

Addressing the Reactiveness Problem in Sensor Networks Using Rich Task Representation

**A thesis submitted in partial fulfillment
of the requirement for the degree of Doctor of Philosophy**

Konrad Borowiecki

December 2011

**Cardiff University
School of Computer Science & Informatics**

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed (candidate)

Date

Statement 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed (candidate)

Date

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed (candidate)

Date

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed (candidate)

Date

**To Friends & Family
for their patience and support.**

To my beloved wife Marta, for her love, support and understanding during this difficult period of time.

To my parents, Waldemar & Maria, for their help and all the opportunities they have created for me throughout my life.

To Alun Preece for being my mentor, giving me the opportunity to see the 'world of research' and all his patience lost reading my English.

To my friends and colleagues: Diego Pizzocaro, Lorenzo Moncelsi, Matt Williams, Stefanie Walch, Przemek Woźnowski, Chris Gwilliams, and many others whom I met during these years of research.

Great Many Thanks To You All.

Abstract

Sensor networks are increasingly important in many domains, for example, environmental monitoring, emergency response, and military operations. There is a great interest in making these networks more flexible, so they can be more easily deployed to meet the needs of new tasks. The research problem is lack of reactivity of a system utilising a sensor network in a dynamic real-time domain, where the state of sensors and tasks might change many times (e.g. due to a sensor malfunction, or a change in task requirements or priorities). In such domains (e.g. firefighting or the military) we want to minimise the time spent manually configuring the sensor network, as any delay dramatically endangers the outcome of a task or a delay's effects might be unacceptable, e.g. the loss of a human life. The current way of deploying sensors in the problem context involves four consecutive steps: Direction, Collection, Processing and Dissemination (DCPD). These steps form a cycle, called the DCPD loop. Automating this loop as much as possible would be a big step towards solving the reactivity problem. Service-Oriented Sensor Networks (SOSN), allow sensors to be discovered, accessed, and combined with other information-processing services, thus enabling an efficient sensor exploitation. They are only a partial solution to the problem, as they don't employ explicit representations of a user's information-requiring tasks. Therefore, a machine processable expression of a user's task (task representation, TR), allowing automation of the DCPD steps, is needed. We showed that, currently, there is no TR that can completely automate the loop, but that we can create such a hybrid of current TRs (called HTR) that automates the loop more than the individual TRs. Our

literature review revealed four TRs. Using the identified TRs, we formed three high level designs of task representations. None of them covered the loop completely thus by enrichment of one of the built HTRs with the missing concepts, we finally obtained one that covers the DCPD loop fully. We tested the four hybrids in a simulation run for four scenarios with distinctive likelihoods of change of task and platform states. It showed that significant benefits are gained just by reusing existing technologies and that the reactiveness problem can be effectively tackled by that approach, particularly visible in the emergency response scenario, characterised by low task and high platform changeability.

Acknowledgements

This research would not have been done if not for Professor Alun Preece who supervised this work and during its time was always happy to assist me with his expertise and time. Also very positive influence on the work had: Christopher Gibson who offered his time when integrating the IBM Sensor Fabric with the SAM application; Dr. Mario Gomez & Geeth de Mel who helped integrating the SAM reasoner with the SAM application; and Chris Gwilliams who gave his time to proof-read the thesis.

I want to thank them all.

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Contents

Abstract	iii
Acknowledgements	v
Contents	vi
List of Publications	xi
List of Figures	xiii
List of Tables	xvi
List of Algorithms	xvii
List of Acronyms	xviii
1 Introduction	1
1.1 Research Context	1
1.2 Problem and Hypothesis	5
1.3 Key Elements of the Research	6

1.4	Contributions	13
1.5	Publications and Their Relevance to the Research	14
1.6	Thesis Structure	15
2	Previous Work	16
2.1	The Reactiveness Problem and the DCPD Loop	16
2.2	SOA and SOSN	20
2.3	Task Representation	22
2.3.1	Open Geospatial Consortium Sensor Web Enablement TR	23
2.3.2	Semantic Streams TR	27
2.3.3	Goal Lattice TR	29
2.3.4	Sensor Assignment to Missions TR	32
2.4	Other Related Work	35
2.4.1	Sensor and Sensor Network related Ontologies	35
2.4.2	Sensor Allocation Problem	37
2.5	The Research and Our Approach	39
2.5.1	Identification of the Research Problem	39
2.5.2	Presentation of the Testbed Application	42
2.5.3	Our Approach	44
3	Task Representation	46
3.1	The Features of the DCPD Loop	46
3.2	Formalisation of Four Existing Task Representations	49

3.2.1	Open Geospatial Consortium's Sensor Web Enablement TR	51
3.2.2	Semantic Streams TR	53
3.2.3	Goal Lattice TR	54
3.2.4	Sensor Assignment to Missions TR	56
3.3	The TRs in Context of the DCPD Loop	59
4	Hybrid Task Representation	64
4.1	Methodology of Hybrid Task Representation Creation	65
4.1.1	Mapping Between the TRs	68
4.1.2	General HTR	70
4.2	The Three HTRs	73
4.2.1	Max SAM HTR	73
4.2.2	Max SWE HTR	75
4.2.3	Max SS HTR	77
4.3	The Three HTR Models in Relation to the DCPD Loop	79
4.3.1	Direction	80
4.3.2	Collection	81
4.3.3	Processing	81
4.3.4	Dissemination	83
4.3.5	HTRs Analysis Conclusions	84
4.4	The Missing Concepts	85
4.5	Max SAM Plus HTR	88

5	Performance Evaluation of Hybrid Task Representations	91
5.1	Experimental Approach and Scenarios	91
5.1.1	Experimental Approach	92
5.1.2	Scenarios	96
5.2	Experimental Environment Design	99
5.2.1	Test Parameters & Assumptions	100
5.2.2	Method of Reasoner Use in Simulation	105
5.2.3	Assignment Cost Calculation Methodology	108
5.3	Performance Experiment Results and Analysis	111
5.3.1	Results for Task LO Platform LO Scenario	112
5.3.2	Results for Task LO Platform HI Scenario	113
5.3.3	Results for Task HI Platform LO Scenario	114
5.3.4	Results for Task HI Platform HI Scenario	117
5.4	Proposed Future Design of the Experiment	119
6	Conclusions & Future Work	121
6.1	Conclusions	121
6.2	Instantiation of Max SAM Plus HTR Model	125
6.3	Future Work	126
A	Ontologies of Task Representations	131
A.1	SWE TR Ontology	131
A.2	SS TR Ontology	141

A.3	GL TR Ontology	150
A.4	SAM TR Ontology	156
B	Ontologies of Hybrid Task Representations	161
B.1	GHTR Ontology	161
B.2	Max SAM Ontology	166
B.3	Max SWE Ontology	168
B.4	Max SS Ontology	171
B.5	Max SAM Plus Ontology	173
	Bibliography	184

List of Publications

Parts of the work in this thesis were previously published in the following:

Refereed Conference Papers

- M Gomez, A Preece, M P Johnson, G de Mel, W Vasconcelos, C Gibson, A Bar-Noy, K Borowiecki, T La Porta, D Pizzocaro, H Rowaihy, G Pearson, & T Pham, An Ontology-Centric Approach to Sensor-Mission Assignment, Proc 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008), Springer, pages 347-363, 2008.
- A Preece, D Pizzocaro, K Borowiecki, G de Mel, M Gomez, W Vasconcelos, A Bar-Noy, M P Johnson, T La Porta, H Rowaihy, G Pearson, & T Pham, Reasoning and Resource Allocation for Sensor-Mission Assignment in a Coalition Context, Proc MILCOM 2008, 7 pages, 2008.
- A Preece, D Pizzocaro, K Borowiecki, G de Mel, W Vasconcelos, A Bar-Noy, M P Johnson, T La Porta, & H Rowaihy, Knowledge-Driven Agile Sensor-Mission Assignment, Proc 3rd Annual Conference of the International Technology Alliance (ACITA 2009), 8 pages, 2009.

Refereed Conference Demo Papers

- A. Preece, D. Pizzocaro, K. Borowiecki, G. de Mel, M. Gomez, W. Vasconcelos, A. Bar-Noy, M. P. Johnson, T. La Porta, H. Rowaihy, G. Pearson & T. Pham
Sensor Assignment to Missions in a Coalition Context: The SAM Tool, Adjunct Proceedings of INFOCOM 2009 demo session, Rio de Janeiro, Brasil, April 2009.

Refereed Conference Poster Papers

- K. Borowiecki & A. Preece, An Ontological Approach to Integrating Task Representations in Sensor Networks Proc 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010), Lisbon. Portugal, October 2010.

List of Figures

1.1	A user utilising a sensor network via a system in a dynamic domain (sensors are marked as ovals)	3
1.2	The DCPD loop.	7
1.3	User- and sensor-level tasking in a system. Expressiveness of task representations (sensors are marked as ovals)	9
2.1	SOA technologies in relation to a sensor and task.	21
2.2	Relation of TR and the DCPD loop.	23
2.3	Chaining of semantic services for satisfaction of users' queries.	28
2.4	Graphical representation a Goal Lattice.	31
2.5	Overview of the Sensor Assignment to Missions application	40
2.6	Framework application user interface	43
2.7	Interface elements for a vehicle detection task	44
3.1	Ontology for the Sensor Web Enablement TR	52
3.2	An example SWE car detection task	53
3.3	Ontology for the Semantic Stream TR	54

3.4	An example Semantic Stream car detection task	55
3.5	Ontology for the Goal Lattice TR	56
3.6	An example Goal Lattice including a car detection task	57
3.7	Ontology for the Sensor Assignment to Missions TR	58
3.8	An example Sensor Assignment to Missions car detection task	59
4.1	Ontological representation of relations between TRs; highlighting general concepts, from GHTR ontology.	69
4.2	Model of the general HTR.	71
4.3	Model of the Max SAM HTR.	75
4.4	Model of the Max SWE HTR.	77
4.5	Model of the Max SS HTR.	79
4.6	Model of the Max SAM Plus HTR.	89
5.1	Flow of operations in a task's state verification algorithm in a single time-step. Rectangles represent operations; diamonds represent conditional change of operation, where Y is for Yes, N is for No	95
5.2	Example of parameters for the experiment run using Parameter Sweep option	101
5.3	Task representation assignment cost scale.	109
5.4	Test results for Task LO Platform LO scenario.	113
5.5	Test results for Task LO Platform HI scenario.	115
5.6	Test results for Task HI Platform LO scenario.	116
5.7	Test results for Task HI Platform HI scenario.	118

6.1	Max SAM Plus HTR instantiated for Capture Speeding Car Image Task	127
A.1	SWE TR class hierarchy	131
A.2	SS TR class hierarchy	141
A.3	GL TR class hierarchy	150
A.4	SAM TR class hierarchy	156
B.1	GHTR class hierarchy	161
B.2	Max SAM Plus class hierarchy	173

List of Tables

3.1	Four TRs versus the Direction step of the DCPD loop.	60
3.2	Four TRs versus the Collection step of the DCPD loop.	60
3.3	Four TRs versus the Processing step of the DCPD loop.	61
3.4	Four TRs versus the Dissemination step of the DCPD loop.	62
4.1	Three HTRs versus the Direction step of the DCPD loop.	80
4.2	Three HTRs versus the Collection step of the DCPD loop.	81
4.3	Three HTRs versus the Processing step of the DCPD loop.	82
4.4	Three HTRs versus the Dissemination step of the DCPD loop.	83
4.5	Three HTRs versus information needs of steps of the DCPD loop.	84
5.1	Scenarios in relation to sensor and task levels of change probability.	97
5.2	HTRs' costs evaluation table. Cost of Interaction Point for each HTR	111

List of Algorithms

4.1	Algorithm of HTR creation	66
-----	-------------------------------------	----

List of Acronyms

UAV Unmanned Aerial Vehicle

NIIRS National Imagery Interpretability Rating Scale

DCPD Direction, Collection, Processing, and Dissemination

TR Task Representation

HTR Hybrid Task Representation

SWE Sensor Web Enablement

SS Semantic Streams

GL Goal Lattice

SAM Sensor Assignment to Missions

Introduction

This chapter introduces the context of our research, where we state what is its focus and the problem. We summarise our thinking and explain the aspects of our approach to solving the problem. We also provide a brief description of the key elements for proper understanding of our research. Towards the end of the chapter we write contributions of our research. We state our publications together with an explanation of how they relate to our work. Finally, we present the structure of the thesis with a short summary of coming chapters.

1.1 Research Context

Sensor networks are now used with a benefit in many domains, for example, science [39], agriculture [20, 78], the military [37], geospatial systems [50], firefighting [14] and many others. Effective use of sensor networks is thus a generic problem and not limited to any particular application domain. This problem becomes more complicated if we consider dynamic domains, such as, firefighting, emergency response, or the military. In these domains the need for the continuous information delivery is so important that any delay, e.g. caused by a user manually responding to environment changes (being it a sensor's and/or a task's status change) is vital for a task. This dynamic aspect requires appropriate consideration when modeling a task. Thus, for a task, to be effective, a representation that would support an automatic operation of

a system is required. This can only be achieved by understanding the way a system operates in a sensor network and its capture within a task representation. Sadly in the current task representations this is not the case.

There is considerable and growing interest in developing approaches that allow sensors to be treated as data-providing resources, and integrated within Web and Semantic Web information architectures (for example, [9, 41, 82]). A key issue in this is making these networks more flexible, so they can more easily be deployed to meet the needs of new tasks [28, 71]. We identify two aspects of the sensor tasking problem: *user-level tasking* involves the representation of a user's tasks in a form allowing their interpretation as operations executable on a sensor network; *sensor-level tasking* involves arrangement and execution of operations on sensors and sensors' generated data which, after processing, answer to a user's tasks. For example, user-level tasking is concerned with tasks such as the detection of vehicles or identification of people, whereas sensor-level tasking is concerned with operations such as collecting video or audio data of a particular quality. Another way of looking at this is to say that user-level tasking focuses on issues of "what" whereas sensor-level tasking focuses on issues of "how". In practice, a complete task specification needs to include both aspects, because users will be concerned with both what they want to know, and how they get the supporting sensor data [28].

To see the issues of sensor tasking more clearly, Fig. 1.1 presents a user utilising a sensor network via a system in a dynamic domain, where the state of sensors and tasks might change many times (e.g. due to a sensor malfunction, or a change in task requirements or priorities). The figure intends to show the interaction processes taking place between a user and a system and a sensor network, and to identify the locations where changes might occur (on the sensor and task sides). Here a system is a software application that provides access to a sensor network. It is composed of components that allow it to take an input from a user (i.e. it has some interface), to find sensors (i.e. it has access to a sensor network), to interpret sensor data (i.e. it has access to a fusion

algorithm), and to deliver information to the user (i.e. it provides sensor data that satisfies the user's task). Thus, the functionalities of such a system are: reception of orders (tasks) from a user then use of a sensor network to gather data (control of sensor data collection); processing of the data; and finally delivery of information (sensor data that satisfies a task) to the user.

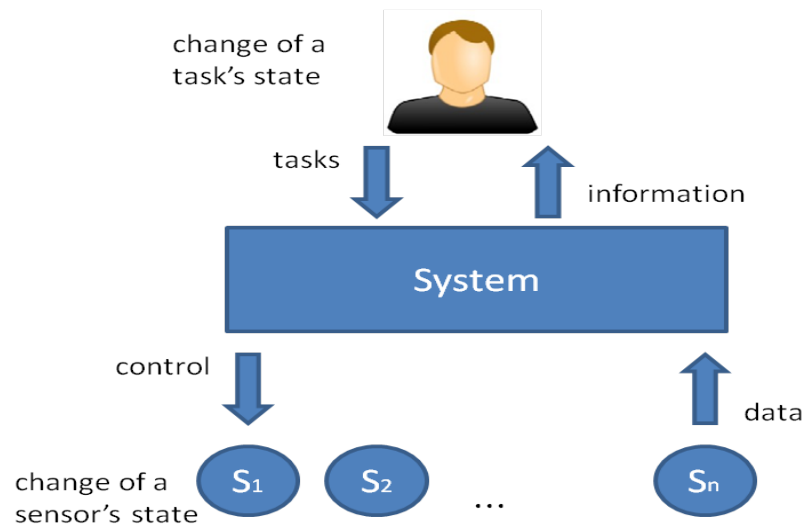


Figure 1.1: A user utilising a sensor network via a system in a dynamic domain (sensors are marked as ovals).

Before continuing with the presentation of the problem let's first take a look at the definitions used and relations between them. A *task* is a set of information requirements which express what a user wants to accomplish. A task's *information requirements* are a set of information necessary for satisfaction of a task, obtained by processing of sensor-provided data. A *sensor* can be a hardware and/or a software resource that provides data describing an event.

The relations between the aforementioned definitions are presented in the following example from a vehicle tracking domain:

- Tasks:

T1 Detect a suspicious vehicle.

T2 Track a vehicle.

T3 Detect a vehicle breaking traffic regulations.

- Information requirements of the 1st task (T1):

IR1 Is the vehicle stolen?

IR2 Who is inside the vehicle?

IR3 Is the vehicle carrying explosives?

- Sensors:

- Although for the first information requirement (IR1), not many types of sensors might be usable, still a rich range of cameras would apply. Particularly these integrated into specialised systems, such as e.g. TALON ANPR [74], dedicated to licence plate recognition, could quickly determine if, for example, a vehicle is marked as stolen in a police database. A camera, especially an infrared one, could also be applied, to satisfy the 2nd information requirement (IR2), to identify the number of passengers inside a suspicious car. For this also useful would be acoustic sensors, as they would allow to listen to what is going on inside the car, possibly identify who the passengers are, what they are talking about or what their intentions might be.
- In case of the 3rd information requirement (IR3) the cameras are not very efficient since these sorts of materials are usually concealed. Therefore, to deliver information for this requirement we would employ special set of sensors which are specifically designed for this purpose, such as explosive trace detectors [79] or X-ray imaging systems like Rapiscan Eagle M60 [73].

1.2 Problem and Hypothesis

Problem

The research problem is addressing the lack of reactivity to changes in the state of a sensor or a task that would allow a system utilising a sensor network to cope with the changes, with minimal user intervention. This problem is of the most importance for dynamic domains, especially for those where the response time is critical (e.g. fire-fighting [14] or the military [37, 61]). In such domains there is simply no time to wait for a user's action (e.g. to reconfigure a task or to select other sensors), as any delay dramatically endangers the outcome of a task or a delay's effects might be unacceptable. We call this problem the "reactiveness problem in sensor networks from the perspective of user-level tasking". For short, in the document we refer to it as the "reactiveness problem".

Additionally, current approaches to the creation of this kind of system are to engineer a solution for particular user-level tasks, using particular sensor-level choices, integrated by means of Service Oriented Architectures (SOA) [19, 38, 54]. Often, the representation of tasks (user- and sensor-level) is implicit. We aim to go further, by building on and integrating existing explicit representations of tasks, in order to allow better integration of sensor planning, operation, and delivery services, as well as encouraging an open, standards-based approach to task representation in sensor networks.

We can write the problem more formally as, at a given point in time: Let $T = \{t_1, \dots, t_n\}$ be a set of tasks deployed in a sensor network. Let $S = \{s_1, \dots, s_m\}$ be a set of services available for a system either delivered by the network or being internal to the system. S includes sensors, processing and delivery services. Let g_i be the (average) time of a task t_i 's solution **(re)generation** by the system. Let e_i be the (average) time of a user's **effort** taken editing a task t_i , $e_i = ei_i + er_i$, where ei_i is the (average) time of a user's effort of **initial** editing of a task t_i , and er_i is the (average) time of a user's effort taken

editing a task in **response** to later changes in the network.

$sol(t_i, S_j)$, where $t_i \in T$ & $S_j \subseteq S$, $isvalid \iff$ the services in S_j provide a **solution** to t_i , where each solution has a **cost** $g_i + e_i$.

We expect that by having richer knowledge captured in a task model we can then perform more operations in an automatic way. Thus, by increasing initial effort e_i to create each task t_i , we can reduce the cost of (re)generating $sol(t_i, S_j)$ pairs. In result we obtain a task which can adopt to changes more rapidly, which is of key importance for dynamic domains.

Hypothesis

“A rich task representation model, mapping existing technologies, significantly reduces a task’s reaction time to a sensor’s and task’s state changes, thus, improving handling of the reactivity problem in sensor networks from a task’s perspective.”

1.3 Key Elements of the Research

All the steps that are present in an information flow of a system facing the reactivity problem can be gathered and simplified into the four distinctive steps: Direction (D_1), Collection (C), Processing (P) and Dissemination (D_2), called later the DCPD loop. The steps appearing in the loop are involving operations on tasks, sensors and sensor-provided information from description of a task’s objectives, through gathering of sensor data, its analysis and delivery of processed information that satisfies the task’s objectives. This is presented in Fig. 1.2 (taken from [86]), where: Direction is determination of tasks’ information requirements; Collection is gathering of data from sensors and delivery to processing services; Processing is analysis of collected data, which satisfies tasks’ information requirements; and Dissemination is delivery of processed information to those who need it.

In our research we are focusing on the steps of the DCPD loop that represent the

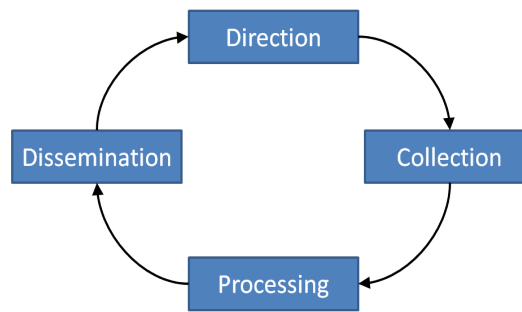


Figure 1.2: The DCPD loop.

information flow leading to satisfaction of a task's information requirements, using sensor-provided data. We are only worried about the information needs, required by each step, that would allow for information to travel around the whole loop. Our research problem is more general than the problems of Sensor-Task Assignment [28] (the problem of allocating a set of sensors to a set of tasks) and Sensor Management [47] (the problem concerning efficient information collection for improved information fusion), as these two problems neglect the issues of why the loop is there and how to deliver the information. We are not considering problems of creation, deployment and configuration of a sensor network or complex interaction between preceding steps in the loop. In our view, maximising the automation of the DCPD loop is key to addressing the reactiveness problem in sensor networks, the problem being especially important in highly dynamic domains where continuous task satisfaction is essential. Some emerging technologies appear to offer help here.

Service-Oriented Sensor Networks (SOSN) [11, 26] can be seen as an enabler of our approach as they allow a system working with a sensor network to control sensors and collect their data (Fig. 1.1). This approach to development of sensor networks, based on a Service-Oriented Architecture (SOA) [38, 54, 19], makes them more flexible, thus helping a system to easily discover, access, and utilise sensors. Here, a service means a function provided by something, whether it is a piece of software or hardware. In the case of SOSN, a sensor's capability is a service. However, SOSN is only a partial solution to the reactiveness problem, as utilisation of sensor capabilities is only one

part of the problem (control and data collection). SOSNs do not consider (at least) the user's information-requiring tasks.

Therefore, we assert that a machine processable expression of a user's task (task representation, TR) that captures user-level tasking requests and links these to sensor-level tasking requests, is needed. Such a TR would provide all the necessary input to a system that would operationalise a user's request in terms of necessary "what" and "how" requirements. It would capture all of the information needs that the four DCPD steps require to operate, then it would allow automation of the loop without further user input. This would be a significant step towards solving the reactivity problem in sensor networks.

In order to identify a set of requirements for our task representation, we conducted a literature review and identified four existing representations addressing aspects of user-level and/or sensor-level tasking, for which there were reasonably detailed descriptions of the TR formalism. Though the main reason for selection of these particular TRs was consideration of user-level issues in their description, since, as we mentioned in the problem definition, we are focusing on the reactivity problem from the perspective of user-level tasking:

- Open Geospatial Consortium Sensor Web Enablement (SWE) enables tasking on a sensor-level, allowing for discovery, access and setting of sensors through Web service standards [9, 71].
- Semantic Streams (SS) are useful for both user- and sensor-level tasking, as they enable the creation of streams representing the flow of sensor-generated information and processing required in order to satisfy a task's information requirements [41].
- Goal Lattices (GL) assist in user-level tasking, during task planning, by defining a lattice of goals and weights, where sub-goals contribute to the satisfaction of

super-goals in terms of their relative weight, allowing for goal prioritisation [32, 34].

- Sensor Assignment to Missions (SAM) connects user- and sensor-level tasking, as it enables matching between tasks and sensor types, by mapping of a task's information requirements to a set of sensor capabilities satisfying them [28, 62].

Figure 1.3 shows a user accessing a sensor network via a software system, much like Fig. 1.1. This one though focuses on the positions that the four TRs listed above take in relation to user- and sensor-levels of tasking.

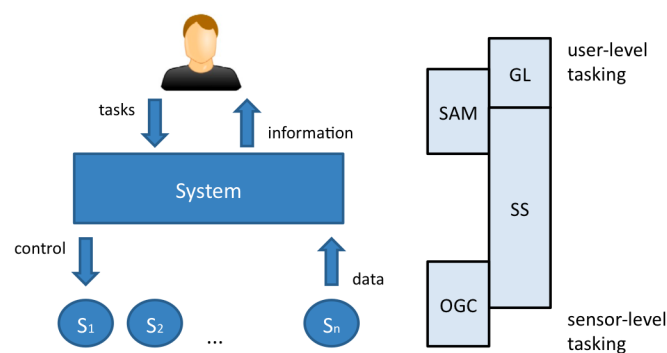


Figure 1.3: User- and sensor-level tasking in a system. Expressiveness of task representations (sensors are marked as ovals).

As an example of the interaction steps shown in Figure 1.3, consider a user — let it be a police officer — who has a need to track a vehicle suspected of being involved in a crime. The user's high-level task would be to identify and track the particular vehicle. The software system would locate individual sensors capable of satisfying the task's requirements (e.g. discovering CCTV cameras in the area of interest) and issue control requests to those sensors to obtain data of the appropriate quality (e.g. sufficient to identify a particular vehicle) such that the data can be fed into data-processing and/or fusion software. Finally, the system delivers the resulting information back to the user (and possibly others who might have interest in the results).

As we read above each of the TRs has its own specific characteristics, SAM & GL

being closer to user level (i.e. telling “what” is the task) and SS & SWE to sensor level description of task (i.e. telling “how” to satisfy the task). Using Web Ontology Language (OWL) [4] we constructed ontologies capturing their concepts. Once conceptualised we analysed them in context of the DCPD loop. At this stage we were observing what information needs of the loop the models deliver. Since none covered it completely we decided to create a mix of the concepts of the four TRs in order to gain a more complete coverage of the loop. This resulted in creation of models of hybrid task representations or HTR. In our case a hybrid model represents a combination of existing TRs such that overlapping of concepts between the included TRs is minimal, i.e. gaining a functionality from one representation without introducing redundant features from another TR. For example, if we need sensor parameter manipulation then we need a particular sensor concept from an ontology even though we already have a concept of sensor e.g. with concepts assisting in the processing step of the loop.

Once we had the hybrid models we performed experimental analysis on them to observe their performance, measured by the ratio of satisfied tasks. Because none of the combinations of TRs captured all information needs of the DCPD loop we created the missing concepts and mounted them into one of the models that performed better than the rest. This is how we obtained a rich enough model of task representation which captures all information needs of the loop. We think that such a model, which has all concepts needed by the steps of the DCPD loop, allows for its full automation. Thus, a system using it is capable to effectively handle the reactivity problem which is critical for dynamic domains.

Throughout the research we made all attempts to build our systematic approach which would allow, having an available set of TRs, to create such hybrids of TRs that are capable to effectively handle the reactivity problem. Additionally, our approach, we believe, is sufficiently general to enable potential further building on top of it or its extension either by use of a new TR or a completely different set of task representations.

In this way we want to make possible further improvement and repeatability of our results.

During the research so far, the Sensor Assignment to Missions (SAM) application was built, to implement the DCPD loop, with an aim to use it as a vehicle that would allow testing and demonstration of the dynamic issues of the problem and their handling with help of a TR. In it we demonstrated that such handling is possible and might be effective if a proper knowledge base is available, for example, as that utilised by Sensor Assignment to Missions reasoner described in [27, 60]. The SAM reasoner is capable to infer logical conclusions from the concepts and properties within the SAM ontology. This way it allows to obtain platform and sensor types which are capable to satisfy a specified task type. The SAM application was developed in close cooperation with IBM UK, particularly during the phase of integration of the application with their sensor middleware, called the IBM Sensor Fabric [6, 83]. It is a sensor infrastructure that provides, among other functionalities, access to and use of sensors, both real and simulated (for details see [6]). The SAM application was described in paper [62]. It was demonstrated live at the International Technology Alliance (ITA) Conference in London September 2008 to Ministry of Defence and USA Army attendees, where it received positive feedback. The feedback and a very warm reception of the testbed application led us to believe that the results we observed in it and ideas it presents are in fact very desired and realistic, as it was expressed by the domain experts who were kind enough to take a look at it.

We are finishing this section with the presentation of assumptions specific to our approach. They are related to our research problem characteristics, which we highlighted in the problem definition, in Section 1.2.

In our research we are focusing on user-level issues of the reactivity problem. We chose to focus on the issues avoiding introduction of the unnecessary complexity re-

lated to the whole problem. Therefore, we considered the low (sensor-) level issues as beyond the scope of the research. Though we did make the effort to understand the whole complexity of the problem, which we further discuss in the related work presented in Chapter 2. We read about them in order to be aware of them when considering a solution for the problem from the user-level perspective. We wanted to create a task representation that would be general enough to have a capability to capture these issues too. For example, one of these low (sensor-) level related issues is information quality a sensor is capable of producing. We offer a support for it by use of an appropriate task representation which can classify tasks in terms of minimal information quality required to satisfy them (but such issues as e.g. setting of a camera's parameters to obtain an optimal picture are beyond our scope). Continuing, we see it is also related to information sharing since, e.g. if users have tasks in the same region and one is less specific than the other, thinking in terms of information quality, there is a possibility that the information that would be consumed by a more specific task can be used to satisfy both tasks (such a relation is observable in the National Imagery Interpretability Rating Scale [44]). The other low level issues are, for example, the sensor assignment problem or the management and control of information processing. We consider these as well by selecting appropriate task representation models.

We also do not put a particular interest, in our research, at the influence of such low level issues as battery life of a sensor or issues related to time, such as, a sensor's availability time, sensor queuing etc. This is because our approach assumes that a user needs a solution to his/her task to be delivered instantaneously. For this reason we consider only sensors available at a particular moment. We made the assumption about the time since we want to take a particular highlight on dynamic domains where task and sensor states might change. We are focusing on the impact of the network state changes on the overall task satisfaction. For this reason every delay of information delivery is unacceptable and is counted against the overall satisfaction of a task.

We assume that our sensor network is heterogeneous where we have many different

platform and sensor types operating in it, with different sensing capabilities. They might be mobile and static but what we really care about, in our approach, is their availability and applicability for a task.

Let's remind that our aim is, by having a rich model of task representation, to make available more of alternative solutions for a task thus increasing its satisfaction.

1.4 Contributions

The main contribution is our novel way of thinking about the reactivity problem in context of the DCPD loop. It is a systematic approach that enables creation and analysis of performance in terms of the number of satisfied tasks of hybrids of different models of task representations. This allows for alignment of the concepts from ontologies of the used TRs, captured in Web Ontology Language (OWL) [4], thus forming a richer model which combines their capabilities.

The second of our contributions is the capturing of all information needs of the DCPD loop in a single model of hybrid task representation. This HTR model is a step towards a solution to the problem of reactivity. As we show, the model offers a significant improvement in handling of the problem. We prove usefulness of a rich model combining already existing task representations as a means to confront dynamic aspects of sensor networks where resources are expected to change often.

This leads to our final contribution which is the alignment of the four identified task representations in the form of an ontology that relates their works together including relation to the general assignment operation process in sensor networks represented by the DCPD loop. As part of this we formalised ontologies for each of the representations (except the SAM for which we adopted ontologies created by its authors). This allowed to observe relations between their concepts and better understand specifics of the representations and their role in a system.

Our contributions are of importance for the sensor network domain of science as, we believe, they are the first attempt at addressing the reactivity problem in sensor networks with respect to issue of task representation. Though we must admit the problem of reactivity in itself is not a new one, i.e. it was mentioned in other fields, e.g. robotics [21, 85], or agents field [48] being it hardware (robot) or software agents as in case of the path finding algorithm testing in game benchmarks [53]. We must say that though in these papers authors talk about the reactivity problem in their approach they do not consider the role of the task representation.

