2] C. Hellert, D. Smirnov, M. Russ, and P. Stuetz, “A High Level Active Percpetion Concept For UAV Mission Scenarios,” in *Deutscher Luft- und Raumfahrtkongress 2012*, pp. 1–9, 2012.

[3] C. Hellert, D. Smirnov, and P. Stuetz, “Ontologiedesign für Sensor- und Perzeptionsfähigkeiten von UAVs,” in *Deutscher Luft- und Raumfahrtkongress 2014*, 2014.

[4] M. Russ and P. Stütz, “Airborne sensor and perception management: A conceptual approach for surveillance UAS,” in *Proceedings of the 15th International Conference on Information Fusion (FUSION2012)*, pp. 2444–2451, 2012.

[5] M. Russ and P. Stuetz, “Application of a probabilistic market-based approach in UAV sensor & perception management,” in *Information Fusion (FUSION), 2013 16th International Conference on*, pp. 676–683, 2013.

[6] W3C OWL Working Group, “OWL 2 Web Ontology Language Document Overview,” 2013. [Online]. Available: http://www.w3.org/TR/owl2-overview/. [Accessed: 08-May-2013].

[7] M. Gomez et al., “An ontology-centric approach to sensor-mission assignment,” *Knowl. Eng. Pract. Patterns*, pp. 347–363, 2008.

[8] M. Horridge et al., “A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools,” Manchester, 2011.

[9] I. Horrocks, P. F. Patel-schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, “SWRL : A Semantic Web Rule Language Combining OWL and RuleML,” *W3C Memb. Submiss. 21*, no. May 2004, pp. 1–20, 2004.

[10] “Protégé project.” [Online]. Available: http://protege.stanford.edu.

[11] C. Ruf and P. Stütz, “Model-driven Sensor Operation Assistance for a Transport Helicopter Crew in Manned-Unmanned Teaming Missions : Selecting the Automation Level by Machine Decision-making,” in *7th International Conference on Applied Human Factors and Ergonomics (AHFE2016)*, 2016.