The problem is generally similar among all research fields where it occurs, i.e. it relates to similar sorts of issues involving response to changes in the environment of an agent, where the faster it can respond to them the better results we get. Whether it is planning of a robot's motion [85], or re-planning of a robot's path [21] where authors improve their algorithm on the previous work of others showing similar approach to us i.e. focusing on dealing with the immediate need of information of a robot (task in our case) for path assignment (for sensor assignment in our case).

1.5 Publications and Their Relevance to the Research

Parts of the work described within this thesis were published, for example, initially in [28]. Here the SAM application was first introduced as an interface for a system working on a sensor network, where authors presented their approach using mapping of sensing resources to tasks by use of the concept of capability of a sensor thus influencing the process of sensor task assignment. The SAM application demonstrates a system operating in a sensor network environment exposed by a sensor middleware, in this case the IBM Sensor Fabric [6]. It was used in the research as a proof-of-concept of our research and approach.

Later in [62] the SAM application was also used as a means to illustrate the concept. The authors discuss in it sensor assignment in a coalition context where effective sensor

resource sharing is the key problem. The ideas captured in the SAM application were demonstrated in [63], there the application also gathered positive feedback.

In paper [64] we showed our findings after experimenting with a richer task concept. It was implemented into the SAM application where, using NIIRS imagery rating scale [44], it was used to demonstrate how the quality of sensing data can be captured to influence bundling of platforms (specifically sensors mounted on them). In this way it enables to find appropriate sensing resources satisfying the task's requirements.

More recently the work was presented in [8] where we presented our conceptualisations of the four TRs and mappings between them forming a hybrid task representation (or HTR). We presented the benefits of use of such model by integrating it into the SAM application, thus giving our system a capability of automatic selection of an alternative solution and support of various delivery methods of information to the user. We showed it in an example described in the paper, where the system automatically responds to a loss of video stream (the static visible on the video output) and it changes to an acoustic solution and locates the detected vehicle (the jeep icon) presenting it on a map. In Fig. 2.6 we show the current interface of the SAM application in a similar situation.

1.6 Thesis Structure

The thesis is structured in the following way: Chapter 2 reviews the current state-of-the-art concerning the problem and its context, including our previous work. Chapter 3 highlights the DCPD loop and task representations, finally exploring the relation between them in detail. Chapter 4 discusses details of the creation of hybrid task representation models, their analysis in relation to the DCPD loop and its findings. Chapter 5 presents evaluation of the research, discussing the results of the performance simulation of the hybrid models. Finally, Chapter 6 concludes our research and discusses our future work.

Previous Work

This chapter describes our analysis of the current state-of-the-art in the research context, as well as our past work that directed our later research. Here we talk about our problem and we tie it to our approach, offering a solution. The following sections are going to focus on, respectively, the reactiveness problem and the DCPD loop, SOA and its implementation in sensor networks (SOSN), the task representation (TR), and our approach.

2.1 The Reactiveness Problem and the DCPD Loop

The research problem is lack of reactiveness of a system utilising a sensor network in a dynamic real-time domain. This means that currently, a system is not able to react automatically to changes of the state of a sensor or a task, thus a user's action is required. The lack of reactiveness makes a system using a sensor network ineffective, especially, in the situation where the time is critical, changes are frequent, and the response requires a complex action. It is a problem of high complexity, resulting from a huge variety of a task's information requirements and sensor types that it is required to operate with, and a large number of sensor and task related changes (e.g. a sensor information quality decreases; task operational area increases, thus a task requires additional sensors). It involves many complex issues, e.g. sharing, reuse, and allocation of sensors to satisfy tasks of many users, processing, and presentation of information.

Although a solution to the problem might benefit also static domains, it is definitely of utmost importance for the dynamic ones (e.g. firefighting [14] or the military [37, 61]), where the dynamic character of such domain assumes that changes of sensor and task states are possible, and they might happen often, e.g. a sensor malfunction, a change of tasks' requirements and/or priorities. Hence a rerun of the DCPD loop and new assignment of sensors to tasks is required. In dynamic domains there is simply no time to wait for a user to reconfigure a task or to select other sensors, as any delay dramatically endangers the outcome of a task or its effects might be unacceptable, e.g. the loss of a human life.

Therefore, it is important to look closer at the way information flows in a system working in these domains. It is represented by the DCPD loop, as its sole purpose is meeting of information requirements through information obtained after processing sensor data and its delivery to the user. The DCPD loop represents a well known and accepted way of serving sensor data, in the military domain of many countries, including United Kingdom, and United States of America. It is easy to observe that it is also present in non-military domains, not necessarily in the same form, but still describing the same functionalities as that captured in the four steps of the DCPD loop. The differences might sometimes concern presence of an additional relation between steps, as in the Sensor Management system framework, presented in [47], where the Processing (Sensor Data Fusion) step is additionally directly forwarding information to the Collection (Sensor Management) step, or in the loop presented in [51], where the Direction step was split into two steps Information Need (description of user requirements) and Resource Tasking (selection of sensors that match a task's information requirements, the problem of Sensor-Task Assignment).

Let's now examine the four steps of the DCPD loop (from Fig. 1.2) in more detail:

1. Direction - This step determines, for each task, what its goals are, and what information requirements must be satisfied for it to be successful, e.g. locate an

object.

2. Collection - It is a process that connects the source of data with the data consumer, which then works with it, in order to produce information satisfying requirements of a task, e.g. acoustic signals delivered to a signal triangulation processor.
3. Processing - This step concentrates on conversion of gathered sensor data into information that is meaningful in context of a task. It connects new information with information already collected this way generating new information that satisfies a task's information requirements, e.g. fusion (triangulation) of acoustic signals, in order to locate an object.
4. Dissemination - This part of the loop considers delivery and presentation of processed information to those who need it (as the loop initiator might be not the only one who is interested in this result, e.g. a fire squad would like to know where the safe evacuation path is) in the most appropriate and useful form, e.g. object located and presented on a map.

In the previous vehicle tracking example the DCPD loop, e.g. for the 1st task (Detect a suspicious vehicle.), would involve: statement of information requirements (Direction), determination of required sensors (e.g. CCTV cameras and/or licence plate recognition systems) and ways in which they are to be utilised (Collection), in order to forward data from sensors to data processors (Processing) that will analyse it (e.g. to determine if a car is acting suspicious, i.e. is breaking traffic regulations, passengers act strange and/or the car is stolen). Then in the final step (Dissemination) results are delivered to a police officer sitting in front of a laptop, and presented on a map of the highway as a red car. A need for recalculation might appear, e.g. the car will notice that it is now being chased and it will exit the highway into a road where there are no CCTV cameras, thus to enable continuation of the car tracking an alternative data source would need to be obtained, e.g. a satellite's or a helicopter's camera.

Below two examples of the DCPD loop in other domains are presented, respectively, science [39], and the military [37].

The four steps in the science domain, where the task is monitoring of the global warming effect, would describe (Direction) information requirements (e.g. a change of temperature in polar regions; size reduction of polar regions), select sensors (e.g. thermometers for polar conditions) and choose processing service (Collection), which will analyse sensor data (e.g. comparing with data previously collected to observe temperature changes over time) (Processing), that may be stored in a database and displayed on Earth's map (Dissemination).

In the military domain, the DCPD loop for the task of securing the path between two islands for a ship would involve determination (Direction) of its information requirements (e.g. suspicious activity on the sea; suspicious activity in the air), then sensors capable of ship detection (e.g. RADARSAT SAR imagery system [12]) could be assigned and the sensor generated data (Collection) would be delivered to processing services (Processing) for analysis (e.g. classification of ships, like in [84]), finally forwarding the results to the task's commander and presenting them on the sea map. The loop might run again as, e.g. an additional confirmation might be required, thus involving additional sensor types into the process (e.g. a video camera on an airplane), in order to scan the suspicious ship; or the current solution can become invalid, e.g. u-boat activity has been spotted, thus a solution using a sonar is needed.

Observing the examples given in this section it can be noticed that the DCPD loop is present. There is the initialisation step, where a user describes his task stating its information requirements, then sensors satisfying them are located, then data from sensors is processed, and in the final stage processed information is delivered back to the loop initiator.

2.2 SOA and SOSN

The goal of SOA is to interpret software or hardware functionality as a set of services, thus allowing for interoperability between implementation of functionalities independent from the conditions for which (e.g. operating system) or tools using which they were developed (e.g. programming language). Even though SOA was not specifically intended to solve all issues of the problem, it is useful for developers of sensor systems. Currently, in the research context there are two significant approaches that are applying SOA. The Sensor Web Enablement (SWE) suite of standards [52] aims to support sensor discovery, execution, composition and interoperability. These standards are being applied, for example, in business companies e.g. Northrop Grumman [2], that were prototyping a real-world global sensor web that allowed discovery, access and subscription to a sensor service; or in government organisations e.g. NASA [2, 40], where they were using the SWE standards in various Earth observation projects, in order to simplify communication process with satellites by its standardisation. Systems built on SWE standards allow their user to operate on sensors, however, they are currently largely manual, as they require a user's action in each process of the information processing loop.

The second approach is based on Semantic Web technologies, for example, Semantic Web Services (SWS) [10, 45] (these aim to increase the ability of a standard Web service, enriching it with a semantic description of what it can do), Sensor Ontologies [3, 28, 69] (use of ontologies, in order to express relations between sensor types that allows software to reason about alternative solutions for a user's task) and Semantic Streams (SS) [76] (a framework that is using semantic technology, in order to decrease the amount of a sensor-domain specific knowledge required from a user of a sensor network). The last mentioned approach is actually adopting concepts from SWS into the concept which authors of Semantic Streams are calling macroprogramming. An approach to sensor networks specifying its global behaviour instead of usual

specification of behaviour for each node of a sensor network, but it still follows the Semantic Web approach.

How the aforementioned SOA technologies relate to each other, and how they relate to a sensor and task, is illustrated in Fig. 2.1.

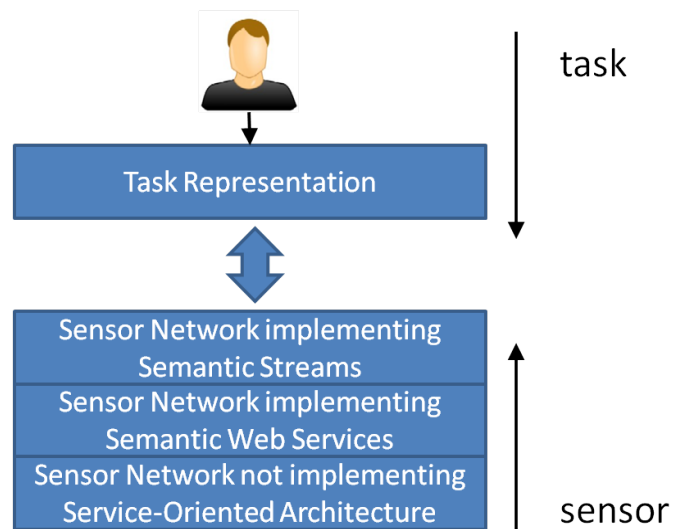


Figure 2.1: SOA technologies in relation to a sensor and task.

Figure 2.1 shows how each of these technologies moves a sensor closer to a user's expressions of their tasks (task representation). Each of the technologies allows for easier exploitation of sensors, by moving away from a level of representation closer to the sensors themselves.

- SWE supports creation and discovery of sensor services, by standardisation of their description;
- SWS provides the capability to express the semantics of a sensor in a service description, thereby indicating what tasks the sensor could be useful for, and how the sensors can interoperate with other components (for example, what type

of data sensor needs as an input, what data types it returns, and what operations it performs before it obtains the result);

- SS allows users to express simple information requirements (for example, determine the speed of moving vehicles), represented using a query language, and then sensor types and data processing services are decided by the system implementing the SS framework.

2.3 Task Representation

Task representation (TR) defines a machine processable expression of a user's task. This means an expression that a computer system is able to work with. Continuing, TR in the research context means a machine processable expression of a user's task, where a task has a set of information requirements, that can be satisfied by sensor-provided data. Therefore, a TR that allows an automated solution to the problem would involve such an expression of a user's task, where a machine, having a task's information requirements, is able to respond to changes, whether sensor or task related, automatically, without a user's action, e.g. finding a substitution for sensors, if previously assigned ones become unavailable or reassigning sensors in presence of a new task with higher priority.

In Fig. 2.2 you can see that a TR is a union of sets of information needs that it provides to the DCPD steps. It is represented by the following quadruple:

$$TR = \langle IND_1, INC, INP, IND_2 \rangle,$$

where INX - means information needs of step X.

Therefore, the definition of a TR in the problem context that expresses the relation of a TR to the DCPD loop, says that a TR describes information requirements of a task in a machine understandable form that satisfies information needs of the four steps of the

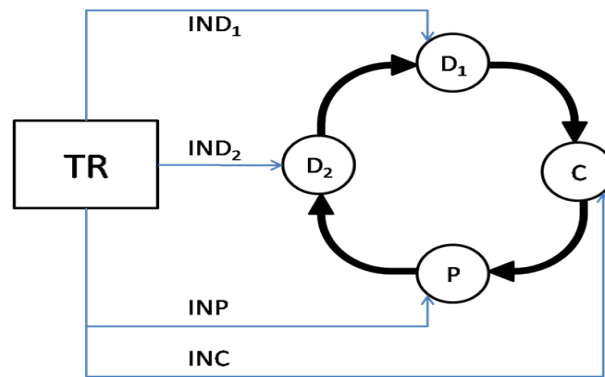


Figure 2.2: Relation of TR and the DCPD loop.

DCPD loop.

In the following subsections we present four task representations (SWE, SS, GL, and SAM) that have been researched in detail. We present what they are as a technology as described by their authors, and how aspects of the technology relate to the problem. In the second subsection we present them in terms of aspects related to task representation. We will focus on these when conceptualising them later on in Chapter 3.

2.3.1 Open Geospatial Consortium Sensor Web Enablement TR

The Technology

The Open Geospatial Consortium's Sensor Web Enablement (SWE) is a set of standards applying to sensors and services processing sensor data. It includes description of their use and their characteristics. Among all the SWE standards we distinguish five key groups of standards, they are:

SensorML (Sensor Model Language) [7] is a XML based language which is used for describing of the sensor network important elements (e.g. sensors and process models, i.e. a sensor data processing service) in a standard format enabling their independent use by different consumers. This allows for sensor and process model automatic

combination by having a common standardised description.

TML (Transducer Markup Language) [30] is XML encoding for exchanging data supporting live streaming of observations and commands from and to a sensor. In other words it is a very low (sensor) level specification describing a protocol of data exchange between elements of a sensor network.

SOS (Sensor Observation Service) [46] gives API for serving and use of data gained from sensors being it stationary (e.g. thermometer or wall mounted camera) or mobile sensors (e.g. a satellite or an Unmanned Aerial Vehicle carrying a bunch of sensors). This service in summary serves to enable a standardised way of consuming sensor generated data. For this reason, it is closely related to the O&M (Observations & Measurements) [15] set of standards. This one captures standards used for XML encoding of sensor data (measurement) but also results of its processing (observation) in a standardised fashion. It defines XML schemas for capturing and exchanging of information.

SPS (Sensor Planning Service) [71] provides standards for describing how to task a sensor (here meaning the setting of a sensor's parameters for needs of a particular task) thus readying it for production of a particular task related information. It involves such aspects as verifying feasibility of a submitted planning request, reservation or commission of a request thus 'blindly tying' sensors to a task, because a relation between a task and a sensor is not expressed within the standards, thus no sensor nor a task knows where it can be useful for. It is up to the user to know and decide what sensors he/she wants to request and for which task he/she is going to use them. The SPS also allows for other types of requests, for example, status updates, cancellation, etc.

The practical use of SWE standards, in summary, is to organise execution of sensing effort describing which and how sensors are to be used for a task. It is up to the user to specify what and how the sensor data will be utilised, how and if it will be processed to obtain needed information, finally user uses the standards to express the way of

delivery of the results. Usually in the delivery process, the user is notified when the results are calculated then he needs to invoke SWE service to collect them.

The SWE approach advocates open standards for the sensor network space. Thus, sensors or sensor data related services in general (such as data processing services) would benefit by use of common standards allowing to use services independently of their providers. This also is important from the perspective of the reactivity problem as it indirectly increases satisfaction of tasks by offering more alternative services.

The Task Representation

The SWE TR, is the one appearing in applications that are using Open Geospatial Consortium Sensor Web Enablement standards. SWE aims primarily to support discovery and access to sensors [9], thus in our terms it is focussed on sensor-level tasking. It is applicable in asset tasking (i.e. setting of parameters for platforms and their sensors readying them for production of data required by a task), as it allows description of its parameters (e.g. resolution of an image captured by a camera, speed of a platform).

This TR lacks a notation of a task's objectives or a sensor's capabilities that would include reasons for data collection, meaning what sensor data is for, e.g. what is the average water pressure in the region X? It only expresses exactly what data is needed, often from which sensor type (e.g. give me a water pressure sensor in the region X) and what to return (e.g. average), limiting a system functionality to sensors and tasks it was designed for, rather than capturing which task can be satisfied with what sensors or which sensor could be useful for which tasks.

Applications using SWE TR are normally very manual, although processing of sensor data, certainly for a particular case, could be automated, as presented in [29]. In more complex tasks the approach used in SWE TR forces high level of details to be required from a user to deliver. A system being 'unaware' of a user's plans cannot assist him much, for example, it may not provide an optional solution, if a water pressure sensor

is not available, a salinity sensor could ‘do the trick’ as it is also capable of water pressure measurement [68].

The Listing 2.1 shows an example of use of a SWE TR’s Submit Request where a user wants to turn and zoom a camera in preparation for his/her task (as seen in [71]). We see here what we discussed above, i.e. it is very low (sensor) level expression of a user’s task. It means that there is lack of relation between the tasked sensor and a task. Therefore, only the user knows how a sensor is going to be used.

```
<sps:Submit service="SPS" version="2.0.0" xmlns:sps="http://www.
  opengis.net/sps/2.0" xmlns:swe="http://www.opengis.net/swe/2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <sps:procedure>http://www.ogc.org/procedure/camera/1</sps:
    procedure>
  <sps:taskingParameters>
    <sps:ParameterData>
      <sps:encoding>
        <swe:TextEncoding tokenSeparator="," blockSeparator="@@" />
      </sps:encoding>
      <sps:values>2010-08-20T12:37:00+02:00,2010-08-20T14
        :30:00+02:00,Y, pointToLookAt,51.902112,8.192728,0,Y,3.5</
        sps:values>
    </sps:ParameterData>
  </sps:taskingParameters>
</sps:Submit>
```

Listing 2.1: Example of a Submit Request with setting of a camera’s tilt and zoom parameters.

2.3.2 Semantic Streams TR

The Technology

Semantic Streams (SS) is a technology offering efficient processing of sensor data from the perspective of information reuse for satisfaction of needs of many users' tasks. The technology was created to enable reuse of existing services for forming of new applications consisting of them, reusing them by combining them into functionally new services.

The intended use of the SS technology is to enable efficient sharing and reuse of information saving to repeat processing of sensor data. This way tasks of different users which are collocated in space and time will reuse data from the same resources to satisfy them. This is because a user is subscribing to specific types of information, rather than to the sensors directly (as for example in SWE).

In their programming model each Service is a process that infers semantic information and incorporates it into a Semantic Stream, where a Semantic Stream represents a flow of information of a particular event type having a timestamp and a set of properties. In more detail a Semantic Service has post- and pre- conditions. Matching these to a user's query (expressed in form of the final service's output) [41], we obtain a chain of services combined using a backward chaining algorithm, down to the sensor level which, from their perspective, is a specific type of Semantic Service that has no pre-conditions.

Figure 2.3 (adapted from [77]) shows how the same chain of Semantic Services [41] (old name Inference Units) could be (re)used by many queries (tasks) of different users. As such they focus on organising of processing of sensor data for increase of satisfaction of many tasks via reuse of the same information obtained from the same instances of sensors. Their approach is therefore interesting for us as it offers increase of coverage of more tasks by reuse of already produced information. This indirectly offers

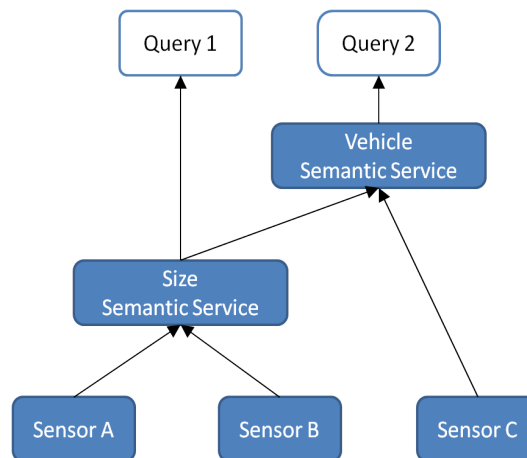


Figure 2.3: Chaining of semantic services for satisfaction of users' queries.

alternative solutions to tasks, thus it is important from the perspective of the reactivity problem handling.

The Task Representation

Task representation presented in the paper [77] is called here Semantic Streams (SS) TR. The authors are demonstrating SS service programming model and query language, that supports it.

Though all that SS TR offers is very beneficial to a user, it might be problematic to use, since the user is required to know the Semantic Streams query language. Thus, when there was no query written before, that would correspond to a user's task. The only option left, for the user, in this case, is to write the query him/herself.

In the SS query language (described in [77]), a detect a vehicle in night conditions query would look like:

```

stream(X, object),
isa(X, vehicle),
property(X, Condition, night).
  
```

The same query written in the currently used notation (introduced in [41]) would look like:

*Stream(VehicleStream,
property(X, Condition, night)).*

2.3.3 Goal Lattice TR

The Technology

The intent of the Goal Lattice (GL) technology is to identify goals, order them, evaluate them and provide means of progressing from measurable goals to non-measurable goals. This technology was originally developed for sensor management and it was aimed to be applied as a decision aid for a task analyst whenever there are many goals competing for resources.

Therefore, in practice it is used for management and organisation of effort in complex tasks. It allows to prioritise the execution of lower level goals with respect to the benefit their satisfaction brings to the top most goal (also called mission). This is done by distribution of the weight of a parent goals to the weight of their children goals. This is used not only to prioritise assignment of sensors to the bottom most goals but also to optimise overall satisfaction of the mission.

The lattice expresses how goals relate to each other, expressing how important a sub-goal is to its super-goal, which is captured by a goal's Weight value. In other words how much a goal contributes to satisfaction of its super-goals (or how much influence an included goal has on the including goal's satisfaction). A goal's Weight relation between goals in a goal lattice is well expressed in those words of its authors: "At

each layer in the lattice each goal's value is computed as the sum of the (higher) goals in which it is included and its value is apportioned among the (lower) goals which it includes." [34]. This allows us to mathematically express values for competing goals to assist a task analyst to prioritise their execution, allowing to control management of sensing resources. This is of high importance for scenarios, where resources are scarce or are expected to dynamically change their availability. Because it allows a system to continue satisfaction of a task, although partial, but from the user's perspective this is still sufficiently beneficial.

Goal Lattice does not assume a level of goal expressiveness to which it can be applied. For example, in [31] the author presents goal lattices containing another goal lattice. Basically, a goal of one goal lattice can connect to a goal of another goal lattice, allowing for expression of various task complexity.

In their later papers [32, 33] authors showed how goals can be dynamically created for an already existing running goal lattice which represents temporal needs of a user not scheduled a priori, which are instantiated from a set of previously defined goals. As such a goal lattice is not a static structure, as it was previously perceived, and it can evolve depending on temporal user needs.

Because it expresses various levels of goals from high (e.g. success of a mission) to low level goals (e.g. detect a vehicle), it allows for expression of relation of all sub goals leading to satisfaction of the task, while placing values of weight to express importance of the sub-goals for their super-goals. Allowing, as authors claim in [34], optimisation of the system from perspective of all goals knowing their relation to the top goals. This sort of approach gives a means of indirect, rather than requiring user to directly, control evolution of a task by the system. It agrees with our thinking about approaching the reactivity problem, as it opens more alternative solutions for a task by allowing an acceptable partial satisfaction of a task. Also one can imagine how it could be utilised for reusing of previously used goal lattices, following what author says in [31].

The Task Representation

The goal lattice (GL) TR, detailed in [34] allows the capture of relations between goals (information requirements) of a task and it expresses their importance, priority, influence that it has for the success of the main goal, e.g. the success of a task.

The latest approach in goal lattices (detailed in [33]), includes a dynamic goal instantiation, that assumes existence of a set of predefined goals. It allows for an indirect control over changes, thus improving effectiveness of goal lattices in the dynamic domains.

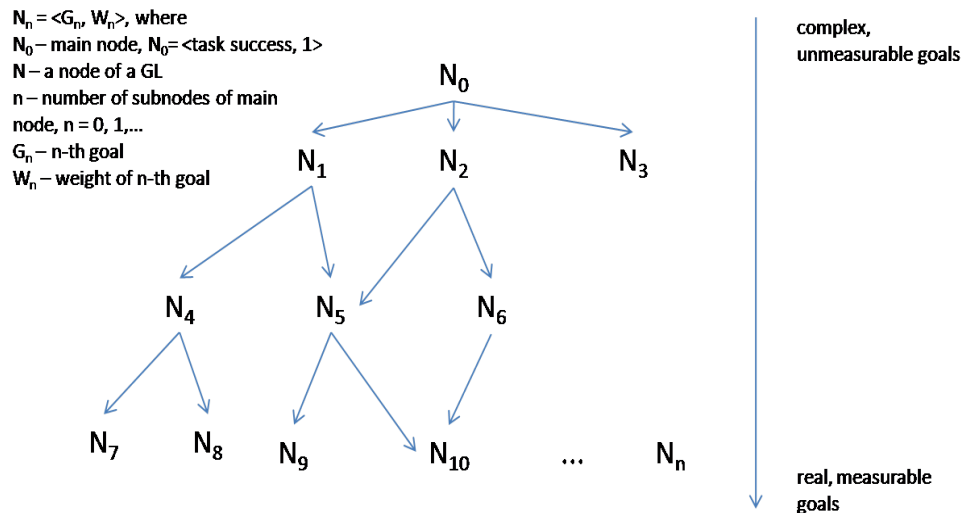


Figure 2.4: Graphical representation a Goal Lattice.

Fig. 2.4 shows an example of a goal lattice graph, where N_n is a node of a GL that has $n + 1$ nodes. A node is described by a pair of a goal (G_n) and its weight (W_n) that expresses its importance for the main goal (G_0), where a node $N_n = \langle G_n, W_n \rangle$. Weight of a parent node equals the sum of weights of its children nodes, e.g. $W_0 = W_1 + W_2 + W_3$ or $W_1 = W_4 + W_5$. The relation which is used to order the goals in a goal lattice is expressed by the words: “(this goal) is necessary to achieve (this other goal)” [34]. Later in [31] this relation was in fact relaxed to effectively express any type of sub- to

super- goal relation, authors are saying it exactly in these words: "... ordering relation, ..., is variously defined as "being necessary for the accomplishment of" or "contributes to accomplishments of" (a super-goal)...". Top goals are unmeasurable by sensors, e.g. task success, gain control over a region, minimise losses. The further from the main node the goals are the more measurable by sensors they are, e.g. minimise personnel losses, track personnel, detect hostile activity.

2.3.4 Sensor Assignment to Missions TR

The Technology

The Sensor Assignment to Mission (SAM) technology aims at controlling assignment process of sensors to missions (tasks) via capturing of knowledge allowing for mapping of sensor provided and task required Capability. The formulation of relations between sensor and task follows closely, as the authors admit in [64], to that presented by the Military Missions and Means Framework (MMF) [70], which SAM authors captured in their ontology based on MMF. They use it with an important generalisation, i.e. the concept of Task is used to represent Missions, Operations and Tasks since, as they claim, in practice such distinction between complexity of those three concepts is vague. For this reason the Task concept can have sub tasks which are also of class Task thus omitting distinction between these three concepts [64].

The practical application of the SAM technology is optimisation of sensor assignment to maximise resource use and satisfaction of tasks. It is done by providing a knowledgebase which maps task types against required capability which specific sensor types can provide. Thus, the SAM technology also allows a user to know what the alternative solutions for a task type are. This effectively improves task satisfaction in domains where task and/or sensor state can change."

The work presented in [64] looks at the sensor-task matching process with relation to actual resource allocation algorithm, particularly combinatorial auction algorithm,

which was further discussed in [67]. They presented the notion of a Bundle type representing a matching of a set of sensor types to a set of task types they might satisfy. In other words Bundle types are, effectively, a collection of resource (platform and sensor) types which instances form the actual solution for a particular task type.

In their later paper [64] authors integrate into their approach quality of data produced by sensors to data quality required to satisfy a task, which is captured in National Image Interpretability Rating Scale (NIIRS) [44]. It aligns data quality to what could be achieved with it. Thus, by having no direct relation between task types and sensor types, it opens search space for alternative solutions for a task, effectively letting a system find sensor types capable of a particular task type satisfaction. To represent this concept they built the NIIRS Task concept as a specific type of Task whose required capabilities are expressed in terms of values corresponding to the NIIRS ratings. Thanks to the knowledge captured in NIIRS, the NIIRS Capabilities may be broken down to matching Asset types providing them. The NIIRS Tasks are related to the Detectables ontology, first introduced in [16], where are defined concepts and relations between objects detectable by sensor means. There are mentioned three types of NIIRS Tasks, i.e. Detecting, Identifying a Detectable or Distinguishing between few Detectables.

This sort of approach allows us to treat satisfaction of a task from the perspective of information, while enabling optimisation of information sharing among tasks satisfiable by the same bundles. It certainly depends on the network environment or particularities of a scenario (e.g. if it allows for information sharing among different tasks) and it can be controlled, for example, by use of policies such as presented in [75, 83]. The SAM's approach, effectively adds to a solution space more options not only of type but also of instance level, by reuse of sensor provided data. For example, we might use this approach to limit engagement of instances of resources by reusing of the same bundles. It is beneficial from the perspective of the reactivity problem tackling, since it allows us to find alternative solutions for a particular task type.

The Task Representation

In the paper [62] Sensor Assignment to Missions (SAM) TR is presented as a union of two sets, a set of interpretation tasks and a set of required sensor capabilities. They currently use a very narrow set of NIIRS (National Imagery Interpretability Rating Scale) interpretation tasks, called later NIIRS tasks. The SAM TR could be described in a mathematical form as:

$$SAMTR = (NT_1, \dots, NT_n) \cup (RSC_1, \dots, RSC_m),$$

where NT (NIIRS Task) describes the objectives of a task in terms of what it intends to achieve, meaning, what operation a task is to perform on a physical event or a thing. It is represented by the following tuple $NT = \langle action, detectables \rangle$, where an action determines what to do on detectables, and a detectable is an event or a thing that might be observed by sensor resources. Examples of a NT could be: $\langle detect, cars \rangle$, $\langle identify, plane \rangle$. n is the number of NIIRS tasks. RSC (Required Sensing resource Capability) means a capability that a task requires a resource to provide, before it can become useful for this task, e.g. ACINT, IMINT, RADINT (respectively: acoustic, imagery, and radar intelligence). m is the number of RSCs.

SAM TR is using a query language to express a task's information requirements. The example below shows how a query used in SAM TR might look like, when a task is to Detect a vehicle in night conditions (written in the SPARQL query language [81]):

```

PREFIX niirsOnto: <http://example.com/niirsOntology#>
SELECT ?vehicle
WHERE {
    ?x niirsOnto:vehicletype ?vehicle ;
    niirsOnto:inCondition niirsOnto:Night .
}

```

2.4 Other Related Work

2.4.1 Sensor and Sensor Network related Ontologies

The ontology described in the paper [18] is relevant to our work in general. It is relevant to us especially because of what it offers, i.e. an optimisation of searching for sensor or data processing services for improvement of precision of search. Yet it is too low level and it talks about the relation of sensor (and processing service) and data not touching the role of a task in a system.

In our approach, in order to keep it domain independent, we tried not to focus much on ontologies that are concentrating on a particular domain, for example, OOSTethys [13] ontology for oceanographic observations created by the OOSTethys community¹ or Maritime Metadata Interoperability (MMI) [36] ontology for oceanographic devices. Our approach is general and as such we stayed away from inclusion of domain specific concepts. Though we were checking if the concepts that we see as general are also fitting to concepts in these ontologies.

The Semantic Sensor Network (SSN) ontology [3], created by the World Wide Web Consortium's (W3C) Semantic Sensor Network Incubator Group (SSN XG), can be briefly described as capturing sensor network related concepts, enabling its domain-independent use. In our terms we would also describe it as being a low (sensor) level ontology, since it captures sensors and other processes (such as processing services) related to use of sensors in sensor networks, but it doesn't relate them to a user's task. In fact the SSN ontology is built to be compliant with the OGC SWE standards. Thus, they want to allow applications already built using SWE standards to benefit from semantic approach while continuing to comply with the SWE standards. The very

¹<http://www.oostethys.org/>

significant difference between the work of SWE and SSN is that the SSN follows the Semantic Web approach thus it offers potential to categorise and reason about the concepts it uses. Its similarity to the SWE work is one of the reasons for not introducing the SSN XG's work as another TR. The main reason though is lack of relation of its concepts to a task. Because of it we needed to analyse the SWE standards ourselves to look for reasons of this. As a result we created our ontology of the SWE TR. Our mapping of SWE standards corresponds to that presented by the SSN XG group, though by no means ours is that detailed, as it was beyond the scope of our research.

After research there appear to be very few ontologies that include task and specifically its relation to other sensor and sensor network important concepts. We know that having these in one ontology would allow reasoning, for example, about task satisfaction as it is possible with the ISATR ontology discussed in [28, 59]. This is the ontology that the SAM technology makes use of to enable mapping between sensor types providing and task types requiring capability.

OntoSensor ontology, presented in [69], offers a Semantic Web compliant approach adopting SensorML [7] part of OGC SWE standards, extending concepts in IEEE SUMO² ontology and referring to ISO 19115³. Its aim, as author claims, is to support fusion of data coming from different sensors by conceptualisation of relations between the concepts which construct the ontology. We didn't include the OntoSensor ontology, because it effectively captures some of the concepts from SAM (sensor type details) and (part of) OGC SWE. Since our approach is more general than that which is shown by the author (i.e. the role of task is not expressed in the ontology) we decided to make our own ontology of concepts related to the SWE standards in general, which we captured in SWE TR discussed in more detail in Section 3.2.1.

The SWAMO framework [80], and the ontology that comes with it, offers a variety of functionalities allowing for creation of a system that would be able to work with

²<http://www.ontologyportal.org/>

³http://metadatos.ingemmet.gob.pe/files/FDIS_19115.pdf

other systems (implementing the same framework) to connect into a so called ‘system-of-systems’ for a mutual benefit. It would be able to discover, find and set sensing resources depending on a user’s request. However, it lacks, for example, rich relation of task to other concepts. Thus, tasks of a user can only be expressed in form of request, e.g. “I want XXX [sensor]” [80], rather than i.e. my task is YYY, and let the system then worry how to satisfy it. We suspect it might be the result of focusing on the OGC SWE standards explicitly and not capturing a richer relation between a task and sensing resources. Anyway the SWAMO framework looks very promising as it offers use of intelligent agents (each playing a role of system in their architecture of ‘system-of-systems’) to assist in management and control of sensors, hiding (at least partially) tasking complexity from a sensor network user.

2.4.2 Sensor Allocation Problem

We mentioned already that observable is the similarity between research in other domains, particularly in the domain of robotics. In paper [24] the authors present a similar problem conceptually to that of assigning sensors to a task, here robots to a task. If we see the robots as a platform carrying sensors, then the problem effectively becomes the same. Though we are focusing more on their sensing capabilities rather than their physical capabilities, e.g. transportation of a thing by a robot (i.e. coordination of robots). This similarity is also mentioned in the sensor network literature, for example, in [67] authors build their problem basing on that of robotics. Later in [58] authors formalise their problem as Multi-Sensor Task Allocation (MSTA) as to highlight its similarity to that of Multi-Robot Task Allocation (MRTA) presented in [24]. The first take on the formalisation of the MSTA problem was shown in the paper [57] where authors presented a taxonomy of MSTA problem relating it to that of MRTA. Our approach to the sensor tasking problem is similar to that in [67] with the exception that we are considering a more general problem, which the sensor allocation is a part of. We also assume the user wants a solution to his/her task as soon as it is available.

Additionally, we do not want a user to worry about all the actual details of sensor assignment or parameter setting. We aim to enable a user to express what his/her sensing task is and the rest should be taken care of by the system. In the paper authors also discuss how a need for an alternative solution for a task can be handled from perspective of their problem. They explored the mechanism of preemption of resources serving a task as a means to allow treatment of tasks of different priority. The other mechanism they explored is called the rebidding mechanism (meaning "...the ability for a task to request a different bundle of sensors in case the resources requested the first time were not available..." [58]). They also showed that such an approach combining rebidding and preemption mechanisms offers a greater total profit, i.e. more satisfied tasks; than allocation without these mechanisms.

The allocation problem is presented here [57] in the context of an emergency response scenario. It is important for us as one of our problem's characteristics is a dynamically changing environment. In this case many changes are expected in the set of available sensors. They highlight how important it is to timely deliver required information to a task and to continue its delivery, and also that a typical user (rescuer) usually lacks expertise to specify what sensors he/she needs to use, or how to use them. Thus, they offer here an automated approach to allocation of sensing resources. In relation to our work we also believe it is important to allow a user to express his/her task in higher terms of expression, e.g. my task is to monitor the accident site, rather than specifying what sensors he/she needs and how; while ignoring details of tasking specifics not only those related to resource assignment. Thus, our problem in comparison to that described by the authors is more general, as we are focusing on the role of a task in such scenarios. This is what distinguishes our approach.

Authors of [66] present analysis of the field which they call sensor selection. In other words it is a part of the sensor allocation problem but more focused on determination of which sensor to choose depending on its availability, its detection range, battery life, bandwidth available for information transportation, area coverage and many other

details related to sensor itself or a sensor network. This work is interesting for us as it provides an overall view on the sort of low (sensor) level related issues of the sensor allocation to a task. Thus, we used it to familiarise ourselves with the low level issues. Though it does not discuss the role of a task in the process, nor how complexity of task model can influence it which is our main focus.

Finishing this section we want to highlight that the problem of sensor allocation and the approaches we described above are relevant to our research. Familiarising ourselves with this problem gave us the understanding of the range of parameters and issues that a system and sensor network might be concerned with. Particularly the low level details, related mostly to sensor and information, such as, sensor availability, range, information quality, information sharing, task priorities, etc.; which we tried to capture within our models. We experimented with some of these issues in our framework, the SAM application (described in 2.5.2).

2.5 The Research and Our Approach

This section describes the research, the problem and the way we approached it. Subsection 2.5.1 identifies the research problem. Subsection 2.5.2 contains description of the SAM application, which serves as a testbed in this research. Subsection 2.5.3 describes our research approach in general.

2.5.1 Identification of the Research Problem

The task representation in the research context, as it was presented in Section 2.3, can be put, in other words, as the problem of how to represent a task so it can be broken into a set of information requirements, and how to express them so a system can work with them, automatically making some decisions (e.g. in the vehicle tracking example, when a suspicious car enters a tunnel, thus rendering a helicopter camera's data tem-

porarily useless, a nearby CCTV camera located inside the tunnel is assigned instead) without necessity of a user's action, in contrary to the current solutions ([50]), where a manual intervention is required.

To observe what the influence of TR is on the steps of Direction, Collection, Processing and Dissemination, the DCPD loop implementation was created, see Fig. 2.5. This figure highlights in which points DCPD steps are present (round areas) and shows implementation specific parts (rectangular areas) of our loop.

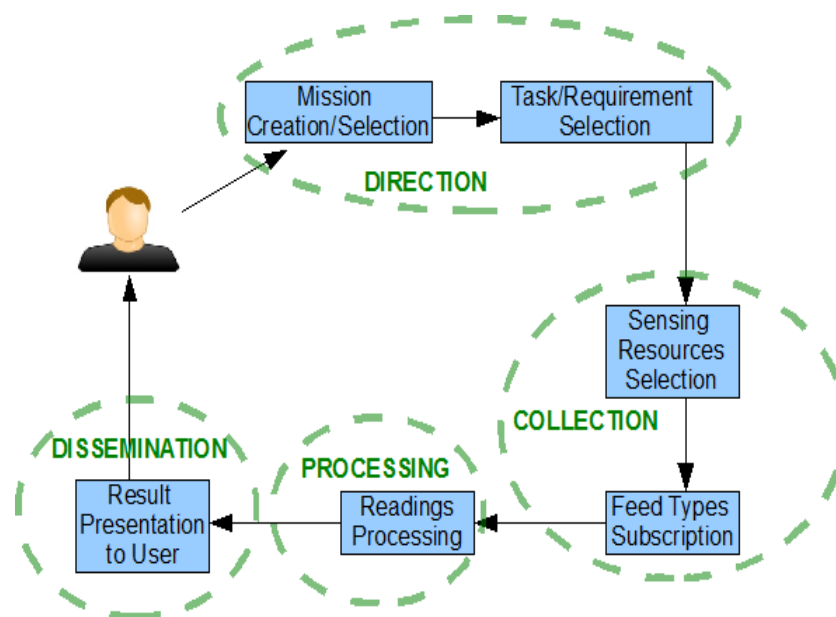


Figure 2.5: Overview of the Sensor Assignment to Missions application

The steps presented in Fig. 2.5 are explained below:

1. Mission creation/selection - (Direction) A high level description of a mission. It contains a statement of a mission's objectives, that requires further interpretation and breakdown into more focused tasks.
2. Task/requirement selection - (Direction) Selection of regions, where tasks are going to take place and specification of information requirements for each task.

Tasking could progress, like in [61], where high level of information (not measurable by sensors) is broken down to scenario specific information requirements, and finally broken into more specified requirements for which sensor-provided data could be useful.

3. Sensing Resources Selection - (Collection) This step involves finding of the resources that are able to satisfy previously selected information requirements for each task, matching of information requirements to types of available sensors (matching examples could be found in [27, 28, 59, 60]), which limits the search space for a process of allocation of sensors to tasks (for an example of an allocation algorithm see [56]), making it faster and more efficient.
4. Feed Types Subscription - (Collection) After a sensor's data format (feed type) and data fusion mechanisms are selected the actual reservation of sensors for a task is happening.
5. Readings processing - (Processing) In this step, operations are executed on readings obtained from sensors, i.e. readings interpretation and understanding of their meaning for a task. This involves data fusion processes, their details could be found in [22, 55]. These processes are intended to exploit information, so it can satisfy as many information requirements of tasks as possible. This considers various aspects of a relation between an information and a task, the authors of [22] are classifying it as Temporal, Spatial, Context, Content, Priority, Historical, Usage.
6. Result presentation to user - (Dissemination) Presentation of results to a user in an informative way i.e. if a user is supervising many tasks, the amount of information delivered to him must allow him to cope with it. In paper [22] the author highlights the importance of a fusion processing (the previous step of the loop) and its close relation to this step. Thus, the form in which results are presented is influenced by the way in which they are processed.

We have presented our loop which is implemented by the SAM application, described in Subsection 2.5.2, which is meant to serve as a testbed for various TRs.

2.5.2 Presentation of the Testbed Application

The Sensor Assignment to Missions (SAM) application implements the DCPD loop in the form presented in Subsection 2.5.1, and presented in Fig. 2.5. The application was developed with the intention to serve as a testbed for the research of the TR in the research context. To experiment with the reactivity problem, and to demonstrate the utility of a TR in a system working in a sensor network, we implemented the framework application that serves as the system shown in Figure 1.1. This follows work previously presented in [62]. Our approach follows the principle that many tasking details, especially those related to the sensor-level tasking (e.g. selection and setting of parameters for an asset) should be inferred, where possible, from higher-level descriptions of a task (i.e. in terms of Section 1.1: “what”, not “how”). However, we don’t want to prevent users from choosing the types of solution they are interested in (i.e. “how”) because there may be situations where a user prefers one kind of solution over another. For example, a vehicle detection task may be solvable by acoustic and imagery means, but the user is really interested only in the imagery solution as they can then use the images of detected vehicles as a proof that they breached a speed limit and penalise their drivers.

The application can find sensors for user-specified requirements, by use of the SAM ontology and reasoner [28] (it provides a mapping between a task’s information requirements and sensor types); it works with sensors and the environment in which they are deployed (the IBM Sensor Fabric, presented in [6], a sensor infrastructure, that among many functionalities, allows to access and use sensors, both real and simulated); it can process (fixed processing of acoustic information, triangulation of the signal); and deliver & present to a user information which satisfies his/her task.

The latest interface of our framework application is shown in Figure 2.6; more readable elements of this interface appear in Figure 2.7. The figures show a vehicle detection task using acoustic sensors (e.g. acoustic array) mounted on a platform (e.g. iRobot PackBot), since the previously used video feed was recently lost (presented by static image in the pop-up window to the left). It also shows a simple goal lattice diagram expressing priority and relation of our task of vehicle detection to mission success. Figure 2.7a shows the interface using the SAM TR. The tabs allow a user to select from currently defined NIIRS Tasks (e.g. detect vehicle) and specify Required Sensing Capabilities in terms of intelligence types (e.g. ACINT, IMINT, RADINT). The list in Figure 2.7b shows platforms with sensors mounted on them bundled together into a complete solution for the task (bundle no.2 is selected, containing the PackBot platform, P2, with an acoustic array installed on it). The third element of the framework, shown in Figure 2.7c is a map that serves three purposes: to allow a user to specify the area of a task, to present the location of assets providing some of the capabilities required by a task, and to deliver processed sensor information where appropriate (the jeep icon, representing detected vehicle by the acoustic array on the PackBot, P2).



Figure 2.6: Framework application user interface

Currently, the framework’s task representation integrates the SAM TR — which provides

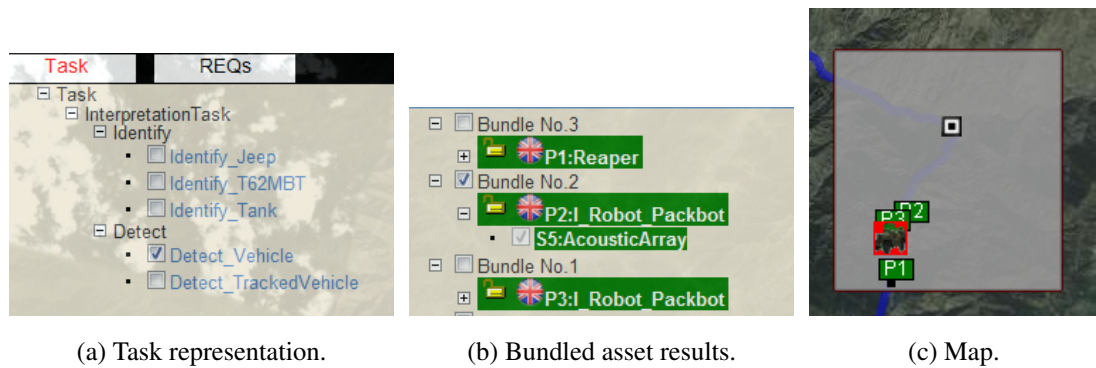


Figure 2.7: Interface elements for a vehicle detection task

sensor-task matching — and uses an SWE-like sensor network layer, that has capabilities allowing discovery, management, and retrieval of data from sensors. It also uses the GL TR to allow expression of a task’s priority. The processing of sensor data, though present in the framework, is fixed (e.g. acoustic data is triangulated to locate an object), but this could be addressed by incorporating the Semantic Streams features. By extending the current framework to include these additional capabilities, we would have a complete implementation of the system illustrated in Figure 1.1.

In experiments using the SAM framework we observed potential benefits of addressing both (user and sensor) levels of tasking, which task representation is meant to capture. Thus, we want to summarise this section by saying that: Having an open, explicit task representation that addresses both user- and sensor-level tasking also allows the possibility for the creation of new sensor planning, operation, and delivery mechanisms and tools that use the existing TRs, or further extend it with additional features.

2.5.3 Our Approach

Identification of the research problem and its context (Section 2.5.1), and testing of the DCPD loop in the SAM testbed application (Section 2.5.2) highlighted the very important role of the loop in the context of the problem and its close connection with a TR. In order to present it and to verify how a TR can automate solutions of the prob-

lem, it is important to find what are the information needs of the DCPD loop, i.e. input and output of each step. It will allow us to compare the TRs in the context of the DCPD loop, in other words, to verify the amount of information needs of each step of the loop are provided by each TR. Observation of differences and similarities between TRs in this context will allow to show if there is a TR that satisfies all the information needs of the DCPD loop. If there is no TR that satisfies all the information needs then it will be required to verify if a combination of TRs can satisfy them or at least can satisfy more than TRs separately.

After building such hybrid of TRs (or HTR), we will observe what is the coverage of the DCPD's steps now. Then having many HTRs we will test them in a simulation, where we will simulate a sensor network, its behaviour and changes in its state. From these results we want to obtain a HTR that delivers all information needs. If such a HTR is not present we will create the missing concepts and integrate them into one that performs the best of them all. Effectively, we will create the HTR that is the best fit for dynamic environments where the likelihood of a sensor's state change is high. The final HTR will be evaluated under the same conditions in our simulation where we will measure percentage of gain in task satisfaction when using this model versus the satisfaction of other models. This way we will evaluate if our approach was successful.

Task Representation

We start this chapter with a description of the features of the DCPD loop (Section 3.1). We continue, in Section 3.2, with a description of the four existing models of task representations we identified. Each represents a particular approach that people use in creation of a task submitted onto a sensor network. We show conceptualisations of their concepts which we captured in the form of a set of ontologies using Web Ontology Language (OWL) [4]. They can be seen as our base elements which we use in later research, presented in Chapter 4 which deals with further analysis of our findings and creation of a hybrid of the task representations presented here. In the last section (3.3) we present the role of the models in the context of the loop, i.e. what information needs of the steps of the loop they deliver.

3.1 The Features of the DCPD Loop

Here we present the features of the DCPD loop [86] described for each of the four steps. These features can be viewed as a set of information needs required to go around the loop in an automatic way, i.e. without the need for user intervention. Knowing what is needed in each step of the loop and delivering it once, at start, with a description of a task makes an automatic operation of a system possible. All the features have been taken from the literature describing a system operating in a sensor network. These papers are [86] introducing the concept of the DCPD loop, [37, 51] are presenting

problems and issues involved in automation of a system operating in a sensor network of a critical time in the military domain, [47] where the authors describe general issues of sensor management with a focus on the processing step, [61] here the authors present the problem of operation on a sensor network in a step by step walkthrough, where functionality looks like that which is captured in the loop, [14] though in this paper the authors are not talking directly about the loop, they are describing its function in the form of requirements of a potential system that would work with a sensor network in the firefighting domain.

The DCPD features are effectively possible user interaction steps, and this is how we present them below. The number, and type, of actions taken in an interaction step, presented below, answers to the specifics of the information needs of the DCPD loop. These actions represent a minimal set of requirements needed by a task, from the perspective of the information needs of the steps of the loop [86]. It is important to remember that they are not necessarily implying the order, or number, of actions a user will take in an implemented system. They are merely meant to present general steps an average user of a system, working with a sensor network, might need to take in order to describe his/her task in sufficient detail, thus giving a system enough information for automated operation.

- Direction (D_1)
 - Task Description - Describe the task, e.g. select the region of action, task requirements, express its importance (priority) etc.
 - Sensor Type Selection - Select sensor types, from a set of sensors having the selected requirements.
 - Sensor Instance Selection - Locate & select sensor instances for the sensor types.
 - Sensor Setting - Set parameters for the sensors.
 - Monitoring - Monitor the task status. Most likely a user will not be involved

in this step at the beginning, otherwise he/she might specify the conditions for the monitoring, e.g. when to change the task's solution. The role of this step is to enable a user to express how flexible his/her task is, meaning what the user sees as a minimal solution for the task. Thus, allowing for some degradation of a task solution over time but still satisfying it, applicable in situations where no sensors producing a better solution are available. For example, a user wants to identify a car but is also happy with just its detection, e.g. the user might have another method of identifying a car, knowing it was detected.

- Collection (*C*)
 - Data Collection - Collect data from the sensors. Here a user might need to select how the data should be collected, e.g. stored in a database or, maybe, involving the next step, passed straight to the processing services.
 - Data Forwarding - Forward the gathered data to processing services, which can extract information, solving the task from the data. In this step the role of a user could be to state to which processing service the collected information should be forwarded.
- Processing (*P*)
 - Data Processing - Process the data into meaningful, for the task, information. A user here may need to express a flow of data and specify the algorithms which should be applied for the received data.
 - Information Forwarding - Further process or forward to delivery services, e.g. information which may be a good solution for one user might not be satisfactory for another, for example, only detecting a vehicle, is not enough for a user who wants to locate it, and both might be satisfied by the same information if it is rich enough and it has the detected vehicle's coordinates.
- Dissemination (D_2)

- Recipients Selection - Determine recipient or recipients of the task's solutions. Often more people than just the task's author might benefit from task results, e.g. the task might be created by an analyst and the actual information consumer might be the rescue team on the site of an incident.
- Delivery Medium Selection - Select the way in which to deliver the solution to a recipient, e.g. live stream, email notification, packed to save bandwidth etc.
- Presentation Format Selection - Select the format in which the solution will be presented to a recipient, e.g. map location, image, etc. This might also be dependent on the device type of the information recipient (or conditions in which the recipient operates). For example, a police officer taking part in a car chase would benefit more from a representation showing his/her and a suspect's location on a map, rather than just the suspect's location. Since he/she needs to operate quickly.

3.2 Formalisation of Four Existing Task Representations

As introduced in Chapter 2, we conducted a literature review to identify task representations (TRs) that satisfied two criteria: (1) inclusion of features relevant to user-level and/or sensor-level tasking and (2) a sufficiently-detailed description in the public domain to allow us to analyse and compare the features with other TRs. To perform our comparative analysis, it was useful to explicate the features of each representation in a common format. For this purpose, we chose the Web Ontology Language, OWL to capture the features as class and property definitions. OWL is a formalism designed to support machine processing of the content of information [4]. Use of OWL in creation of an ontology brings, among many useful capabilities, openness and extensibility, compatibility with Web standards and scalability. Wanting to capture concepts of the

TRs in a standard representation that is extensible and scalable and which would allow to reuse the created ontologies in future, we decided to use OWL in our research. Additional benefits arising from the use of OWL were that we would then be able to use ontology-alignment techniques to define relationships between the four representations, and also potentially use the ontologies as schemas for representing task instances.

In this section we discuss the four TRs, introduced in Section 2.3, in the following order: Open Geospatial Consortium's Sensor Web Enablement (SWE), Semantic Streams (SS), Goal Lattices (GL), and Sensor Assignment to Missions (SAM). We describe their characteristics and introduce our ontologies (available in Appendix A or¹). Additionally, in each subsection we demonstrate the ontological concepts using the example of a car detection task. This task represents a situation where a user's intention is to monitor an area of interest for any presence of a car in that region. We will use this example to show the sort of details can be expressed with concepts of each TR and to which (user and/or sensor) level they apply. In Figures 3.1 to 3.7 below, we use the following notation: classes are depicted as ovals, and instances as rectangles; data properties are the text next to corresponding instances (rectangles); subclass relations and all other properties are shown as labelled solid arcs.

The notions might appear confusing at first as they are used on two different figures (the first presenting relations between classes the second between instances of the classes) shown for the same TR. Therefore, we present here a sample of each notion in order to help understanding the figures better. We explain them giving examples as seen on the two figures of the first TR we discuss below, i.e. Fig. 3.1 (presenting relations between ontology classes) and Fig. 3.2 (showing relations between instances).

- Oval - represents a class, e.g. Asset in Fig. 3.1.
- Rectangle - represents an instance of a class. Its name always, among instance specific details, explains which class it represents. For example, Predator A Platform, in Fig. 3.2, represents an instance of Platform class.

¹Available from: <http://users.cs.cf.ac.uk/K.Borowiecki/Thesis/Ontologies.zip>

- Data properties - are texts next to an instance (a rectangle) they relate to. For example, in Fig. 3.2, Location parameter has data properties of latitude and longitude which represent coordinates of the position of Predator A Platform.
- Labelled dashed arcs - have a title which explains what is the relation that they represent. For example, Platform *carries* Sensor, in Fig. 3.1, means that an instance of the Platform class can carry (have mounted on it) only instances of Sensor class, e.g. in Fig. 3.2 Predator A Platform carries Camera A Sensor.

3.2.1 Open Geospatial Consortium's Sensor Web Enablement TR

The Open Geospatial Consortium's Sensor Web Enablement (SWE) suite of standards is large and complex. We have tried to identify its features that are most relevant to the representation of tasks at the user and sensor level. Our resulting ontology is shown in Figure 3.1. The ontology includes the concepts which the Sensor Planning Service (SPS) makes use of when setting parameters of sensors, and also those that allow retrieval of sensor data via the Sensor Observation Service (SOS). The SWE approach defines SensorML [49] for the representation of sensors and measurement processes, thus allowing for interoperability between the services of the SWE suite (e.g. SPS to task a sensor and SOS to retrieve data from it). Additionally, SensorML defines Process Models, representing transformation and processing of sensor data.

In the SWE TR ontology, by an *asset*, we mean both a sensor and platform. In the SPS specification document [71] the authors define the word "asset" as synonymous with a sensor or simulation, though in the examples in the document they use it while presenting the service's functionality, where they also state that it might be applied in tasking of both platforms and sensors. The use of the SPS service for tasking of both a platform and a sensor mounted on it was demonstrated in [35]. Therefore, in the ontology we use Asset as super type of a Sensor and Platform. Between these two is a *carries* relation representing that a Platform might have some Sensors mounted on it.

A Submit Request describes tasking details for one or more Assets to which it assigns tasks. Each Asset has some Parameters which might be instances of subclasses of the Parameter class: Non Taskable Parameter (which is used to describe characteristics, capabilities or state of an asset) and Taskable Parameter (which represents parameters that might be modified e.g. zoom level for a camera, its facing angle or speed of a platform). In SensorML, sensors are modelled as processes that “convert real phenomena to data” [49]. Therefore, a Sensor is a specific type of Process Model in SensorML, defined as a processing block that performs operations on data, and has its input, output and all parameters precisely defined.

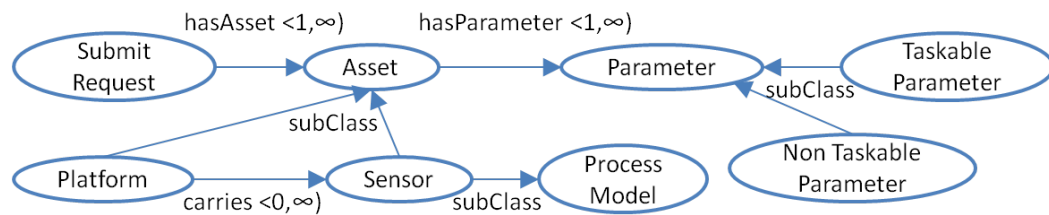


Figure 3.1: Ontology for the Sensor Web Enablement TR

Figure 3.2 shows the use of the SWE TR in the example car detection task. Here we observe how low level the TR is, lacking the capability to express the relation of the submitted request to the task’s needs, which is to detect a car.² Using the TR we can set values of taskable parameters for a platform and sensors, on that platform, that are capable of satisfying our task. It is up to the user to know what sensors and/or platforms might be capable of satisfying a task. Here a user has selected a Predator (an Unmanned Aerial Vehicle) platform with a camera that has infra red intelligence (IRINT) capability, and has assigned values to other parameters of the assets.

²We note that it is possible in SWE to represent how information can be combined, by specification of proper process models. For example, it would be possible to represent a model for image processing that can observe presence of a car in an image.

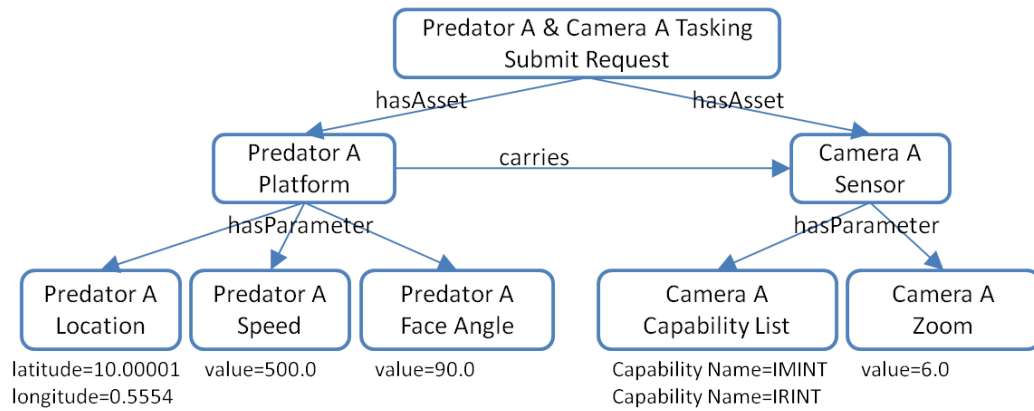


Figure 3.2: An example SWE car detection task

3.2.2 Semantic Streams TR

Our ontology for the Semantic Streams (SS) task representation is shown in Figure 3.3. The SS TR is useful for both user- and sensor-level tasking. In fact, its applicability lies more in the middle between high and low-level task expressions. This approach allows creation of an information flow, called a Semantic Stream [41] (previously called an Event Stream [76]), which represents a sequence of events of the same Event Type, and originates from a Sensor, then the stream may pass through a series of Semantic Services which add new semantic information about the event type of the stream, new Semantic Properties, and/or modify the Event Type of the stream. A Semantic Service (previously called an Inference Unit [76]) represents a data processing operation that is performed on one or more input Semantic Streams, and outputs (creates) one or more Semantic Streams which now are enriched with additional semantic information. As a result, the event type of the output stream receives new Semantic Properties and/or is modified, e.g. a vehicle detection service could take a semantic stream having an “object detection” event type and, on successful determination that the object is a vehicle, change the event type to “vehicle detection”.

A Sensor is a specific type of Semantic Service which has no input streams [41].³ A

³We note that the recent definition of a sensor is richer than the previous one and is equivalent to combination of previous concepts of Sensor and Wrapper Inference Unit [76].

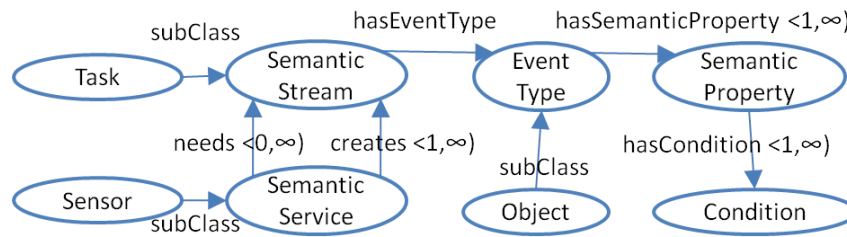


Figure 3.3: Ontology for the Semantic Stream TR

Task is a specific kind of Semantic Stream which is not an input into any Semantic Service; in other words, it is a final output stream [41] which provides the information specified in a user’s query. A Semantic Property contains specific information about the Event Type it describes, e.g. the location, observation time, or anything else that tells us more details about the nature of the observed object, such as the speed of a car. Each semantic property might have some Conditions, which could put constraints on the property value, e.g. speed of a car greater than 100.

The example in Fig. 3.4 shows how concepts of the SS TR’s ontology can be instantiated to express the car detection example. In order to present the capability of the TR to put constraints on the result space in the data processing stage, the example was made more specific: to detect a car with speed greater than 100 km/h, similar to the example in [76]. For brevity, we omit *hasEventType* properties for each stream, and *hasSemanticProperty* (for example, for the “Car breaching 100 km/h speed limit” stream the event type would say it is a car that has property speed which satisfies a condition, greater than 100).

3.2.3 Goal Lattice TR

The Goal Lattice (GL) task representation was originally conceived for task planning. A GL consists of a collection of goals, which might be ordered, for example, by the following relation “(this goal) is necessary to achieve (this other goal)” [34]. The ordering relation was relaxed in later paper [31] expressing now that a sub-goal ‘con-

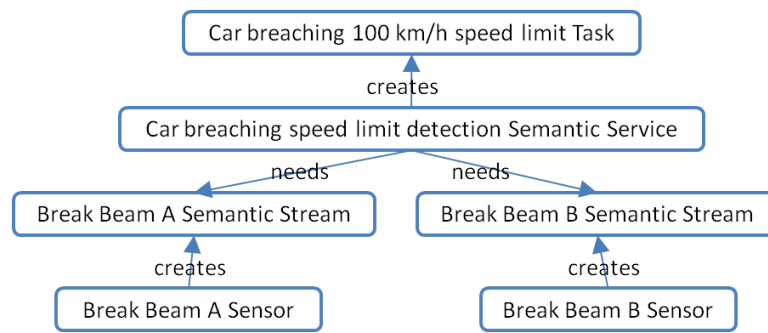


Figure 3.4: An example Semantic Stream car detection task

tributes' to its super-goal. Top goals are unmeasurable by sensors, e.g. task success, gain control over a region, minimise losses. The further from the main goal the goals are, the more they are measurable by sensors, e.g. minimise personnel losses, track personnel, detect hostile activity. Our GL TR ontology is shown in Figure 3.5. Each Goal has a Weight, expressing the importance of that goal to the top-most goal (which represents the success of a task). The weight values might be used to prioritise the goal satisfaction process. Weight values are in the range between 0 and 1 inclusive, and for the top-most goal the weight always has value 1, meaning a goal that must be satisfied otherwise the whole task fails completely [34]. A value of 0 means a goal does not have influence on the task and its satisfaction could only bring additional information that might be interesting, but is not crucial to the task (e.g. having already identified a car, receiving additionally its image) and it will only have resources assigned if there are surplus resources after satisfying all non-zero weighted goals.

According to the more recent literature on the GLs [32] there are two types of Goal: Static Goal and Dynamic Goal, which extend the earlier definition [34] where there was no distinction between goal types. The difference between these is that a dynamic goal has no sub-goals, thus is located at the very bottom in the lattice structure, and is created when a need arises. Noticeable is also a sub type of Static Goal to which author in [32] refers to as Sensor Task and it represents all static goals which only have children that are instances of dynamic goals. There are three types of sensor task that were mentioned in [32]: tracking (monitoring of a detected object's current position),

identifying (determination of an object’s characteristics and intentions) and searching (detecting presence of an object in a location). They are represented, respectively, in our ontology as: Track, Identify and Search.

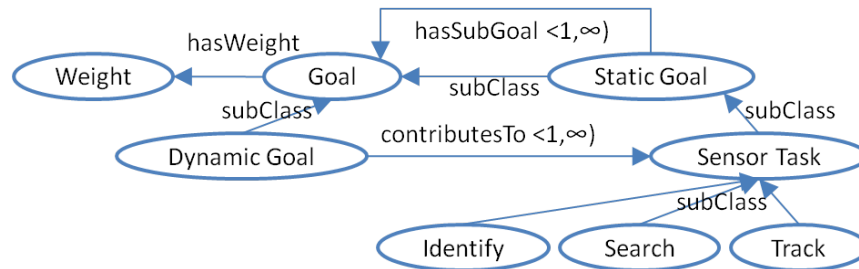


Figure 3.5: Ontology for the Goal Lattice TR

Figure 3.6 shows a car detection task as part of an emergency response scenario, where a user’s task is to monitor the activity of people and cars in a particular area. Detecting trucks is an additional goal with weight of 0 as the only trucks that might be present on the scene are assumed to belong to rescue teams. This allows us to observe a breakdown of weight value as we propagate from the top most goal through static goals, finally reaching at the very bottom dynamic goals of Truck and Car Search, which have been instantiated for a Vehicle Search static goal. Here, Human Search is a sensor task as, depending on the need, it might be further broken down into dynamic goals, for example, search for victims or rescue personnel.

3.2.4 Sensor Assignment to Missions TR

In the Sensor Assignment to Missions (SAM) TR, a user’s task (e.g. search for victims) is represented as part of a set of tasks called an operation (e.g. rescue people); a set of operations is called a mission (e.g. a rescue mission) [28]. Every task, operation and mission can have a set of sensing Capabilities. The notion of “interpretation task”, introduced in [62], is a specific type of a task that derives from the National Imagery Interpretability Rating Scale (NIIRS) [44] which defines the kinds of intelligence tasks that can be achieved by imagery of different qualities. For example, NIIRS indicates

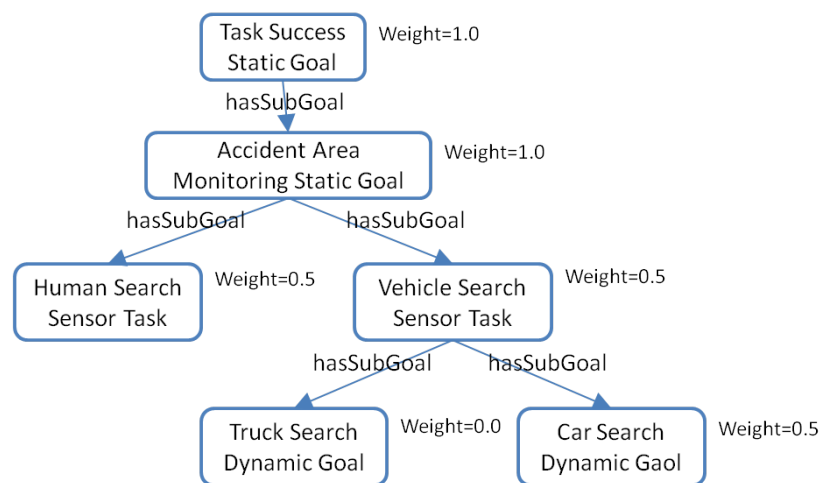


Figure 3.6: An example Goal Lattice including a car detection task

the quality of image needed to allow the detection, identification, and distinguishing of vehicles and buildings of various kinds, in various contexts. The SAM approach defines a knowledge base that allows a user to specify their (NIIRS-based) interpretation tasks as tuples of the form $\langle action, detectables \rangle$, where an action determines what to do on detectables, and a detectable is an event or a thing that might be observed by sensor resources, e.g. $\langle detect, car \rangle$, $\langle identify, aircraft \rangle$.

As described in [62], the SAM approach is to infer a set of more primitive required sensing Capabilities from the Interpretation Tasks, and use these Capabilities in a matchmaking process to identify suitable sensors (where sensors are defined in terms of the Capabilities they provide). Capabilities defined in the SAM knowledge base include kinds of intelligence data (e.g. acoustic, imagery, and radar intelligence, or ACINT, IMINT, RADINT respectively) and also requirements due to the environment such as fog or foliage penetration.

The SAM TR is applicable for user-level tasking as it enables creation of mappings between user-defined task types and sensor types, thus allowing identification of sensors which are able to satisfy a user's tasks. This is done through matching of sensor-provided information capabilities which are sensor-type specific to a task's information requirements. SAM TR's characteristics can be seen in Figure 3.7.

Following the aforementioned definition each Task can require many Capabilities. An Interpretation Task, which is a sub class of a Task, has three sub classes (corresponding to NIIRS actions) describing what it aims to achieve, i.e. what things it wants to: Detect (i.e. to observe presence of an object in a region), Identify (i.e. to classify objects characteristics and intentions), Distinguish (i.e. to differentiate one thing from other, e.g. a vehicle from a person); and one or more Detectables, which represent objects that are discoverable by sensing.

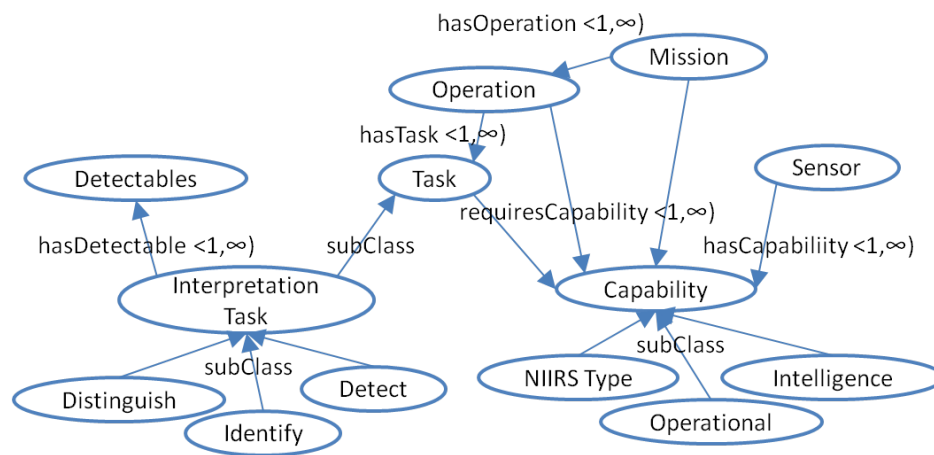


Figure 3.7: Ontology for the Sensor Assignment to Missions TR

Figure 3.8 shows a use of the SAM TR in the car detection example, illustrating the use of the NIIRS classification scale, expressed by the car detection interpretation task with a detectable set to be a car, and a required sensing capability of infra red intelligence (IRINT). This example also presents how a SAM TR can be applied to constrain the result space to sensors of a very specific type in relation to aims of the task, expressed by the detect car interpretation task. Normally the interpretation task would return acoustic and imagery intelligence as two possible sensor capabilities satisfying it, but as the user has also specified the IRINT capability (which is a sub class of IMINT) the solution to the task will involve only sensors providing that capability.

Having concluded our independent analysis of the four task representations, we now turn our attention to identifying possible alignments between them, to allow us to create

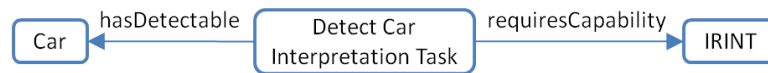


Figure 3.8: An example Sensor Assignment to Missions car detection task

an integrated representation. Therefore, in the next section we look at how the models discussed here relate to the DCPD loop.

3.3 The TRs in Context of the DCPD Loop

In this section we describe the features of the DCPD loop vs. the features provided by the four TRs, i.e. Sensor Web Enablement (SWE), Semantic Streams (SS), Goal Lattice (GL), and Sensor Assignment to Missions (SAM). This part of research analysis is an entry to the creation of a hybrid of task representations described in the next chapter. The goal of this part was to observe how the models relate to each other and how they relate to the steps of the DCPD loop. We want to identify to what extent the existing representations cover these steps. We also want to observe the parts of the steps where the representations overlap. On the basis of these findings we lead our next stage of the research.

We use the below notation to graphically express how each information need of the steps of the DCPD loop relates to a TR capabilities. We use it in the tables below, presented and discussed for each step separately.

- A TR satisfies the information need of a step completely.
- A TR satisfies the information need of a step but only partially.

The task representation of SAM, as presented in Table 3.1, showing TRs in relation to the Direction step, influences the two first points of the step where in the Task Description it only lacks priorities to serve it completely. GL TR only deals with priorities thus its role is only in the initial points of the step. From characteristics of the SWE TR we

DCPD steps	TRs			
Direction(D_1)	SAM	GL	SWE	SS
Task Description	○	○		○
Sensor Type Selection	●	○		
Sensor Instance Selection			●	○
Sensor Setting			●	
Monitoring			○	○

Table 3.1: Four TRs versus the Direction step of the DCPD loop.

DCPD steps	TRs			
Collection(C)	SAM	GL	SWE	SS
Data Collection			●	●
Data Forwarding			○	●

Table 3.2: Four TRs versus the Collection step of the DCPD loop.

know that it deals mostly with low level operations i.e. those relating to sensors thus we observe its influence in appropriate points, i.e. setting of sensors and collection of their data [9]. The SS TR doesn't operate on sensor types (as discussed in Section 3.2.2 and presented in the paper [41]), since the sensors are seen only as the source of a stream. Therefore, in the Sensor Type Selection point we have no mark. The task in SS is expressed using low level technical language, which an analyst creating ('programming') a task must know. The SS TR in contrary to SWE TR doesn't support sensor discovery, it only deals with what it has given in a database.⁴ We also observe that none of the TRs handle completely issues of Monitoring of environment evolution, i.e. monitoring of sensors and tasks for their state change.

In the Collection step, shown in Table 3.2, we see that SAM & GL task representations

⁴Though we do imagine it is possible to handle sensor discovery in a similar fashion as in SWE, since both SWE & SS are service oriented architectures, the authors of SS don't state their approach to this problem.

DCPD steps	TRs			
Processing(P)	SAM	GL	SWE	SS
Data Processing			●	●
Information Forwarding			○	●

Table 3.3: Four TRs versus the Processing step of the DCPD loop.

are not helpful in this step, because of their high level related characteristics. Thus SWE & SS TRs are covering the step completely, with the exception that the SWE offers only a partial coverage of the Data Forwarding part of the step. The difference between SS & SWE TRs, in this step, is a result of not relating the processing services to sensors or more precisely the information one produces, by the SWE. It is done in the case of SS TR, where Semantic Services are forming together a chain of data processing services by combining appropriate input and output Semantic Streams [41, 76]. Thus, building a full solution, which in their case is derived from top to bottom, i.e. user expresses a task in the form of a final stream which is then decomposed to a set of streams and services producing it, as mentioned in [41].

We observe a similar relation between the points of the Processing step and the TRs (see Table 3.3) as it was for the Collection step. The SAM & GL TRs have nothing to offer for this step. Where SWE & SS TRs have complete coverage of the step, with the exception that the SWE lacks the full coverage of the second part of the step, i.e. Information Forwarding. Similarly, as we mentioned in the previous step, we claim that using SWE or SS TR and its technologies you can achieve the same results. Yet SWE TR doesn't show such close connection between the task and the processing services as it is observable in case of the SS TR. In its case the task is effectively described as the final result (the final Semantic Stream) which is returned by a more complex chain of Semantic Services.

Since the Dissemination step, visible in Table 3.4, deals mostly with the way of delivery, and presentation, of results to the user(s) thus SWE TR, since it is very low level,

DCPD steps	TRs			
Dissemination(D_2)	SAM	GL	SWE	SS
Recipients Selection				○
Delivery Medium Selection				○
Presentation Format Selection				○

Table 3.4: Four TRs versus the Dissemination step of the DCPD loop.

has nothing to offer here. Also SAM & GL TRs don't address issues involved in this part of the loop. Only SS TR has partial influence at this step. Covering, implicitly, parts of the points involved in this step. For example, the default protocol, the default recipient of a task solution and forms of delivery directly expressed by the user, during task creation.

To summarise the chapter we will discuss now the strengths and weaknesses of the TRs referring again to the example of car detection task. For example, we see that SWE TR covers the low level details concerning the sensor tasking i.e. it, as opposed to others, allows for control over sensor parameters. Additionally, it offers a strong support for the steps of Collection and Processing of sensor data. On the other hand visible is lack of direct relation between a task and sensors, thus from the SWE TR ontology's concepts we cannot establish which sensors can fit to which task. This weakness is not present in SAM TR, where task is connected to sensors by information that can be obtained by processing of sensor generated data and which the task requires. This is in fact where the strength of this TR ends as it is very specific. Similarly is with GL TR, where its only strength is capability to organise the sub-tasks of a complex task, breaking the complex task down to sub-tasks which are satisfiable by sensors-originated information and prioritise their satisfaction according to their importance to the main task. On the other hand though SS TR lacks this capability or the capability to map sensor types to task types (provided by the SAM TR), it offers coverage of almost all steps of the DCPD loop though not always complete. It offers more detailed

task description including the form of result representation and complete support of Collection and Processing steps.

Our next step is to attempt the mixing of the models, since, as we observe, none of the existing TRs delivers all information needs to the steps of the loop. On the other hand, when looking at the TRs together we can notice that the models are actually fitting well together, to some extent, they are filling each other weaknesses. The intended result of the mixing being the creation of a “better” TR, one that combines their approaches and covers the whole DCPD loop.

Hybrid Task Representation

This part of the research resulted from the observed overlappings between the TRs and the DCPD loop, which we mention in Chapter 3. We present here the approach, in which we created hybrids of existing task representations, called hybrid task representations or HTRs. In our case a hybrid model represents a combination of existing TRs such that overlapping of concepts between the included TRs is minimal, i.e. gaining a functionality from one representation without introducing redundancy from other TRs.

We show here the models of hybrid task representations we created, each with its specific characteristics (the ontologies are available in Appendix B or¹). They are discussed in context of the steps of the DCPD loop. This was done so we could observe which of the steps, and to what extent, can be satisfied with each model. Towards the end of the chapter we present the creation of the fourth model of hybrid task representation, that was built on top of the first model (Max SAM HTR). We explain its details and reasons for its creation.

On the ontology graphs throughout the chapter (i.e. the figures from 4.1 to 4.6), we use the following notation: classes are depicted as ovals, the OWL *sameAs* property (represented by a solid bidirectional arc) to align common concepts (using namespace notation to indicate which TR ontology the concepts are from), subclass relations and all other properties are shown as labelled solid arcs.

¹Available from: <http://users.cs.cf.ac.uk/K.Borowiecki/Thesis/Ontologies.zip>

4.1 Methodology of Hybrid Task Representation Creation

The HTRs were created combining the four task representations (TRs), described in Chapter 3, i.e. Sensor Web Enablement (SWE), Semantic Streams (SS), Goal Lattice (GL) and Sensor Assignment to Missions (SAM).

The approach leading to the creation of the three HTRs can be described by these three general steps:

1. Initially, we analyse the concepts in the ontologies of the TRs, looking for common concepts and relations between them.
2. Following the findings from (1), we create a general model of a hybrid of task representations (called GHTR for short). It is built to capture all of the concepts contained in the used TRs. The graph depicting its ontology is shown in Figure 4.2 in Section 4.1.2.
3. Finally using the relations between the DCPD loop and the TRs (as presented in the tables 3.1, 3.2, 3.3 & 3.4 of Chapter 3), we create mappings between the concepts of the used models and that of the GHTR. Doing this we try to maximise coverage of the steps of the loop while also avoiding unnecessary redundancy. For example, if we have complete coverage of the Processing step served by one TR we will not merge concepts of other TRs related to the step.

The third step could be represented, in a more detailed way, by the following algorithm 4.1:

1. Start the creation of a new HTR by importing the GHTR into its ontology. It will serve as the base of the created HTR on top of which TRs concepts are going to be mapped.
2. Select the focus of this model, i.e. set initial concepts and conditions the model will respect. The set of these options is obtained by analyses of overlappings between the TRs, which we did in Chapter 3, where we put all TRs in relation to the steps of the DCPD loop. For example, in our case we have, one technology doesn't use sensor types where others do; and we have two techs offering coverage of the processing step.
3. Integrate concepts of the TRs applying the following pseudo code:

We assume the variable `trs` stores the list of TRs, and the variable `htr` stores the HTR model we are building.

{To enter the while loop at start we set it to true.}

`boolean isConceptAdded = true`

while `isConceptAdded == true` **do**

`isConceptAdded = false`

for `tr IN trs` **do**

`List trConcepts = tr.getTRConcepts()`

for `trConcept IN trConcepts` **do**

{Make sure the concept is only added if its inclusion will result in a more complete coverage of the DCPD or if it is required by the concepts in htr.}

if `trConcept.coversPartOfTheLoopNotCoveredInModel(htr)` **or**

`trConcept.isNeededByConceptsInModel(htr)` **then**

`htr.addConcept(trConcept)`

`isConceptAdded = true`

Algorithm 4.1: Algorithm of HTR creation

To visualise the relations between the concepts of these TRs we created Figure 4.1. It shows the concepts from TR ontologies and their relations mapped together with

general concepts common among the TRs, important from the perspective of the DCPD loop, such as Processing Services (i.e. hardware and/or software units responsible for data manipulation), Task, Sensor, Information or User.

To obtain the base ontology on top of which we will map the TR concepts, we extrapolate the common concepts into a general hybrid model, called GHTR (presented in Figure 4.2). Having constructed our base ontology we proceed with the step of HTR model creation.

Creation of these three HTRs from the four TRs was influenced by their specifics and characteristics. From Section 3.3, presenting TRs in relation to the DCPD steps, we know that some of the TRs offer similar coverage of the same steps. Particularly important for us is overlapping of similar concepts in case of e.g. SS and SWE. Even though implementation differs between these two, both of them offer similar functionality when it comes to processing of sensor data for satisfaction of a users's tasks. The other important factor was the importance of a particular concept within ontology of a TR, as it is, for example, with the role of sensor type in SAM and SS TRs. In this example they differ completely, where as for SAM sensor type is the key concept essential for the consistency of its ontology, for the SS it has no meaning (here we observe that a similar role plays the type of Stream that a Sensor produces). Wanting to avoid concepts that are not essential for the newly created hybrid TR we chose to limit redundancy of similar concepts to only those necessary for a model's completeness (in terms of the DCPD loop's coverage).

Applying the above rules we obtain three models: Max SAM (Section 4.2.1), where sensor type plays the key role; Max SWE (Section 4.2.2), where SWE technology is utilised in sensor data processing step; and Max SS (Section 4.2.3), where SS is used for processing and since type of sensor is of no importance in this case SAM TR is not present.

4.1.1 Mapping Between the TRs

Our primary motivation for creating the mapping between the TRs was to enable integration of the underlying tools and mechanisms that support each individual TR. As the four approaches, to a large extent, address different parts of the tasking problem, their simultaneous use would thus offer a more complete solution to the problem. Our proposed mapping of concepts between the four ontologies from Section 3.2 is shown in Figure 4.1. It also pictures their relation to general (common) concepts, which are further discussed in Section 4.1.2.

The HTR is built around two concepts: Sensor and Task. We observe that a SAM:Sensor models the same real-world thing as both a SS:Sensor and a SWE:Sensor, despite the Semantic Streams and SWE TRs dealing with different aspects of the sensor-level tasking problem. Therefore, the Sensor concept in our HTR is intended to be the union of the Sensor concepts in SAM, SS, and SWE ontologies.

Three of the four ontologies above have an explicit notion of Task: GL, SS, and SAM. In SAM terminology, a Task is something that can be satisfied by information obtained from a Sensor's data. This equates to the GL:Dynamic Goal concept, as both are directly satisfiable by means of sensors. Relationships exist between subclasses of SAM:Interpretation Task and subclasses of GL:Sensor Task because of the relation in GL between a sensor task and the dynamic goals contributing to it. The three subclasses of a GL:Sensor Task will in fact be determining the types of the dynamic goals, as these are only instantiated for sensor tasks. Therefore, the three subclasses will correspond to subclasses of an interpretation task. In principle it would then be possible to use the SAM knowledge base to infer Capabilities for a GL:Dynamic Goal.

The GL's Static Goal represents the same concept as an Operation in the SAM TR. Some Static Goals have very wide meaning, e.g. control a region; some (Sensor Tasks) can be broken down into satisfiable by information obtained from sensors' data Dynamic Goals (Tasks in case of SAM TR), e.g. detecting people, broken into rescue

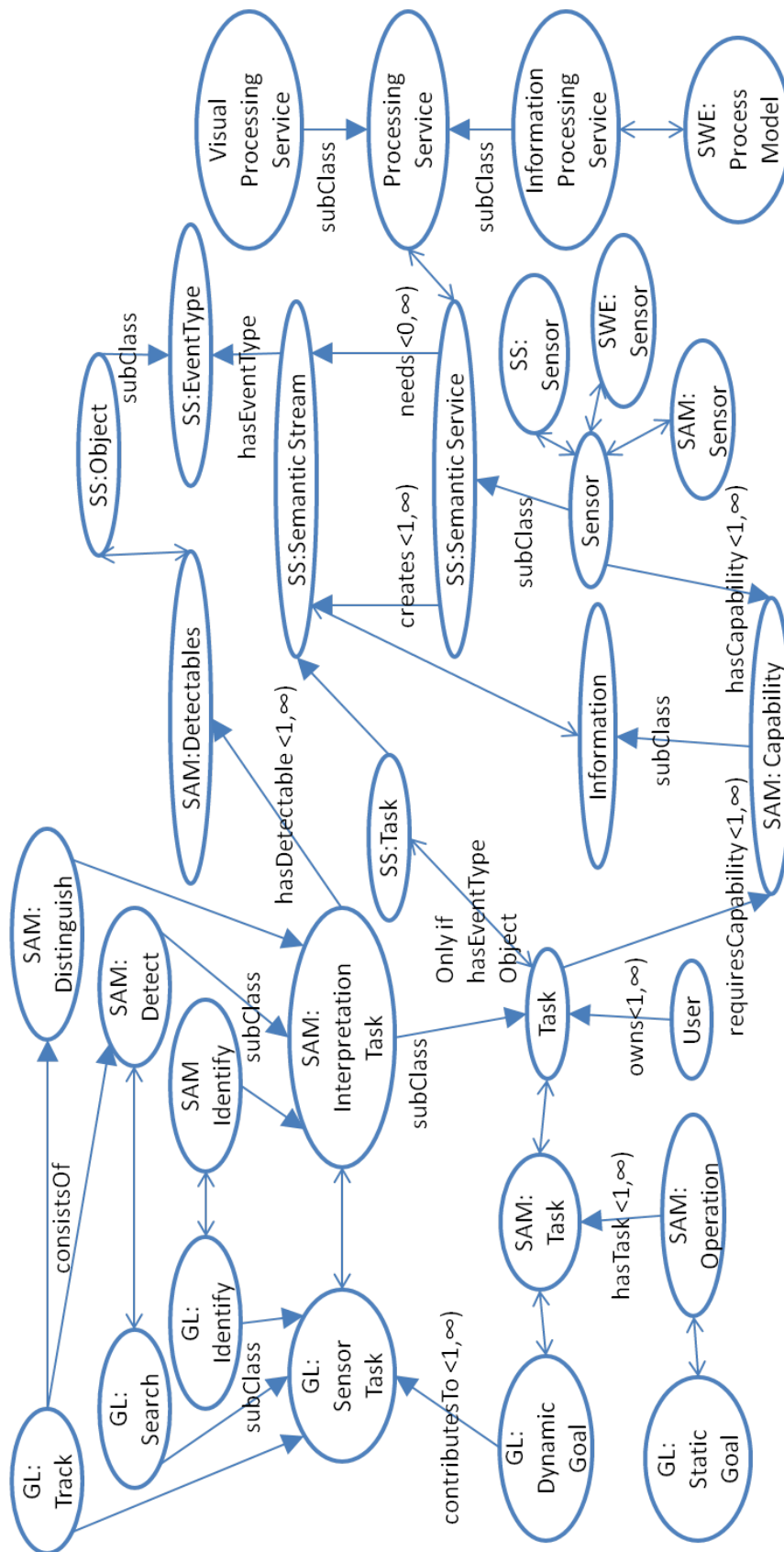


Figure 4.1: Ontological representation of relations between TRs; highlighting general concepts, from GHTR ontology.

personnel and victim detection. For this reason we can say that a SAM:Operation is the same as a GL:Static Goal. Though SAM TR does not have a notion expressing relationship between operations, in the HTR it will be provided by the *hasSubGoal* property, with important notice that this property, when used between Sensor Task and its Dynamic Goals will correspond to Operation's *hasTask* relation, because a Task is a single sensor means satisfiable concept of the SAM TR and the Dynamic Goal has exactly the same role in the GL TR. Respecting these relations a goal lattice could be used to represent a single SAM:Mission.

A SS:Task is a kind of SS:Semantic Stream, and represents only one direct need of a user (e.g. images of vehicles traveling faster than 15 mph [41]). Therefore, a semantic stream's role is more like that of a SAM:Interpretation Task but with the important difference that SAM:Detectables do not have as wide a meaning as an SS:Event Type (we consider SS:Object to be the same thing as SAM:Detectables, because both are describing sensor-measurable things; for this reason its relation to SS:Event Type is exactly the same as that of Object to Event Type, i.e. it is its subclass). For these reasons, a SS:Task can be perceived as the same as a SAM:Task only if its event type is within the space of the Object class. If this is the case, a SAM:Task could be used as the final output stream or SS:Task, for which services are being composed.

4.1.2 General HTR

Wanting to have a base that we may use for creation of the HTR models we extrapolated general concepts from the model mapping the four TRs together. As a result we obtained a model which we call General Hybrid Task Representation, or GHTR for short. Thus, when creating the hybrid models we only need to express mappings between the used concepts of TRs and that of the GHTR. This way, as long as we remember to take all the essential concepts from the integrated TRs, we are sure that they will create a concise ontology.

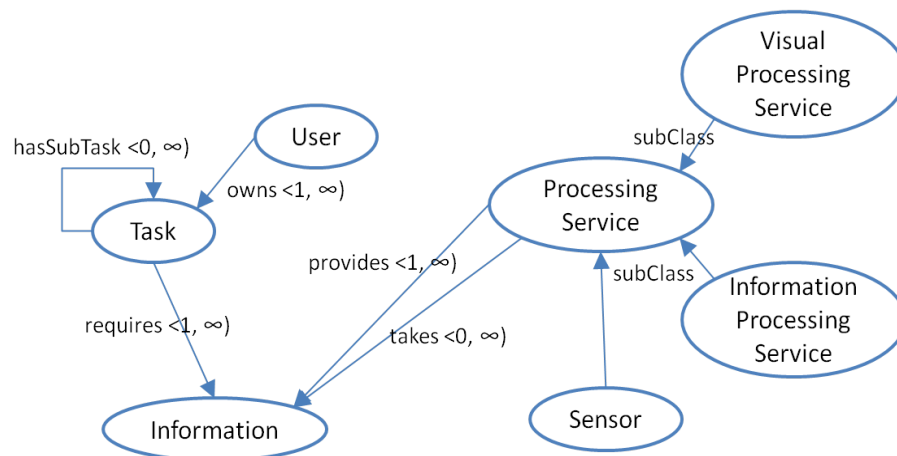


Figure 4.2: Model of the general HTR.

In Figure 4.2 we observe five concepts, User, Sensor, Task, Information, and Processing Service. If the User, Sensor, Task and Information were detailed previously we have only mentioned briefly what a Processing Service is. Therefore, here we only describe what it is and how it relates to the other concepts.

Processing Service is a piece of hardware and/or software which can combine and/or transform information in result creating another. We distinguish two subclasses of these services, i.e. Visual and Information Processing Services.

Visual Processing Service is a more specialised mechanism of processing which is responsible for presentation of a certain information to a user in an informative form, e.g. in case of car detection task, turning of plain numbers representing latitude and longitude of the detected car's location to a point on a map. In other words it deals with formatting of information already satisfying the user's task not yet represented in a user readable form.

Information Processing Service is a specific type of processing mechanism that is responsible for processing of information the final result of which is not user readable. This is often the simple data that is produced by a Sensor, which lacks context in relation to a user's task, thus, by itself, it is meaningless. One could argue that, after further analysis, a user may find a solution to his/her task, but the information at this

stage does not completely satisfy the task. For example, a simple signal from a break beam sensor, or it could be even a complex information such as a camera image but is meaningless for the user and without further analysis by the user it cannot answer his/her task's needs.

Even though pure Sensor data is often not enough to solve a Task, we can imagine Tasks for which the image or sound could be what a user actually wants. For example, even in a complex task, where the user wants images of cars breaking a speed limit, the image would be the final solution though many operations would be executed on the information, in the processing stage, in the background. Therefore, we see a Sensor as a Processing Service and, depending on a user's needs, it could play part of Information or Visual Service, for this reason these two concepts cannot be disjoint, and often might overlap. The Sensor here, similar to the Sensor concept defined in Semantic Streams, is a particular Processing Service which *needs* no input Information (SS:Semantic Stream), since it turns a real physical event into a computer processable one. This representation of Sensor agrees with SAM understanding of Sensor concept and its relations to Task and Information (SAM:Capability), i.e. Task *requires* some Information and Sensor *provides* it (or in SAM terms *hasCapability*). Though in our understanding Information is a broader concept representing any type of information not only that which originates from a Sensor i.e. it represents also that being the result of processing of various sensor information types (SAM:Capability). This in turn matches the representation captured in the form of the DCPD loop, i.e. the sensor information passes through the processing stage before it satisfy a user's needs.

It is important to mention here that we decided not to include such concepts as Operation or Mission and its relations when representing a Task in the GHTR ontology. We substituted them by broadening the concept of Task and placing the *hasTask* relation on it. This follows similar thinking to that of the authors of the SAM reasoner. In their paper [64] you can read: "... Because the definitions of mission, operation and task are somewhat subjective in practice, we adopt a simple model of hierarchical trees of

tasks, where a Task can be broken down into sub-tasks (which are also Tasks) ...”. This agrees with our point of view since we are focusing on the means of satisfying a task rather than on how a task breaks down, and how it relates to other tasks in a structural understanding i.e. which task is sub task of which. We care how tasks relate to each other but in a priority sense i.e. which task we care about and how much. This sort of approach allows then, in the assignment stage, to focus resources on satisfaction of the most important task first. In our models this sort of relation is expressed by use of the GL TR. Therefore, such concepts as Operation or Mission have no real meaning from our perspective and are thus substituted by a more general concept of Task.

4.2 The Three HTRs

In this section we present three models of hybrid task representations, which are the result of applying the HTR creation methodology. The first HTR takes full use of SAM TR, the second utilises SWE TR for processing, and the third HTR uses SS’s entire capability.

4.2.1 Max SAM HTR

This HTR is modeled to make full use of SAM TR’s capability to control a system behaviour dependent on types of sensors, hence its name Max SAM HTR. This model enables reduction of search space, e.g. for sensor discovery algorithms, assignment algorithms etc., by limiting it only to sensor types having capabilities applicable to specific task types selected by a user.

Following the aforementioned method of HTR creation, in this case we chose to use sensor types and, for processing, we used Semantic Streams since their understanding of a task is similar to that of SAM.

Creating the HTR we thought about maximising use of the benefits that the SAM reasoner technology has to offer, e.g. it enables finding of appropriate sensor types that are capable of satisfying a particular task type or types. It brings, as we think, improvement of resistance (i.e. increase of adaptability) of this model to changes within the network it operates. The adaptability is improved, especially, to the changes within a sensor's and platform's states or availability. These resources, in dynamic environments, are often susceptible to changes that can render a task, using an unprepared model, useless in such case.

The approach has certain benefits (\oplus) & limitations (\ominus):

\oplus Increase of adaptability of a task applying the model, to sensor level changes, through mapping of task types to sensor types. Thanks to this we already know alternative solutions and are ready to use them once the current one becomes unavailable.

\oplus Close connection between SAM and SS perception of task, i.e. seen as the final result expressed as a set of requirements requested by a user. Also SS allows a user to express the form in which he/she wants the result to be delivered, which is an important part of the dissemination step.

\oplus If we assume that we can treat Processing Model of SWE exactly as Semantic Service of SS (assuming there exists some mapping between these two), the model would be able to use them interchangeably or mix them (the second we do in the model of Max SWE HTR described in Section 4.2.2).

\ominus The processing does not follow the emerging standards of SWE, though it could be fixed by additional merging of SWE concepts into the HTR. We chose not to do so and suffer this deficiency, since our approach was to avoid concepts redundancy that is unnecessary from a model's completeness point of view.

Figure 4.3 presents mappings between concepts of the used TRs and that of the general HTR which together form the Max SAM HTR. It shows that SAM:Task and SS:Task are the same thing, in this ontology. The Information concept is the same thing as

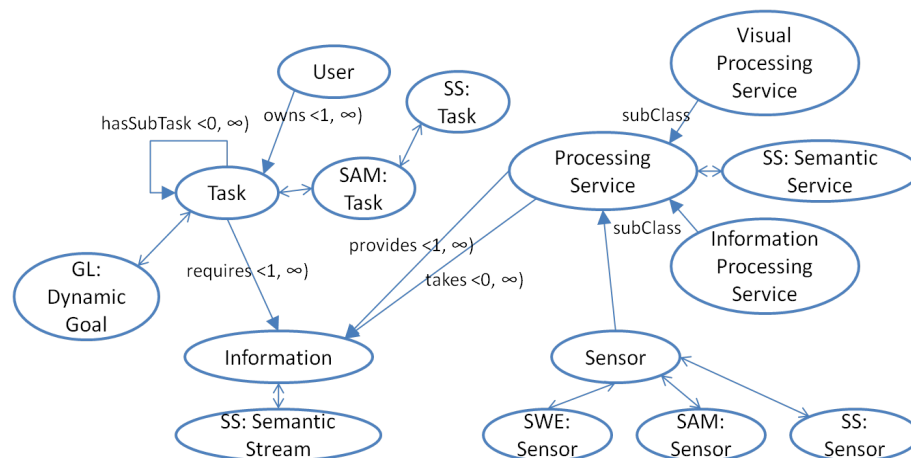


Figure 4.3: Model of the Max SAM HTR.

SS:Semantic Stream and the Sensor is a combination of all three Sensor concepts from SWE, SAM and SS task representations. This is because we need to assure a sensor's usability by the Processing Service whose role plays SS:Semantic Service; we want mapping between sensor types and task types delivered by SAM; and we need the ability to control a sensor provided by SWE.

4.2.2 Max SWE HTR

The Max SWE HTR focuses on the usage of SWE standards, thus it maximises its presence within our model, reducing the influence of, for example, Semantic Streams in the processing phase (SS is used to capture the final processing step, i.e. visualisation). The aim here is to make creation of implementations, which closely follow the standards developed by the OGC community (authors of SWE), possible.

In this model our approach was to use SAM thus we make use of sensor types and task type mapping and use SWE as the other technology for processing, thus we observe increased presence of SWE concepts in this model's ontology.

The model enables wider operability between various sensors, platforms & processing models through use of the standards in their description. This approach has potential to

enable easier usage and mixing of various components (sensors & processing services) built by different vendors. This way a larger sensor network built from various sensors would be supported by the model. Therefore, we observe the increased presence of concepts native to the SWE task representation.

The approach has certain benefits (\oplus) & limitations (\ominus):

\oplus Increase of availability of components delivering and operating on information assuming they are built reusing the SWE standards. This indirectly increases accessibility of alternative solutions, which could serve a task once the current solution becomes no longer valid.

\oplus Close connection between the Sensors and Processing Services, both served by SWE, enables increased interoperability between these two.

\ominus The dependency on a mapping between SS:Semantic Services serving the final processing of information for delivery to a user and SWE:Process Models processing the initial data generated by sensors. This can be fixed either by the creation of mapping between the form of output and Processing Model and further treatment of processing, as done by SWE (as a result substituting use of Semantic Streams by SWE and treatment of Processing Models as Processing Services); or the creation of mapping between the Semantic Services and Processing Models allowing SS to see a Process Model as part of their system (as another Semantic Stream). However, we are not aware of any work researching these particular issues.

The main parts of the model are similar to those of Max SAM HTR described in 4.2.1. The role of the Task is served by SS:Task and SAM:Task which requires Information which Sensor is capable of delivery. Since both SS and SAM perceive Task in a similar fashion, as described above, the SS:Semantic Service performs the role of the Visual Processing Service. Therefore, the SWE:Process Model is used as an Information Processing Service only. We already argued that there is no strong differentiation between these two classes of Processing Services and it strongly depends on Task type, as such

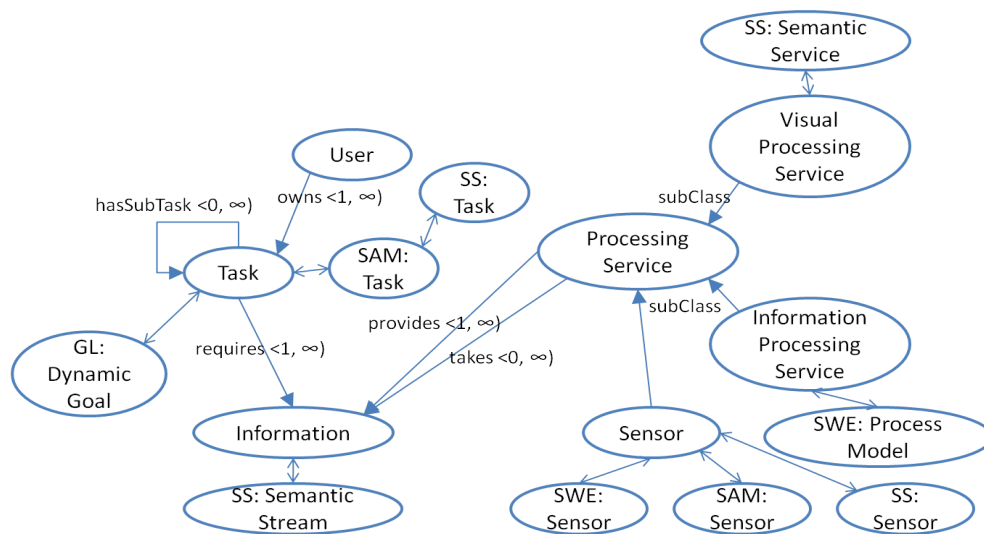


Figure 4.4: Model of the Max SWE HTR.

it might be that formatting of the result data will not involve SS:Semantic Service and effectively the whole processing stage will be served completely by a chain consisting of SWE:Process Models.

4.2.3 Max SS HTR

This hybrid is designed to make full use of SS technology, as it offers a nearly complete solution, with few minor exceptions which we are fulfilling using other TRs. We are following their approach of not relying on sensor types, thus the use of SAM TR is not beneficial here. This ignorance of sensor types is the major differentiator of Max SAM and Max SS HTR models, otherwise they are pretty much the same.

In this model we decided not to make use of sensor types, thus removing presence of SAM TR in the model, and making extensive use of the SS TR. We chose to do so since the model maximises the use of the SS which doesn't work on sensor types.

Creation of the Max SS HTR was meant to allow use of the whole capabilities of the SS technology, by not introducing elements or concepts not essential for operation of the model, even if they would offer potential benefits as, for example, that offered by the

SAM technology and working with sensor types from the start. In the SS technology there is no need for matching of sensor types and task types since the SS operates purely on information. What it means is that ‘everything’ in SS is seen as either a source or consumer or both of information which consists of a stream of events called Semantic Stream [41]. Therefore, in SS sensors are seen as data sources and are differentiated by the type of events carried by the streams they produce.

The approach has certain benefits (\oplus) & limitations (\ominus):

\oplus Use of the SS technology with a minimal interference into its mechanics. The model effectively represents an enhanced version of SS TR, enriched with Sensor concept from the SWE TR and priority concept from the GL TR.

\oplus Native to SS technology operation on sensors and processing of sensors’ data.

\ominus It makes use of the emerging SWE standards but only in a sensor’s description. Thus, in processing stage, it could only use Semantic Services (as Processing Services) which are made available & supported by a system’s authors. This is fixable, as it was mentioned in the benefits & limitations paragraph of the Max SAM HTR in Section 4.2.1, by mixing of SWE TR into the model, but since our goal was to limit redundancy we didn’t do it.

\ominus Not operating on sensor type level. Thus, missing the possibility to initially narrow the search space and capture the knowledge of alternative solutions. Still, finding an alternative solution is feasible, for example, via re-tasking of the task (repeating the sensor task assignment algorithm or creating the task once more). This way the SS assignment algorithm has to perform an exhaustive search of the currently available sensor resources to discover new sensors, as described in [41]. This could be fixed by mixing the SAM TR into the model. This would effectively turn it into the Max SAM HTR.

The important difference between the other two mentioned HTR models and this model is observable dominance of the SS TR concepts in the model. The Task is same as the

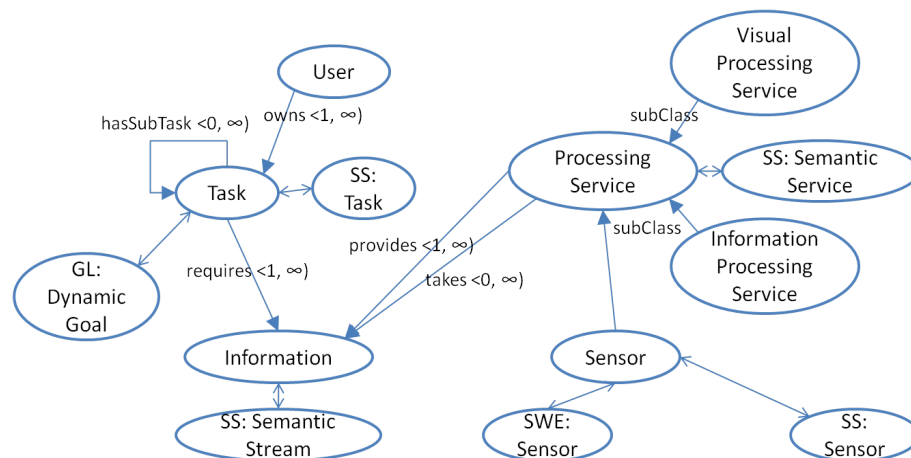


Figure 4.5: Model of the Max SS HTR.

SS:Task and the Sensor is the sum of concepts of SS:Sensor and SWE:Sensor since we want to process the sensor data using SS:Semantic Services while being able to control a sensor. Also as we mentioned the SAM TR concepts do not appear in this model.

4.3 The Three HTR Models in Relation to the DCPD Loop

Our intention in this part of our research is to observe how the concepts, captured by combining the TRs, relate to those needed by the steps of the DCPD loop. Therefore, we have performed analysis of ontologies in respect to the loop, and the result was the creation of tables presenting each HTR in relation to a certain step of the loop. We start with description of the sort of information which might be required in a particular step of the loop, giving examples, where needed, for better clarity.

To graphically express how each information need of steps of the DCPD loop relates to an HTR capabilities we use the below notation. We use it in the tables below, presented and discussed for each step separately.

X An information need does not apply to a TR.

DCPD steps	HTRs		
	Max SAM	Max SWE	Max SS
Direction (D_1)			
Task Description	●	●	●
Sensor Type Selection	●	●	X
Sensor Instance Selection	●	●	●
Sensor Setting	●	●	●
Monitoring	○	○	○

Table 4.1: Three HTRs versus the Direction step of the DCPD loop.

- A TR satisfies the information need of a step completely.
- A TR satisfies the information need of a step but only partially.

4.3.1 Direction

This step of the loop is responsible for capturing of initial details of a task and further monitoring of its satisfaction.

The models satisfy four out of five information needs of this step. The Sensor Type Selection step does not apply to Max SS HTR since the approach this model uses follows that which is presented in Semantic Streams where sensor types are not useful and they work directly with the data a sensor produces.

The Monitoring need is equally not treated extensively in each of the models. Meaning that they are not capable to offer more beyond the default i.e. resubmission of a task. Examples of other options could be: expression of condition for a reassignment and/or offering of an alternative solution. For this reason a task created for those HTR models can only reach a total satisfaction or nothing, while using a different sensor type, or satisfying the task partially, might be feasible and still helpful, for example, detecting a vehicle when there are no sensors available to identify it. Yet a user cannot express

DCPD steps	HTRs		
Collection (C)	Max SAM	Max SWE	Max SS
Data Collection	●	●	●
Data Forwarding	●	●	●

Table 4.2: Three HTRs versus the Collection step of the DCPD loop.

it in the model. This lack doesn't allow for a lot of flexibility in treatment of changes influencing a task satisfaction.

4.3.2 Collection

The step is responsible for gathering of sensor data and forwarding it to appropriate processing services.

Each model satisfies this step wholly either by the use of only SS, as it is in the case of the Max SS or Max SAM, or by mix of SS and SWE technologies, as in the Max SWE HTR. This is supported by the captured knowledge in the concepts of the ontology, i.e. they know what the input and output of Process Models (or Semantic Services) is. Therefore, a system can find appropriate Sensors delivering required Information (or Semantic Stream) type. Since they also describe outputs, a system can build a structure combining many Process Models (or Semantic Services) into a single processing unit which satisfies a Task. In such a fashion all three HTRs support this step by knowing what information to collect and where to deliver it.

4.3.3 Processing

This step's role is to work with received data and production of information from it. Thus, in result receiving a solution satisfying the task for which the data was originally collected.

DCPD steps	HTRs		
	Max SAM	Max SWE	Max SS
Processing (P)			
Data Processing	●	●	●
Information Forwarding	●	○	●

Table 4.3: Three HTRs versus the Processing step of the DCPD loop.

This step is completely satisfied by all the models but not by the Max SWE HTR. This is the result of the lack of a relation between the Process Model and Task in the SWE TR ontology. It is such because this TR is focussed on sensor-level tasking and it doesn't have the Task concept. This is why we do not know the last step before forwarding the processed information to the Dissemination step. For this reason the Max SWE HTR model mixes Semantic Services of SS TR with the Process Models of SWE. This results in a slight deficiency in the final stage of processing where the information is passed from the Information Processing Service (the role of which, in this model, is played by SWE:Process Model) to Visual Processing Service (in this model SS:Semantic Service). The reason is the lack of mapping between these two technologies. Therefore, this model would require additional research and implementation effort to apply.

Since we do not argue how difficult this mapping in reality might be, as it is beyond the scope of this research, we have created our own ontologies for both SWE and SS TRs which allowed for such mapping, for example, when treating SWE:Process Model similar to the SS:Semantic Service, i.e. as producer and consumer of SS:Semantic Streams. Yet knowing that there are differences in how these two are being applied, we expect that this mapping might in fact be more difficult. We are also not aware of any research being done on the topic of mapping these two. Therefore, we want to highlight it by marking this as a deficiency of this model.

DCPD steps	HTRs		
Dissemination (D_2)	Max SAM	Max SWE	Max SS
Recipients Selection	○	○	○
Delivery Medium Selection	○	○	○
Presentation Format Selection	○	○	○

Table 4.4: Three HTRs versus the Dissemination step of the DCPD loop.

4.3.4 Dissemination

This step's responsibility is to make sure that information is delivered and presented appropriately to all task beneficiaries.

All models satisfy this step to the same degree i.e. they offer only a minimal coverage of the information needs of this step. Meaning they do not offer more options related to these information needs. They handle them all but only partially. In the Recipients Selection it is assumed that there is only a single user, the author of a task, and the solution is only directed at him, similar treatment is seen for the other steps, i.e. they are minimal and focused on the task's author. The Delivery Medium Selection, by default, uses the same communication method as that used by the user's system to 'talk' to a sensor network. Generally speaking, this step is controlled completely by the format of presentation the user requested when creating a task. For the Presentation Format Selection the author decides how to deliver the result while creating his/her task. Here is missing, for example, a relation between recipients device and the presentation format, e.g. some formats may not be open on mobile devices, even on the modern ones, like Adobe Flash on iOS devices; or will be difficult to use on them, like a large document or image requiring detailed analysis. In such situations a much better solution would involve tailoring of a task solution for a particular consumer.

This step is, therefore, only partially satisfied by all of the three HTR models. We can imagine that in a real situation there would be in fact more consumers of a task solution

DCPD steps	HTRs		
	Max SAM	Max SWE	Max SS
Task Description	●	●	●
Sensor Type Selection	●	●	X
Sensor Instance Selection	●	●	●
Sensor Setting	●	●	●
Monitoring	○	○	○
Data Collection	●	●	●
Data Forwarding	●	●	●
Data Processing	●	●	●
Information Forwarding	●	○	●
Recipients Selection	○	○	○
Delivery Medium Selection	○	○	○
Presentation Format Selection	○	○	○

Table 4.5: Three HTRs versus information needs of steps of the DCPD loop.

and they will likely have other means to consume it or other format preferences of receiving it (for example, in a collaboration environment as presented in [63, 83]).

4.3.5 HTRs Analysis Conclusions

As shown in the above tables we managed, using three different approaches, to maximise the coverage of the steps of the DCPD loop. We achieved very close capability in the models, which is not a surprise, as they were constructed from the same components. Yet still we can observe some differences between the models, for better visualisation of the models versus the whole DCPD loop we introduced the summary in Table 4.5. It puts all previously presented data in a single place.

Taking a look at Table 4.5 we observe, among the majority of similarities, some small

differences between the models and coverage of the DCPD steps. These are the lack of sensor type step in the Max SS HTR, which is the result of its specific approach; and the lack of mapping between Process Models of SWE and Semantic Services of SS in the Max SWE HTR. These are a direct result of the methodology used for the models creation described above.

Noticeably, there is a lack of complete support for all the steps of the loop in each of the HTR models, but still they offer more complete coverage than the TRs separately. This leads us to believe that, by enriching one of the models, since they have very similar capabilities, would allow us to create a HTR that captures all information needs of the DCPD loop. In effect we want to obtain a hybrid that best fits a dynamic environment. At this stage the plan was to first define what the missing concepts are and then choose the model which we will enrich them with.

4.4 The Missing Concepts

We have created the following concepts, which we found missing from the general HTR, which was a result of generalisation of the created combination of the existing task representations. The missing concepts were created with the intention to enable capturing of all information needs enclosed within the steps of the DCPD loop.

In Table 4.5 we observe that there is a lack of coverage in Direction, Processing and Dissemination steps of the loop. We have already discussed the treatment of the deficiency in the Processing step of the Max SWE HTR in Section 4.3.3. Yet when it comes to the other deficiencies we believe that they can only be overcome by introduction of additional concepts. Therefore, we are doing so here.

The concepts of Beneficiary, Device and Presentation Form that a device can support are meant to fill the gap in the Dissemination step. Where the other six are meant to allow for a very high flexibility of a task description. Thus, filling the missing support for Monitoring in the Direction step, by enabling a user to express what changes to a

task solution he/she is ready to endure. Below is a detailed description of what each of the missing concepts actually represents.

- **Beneficiary** is any user that benefits from the results of satisfying a task. For example, a task's author might be an analyst controlling sensor tasking in rescue operation headquarters and the 'real' task information consumers will be people in the field undertaking the rescue effort.
- **Device** is the concept which captures details of a hardware tool used by a beneficiary to receive the information satisfying a task. Depending on the device a beneficiary uses he/she can benefit from certain Presentation Forms. For example, analyst in headquarters would have a global view of the rescue operation where a rescue team member would need a location in relation to his position. For this to be possible his/her device would require to have appropriate hardware & software installed, e.g. Global Positioning System (or GPS). Another example, which shows that to benefit from a certain solution a user's device must have certain capabilities, could involve a device with a low resolution display where adequate presentation of a task's solution would involve high resolution images.
- **Presentation Form** means the way in which the result of a task is visualised for a beneficiary. For example, we can present a detected object on a map or just display an image of the object. It strongly depends on the details of a task the user has described and what he/she is exactly after.
- **Monitoring Rule** is a general concept representing all the rules which apply to a task and they describe what eventual changes the user can accept. Its specific subclasses can be assigned during task creation to express what a user wants and what a user can bare when no perfect solution is present. It increases the number of alternative solutions for a particular task by expressing acceptable levels of variables of the assignment process. For example, for police officers chasing a suspicious car, it is perfect to know the exact location of the suspect, but it is

sufficiently helpful if they know where the suspect was last seen. This way they know where to focus their efforts, thus being more effective at their task.

- Reassignment is allocation of exactly the same type of sensors thus obtaining the exact same solution just delivered by different instances of the sensing resources. For example, in a situation where there are no alternatives available or other parts of our process depend on the exact solution we cannot be satisfied with any potential alternative solution, e.g. in a situation that a CCTV camera becomes unavailable, destroyed or something will cover its view the task will only be able to be satisfied again if another camera within the same region is assigned instead.
- Alternative Solution involves the change of type of sensor, type of form or quality of the solution. Generally, it represents a solution which gives task satisfaction just by different means. This type of monitoring applies to tasks where a user is after a certain solution but is willing to sacrifice some of its e.g. quality if a perfect solution is unavailable. For example, in a situation that the police are chasing a suspect it is perfect to know his/her exact position, but it is also acceptable to know the region he/she is in, since the police have a large force they still can manage the situation. Such a situation can take place because the currently used resource ideal for tracking, e.g. helicopter with an IRCamera, will be reassigned to a more important task. This in effect doesn't allow for an accurate tracking and the CCTV cameras, which are assigned instead, only allow to observe suspect's last location, therefore, in order not to lose him/her police officers must secure many potential routes, but still are able to control the situation with the alternative sensor type.
- Presentation Form Change means the swap of the output of the result to a different format, for example, it happens when a task degrades from localisation to detection. At first we had exact point location and now we have an area of interest instead.
- Sensor Type Change is substitution of the current solution with sensors of differ-

ent type of which information is capable producing the same result after a certain processing effort.

- Solution Degradation is a particular type of alternative solution which involves various changes causing reduction of the current solution in some sense. It could be either quality of the output, e.g. reduction of a satellite's image quality delivered to beneficiary; or a more likely type of solution, e.g. change of task type from a particular car localisation to vehicle detection.

Figure 4.6 shows how the 'missing concepts' fit into the other concepts of the GHTR. There we present them as part of ontology of the fourth HTR, called the Max SAM Plus, which was created by augmentation of the Max SAM HTR ontology with the 'missing concepts'.

4.5 Max SAM Plus HTR

We claim that having the 'missing concepts' in a model enables servicing of all the steps of the DCPD loop, thus allowing for full system control over a task following its creation. For this reason, it was important to consider the creation of a model which would hold all the concepts. Our choice was to build it on top of one of the hybrid models, we have already created, as oppose to creating a very new one.

Since by the time the fourth HTR model was developed we have already conducted initial experiments that indicated Max SAM was the best-performing of the HTRs, we chose it for enhancement by introduction of the 'missing concepts' into its body. This way we obtained the ontology of the model which we call Max SAM Plus, for it is built on top of the Max SAM HTR. The full sets of experiments are in Chapter 5.

In initial tests (their details are presented in Chapter 5), we compared the capabilities of the three HTR models, which were: Max SAM, Max SWE, and Max SS (for test results see Section 5.3). We observed that Max SAM HTR is the most flexible and

adaptive one in every scenario. Our particular interest was the improvement of a task's adaptability in dynamic networks, i.e. scenarios where there is high platform status change possibility. Therefore, we first analysed these results. Here are two models, i.e. Max SAM HTR & Max SWE HTR, that score high and are very close, yet, when you look at the other scenarios, the Max SWE HTR appears to perform significantly worse than Max SAM HTR.

Thus, from those findings we favoured the Max SAM HTR as the candidate for further improvement. We decided to build our fourth HTR on the base of the Max SAM HTR model and augment it with the 'missing concepts'. The filling of these gaps would allow for more complete control over the Dissemination step and for expression of richer rules of monitoring in the Direction step, thus, obtaining a model which delivers all information required by the steps of the DCPD loop. This is crucial for a system using it in a dynamic scenario, particularly in scenarios where we expect sensing resources to change often their status or availability.

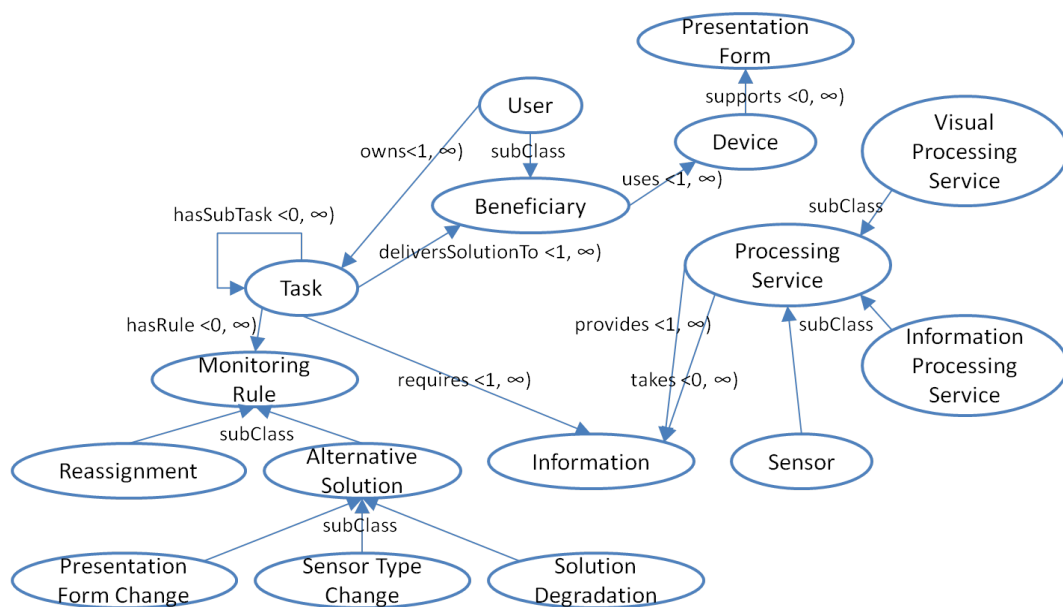


Figure 4.6: Model of the Max SAM Plus HTR.

In Figure 4.6 we see how the 'missing concepts' (from Section 4.4) were introduced into the Max SAM HTR ontology, giving us the model of Max SAM Plus. Task's

Beneficiary concept has a many to many relation with a Task, as a task can have many beneficiaries and many beneficiaries might need a solution for a single task. This is in fact more complex, though the beneficiaries have the same task they might require a slightly different solution (e.g. involving different sensor types). This type of relation we observe in the literature, for example, in the paper [83], where the authors present a policy layer on top of their middleware, which serves as a means of controlling sensor information sharing. A Task can have many Monitoring Rules, for example, a user might be able to accept an Alternative Solution involving Solution Degradation, i.e. loss of information quality, where one information might be insufficient to identify a car it still might be good enough to detect it, which is captured in the NIIRS [44]. In some other situations a user, might still desire the same quality of solution but he/she doesn't care about sensor type, and thus is ready to accept some Alternative Solution being it Presentation Form Change or Sensor Type Change. In situations where the user cares about exactly the same solution but doesn't care which sensor instances are delivering it, he/she would chose Reassignment.

Performance Evaluation of Hybrid Task Representations

This chapter presents our research evaluation. It describes simulation experiments conducted based on four different scenarios. The models from Chapter 4 are analysed in context of the scenarios, to observe their performance in different circumstances. This shows which situations each model is best suited for. Finally, we discuss the results of the experiment, in terms of the relative performance of the models, and make observations of potential further improvements in the case of each scenario. The evaluation shows that, through the use of a rich model, we can handle changes in some dynamic environments significantly better. We are basing the simulation on our framework, the SAM application 2.5.2.

5.1 Experimental Approach and Scenarios

Here we present a higher-level description of the experimental approach and simulated environment. We explain what a single run of the experiment looks like and we detail the scenarios for which it is executed.

5.1.1 Experimental Approach

The aim of the experiment was to verify the performance of each of the created HTR models, to observe how they relate to each other. In order to put them into context we introduced theoretical worst-case and best-case models. This allowed us to compare the performance results of the HTRs in comparison to these theoretical models. The operations and interactions a user would take using a system in the simulation were influenced by the way we implemented them in the SAM application 2.5.2.

The test environment is initialised and populated with tasks and platforms carrying sensors. The archetypes of each are defined a priori (for the types used see Section 5.2.1). Their instances are placed in random locations.

We simulate a user who expresses his/her task details, i.e. selects its type, priority and locates his/her task in an abstract area. Each task is created in an automated fashion, randomly selecting its location, task type from a set of possible types and it receives randomly assigned priority respecting goal lattice rules.

Since we simulate that assignment is influenced by user actions and network delays, in our assignment algorithm we set an abstract number representing how long it takes for a task to successfully assign sensors, the number depending on the model used. We introduced this to capture all that might be involved before a task gets a solution, i.e. a user detailing his/her task, the time the network will take to find resources appropriate for the task and the time which has to pass before the task will start receiving the required information etc. We count down the time, then when the time is up we assign the sensors. Details explaining how we calculated assignment costs for each of the models are explained in Section 5.2.3.

We consider a task to be satisfied only if it has all the sensors required assigned to it or it is considered to be an optional task (its priority, i.e. weight equals 0). This captures situations when, for example, an emergency response worker's main goal is to track victims of an accident and his/her secondary goal is to track the rescue team,

and he/she wants to make sure the resources will not be wasted for the secondary goal, if they are scarce.

In order to introduce dynamicity in our simulated environment we introduced variability of arrival and departure of both platforms and tasks. This is controlled by scenario characteristics (they are detailed in Section 5.1.2). This way at every time-step there exists a chance that a new platform would appear and that the existing one would be removed, and the same for tasks. In such way putting the models versus the characteristic of scenarios we see which model is more than others appropriate to a particular scenario.

If there are any changes within a sensor, task or the network, we simulate a user responding to them, i.e. correcting/re-specifying details of the affected task. We do that by running the timer counting down the time before the task achieves the readiness for sensor assignment again, only then do we connect the required sensors to it; otherwise we let the task operate normally. We simulate a user responding to changes in the network in a similar fashion as during creation of a task.

The simulation is repeated for each HTR, for the same time line (i.e. initial state of the network and changes to it). It is done so we can then use the test results and compare them against each other, since in this way the results are only model dependent.

We put results for each model in context by introducing them next to the best and the worst theoretical models, respectively Theo.Best & Theo.Worst (they are further explained in Section 5.2.3). This allows us to observe how the other models relate to this particular cases. Thus, we look for these models which are scoring closest to the results obtained for the best theoretical case, i.e. Theo.Best.

The algorithm of our experiment's execution, specifically its part concerning a task's state evolution, is presented in the operation flow chart in Figure 5.1. There we see steps the algorithm goes through in a single time-step. We see how it verifies changes to a task's state, and how it responds to them. It shows a more formal representation

of operations we have discussed in this section. A task's state Satisfied means that at some time-step a valid solution was assigned for the task. A Valid Solution consists of a configuration of platform and sensor instances (of types required by the task) where each is fully functional and not yet assigned to other tasks. Assignment Counter keeps track of time (a number of time-steps) for a task, and it says for how long a task was in its current state (e.g. (not) satisfied). The counter is increased at the end of every time-step and it is reset to zero only when the current state of a task changes. In the algorithm we use it to count the time before we can assign resources to a task. We assign the resources only when they are available and when the counter's value is greater or equal to the value of the current cost of the task. Normal and Alternative Assignment Costs (for each model are taken from Section 5.2.3) represent, respectively, time it takes to create a task, and time it takes to switch to an alternative solution. In terms of our problem definition (described in Section 1.2): normal assignment cost is equal to $g_i + ei_i$ and alternative assignment cost is equal to $g_i + er_i$.

On closing of our experimental approach discussion we want to explain why we chose to simplify some of the issues, mostly related to a real life implementation. The reason was that we were focusing our approach on the role of the task representation in handling of the reactiveness problem and this is what we highlight in our approach.

We do not consider, in the simulation, location and status of the processing services, which, in real life situations, could also vary. They might be located within the sensor network, on a user's device or in some other computer network to which the user has access. If we were to introduce the manipulation of processing services in our simulation it would only cause the changes to occur more often and it would affect each HTR similarly. We think that it would have noticeable impact during task execution and it would be the highest for the Max SS HTR, where no alternative solutions are captured, resulting in a more frequent recreation of a task. In fact, if we consider the specifics of the processing services, we could claim that they are (or at least can be made) not very susceptible to a status change as they mostly involve virtual entities

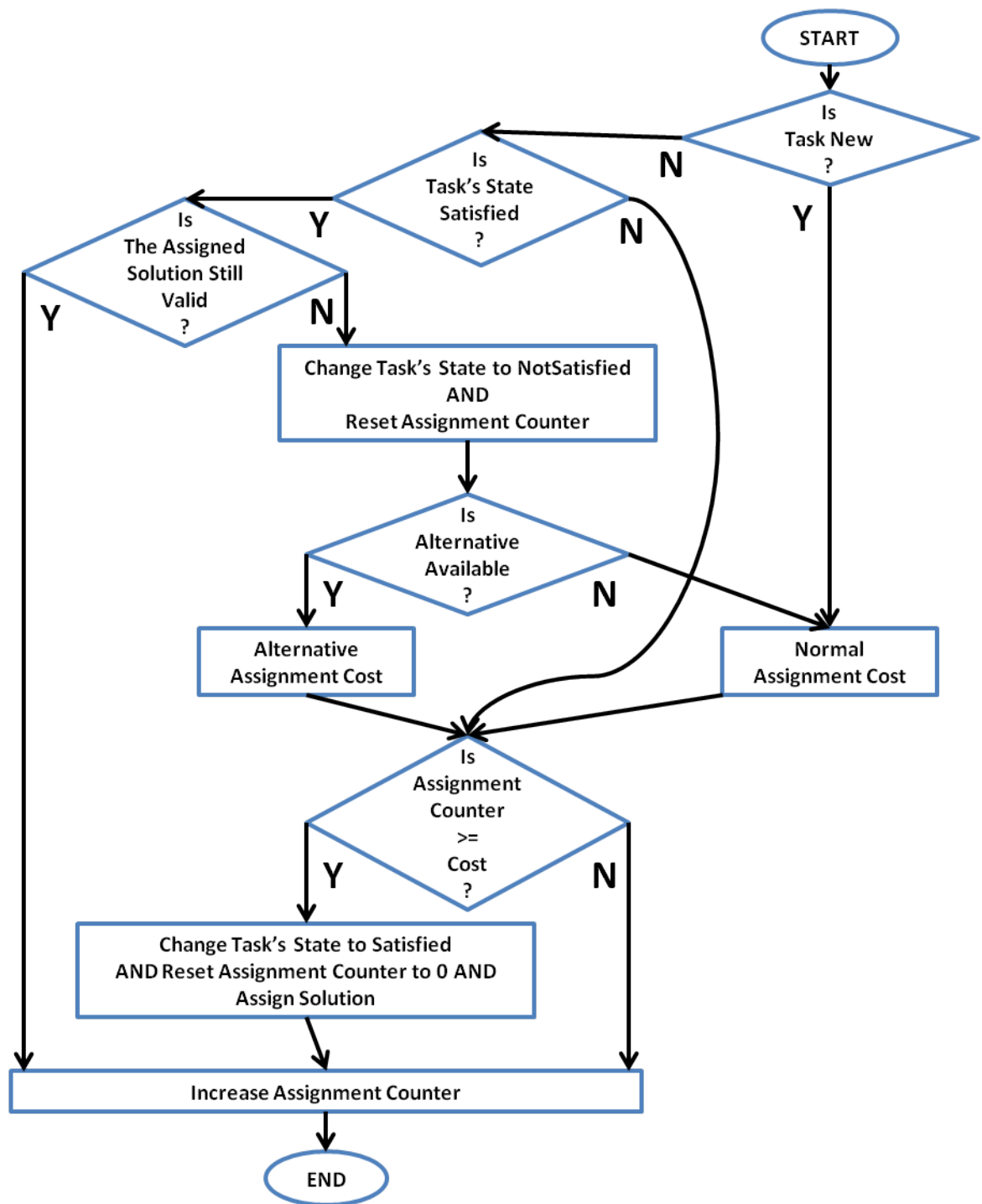


Figure 5.1: Flow of operations in a task's state verification algorithm in a single time-step. Rectangles represent operations; diamonds represent conditional change of operation, where Y is for Yes, N is for No.

(e.g. algorithm of object localisation by use of acoustic signal triangulation). For this reason their replication is much easier, thus we treat them as stable. For example, in a situation when battery life is not an issue we can, theoretically, have them on every node of the network, thus technically always available on a sensing resource.

We want to clearly state that, in our approach, we consider that a platform could be mobile or static. Though we do not simulate movement of platforms in our experiment, we do consider both availability and distance of platform to a task (each sensor has radius of operation). But the fact that the platform is mobile or static is not that interesting for us as whether the platform is available or not. As we stated in the problem definition we are focusing on the reactiveness problem from the perspective of user-level tasking, thus we care about the fact that a task has lost a platform exclusively, independent of the reasons, e.g. it could be destroyed, permanently removed or it could moved out of its sensors range. Therefore, we are simulating platforms appearing and disappearing which in this respect could also mean that a platform has moved away from a task, thus it is no longer useful for it.

5.1.2 Scenarios

Here we present our four different scenarios. They are descriptive examples of the kinds of network environments where assignment of sensors is taking place. They were created for various task and platform change probabilities. Their intention is to show how well suited a model is to a particular type of scenario defined by the degree of potential change in sensor and task states.

We want to state that we had no access to the actual data on platform and task change probabilities in real-world scenarios. Therefore, we had to make assumptions about these in the simulation. To show the significant difference between some of the scenarios, we chose the probabilities accordingly.

We selected two probability levels of state change occurrence, i.e. low (LO) and high

(HI). This way we can present use of models in 4 different extreme scenarios, as presented in Table 5.1. LO represents a chance with the likelihood between 0-5%, and HI means a chance of 95-100% that a change will occur during a single time-step, i.e. either a currently running sensor or task will be removed from the network or a new sensor or task will be added to it. Having these two levels set for sensors and tasks we described four typical scenarios which we later use for better understanding of the experiment's results. Their analysis is presented in Section 5.3.

		Platform change probability	
		LO	HI
Task change probability	LO	Environmental Monitoring	Emergency Response
	HI	Border Monitoring	Urban Unrest

Table 5.1: Scenarios in relation to sensor and task levels of change probability.

- Task LO Platform LO: Environmental Monitoring. This scenario takes place where we employ a sensor network built for a purpose, for example, in order to monitor glacial status or to track animal movements in an area. In this sort of scenario the network and its resources are set for a particular purpose and are unlikely to change. If a sensor's status changes, e.g. its battery is depleted it will not be changed before the period of time a scientist regularly comes and does change the battery. Since the period of time is set to make this unlikely, thus this might never occur. For these reasons we can see this type of scenario as stable on both degrees of freedom. In [42, 43] authors present the scenario specific challenges. They describe that during deployment and life time of such network all steps are taken in order to make sure no changes (e.g. sensor battery depletion etc.) in the network's state will occur.
- Task LO Platform HI: Emergency Response. This type of scenario is, usually, focused on a single task i.e. securing the area affected by a disaster or rescuing people from a burning building. Therefore, the state of tasks is unlikely to change

or new task to occur. Yet in this situations the likelihood of change is high for the resources, since, for example, in emergency response situation there is lots of human intelligence generated by people. The people here are the sensing resource, since they often change their location, thus the resources' availability is changing very often. It was shown in the recent research that the data delivered by mobile users is very useful, particularly in emergency response scenarios. The authors of [5] describe population tracking with use of mobile data in a study after the 2010 Haiti earthquake, [25] where the authors discuss benefits and challenges for efficient use of mobile data in such scenarios, or [72] where authors present analysis of current Mobile Disaster Management System applications. The data generated by people could be effectively utilised to access the situation better and to locate and get to the people in need of assistance faster.

- **Task HI Platform LO: Border Monitoring.** This includes surveillance of a region, this could be a country's border or something more specific as a property, where we have a sensor network deployed with purpose of securing the area. Here the main need is to track the movement of objects (people and/or vehicles), and ensure they do not act inappropriately. For this type of situation the resources are considered fairly stable. No new resource will arrive and the likelihood of existing resource states changing is low, e.g. it is not likely that trespassers will bother to destroy cameras while running through the border. Yet tasks are often submitted, e.g. tracking of a fleeing trespasser, tracking of an aggressive group of fans after a game, or just monitoring of people and vehicle movement, observation for aggressive behaviour, and many many others. These issues are described, for example, in [1] where general description of the scenario issues is provided, in [17] authors discuss their approach to improving security of a sensor network deployed in the scenario.
- **Task HI Platform HI: Urban Unrest.** This scenario takes place when a large mass of people suddenly starts acting aggressively and the people responsible for the

safety are directed to securing the outburst's results. For example, a regular city policing task in the case of a riot become a very highly dynamic scenario. In such a situation there is lots of tasks thrown on the network also resources are more likely to get destroyed plus human generated information is flowing from various sources and places, thus increasing the speed of resource state change. This scenario is a very dynamic one. It is because of its contagious nature that if not contained quickly it can spread to other parts of a city or even beyond its borders. It could intensify in nature, e.g. CCTV cameras get destroyed, buildings get raided or even burned. These scenes we observed, for example, during the recent riot in London, UK ¹. In the book [23] the author describes analysis of past unrest occurrences in US, trying to understand how to contain them and the reasons for one occurrence. The author says that the only thing stopping people from rioting is improvement of their well being, mainly reduction of poverty. Since this involves a large degree of social aspects such a solution is not easy to achieve, and it is, realistically speaking, not possible to predict the severity nor the scale of a public outburst.

5.2 Experimental Environment Design

Here we explain how we built the experiment. We detail our assumptions and present parameters used in it. Following the information enclosed in this section one should manage to easily replicate our results. The aim of this experiment is to observe the performance of each of the designed hybrids in scenarios (detailed in Section 5.1.2) characterised by the likelihood of tasks' and sensors' state change.

The project including the code of the experiment is available at ².

¹<http://www.bbc.co.uk/news/uk-england-london-14439970>

²<http://users.cs.cf.ac.uk/K.Borowiecki/Thesis/SimulationProject.zip>

5.2.1 Test Parameters & Assumptions

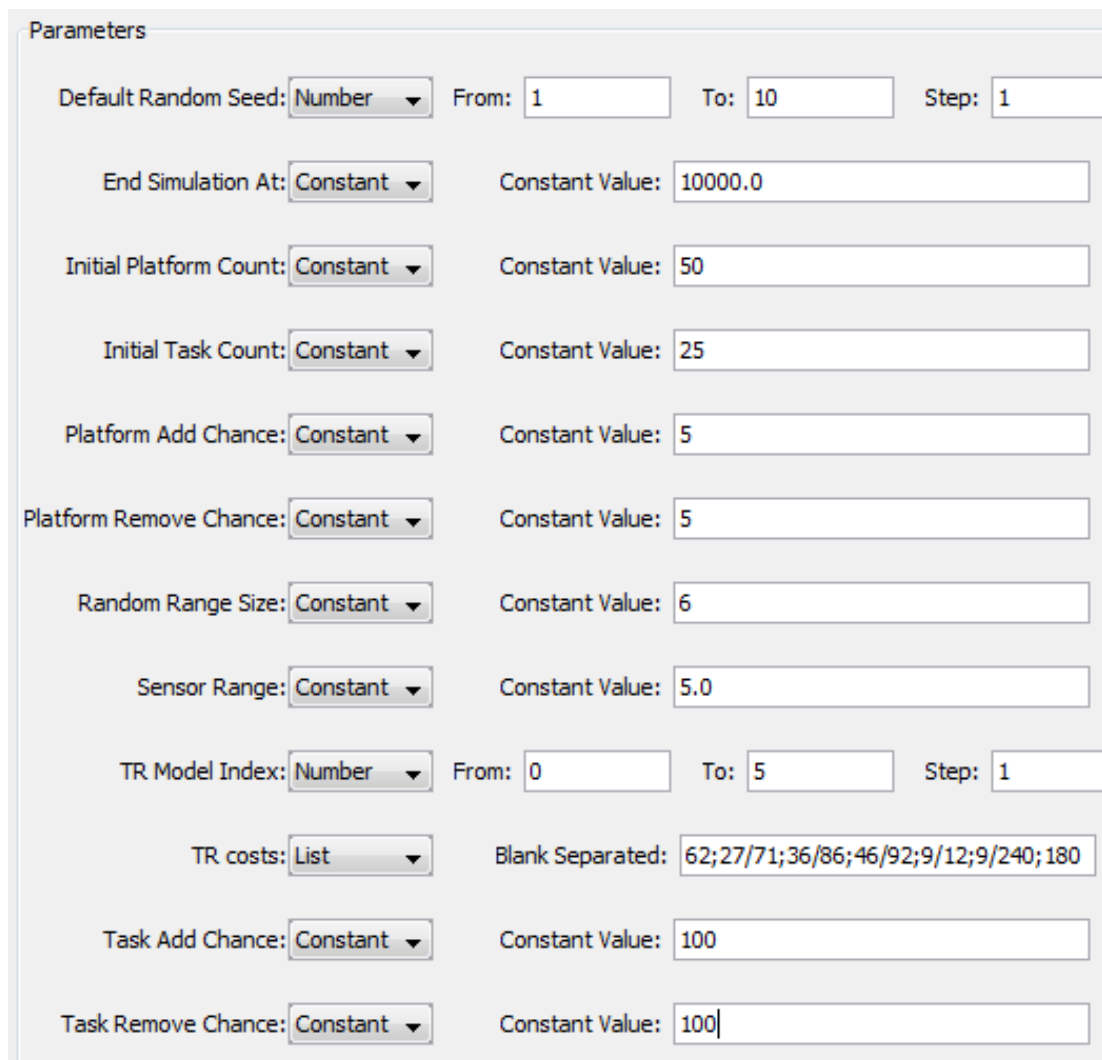
Here are presented parameters of the adaptability performance test, i.e. sensor and task types used, etc. This section details all the parameters required to configure the testing tool that would allow replication of the results, to those interested in it.

The tool used in our experiment is called the Repast Symphony 2.0 beta [65]. This is a cross platform agent-based modeling system written in Java. We wanted to simulate high level behaviour of parts of a sensor network. This tool was perfect for this role, since it had all the elements we needed in order to build the experiments. It is often used for agent behaviour simulation. It has been successfully used in many domains, for example, social science, consumer products or supply chains, etc. [65] In our case we have task, platform and sensor agents which, every time-step, we try to connect together depending on the current state of a task and availability of sensors and platforms in the network. A single time-step in our simulation represents a time including operations on information and propagation of the information between nodes of the network. It involves network latency, network complexity (no. of jumps for a message), size of message, etc.

In Figure 5.2, we present an example of parameters as if they were set for a single run of our experiment using Repast's Parameter Sweep option.

The Default Random Seed parameter is used to allow for repetition of randomly taken actions for every decision in the experiment. Thus, running the simulation with the same seed but, for example, with a different HTR model will effectively result in the same changes within the network (e.g. platform was added or removed etc.), where other changes will be model dependent (e.g. task satisfaction rate). The value of this parameter in our case was a range of integers from 1 to 10 since we repeated each test 10 times (each time with a different seed) for each HTR.

The variable, End Simulation At, controls when (at which time-step) to finish the simulation, in our case its value was always set to 10000. Thus, each single test was



Parameter Name	Type	Value
Default Random Seed	Number	From: 1 To: 10 Step: 1
End Simulation At	Constant	10000.0
Initial Platform Count	Constant	50
Initial Task Count	Constant	25
Platform Add Chance	Constant	5
Platform Remove Chance	Constant	5
Random Range Size	Constant	6
Sensor Range	Constant	5.0
TR Model Index	Number	From: 0 To: 5 Step: 1
TR costs	List	Blank Separated: 62;27/71;36/86;46/92;9/12;9/240;180
Task Add Chance	Constant	100
Task Remove Chance	Constant	100

Figure 5.2: Example of parameters for the experiment run using Parameter Sweep option.

performed for a constant number of time-steps, i.e. 10000.

The Sensor Range Parameter represents a maximum distance a sensor can be apart from a task if it is to be considered useful for it. We took a fairly big value (i.e. 25 units), in relation to our space of simulation (i.e. 50 by 50 units), for this parameter because we wanted to limit its influence on our results. Thus, we wanted to observe mainly variability being the result of a HTR model's characteristics.

Initial Platform Count and Initial Task Count are values expressing the number of,

respectively, platform and task instances that are created on start of the simulation. In our case it is always set to, respectively, 50 and 25. Having these values we wanted to make sure that, at least in the initial stage of experiment, there are enough platforms to satisfy the tasks.

Platform (Task) Add/Remove Chance parameters could have a low (LO) or high (HI) value which, in numerical sense, is a range of values, respectively, from 0 to 5 % for LO and from 95 to 100% for HI. It means a chance that during a particular time-step a task (or a platform) could appear or disappear. The chance is calculated by taking the value of the Random Range Size parameter and the random selection of an integer from 0 to its value (excluding) then subtracting it from both the value of add and remove parameters (since we want to keep them the same). It is done, at the start of a test, once for Task and once for Platform Add/Remove Chance parameters.

The Random Range Size parameter expresses the size of the range of values of Platform (Task) Add/Remove Chance. Its role is to create some variability for these parameters in each test which is within a certain range of values. In our case it is always set to 6, this way we have a modifier from 0 to 5. Two randomly selected values from its range are then subtracted, at the start of a test, one from Task and the second from Platform Add/Remove Chance parameters.

Since a single experiment uses a particular TR model for the whole of its duration, we use the TR Model Index variable to control change of the TR model between experiments. We give this parameter as a range of integers from 0 to 5 (both inclusive) since we have 6 models.

TR Costs is a parameter used to set the values of variables storing costs of assignment and alternative assignment for each TR model. It is done once at the start of a simulation. The costs are integers and they represent a number of time-steps it takes for a task before it can successfully assign resources. The parameter itself is expressed in the form of a string, i.e. normal assignment ‘;’ alternative assignment, and each model’s values are separated by ‘/’. The method explaining how we calculated the costs for

each model is described in Section 5.2.3.

This way we have each time, for a single experiment (executed for a single scenario), a number of tests which is calculated as the number of used Default Random Seeds multiplied by the number of used TR models (expressed by the TR Model Index parameter). Therefore, in our case it is always 60, i.e. $10 * 6$.

As a side note, please also notice how, in Figure 5.2, the ‘types’ for the parameters (the boxes just to the right of parameter names) are set, i.e. all are set to Constant except for the Default Random Seed and TR Model Index which are set to Number; also except the TR Costs which is set to List. This is related to how the Constant and Number types are implemented in the used version of the Repast tool, i.e. Constant can only take double values and Number parameter can only take integer values.

The experiment is run 60 times, i.e. 10 times (since we use 10 Random Seeds) for a single model and we use 6 models, for each of the four typical scenarios (for scenario details see Section 5.1.2). Each of the scenarios is characterised by the percentage chance of occurrence of a change in the state of sensors and tasks in a sensor network, being it arrival or removal of a resource.

We used a set of 10 unique platform configuration types, which means a particular combination of a type of platform and types of sensors it carries. We chose these platforms since in relation to the selected task types (presented below) they can each satisfy a few of task types. This way we make sure that for tasks alternative solutions are available. We did it since our aim is to verify how responsive each model is and how effectively it uses available alternative solutions to improve its responsiveness. The platform configurations are as follows:

- Improved GNAT (I_GNAT³ is an improved version of the original GNAT reconnaissance UAV manufactured by the General Atomics Inc.) with Electro-Optical

³http://en.wikipedia.org/wiki/General_Atomics_GNAT

Camera (EOCamera).

- I_GNAT with Synthetic-Aperture Radar (SAR, for more see ⁴), EOCamera and Forward Looking InfraRed camera (FLIR is a camera which senses infrared radiation, for more see ⁵).
- Predator_A (is a UAV used for reconnaissance and forward observation roles, more available at ⁶) with SAR and TeleVision Camera (TVCamera is a type of a professional camera that can produce images of high quality, for more see ⁷).
- Reaper (is a surveillance UAV, for more see ⁸) with Daylight TeleVision camera (DaylightTV is a high image quality camera which is designed to perform in day conditions) and SAR.
- Reaper with SAR.
- Harrier_GR9 (is a vertical/short takeoff and landing jet aircraft, for more see ⁹) with EOCamera and InfraRed Camera (IRCamera is a type of camera that registers infrared radiation of objects, more available at ¹⁰).
- Global_Hawk (is a large surveillance UAV, for more see ¹¹) with SAR, EOCamera and IRCamera.
- AS_Mote (is a specific type of Mote, i.e. a small wireless sensor which can form with others a sensor network, where each mote serves as a node of the network ¹²; which has acoustic and seismic capabilities) with AcousticSensor (is

⁴http://en.wikipedia.org/wiki/Synthetic_aperture_radar

⁵<http://en.wikipedia.org/wiki/Flir>

⁶http://en.wikipedia.org/wiki/General_Atomics_MQ-1_Predator

⁷http://en.wikipedia.org/wiki/Professional_video_camera

⁸http://en.wikipedia.org/wiki/MQ-9_Reaper

⁹http://en.wikipedia.org/wiki/Harrier_GR9

¹⁰http://en.wikipedia.org/wiki/Thermographic_camera

¹¹http://en.wikipedia.org/wiki/Global_Hawk

¹²<http://en.wikipedia.org/wiki/Motes>

a sensor that is able to capture acoustic signal) and SeismicSensor (is a sensor that can capture seismic signal).

- Raven (is a small launch-able by hand UAV, for more visit ¹³) with DaylightTV, Low Light TeleVision (LLTV, is a camera which provides night vision capability ¹⁴) and FLIR.
- Nimrod_MR2 (is a maritime patrol aircraft, for more see ¹⁵) with IRCamera and EOcamera.

We used a set of 12 unique task types; 6 of detection type i.e. verification of presence of a looked for object in a region; and 6 of identification type i.e. classification of the more specific characteristics of the detected object thus usually e.g. information quality requirements of this sort of tasks are higher, as we can observe in NIIRS [44]:

- Detect: Ground Platform, Tracked Vehicle, Wheeled Vehicle, Aerial Platform (represents any platform that can fly), Helicopter, Airliner (is a particular type of Aerial Platform which includes only planes).
- Identify: Train, Car, Lorry, Thermally Active Vehicle (is a specific type of a ground platform which leaves a heat signature), Frogfoot (is a small military fighter jet, more available at ¹⁶), Bear (is a large aircraft, more available at ¹⁷).

5.2.2 Method of Reasoner Use in Simulation

To substitute live reasoning during simulation (which would otherwise be time consuming) we chose to precompute all the reasoner's results and put them into tables. For the purpose of this reasoning we employed the SAM reasoner [62] which is part of

¹³http://en.wikipedia.org/wiki/AeroVironment_RQ-11_Raven

¹⁴http://en.wikipedia.org/wiki/Night_vision

¹⁵http://en.wikipedia.org/wiki/Nimrod_MR2

¹⁶<http://en.wikipedia.org/wiki/Frogfoot>

¹⁷http://en.wikipedia.org/wiki/Tupolev_Tu-95

technology behind the SAM TR (Section 3.2.4). The tables' names are (each is stored in a separate file with corresponding name and extension '.csv'): Task Types, Platform Solution Types and Solution Types. The first two, respectively, store identifiers of all generated task types and platform solution types. The third one serves as a mapping between these two, i.e. it stores what platform solution type ids are applicable for which task type ids. In each row it maps one task type to a particular solution. Thus, a few rows might represent solutions for the same task type, but obviously having different values of platform solution types.

The tables were created by the following algorithm:

1. Each task type (TT) is defined by type of interpretation task: Detect, Identify or Distinguish; and one or more detectables, depending on the applied interpretation task. In the case of Detect and Identify, only one detectable is required, whereas Distinguish requires two.
2. For each class in the Detectables ontology if it has sub classes we create a detection TT, otherwise an identification TT [16].
3. From a set formed from the newly-created detection and identification TTs we create distinguish TTs combining two TTs, where they respect the following rules: the detectables of both TTs are not equal, and one is not a sub class of the other and vice versa [44]. Thus, distinguishing between a concept and its sub concept is impossible, e.g. distinguish ground platform from car, since the general one includes the sub concept.
4. For all created TTs we randomly assign rating values for scales of NIIRS (and NAIRS (acoustic) and NSIRS (seismic) equivalents of NIIRS scale) depending on how low in the ontology tree hierarchy they are (level/value range/inapplicable chance: 1/1/0, 2/1..2/5, 3/2..4/15, 4/4..7/35, 5/7..9/60). Additionally, for identification tasks, to represent that they are generally harder to satisfy (observable in the NIIRS table [44]) we add an integer for each rating, randomly from

range 0..2. In case that the rating value ends up over the maximum range value (i.e. 9) it is considered not applicable for this TT. A value, for example, of a NIIRS scale for a particular TT means that it requires sensor types which have equal or greater capability of corresponding scale, e.g. a camera with NIIRS 4 will be able to satisfy any TT which NIIRS scale is not greater than 4.

5. For each created TT we invoke the reasoner to obtain platform and sensor types which have capabilities that correspond to the TT's requirements. Thus we obtain a list of solutions with platform configuration types (PCT) i.e. platform type and a list of sensor types mounted on it that satisfy a task type.
6. We map the obtained solutions into the Platform Solution Types table where we add a new row only if such PCT was not yet present in it.
7. At the same time we add them into the Solution Types table where a unique PCT id is entered next to a task type id. This way each row in the task Solution Types table will represent a tuple: $\langle TT, PCT \rangle$.

To see an example of the algorithm's output please see the three listings below, where five first rows of each table are presented.

Listing 5.1 shows the first five rows from the Task Types table. We see here, for example, that a Factory must be the most specific type of a structure therefore we have Identify_Factory type of task, where Structure being its super class will form Detect_Structure type of task.

```
Id ,TaskType
1 ,Detect_Detectables
2 ,Detect_Structure
3 ,Identify_Factory
4 ,Detect_Bridge
5 ,Identify_RoadBridge
...
```

Listing 5.1: First five rows from the Task Types table

Listing 5.2 shows the first five rows from the Platform Solution Types table. For example, the fifth row represents a platform configuration of I_GNAT platform with a FLIR sensor mounted on it.

```
Id , PlatformSolutionType
1 , AS_Mote : AcousticSensor
2 , AS_Mote : SeismicSensor
3 , Raven : FLIR
4 , Raven : DaylightTV /LLTV
5 , I_GNAT : FLIR
...
```

Listing 5.2: First five rows from the Platform Solution Types table

Listing 5.3 shows the first five rows from the Solution Types table. It shows, for example, that all platform solution types presented in the Listing 5.2 above are applicable as a solution for a Detect_Detectables type of task. This is because the Detectables concept is the top most concept in the class hierarchy of the Detectables ontology and it represents any type of detectable, i.e. anything that might be detected by sensor means.

```
Id , TaskTypeId , PlatformSolutionTypeId
1 , 1 , 1
2 , 1 , 2
3 , 1 , 3
4 , 1 , 4
5 , 1 , 5
...
```

Listing 5.3: First five rows from the Solution Types table

5.2.3 Assignment Cost Calculation Methodology

The method used to evaluate the cost of assignment for each of the models can be described in 5 steps:

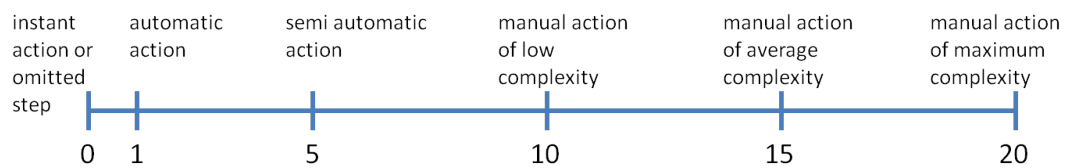


Figure 5.3: Task representation assignment cost scale.

1. Determine possible points of user interaction (discussed in Section 3.1), which can be served by a task representation (TR) model. These points are the result of the information needs of the steps of the Direction, Collection, Processing, and Dissemination (DCPD) loop (Section 3.1). Their satisfaction is required for the automatic operation of a task.
2. Determine which of the points are automatic, and which require user interaction and to what degree. The points result from the concepts which are included in the ontology representing a TR.
3. Apply the scale in Figure 5.3 to determine the cost of assignment for each TR. Remembering to respect relations between TRs, expressing the complexity of a step in a model.
4. Determine which steps have an influence on the cost of an alternative assignment of sensors. Some steps might not be required or they stay the same, e.g. the goal of a task, the recipients of the information satisfying tasks or the way information is delivered.
5. Finally, calculate the cost of the alternative assignment.

Applying the aforementioned methodology we obtain the following table 5.2. It presents the assignment costs calculations for each of the HTR models in relation to the potential interaction points of the DCPD loop (presented in Section 3.1). To make the decisions leading to creation of the table we reviewed the literature concerning the DCPD loop, particularly studying which of the points, authors of various papers were trying

to automate. Considering those we concluded which of the interaction points are influencing alternative assignment or, in other words, which points might require additional interaction from the user before a task can consume the new solution.

All the parameters of the simulation, including the values of the cost scale, were set to show in a statistically significant form the relations between the models and their task satisfaction potentials. We chose relation of auto-manual (1-20) when tuning the experiment to enable us observation of relation between the performance of the models. We tried also with other values (e.g. 1-100 and 1-1000) while other parameters were kept unchanged. The relation of those results, with increased gap between the costs, was proportionally the same though significance of (some of) the results could not have been statistically proven. Therefore, we needed to tune the parameters to a degree where we were able to show the relations between the models in a significant manner. We had to decide on these values ourselves since we were unable to find a definitive answer in the literature that would say how much an action should cost (in terms of time-steps). It very much depends on the situation and is often subjective. Therefore, we chose to make the difference between auto and manual action of right proportion to allow the results to preserve their significance.

It is important to notice, at this point, that Table 5.2 contains two more models than the 4 already presented, i.e. the model Theo.Best & Theo.Worst. The Theo.Best is the best case theoretical model, i.e. where we assume each step is fully automated, thus each operation takes the minimal time of 1 time-step. The Theo.Worst is the opposite of the Theo.Best model, i.e. Theo.Worst is the worst case theoretical model, where we assume that each step requires a complex user's intervention every time, thus the cost of a maximum 20 time-steps per operation.

Interaction Point	Max SAM	Max SWE	Max SS	Max SAM Plus	Theo.Best	Theo.Worst
Task Description	15	15	20	15	1	20
Sensor Type Selection ●	1	1	0	1	1	20
Sensor Instance Selection ●	1	1	1	1	1	20
Sensor Setting ●	1	1	1	1	1	20
Monitoring ●	10	10	20	10/1	1	20
Data Collection ●	1	1	1	1	1	20
Data Forwarding ●	1	1	1	1	1	20
Data Processing ●	1	1	1	1	1	20
Information Forwarding ●	1	10	1	1	1	20
Recipients Selection	10	10	10	20	1	20
Delivery Medium Selection	10	10	10	20	1	20
Presentation Format Selection ●	10	10	20	20/1	1	20
Σ	62/27	71/36	86/46	92/9	12/9	240/180

Legend

●, Interaction point influencing alternative assignment.

X/Y, where X - cost of creation time assignment, Y - cost of alternative assignment.

Table 5.2: HTRs' costs evaluation table. Cost of Interaction Point for each HTR

5.3 Performance Experiment Results and Analysis

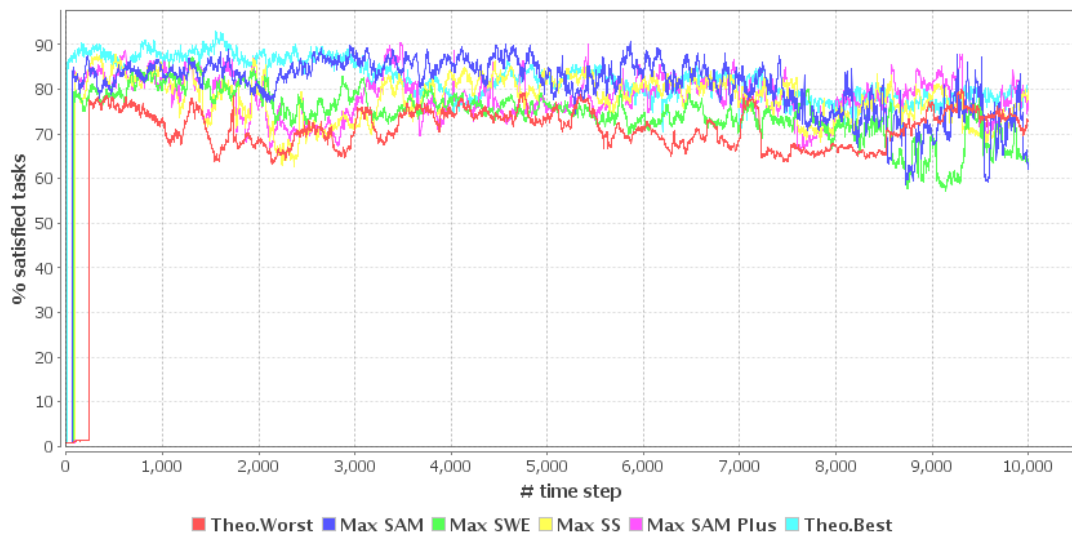
In this section we present the results of the tests. The results are shown on graphs for each of the aforementioned scenarios. They are discussed in the same order as they are presented in Section 5.1.2. As a reminder, LO and HI, respectively, mean low and high likelihood of a change.

For each scenario we are presenting two figures, with values showing the results obtained for it. The first figure, in the form of a line plot, e.g. like Figure 5.4a, presents time-step per percentage of satisfied tasks. In it we can see an average value representing how many tasks are satisfied in a particular time-step throughout the whole simulation time. The other figure, in the form of a bar plot, e.g. like Figure 5.4b, shows percentage of overall satisfaction of tasks and deviation of the results calculated over the whole period of the simulation. In this one we observe the overall performance of all the models used in the simulation.

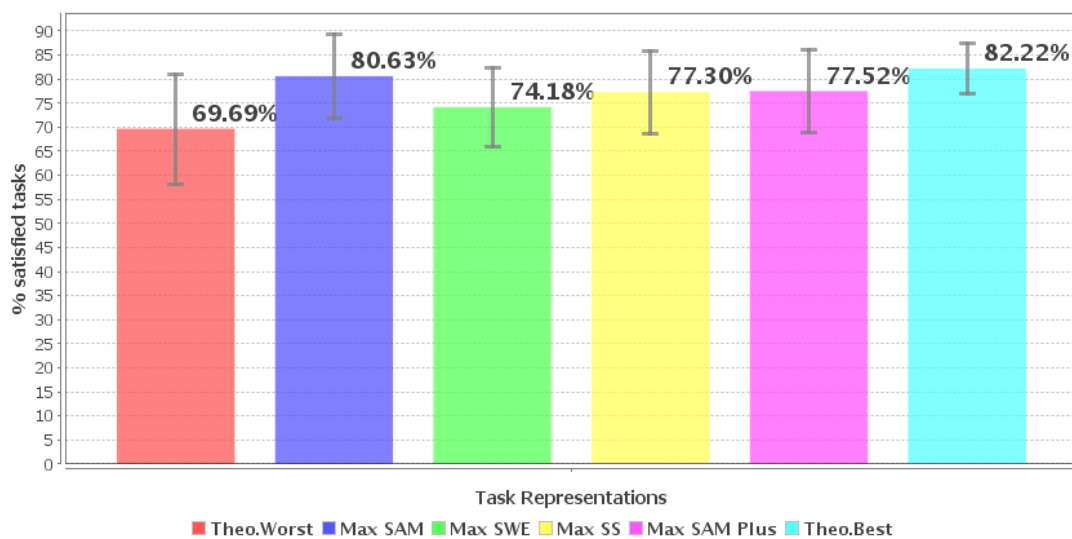
5.3.1 Results for Task LO Platform LO Scenario

In this section we present results obtained from experiments performed for the Environmental Monitoring (Task LO Platform LO) scenario. In this case we observe, for all models, nearly perfect results, i.e. each model scored very close to the Theo.Best model. The reason being the effectively small changeability of the network state. Therefore, once a task receives its information it will, very likely, stay satisfied till the simulation ends. Interestingly out of the four designed HTR models the Max SWE scored the lowest. Also the Theo.Worst model obtained surprisingly high results, presenting very small difference between it and the Theo.Best model.

Figure 5.4 shows the results of a scenario such as, rainforest animal tracking. We expected such results, i.e. that each model would perform well in this scenario, since it is fairly stable, thus once we satisfy a task it will, very likely, stay satisfied till its end. We observe that even in the worst case we achieve task satisfaction of a value not too far from that for the best case, respectively, 69.69% and 82.22%. The other models scored: Max SAM 80.63%; Max SWE 74.18%; Max SS 77.30%; and Max SAM Plus 77.52%. We see here that Max SAM HTR performs best versus the other 3 HTRs. It achieves values nearly that of the best case model, even though the differences are not too big between Max SAM and the other three models i.e. it is around 6% vs Max SWE and 3% vs Max SS or Max SAM Plus.



(a) % satisfied tasks by time step



(b) % satisfied tasks overall (mean and standard deviation)

Figure 5.4: Test results for Task LO Platform LO scenario.

5.3.2 Results for Task LO Platform HI Scenario

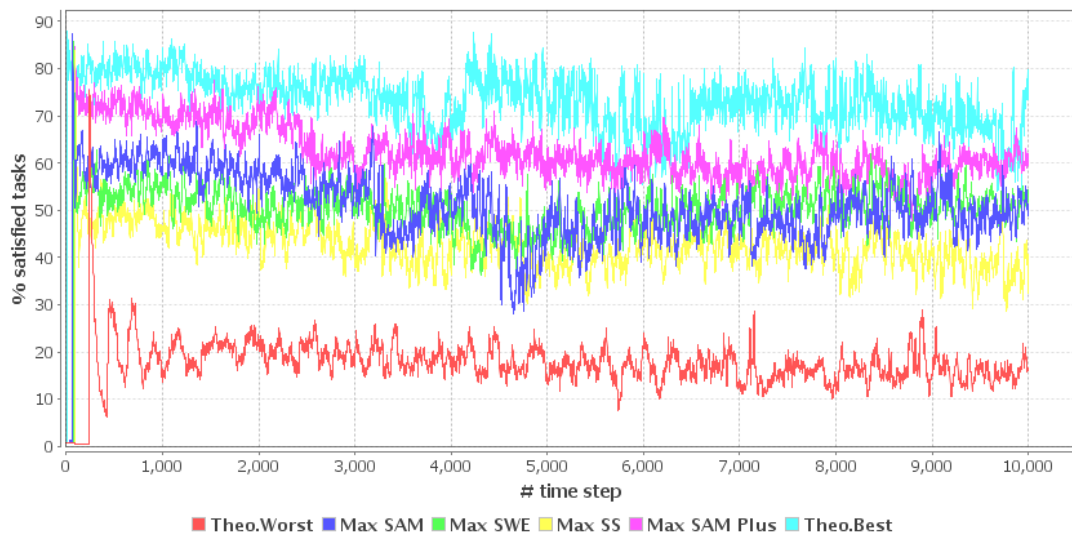
The section presents results obtained from experiments performed for the Emergency Response scenario, characterised by LO Task, and HI Platform status change probability. In this case we observe that all the designed models perform well, we also notice for the Theo.Worst a significant drop of task satisfaction. The results must be the effect

of small changeability of task states and effective handling of the changes in platform states, through use of available alternatives. Therefore, once a task receives its information it will try to adapt to its loss by finding other applicable information generated by an alternative platform instance. Interestingly the highest impact, the increase of possibility of platform status change had, was on the Max SS HTR model (it is nearly half of the results observed for the Environmental Monitoring scenario). We also observe that this had very small effect on the performance of the Max SAM Plus, which is the result we were hoping for, when designing it.

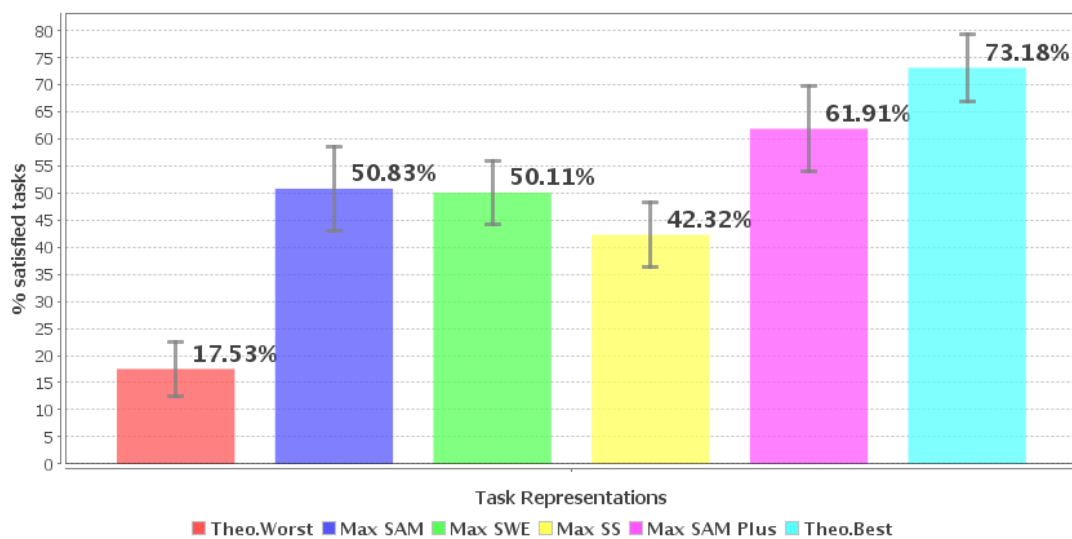
In Figure 5.5 we present results for the Emergency Response scenario. Here we can observe the results we were expecting. They prove our claim i.e. by enriching a model, here Max SAM to Max SAM Plus, we have achieved a significant improvement in task satisfaction rates. This proves that a high level of platform changeability can be confronted via use of a rich model delivering a sufficient amount of information allowing for automatic system operation. This results in a faster selection of alternative solutions for a task, while utilising network resources more effectively, giving us a higher task satisfaction rate (Max SAM Plus 61.91%), which is only 11% worse than that of the best case model (73.18%). It is half the gap between the used model (Max SAM 50.83%), on top of which it was built, and the ‘ideal’ model. Notably other models perform well by themselves, without enrichment (Max SWE 50.11%, Max SS 42.32%), which is much higher than the results for the worst case, 17.53%. The results, after enrichment, show that the use of a richer model gives a significantly more efficient handling of tasks in this type of scenario. Indirectly this provides a more efficient use of the network resources, while directly improving satisfaction of a user’s tasks.

5.3.3 Results for Task HI Platform LO Scenario

The section presents results obtained from experiments performed for the Border Monitoring scenario, characterised by HI Task, and LO Platform status change probability. In this case we observe that all the designed models perform badly. The Theo.Worst’s



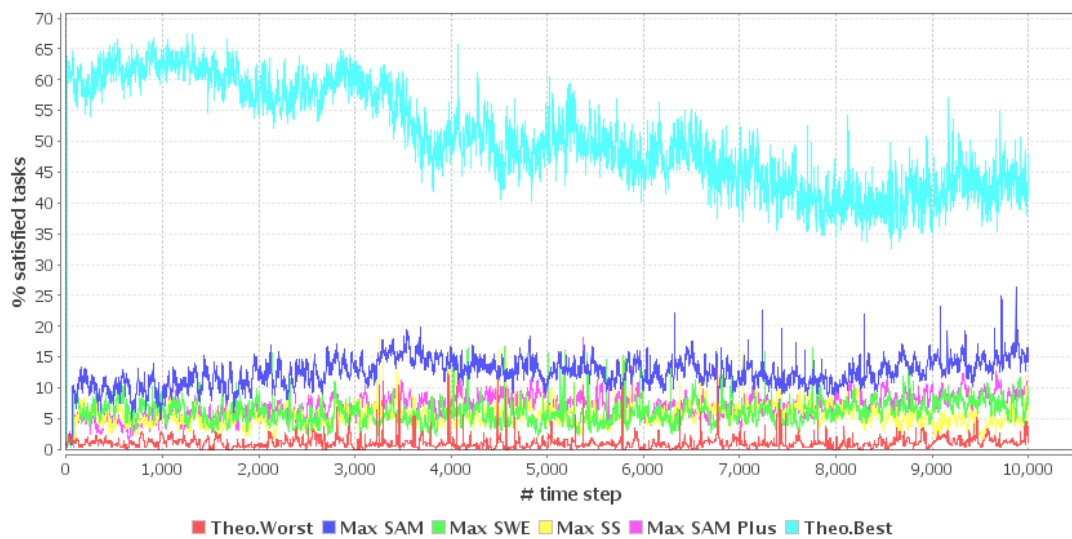
(a) % satisfied tasks by time step



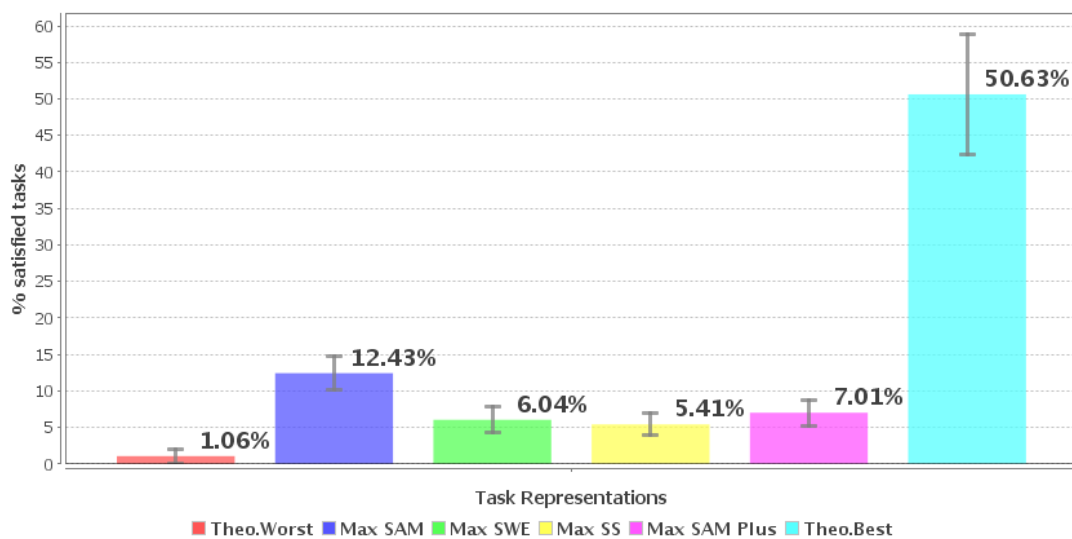
(b) % satisfied tasks overall (mean and standard deviation)

Figure 5.5: Test results for Task LO Platform HI scenario.

performance is even extremely bad. The results are the effect of high changeability of task states. The conclusion from this is obvious, the models are too complex for this sort of scenario. Interestingly, even though all models perform badly, still they in comparison to the Theo.Worst outperform it by at least 5 times. The more interesting might be actually the fact that the Max SAM HTR scored only 4 times lower than the Theo.Best, which is significantly better than other models. The closest to Max SAM



(a) % satisfied tasks by time step



(b) % satisfied tasks overall (mean and standard deviation)

Figure 5.6: Test results for Task HI Platform LO scenario.

in terms of the results was Max SAM Plus.

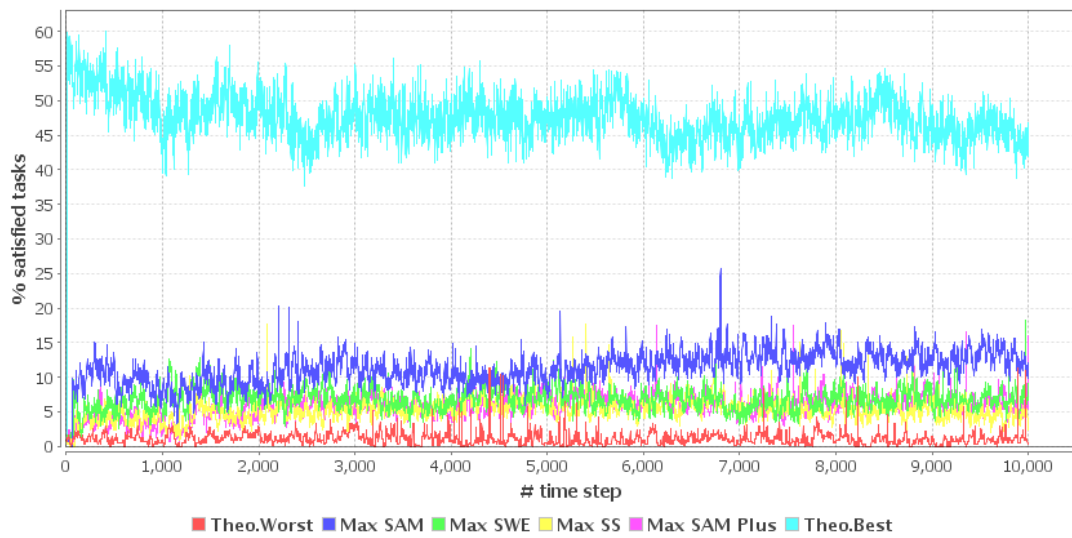
In Figure 5.6, which presents results for scenarios such as border monitoring, we observe that the models are in fact too rich, thus their use takes too long. It means that before we can create a task the task requirements might change several times, effectively that is why we obtained the very low task satisfaction rates. Yet what is worth

saying for the Max SAM is that we have 12.43% which is twice the value of other designed models, i.e. Max SWE 6.04%, Max SS 5.41%, except the Max SAM Plus, in which case it is close to twice its value 7.01%. The value for the worst case model is nearly zero, i.e. 1.06%, where for the best case it is just above 50% (exactly 50.63%). These results show how important it is to find the right model for the right scenario.

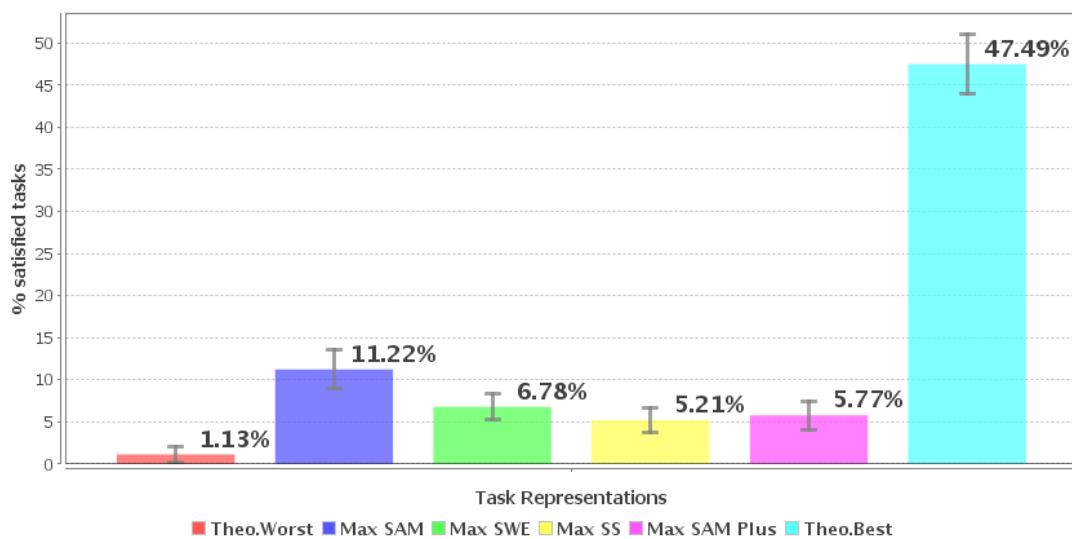
Considering the characteristics of such a scenario we speculate that a more optimal solution could be created. It would involve a system using a low complexity model, but with increased knowledge of predefined tasks. Since the resources are unlikely to change a system could pre-compute all dependencies before a task arrives to the network and store them for later use. This would result in handling of a task using a low complexity model effectively as if it had a rich model. This sort of solution has obvious flaws, for example, it is built for purpose. As such, in case of any change in sensing environment or network resources for which it was built, it would require additional updating of the software of the system in order to adjust to the changes, thus, effectively rendering it useless in certain situations and not applicable in more general situations, e.g. when available sensor types change.

5.3.4 Results for Task HI Platform HI Scenario

The section presents results obtained from experiments performed for the Urban Unrest (Task HI Platform HI) scenario. In this case we observe that all the designed models perform badly. Similarly as it is in the Border Monitoring scenario the Theo.Worst's performance is extremely bad. Just as in the Border Monitoring scenario results are the effect of high changeability of task states. We can say that the models are very susceptible to changes of a task's state as this parameter impacts their performance greatly. These results are also highlighting the importance of selection of appropriate models for scenario specific characteristics. In this case the models are too complex, thus before a simulated user finishes describing his task, new requirements appear which make the old ones obsolete (while it is still counted against the overall satisfaction of tasks).



(a) % satisfied tasks by time step



(b) % satisfied tasks overall (mean and standard deviation)

Figure 5.7: Test results for Task HI Platform HI scenario.

Interestingly, the platform status change rate in this scenario didn't have much effect. It only slightly (by a few percentage) further reduced the overall task satisfaction of the models. The Max SAM in comparison to the other designed models still looks the best.

Figure 5.7 displays results of a scenario such as city policing during a riot. In such a

situation the environment is very highly dynamic on both degrees of freedom. Therefore, as we can observe in Figure 5.6, presenting results of another type of scenario where there is high task change chance, we have very low task satisfaction results for all models. The results are similar for a border monitoring scenario, still Max SAM 11.22% performs better than other three models, i.e. Max SWE 6.78%, Max SS 5.21%, Max SAM Plus 5.77%. We also observe that the dynamicity of such an environment has significant impact on the best case model, i.e. its value reaches slightly below 50%, exactly 47.49%.

In this situation a more optimal solution is not that clear nor easy to imagine, as in other types of scenarios with high task changeability. Since the high likelihood of platform change renders the previous solution useless, potentially its use would return even worse results. This is only a speculation that use of a similar more optimal approach, as in the border monitoring scenario, but still using a rich model, would enhance the obtained results.

5.4 Proposed Future Design of the Experiment

In this section we want to discuss potential for the future changes to the experiment's design. This sort of detail is related to a possible future piece of work but since it is more related to the experiment rather than to the future research we decided to discuss it in this place.

Therefore, if we didn't focus only on the significant parts of the experiment, and decided not to simplify our sensor task assignment in the simulation, the following could take place: sensors' parameters are set accordingly to satisfy the task, e.g. cameras are directed and zoomed as needed. When the sensors are ready to deliver data (task has access to all required sensing instances), the data is collected and forwarded to appropriate processing services. They are discovered in a processing database which serves as a map between the types of tasks to sensor types and to processing service types.

Processing might be simulated, for example, by verification if a required number of input data represents the same event if so then returning true, otherwise false. If the processing is final (i.e. the information is ready to be presented to a user) then it is forwarded to delivery services, otherwise another processing service must be located in a similar way and processing is repeated until it is final. Then in the Dissemination step we check who the task recipients are and deliver them the information in the required form.

Conclusions & Future Work

In this chapter we summarise the work described in the previous chapters and present how the future research might evolve. Section 6.1 summarises our contributions, leading the work towards richer task representation to address the reactivity problem in sensor networks. Section 6.3 discusses our future work, describing where we expect the next steps in the research to take us. We identified at least two related potential research areas to consider in the future. We also discuss how the model of Max SAM Plus HTR could be further enriched in order to improve its usability in scenarios where a task's state can change often, e.g. urban unrest and border monitoring.

6.1 Conclusions

The main contribution of the research, presented throughout the thesis (mostly in Chapter 4), is our methodology and thinking about the problem. It is a systematic approach that enables creation of hybrids of different models of task representations and analysis of their performance in terms of a number of satisfied tasks. It allows for alignment of concepts of different TRs by combining them into a single hybrid offering their combined functionality. The proof and result of our approach are our four HTR ontologies which we created by minimising redundancy among the concepts. We limited overlapping of concepts to those necessary for consistency of a model's ontology. The models were created by mixing the four TRs (SWE, SS, GL and SAM),

where each has their specific approach to the sensor tasking problem. Bound together they offer a more complete solution, from the perspective of information needs of the DCPD loop, i.e. the combinations of TRs capture more information needs than TRs separately. In Chapter 5, we showed the performance evaluation of HTR models created by applying our methodology. The models' performance is tested in relation to the 'best' & 'worst' theoretical models. It shows promising results. We showed that by reusing one of the models (namely Max SAM HTR) and the introduction of additional concepts we obtained a model (Max SAM Plus HTR) capturing all information needs of the DCPD loop, thus allowing maximum flexibility of automatic resource assignment. It increases the number of potential alternative solutions available for each task type.

We see that the more general a task is then the higher is the likelihood that a task will be satisfied. Thus, in order for a system to provide maximum support a user should express his/her task in terms of 'what' he/she wants rather than 'how' he/she wants it. Yet we agree that in an implemented system both options of expression should be available. This is what we have done in our testbed, which was used as a tool for better understanding of the problem. Thus, from an implementation point of view, it is good to leave an option for a user to express not only his specific requirements (e.g. I want a sensor with acoustic capability) but more importantly to give the user an option to state his/her requirements in general terms (e.g. I want to detect a vehicle) while at the same time making him/her aware of benefits that come with the second option, i.e. increased task satisfaction by automatic system's assistance in case of dynamic changes within the network.

Our approach involving representation of a task in the context of the needs of the DCPD loop makes a task well prepared for an automatic use by a system. It is done by capturing all the crucial information needs, required by the steps of the loop to operate, initially at the time of a task instantiation. Thus, further user's intervention becomes no longer needed. Effectively, it allows for the satisfaction of a task as long as there

are resources available that are physically capable to satisfy it (which is obtained from relations captured in a task representation's ontology, e.g. SAM TR's concepts would allow in such situation to obtain alternative solutions every time a task loses its current solution). As a result, a user would observe continuous task satisfaction, as long as it is still possible, while not experiencing the tasking complexities and difficulties as the system would respond to changes. This is in fact the sort of solution that is the most required (appropriate) in dynamic scenarios. Basically, we want a user to be involved as little as possible, particularly in the process of response to changes in the network environment, once he/she states the goals of his/her task.

We believe that our approach is general enough to be applicable for any TR in case a new one was to emerge. Assuming the new TR offers some additional functionality or provides access to a technology beneficial for the problem solution. It would be possible, applying our methodology, to arrive at a hybrid model integrating it with other TRs, enabling a potentially better solution (task satisfaction rate wise) than that of the proposed model of Max SAM Plus HTR.

Our second contribution is creation of the model of Max SAM Plus (Chapter 4) which performs close to the 'ideal' model (in certain scenarios). We showed (in Chapter 5) that it is best for long lasting tasks, where a sensor's status may often change forcing task resource reassignment. Thanks to our model we can derive alternative solutions from its concepts thus automating (in effect minimising) user interactions in presence of changes in the environment which would otherwise cause a task to fail, for the duration of the user's interaction.

We see the HTR as supporting a higher degree of automation in a sensor tasking system than is currently possible. This involves a high-level specification of tasks, sensor selection, composition of chains of processing services and delivery of information satisfying a task to the right consumers. Having rich information captured in the HTR means that a system can make appropriate decisions (without asking the user) and se-

lect appropriate sensor resources for the user's tasks; moreover, the HTR enables finding of substitutions for sensors if the previous choices become unavailable, or other changes occur e.g. a user adds more requirements to his/her task. This is especially important in highly dynamic domains, such as, emergency response or the military operations.

Another of our contributions is creation of ontologies for the four identified TRs. It was a result of our initial exploration of the concept of task representation in sensor networks. The analysis of the current approaches identified four TRs, implemented in separate systems, where each addressed different aspects of the sensor tasking problem. Explicating their concepts in the form of ontologies enabled us to identify mappings between the TRs, allowing the creation of hybrid task representations that had their combined features. This contribution (described in Chapter 3) to sensor networks domain of science lets us treat the concepts according to their semantic meaning. For example, we used the TRs in our research to obtain richer models combining their concepts to offer full(er) coverage of the DCPD steps. This approach allows us to observe relations between the existing technologies, which the TRs represent, allowing for mixing of the capabilities provided by the independent models. As a result, giving a richer model the ability to handle the reactivity problem in dynamic sensors networks performing close to the 'ideal' case model, as we showed in Chapter 5.

Furthermore, we would like to repeat that having an open, explicit task representation that addresses both user- and sensor-level tasking also allows the possibility for the creation of new sensor planning, operation, and delivery services and tools that use the existing HTR, or further extend it with additional features.

6.2 Instantiation of Max SAM Plus HTR Model

In this section on example of the Max SAM Plus HTR we will show how OWL might be used to instantiate a HTR model. The instance of the model which we create here with help of OWL might be used by a system to control its operation.

In order to instantiate a model we need to create instances of appropriate classes of which the model consists and which are needed by the task for which we are instantiating the model. To show how then the instance relates to TRs it integrates, we will use example from car detection domain, which we call Capture Speeding Car Image Task. The goal of this task is to obtain images of cars which break a speed limit.

In our scenario, as presented on Fig. 6.1 (for RDF/XML please see instances section of Max SAM Plus HTR in Appendix B), we have a police officer and a traffic analyst, two beneficiaries who are interested in a solution for this task. The analyst just needs statistics, whereas the officer always needs car pictures independently of which solution is used. We see that there are two potential options available in the region of the task. One involving just speed camera, the other requiring (at least) two break beams to detect a speeding car and a camera to capture its image. The data coming from the sensors is accordingly combined by semantic services which enrich it with more and more semantic information at the end returning the final stream with images of speeding cars, in other words our task. We note that preparation of a solution for the analyst would involve an additional service counting the images taken over time, also before using the sensors would very likely require appropriate setting of their parameters, but for brevity we didn't put it. For the same reason we chose not to put weights on the static and dynamic goals (respectively tasks and sensors), which in this configuration would favour speed camera since it is the best fit for this task.

The TRs which the HTR contains would take details needed for setting of their parameters respectively: Initially the process should use the model's ontology to validate an instance then if it is valid, GL would use weights and hasSubTask relations to form

a goal lattice and prioritise sensor assignment effort. SAM would set task's details including required asset types. SWE would find and set parameters and collect data from currently available sensors. SS would process the sensor data combining it as presented on the figure. Additionally, in the situation that the current solution for the Detect Speeding Car (sub) Task (as only this task has an alternative rule declared) became unavailable, SAM would use its ontology and reason on it to recognise alternatives.

6.3 Future Work

Part of our further work will certainly involve further enrichment of the Max SAM Plus HTR with a richer knowledge base, in order to simplify the description complexity of its steps. In effect we believe, as we speculated in the results section of Chapter 5, it will decrease the gap between our model and the 'ideal' one. This will result in increased robustness of the HTR to changes in a task's state, thus improving task satisfaction performance of the model.

Implementing a system based on the Hybrid Task Representation would allow integration of several useful mechanisms: goal prioritisation using goal lattices; matching of task satisfying sensor types through SAM; use of sensor instances, setting of their parameters for a task using SWE; and processing of sensor data by SS. Such a system would provide a significantly more complete solution to the sensor tasking problem than currently exists.

To demonstrate the utility of a Hybrid Task Representation such as the one presented in Section 4.5 (Max SAM Plus HTR), we are implementing a framework application that serves as the system shown in Figure 1.1. This follows work previously presented in [62]. Our approach follows the principle that many tasking details, especially those related to the sensor-level tasking (e.g. selection and setting of parameters for an asset) should be inferred where possible from higher-level descriptions of a task (i.e. in terms of Section 1.1: "what", not "how"). However, we don't want to prevent users from

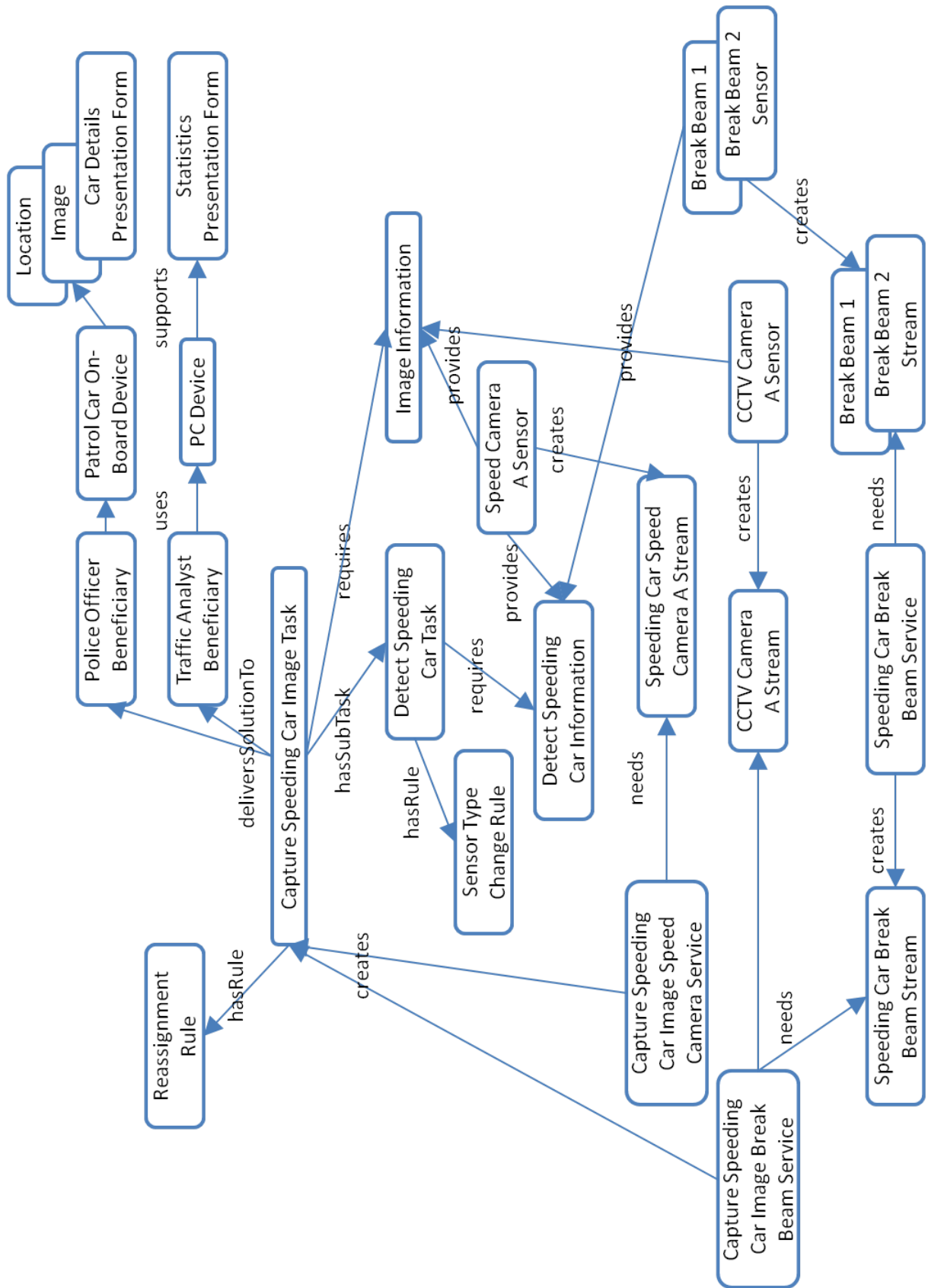


Figure 6.1: Max SAM Plus HTR instantiated for Capture Speeding Car Image Task

choosing the types of solution they are interested in (i.e. “how”) because there may be situations where a user prefers one kind of solution over another. For example, a vehicle detection task may be solvable by acoustic and imagery means, but the user is really interested only in the imagery solution as they can then use the images of detected vehicles as a proof that they breached a speed limit and penalise their drivers.

For this process we will use the technologies of the TRs, captured within the Max SAM Plus HTR model, in the result achieving a complete approach to tasking on both levels, that allows for flexibility in a task description. The upgrade will involve: integration of the HTR into the framework; substitution of the OGC-SWE-like sensor network layer by the OGC-SWE compliant layer; and use of the SS technology which provides sensor data processing, thus, obsoleting the current fixed processing.

Thus, we expect that one of the future goals will involve implementation and integration of the Max SAM Plus HTR model into the current system. While doing so we will remember to make it general enough to avoid binding the system exclusively to a single HTR model. We think that having one ultimate model that would fit all scenarios is not realistic. For proof of this statement, see the results in Section 5.3. There we see that the ‘pure’ Max SAM HTR outperforms others in scenarios where there is high probability of task change, when Max SAM Plus HTR appears to be better fitted for scenarios where platform probability of change is high.

This approach would be a preparation for our next goal which is to allow use of different models in a single system. For this we would need to research the implemented work of the system’s domain further. This way we aim to enable creation of a system that is ready to maximise task satisfaction even further by choosing an adequate model of task representation depending on the environment of operation. This might even involve the temporary characteristics of the environment’s state. This way a system would be able to better assist a user, if the environment was to evolve.

The further stage of the research, as we think, would involve a fine grained form of

optimisation of such a system. We want to form, what we currently call, an approach to a system's implementation supporting adaptive selection of a TR model. It means that we want to create a systematic approach to system modeling that would allow use of many models in a single system. The part that distinguishes it from our second goal is that we want to allow for simultaneous and dynamically evolving use of different models for a task.

The approach of adaptive model selection would enable mixing of models, for example, while initially starting task specification using a 'poor' model at the same time mapping it to a 'rich' one. Thus, having at the start, very few details given about a task a system could, almost synchronously to a user's actions, adapt as the further details are given by the user. Another example could involve a system to assume at first that the only consumer is the creator of a task but still allow the user to specify these details later, then the system would change the model to one capable of it. Thus we think the input (from user's point of view) would somehow have to be prepared to capture the richest model's details.

Effectively, a system would, depending on a particular input in time, choose an adequate model that is 'rich' enough for the data that was input. Such an approach would allow for continuous dynamic satisfaction of a task from its very creation throughout all stages of its evolution, while making the best use of available models.

We would also like to make an additional proof of our approach and methodology. It will be possible when new TRs emerge. It would allow us to further prove our claim about its universality. We believe that we managed to achieve it, yet the only proof we have are our models of HTRs, particularly promising is the Max SAM Plus HTR which captures all the DCPD loop's information needs, its performance was shown in Chapter 5.

We hope to find more TRs that we could use in our future work. One of our future directions assumes that having more models available to use is actually beneficial for

a system, if it is able to effectively use them for managing the tasks. It is important for our adaptive approach to use the model of task representation appropriate for a temporal state of environment. This is enabling implementation of a system that would evolve a task model together with the change of a task's details. We mentioned in the section above that this would further improve a task's satisfaction optimising the system's performance.

The hypothesis capturing the essence of our future research could sound like:

“By having a rich enough model of task representation allowing the capture of all requirements of a particular task, in relation to the temporal characteristics of an environment, optimises the overall satisfaction of tasks in a sensor network by maximisation of a system's assistance role, providing dynamic support for a user creating his/her task”.

Ontologies of Task Representations

Here we show class hierarchy and ontology for each of the TR models.

A.1 SWE TR Ontology

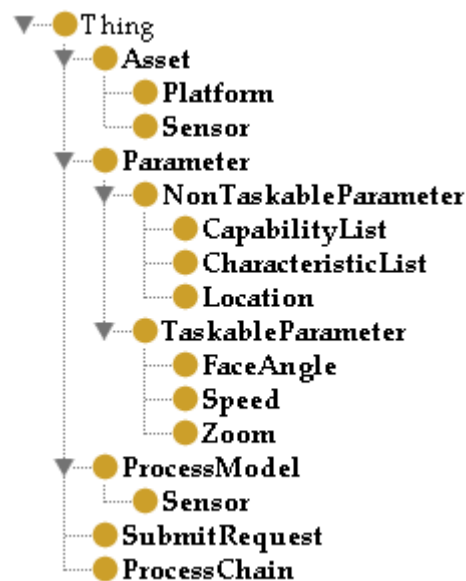


Figure A.1: SWE TR class hierarchy

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <ENTITY dc "http://purl.org/dc/elements/1.1/" >
  <ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <ENTITY owlxml "http://www.w3.org/2006/12/owl2-xml#" >

```

```

<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY OGC "http://www.owl-ontologies.com/OGC.owl#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY SWETROntology "http://www.owl-ontologies.com/SWETROntology.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/OGC.owl#"
  xml:base="http://www.owl-ontologies.com/SWETROntology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:OGC="http://www.owl-ontologies.com/OGC.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:SWETROntology="http://www.owl-ontologies.com/SWETROntology.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
<owl:Ontology rdf:about="">
  <rdfs:comment
    >Created by Konrad Borowiecki 2010</rdfs:comment>
</owl:Ontology>
<!--
//////////////////////////////////////

//
// Annotation properties
//
//////////////////////////////////////

-->
<owl:AnnotationProperty rdf:about="&dc;title"/>
<!--
//////////////////////////////////////

//
// Object Properties
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/OGC.owl#carries -->
<owl:ObjectProperty rdf:about="&OGC;carries">
  <rdfs:domain rdf:resource="&OGC;Platform"/>
  <rdfs:range rdf:resource="&OGC;Sensor"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/OGC.owl#hasAsset -->

```

```

<owl:ObjectProperty rdf:about="&OGC;hasAsset">
  <rdfs:range rdf:resource="&OGC;Asset"/>
  <rdfs:domain rdf:resource="&OGC;SubmitRequest"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/OGC.owl#hasInput -->
<owl:ObjectProperty rdf:about="&OGC;hasInput">
  <rdfs:domain rdf:resource="&OGC;ProcessModel"/>
  <rdfs:range rdf:resource="&OGC;ProcessModel"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/OGC.owl#hasOutput -->
<owl:ObjectProperty rdf:about="&OGC;hasOutput">
  <rdfs:domain rdf:resource="&OGC;ProcessModel"/>
  <rdfs:range rdf:resource="&OGC;ProcessModel"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/OGC.owl#hasParameter -->
<owl:ObjectProperty rdf:about="&OGC;hasParameter">
  <rdfs:domain rdf:resource="&OGC;Asset"/>
  <rdfs:range rdf:resource="&OGC;Parameter"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/SWETROntology.owl#consistsOf -->
<owl:ObjectProperty rdf:about="#consistsOf">
  <rdfs:domain rdf:resource="#ProcessChain"/>
</owl:ObjectProperty>
<!--
////////////////////////////////////

//
// Data properties
//
////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/OGC.owl#capabilityName -->
<owl:DatatypeProperty rdf:about="&OGC;capabilityName">
  <rdfs:domain rdf:resource="&OGC;CapabilityList"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>
<!-- http://www.owl-ontologies.com/OGC.owl#latitude -->
<owl:DatatypeProperty rdf:about="&OGC;latitude">
  <rdfs:domain rdf:resource="&OGC;Location"/>
  <rdfs:range rdf:resource="&xsd:double"/>
</owl:DatatypeProperty>
<!-- http://www.owl-ontologies.com/OGC.owl#longitude -->
<owl:DatatypeProperty rdf:about="&OGC;longitude">

```



```

    <rdfs:domain rdf:resource="&OGC;Location"/>
    <rdfs:range rdf:resource="&xsd;double"/>
</owl:DatatypeProperty>
<!-- http://www.owl-ontologies.com/OGC.owl#value -->
<owl:DatatypeProperty rdf:about="&OGC;value">
    <rdfs:domain rdf:resource="&OGC;Speed"/>
    <rdfs:range rdf:resource="&xsd;double"/>
</owl:DatatypeProperty>
<!--
//////////////////////////////////////

//
// Classes
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/OGC.owl#Asset -->
<owl:Class rdf:about="&OGC;Asset">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&OGC;hasParameter"/>
            <owl:someValuesFrom rdf:resource="&OGC;Parameter"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="&OGC;Parameter"/>
    <owl:disjointWith rdf:resource="&OGC;SubmitRequest"/>
    <owl:disjointWith rdf:resource="#ProcessChain"/>
    <rdfs:comment
        >This might be a platform or a sensor. Eventhough they use asset as a
        synonym of a sensor or simulation, they mention throughout the papers
        and service specifications e.g. in the SPS specification version 1.0
        they say that it is intended to task platforms and sensors, so I
        would say that they have forgotten to update the definition of an
        asset to include platforms.</rdfs:comment>
    </owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#CapabilityList -->
<owl:Class rdf:about="&OGC;CapabilityList">
    <rdfs:subClassOf rdf:resource="&OGC;NonTaskableParameter"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&OGC;capabilityName"/>
            <owl:someValuesFrom rdf:resource="&xsd:string"/>
        </owl:Restriction>

```

```

</rdfs:subClassOf>
<rdfs:comment
  >These are the properties that are measurement connected. They express
    ability of a sensor to take measures, e.g. type of a sensor IR camera
    , as well as properties dependent on an instance as e.g. zoom range
    of a camera.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#CharacteristicList -->
<owl:Class rdf:about="&OGC;CharacteristicList">
  <rdfs:subClassOf rdf:resource="&OGC;NonTaskableParameter"/>
  <rdfs:comment
    >Parameters that are describing physical properties of a sensor, e.g. its
      size, weight, etc.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#FaceAngle -->
<owl:Class rdf:about="&OGC;FaceAngle">
  <rdfs:subClassOf rdf:resource="&OGC;TaskableParameter"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&OGC;value"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
        owl:qualifiedCardinality>
      <owl:onDataRange rdf:resource="&xsd;double"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Location -->
<owl:Class rdf:about="&OGC;Location">
  <rdfs:subClassOf rdf:resource="&OGC;NonTaskableParameter"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&OGC;latitude"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
        owl:qualifiedCardinality>
      <owl:onDataRange rdf:resource="&xsd;double"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&OGC;longitude"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
        owl:qualifiedCardinality>
      <owl:onDataRange rdf:resource="&xsd;double"/>
    </owl:Restriction>
  </rdfs:subClassOf>

```

```

    </rdfs:subClassOf>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#NonTaskableParameter -->
<owl:Class rdf:about="&OGC;NonTaskableParameter">
  <rdfs:subClassOf rdf:resource="&OGC;Parameter"/>
  <owl:disjointWith rdf:resource="&OGC;TaskableParameter"/>
  <rdfs:comment
    >They depend on an asset instance. This class represents all paramteres of
      an asset that might not be tasked. e.g. geographic position of a
      none mobile asset.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Parameter -->
<owl:Class rdf:about="&OGC;Parameter">
  <owl:disjointWith rdf:resource="&OGC;ProcessModel"/>
  <owl:disjointWith rdf:resource="&OGC;SubmitRequest"/>
  <owl:disjointWith rdf:resource="#ProcessChain"/>
  <rdfs:comment
    >It describes the parameters required to taks an asset. e.g. angle to
      rotate a camera, zoom level to zoom it, it also might include a
      period of time the asset is requested for. The parameters set depends
      on a asset instance.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Platform -->
<owl:Class rdf:about="&OGC;Platform">
  <rdfs:subClassOf rdf:resource="&OGC;Asset"/>
  <rdfs:subClassOf
    <owl:Restriction >
      <owl:onProperty rdf:resource="&OGC;carries"/>
      <owl:allValuesFrom rdf:resource="&OGC;Sensor"/>
    </owl:Restriction >
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="&OGC;Sensor"/>
  <dc:title
    >This represents a platform on which a sensor might be installed, e.g. a
      vehicle, meterological station.</dc:title >
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#ProcessModel -->
<owl:Class rdf:about="&OGC;ProcessModel">
  <rdfs:subClassOf
    <owl:Restriction >
      <owl:onProperty rdf:resource="&OGC;hasOutput"/>
      <owl:someValuesFrom rdf:resource="&OGC;ProcessModel"/>
    </owl:Restriction >
  </rdfs:subClassOf>

```

```

    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&OGC;hasParameter"/>
        <owl:someValuesFrom rdf:resource="&OGC;Parameter"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&OGC;hasInput"/>
      <owl:someValuesFrom rdf:resource="&OGC;ProcessModel"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="&OGC;SubmitRequest"/>
  <owl:disjointWith rdf:resource="#ProcessChain"/>
  <rdfs:comment
    >Represents a process that performs transformation and processing of
      sensor information.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Sensor -->
<owl:Class rdf:about="&OGC;Sensor">
  <rdfs:subClassOf rdf:resource="&OGC;Asset"/>
  <rdfs:subClassOf rdf:resource="&OGC;ProcessModel"/>
  <dc:title
    >This represents a sensor (whether physical or simulated). In SensorML,
      sensors are modelled as processes that &quot;convert real
phenomena to data&quot;;, therefore, a Sensor is a specific type of Process Model.</dc:
  title >
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Speed -->
<owl:Class rdf:about="&OGC;Speed">
  <rdfs:subClassOf rdf:resource="&OGC;TaskableParameter"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&OGC;value"/>
      <owl:qualifiedCardinality rdf:datatype="xsd:nonNegativeInteger">1</
        owl:qualifiedCardinality >
      <owl:onDataRange rdf:resource="xsd:double"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#SubmitRequest -->
<owl:Class rdf:about="&OGC;SubmitRequest">
  <rdfs:subClassOf>
    <owl:Restriction>

```

```

        <owl:onProperty rdf:resource="#OGC;hasAsset"/>
        <owl:someValuesFrom rdf:resource="#OGC;Asset"/>
    </owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="#ProcessChain"/>
<rdfs:comment rdf:datatype="&xsd:string"
    >A request for assets sent by user to Sensor Planning Service (SPS), which
        then validates it, if successful then it gets the resources that are
        satisfying that request.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#TaskableParameter -->
<owl:Class rdf:about="#OGC;TaskableParameter">
    <rdfs:subClassOf rdf:resource="#OGC;Parameter"/>
    <rdfs:comment
        >They depend on an asset instance. This class represents all paramteres of
            an asset that might be tasked. e.g. zoom level of a camera.</rdfs:
            comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Zoom -->
<owl:Class rdf:about="#OGC;Zoom">
    <rdfs:subClassOf rdf:resource="#OGC;TaskableParameter"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#OGC;value"/>
            <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
                owl:qualifiedCardinality>
            <owl:onDataRange rdf:resource="#OGC;double"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<!-- http://www.owl-ontologies.com/SWETROntology.owl#ProcessChain -->
<owl:Class rdf:about="#ProcessChain">
    <rdfs:subClassOf>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#consistsOf"/>
                            <owl:onClass rdf:resource="#OGC;ProcessModel"/>
                            <owl:minQualifiedCardinality rdf:datatype="&xsd;
                                nonNegativeInteger">1</owl:
                                minQualifiedCardinality>
                        </owl:Restriction>
                    </owl:intersectionOf>
                </owl:Class>
            </owl:unionOf>
        </owl:Class>
    </rdfs:subClassOf>

```

```

        <owl:Restriction>
            <owl:onProperty rdf:resource="#consistsOf"/>
            <owl:onClass rdf:resource="#ProcessChain"/>
            <owl:minQualifiedCardinality rdf:datatype="&xsd;
                nonNegativeInteger">1</owl:
                minQualifiedCardinality>
        </owl:Restriction>
    </owl:intersectionOf>
</owl:Class>
<owl:Restriction>
    <owl:onProperty rdf:resource="#consistsOf"/>
    <owl:onClass rdf:resource="&OGC;ProcessModel"/>
    <owl:minQualifiedCardinality rdf:datatype="&xsd;
        nonNegativeInteger">2</owl:minQualifiedCardinality>
</owl:Restriction>
</owl:unionOf>
</owl:Class>
</rdfs:subClassOf>
<rdfs:comment
    >It describes a structural block which consists of at least 2 process
        models or other process chains and process models.</rdfs:comment>
</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing -->
<owl:Class rdf:about="&owl;Thing"/>
<!--
//////////////////////////////////////

//
// Individuals
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/OGC.owl#CameraA -->
<owl:Thing rdf:about="&OGC;CameraA">
    <rdf:type rdf:resource="&OGC;Sensor"/>
    <hasParameter rdf:resource="&OGC;CameraA_CapabilityList"/>
    <hasParameter rdf:resource="&OGC;CameraA_Zoom"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/OGC.owl#CameraA_CapabilityList -->
<owl:Thing rdf:about="&OGC;CameraA_CapabilityList">
    <rdf:type rdf:resource="&OGC;CapabilityList"/>
    <capabilityName rdf:datatype="&xsd:string">IMINT</capabilityName>
    <capabilityName rdf:datatype="&xsd:string">IRINT</capabilityName>

```

```

</owl:Thing>
<!-- http://www.owl-ontologies.com/OGC.owl#CameraA_Zoom -->
<Zoom rdf:about="&OGC;CameraA_Zoom">
  <rdf:type rdf:resource="&owl;Thing"/>
  <value rdf:datatype="&xsd;double">6.0</value>
</Zoom>
<!-- http://www.owl-ontologies.com/OGC.owl#PredatorA -->
<Platform rdf:about="&OGC;PredatorA">
  <rdf:type rdf:resource="&owl;Thing"/>
  <carries rdf:resource="&OGC;CameraA"/>
  <hasParameter rdf:resource="&OGC;PredatorA_FaceAngle"/>
  <hasParameter rdf:resource="&OGC;PredatorA_Location"/>
  <hasParameter rdf:resource="&OGC;PredatorA_Speed"/>
</Platform>
<!-- http://www.owl-ontologies.com/OGC.owl#PredatorA_FaceAngle -->
<FaceAngle rdf:about="&OGC;PredatorA_FaceAngle">
  <rdf:type rdf:resource="&owl;Thing"/>
  <value rdf:datatype="&xsd;double">90.0</value>
</FaceAngle>
<!-- http://www.owl-ontologies.com/OGC.owl#PredatorA_Location -->
<owl:Thing rdf:about="&OGC;PredatorA_Location">
  <rdf:type rdf:resource="&OGC;Location"/>
  <longitude rdf:datatype="&xsd;double">0.5554</longitude>
  <latitude rdf:datatype="&xsd;double">10.0001</latitude>
</owl:Thing>
<!-- http://www.owl-ontologies.com/OGC.owl#PredatorA_Speed -->
<owl:Thing rdf:about="&OGC;PredatorA_Speed">
  <rdf:type rdf:resource="&OGC;Speed"/>
  <value rdf:datatype="&xsd;double">500.0</value>
</owl:Thing>
<!-- http://www.owl-ontologies.com/OGC.owl#PredatorAandCameraA_Tasking -->
<owl:Thing rdf:about="&OGC;PredatorAandCameraA_Tasking">
  <rdf:type rdf:resource="&OGC;SubmitRequest"/>
  <hasAsset rdf:resource="&OGC;CameraA"/>
  <hasAsset rdf:resource="&OGC;PredatorA"/>
</owl:Thing>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->

```

Listing A.1: The SWE TR ontology in RDF/XML format

A.2 SS TR Ontology

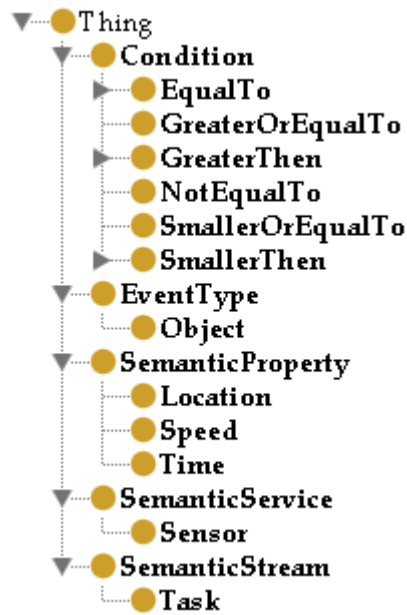


Figure A.2: SS TR class hierarchy

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY SS "http://www.owl-ontologies.com/SS.owl#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY SS2 "http://www.owl-ontologies.com/SS.owl#100" >
  <!ENTITY km "http://www.owl-ontologies.com/SS.owl#100km/" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY CarBreaching100km "http://www.owl-ontologies.com/SS.owl#CarBreaching100km/" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/SS.owl#"
  xml:base="http://www.owl-ontologies.com/SSTRontology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:km="&SS;100km/"
  xmlns:SS2="&SS;100"
  xmlns:CarBreaching100km="&SS;CarBreaching100km/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:SS="http://www.owl-ontologies.com/SS.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

```



```

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#">
<owl:Ontology rdf:about="">
  <rdfs:comment
    >Created by Konrad Borowiecki 2010</rdfs:comment>
</owl:Ontology>
<!--
//////////////////////////////////////

//
// Object Properties
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/SS.owl#creates -->
<owl:ObjectProperty rdf:about="&SS;creates">
  <rdfs:comment rdf:datatype="&xsd:string"
    >It tells what is the output of an inference unit. It tells what will the
      resulting event stream look like, and what will be its properties,
      after operations has been applied by the inference unit.</rdfs:
      comment>
  <rdfs:domain rdf:resource="&SS;SemanticService"/>
  <rdfs:range rdf:resource="&SS;SemanticStream"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/SS.owl#hasCondition -->
<owl:ObjectProperty rdf:about="&SS;hasCondition">
  <rdfs:range rdf:resource="&SS;Condition"/>
  <rdfs:domain rdf:resource="&SS;SemanticProperty"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/SS.owl#hasDetected -->
<owl:ObjectProperty rdf:about="&SS;hasDetected">
  <rdfs:comment
    >Property used by DetectionEventType to mark that it detects an individual
      of the Object class.</rdfs:comment>
  <rdfs:domain rdf:resource="&SS;EventType"/>
  <rdfs:range rdf:resource="&SS;Object"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/SS.owl#hasEventType -->
<owl:ObjectProperty rdf:about="&SS;hasEventType">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:comment rdf:datatype="&xsd:string"
    >This property describes an event stream and it tells what type of event
      it streams.</rdfs:comment>

```

```

    <rdfs:range rdf:resource="&SS;EventType"/>
    <rdfs:domain rdf:resource="&SS;SemanticStream"/>
  </owl:ObjectProperty>
  <!-- http://www.owl-ontologies.com/SS.owl#hasSemanticProperty -->
  <owl:ObjectProperty rdf:about="&SS;hasSemanticProperty">
    <rdfs:comment rdf:datatype="&xsd:string">
      >This property describes a detectable. It tells what are its constraints
        and charactersitics, where it ocured, etc.</rdfs:comment>
    <rdfs:range rdf:resource="&SS;SemanticProperty"/>
    <rdfs:domain rdf:resource="&SS;SemanticStream"/>
  </owl:ObjectProperty>
  <!-- http://www.owl-ontologies.com/SS.owl#needs -->
  <owl:ObjectProperty rdf:about="&SS;needs">
    <rdfs:comment rdf:datatype="&xsd:string">
      >It tells what type of an event stream, and what properties it must have
        in order to use the inference unit. This describes input of an
        inference unit.</rdfs:comment>
    <rdfs:domain rdf:resource="&SS;SemanticService"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="&SS;SemanticStream"/>
          <rdf:Description rdf:about="&SS;Sensor"/>
        </owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="&SS;SemanticStream"/>
          <rdf:Description rdf:about="&SS;Sensor"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:unionOf>
  </owl:Class>
</rdfs:range>
</owl:ObjectProperty>
<!--
//////////////////////////////////////

//
// Data properties
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/SS.owl#conditionValue -->
<owl:DatatypeProperty rdf:about="&SS;conditionValue">

```

```

    <rdfs:comment
      >This property is used by the Condition class , and its stores a right side
        value of a condition.</rdfs:comment>
    <rdfs:domain rdf:resource="&SS;Condition"/>
    <rdfs:range rdf:resource="&xsd;double"/>
  </owl:DatatypeProperty >
<!--
//////////////////////////////////////

//
// Classes
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/SS.owl#Condition -->
<owl:Class rdf:about="&SS;Condition">
  <rdfs:subClassOf>
    <owl:Restriction >
      <owl:onProperty rdf:resource="&SS;conditionValue"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
        owl:qualifiedCardinality >
      <owl:onDataRange rdf:resource="&xsd;double"/>
    </owl:Restriction >
  </rdfs:subClassOf>
  <rdfs:comment
    >It represents , a conditional statement that might be placed upon a
      semsntic property . Often it is used to restrict a renge of the input
      event stream that enters an inference unit , e.g. property(S, V,
      speed), {V > 100 km/h}, read it as the speed of observed object is
      greater then 100 km/h.</rdfs:comment>
  </owl:Class >
<!-- http://www.owl-ontologies.com/SS.owl#EqualTo -->
<owl:Class rdf:about="&SS;EqualTo">
  <rdfs:subClassOf rdf:resource="&SS;Condition"/>
</owl:Class >
<!-- http://www.owl-ontologies.com/SS.owl#EventType -->
<owl:Class rdf:about="&SS;EventType">
  <rdfs:subClassOf>
    <owl:Restriction >
      <owl:onProperty rdf:resource="&SS;hasSemanticProperty"/>
      <owl:allValuesFrom rdf:resource="&SS;SemanticProperty"/>
    </owl:Restriction >
  </rdfs:subClassOf>

```

```

    <rdfs:comment
      >It describes what type of event a stream carries , e.g. it might be a
        detection of an object.</rdfs:comment>
  </owl:Class>
  <!-- http://www.owl-ontologies.com/SS.owl#GreaterOrEqualTo -->
  <owl:Class rdf:about="&SS;GreaterOrEqualTo">
    <rdfs:subClassOf rdf:resource="&SS;Condition"/>
    <rdfs:subClassOf rdf:resource="&SS;EqualTo"/>
    <rdfs:subClassOf rdf:resource="&SS;GreaterThen"/>
  </owl:Class>
  <!-- http://www.owl-ontologies.com/SS.owl#GreaterThen -->
  <owl:Class rdf:about="&SS;GreaterThen">
    <rdfs:subClassOf rdf:resource="&SS;Condition"/>
  </owl:Class>
  <!-- http://www.owl-ontologies.com/SS.owl#Location -->
  <owl:Class rdf:about="&SS;Location">
    <rdfs:subClassOf rdf:resource="&SS;SemanticProperty"/>
    <rdfs:comment rdf:datatype="&xsd:string"
      >Location where the event has occured.</rdfs:comment>
  </owl:Class>
  <!-- http://www.owl-ontologies.com/SS.owl#NotEqualTo -->
  <owl:Class rdf:about="&SS;NotEqualTo">
    <rdfs:subClassOf rdf:resource="&SS;Condition"/>
  </owl:Class>
  <!-- http://www.owl-ontologies.com/SS.owl#Object -->
  <owl:Class rdf:about="&SS;Object">
    <rdfs:subClassOf rdf:resource="&SS;EventType"/>
  </owl:Class>
  <!-- http://www.owl-ontologies.com/SS.owl#SemanticProperty -->
  <owl:Class rdf:about="&SS;SemanticProperty">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&SS;hasCondition"/>
        <owl:allValuesFrom rdf:resource="&SS;Condition"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="&xsd:string"
      >(aka Constraint) It describes properties of an event stream such as where
        it has happened (location), and what was it (e.g. what was detected),
        and other detection details e.g. quality of information (QoI) [SS
        authors are not using the words QoI, but in an exaple they give as
        one of properties &quot;speed estimate greater then 90%&quot;).</rdfs
        :comment>
  </owl:Class>

```

```

<!-- http://www.owl-ontologies.com/SS.owl#SemanticService -->
<owl:Class rdf:about="&SS;SemanticService">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&SS;needs"/>
      <owl:allValuesFrom rdf:resource="&SS;SemanticStream"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&SS;creates"/>
      <owl:onClass rdf:resource="&SS;SemanticStream"/>
      <owl:minQualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger"
        >1</owl:minQualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:versionInfo rdf:datatype="&xsd:string"
    >(in older papers referred as Inference Unit) This is a process that
      operates on event streams. It infers semantic information about the
      world and incorporate it into an event stream. e.g. establishing a
      type of an object, i.e. checking if this event was caused by a car.</
    owl:versionInfo>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticStream -->
<owl:Class rdf:about="&SS;SemanticStream">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&SS;hasEventType"/>
      <owl:onClass rdf:resource="&SS;EventType"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
        owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="&xsd:string"
    >It represents a flow of asynchronous events, each of which represents a
      world event such as object, person, or car detection and has
      properties (constraints) such as time or location it was detected,
      its speed, direction, and/or identity. [definition taken from their
      paper]</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Sensor -->
<owl:Class rdf:about="&SS;Sensor">
  <rdfs:subClassOf rdf:resource="&SS;SemanticService"/>
  <rdfs:subClassOf>

```

```

    <owl:Restriction >
      <owl:onProperty rdf:resource="&SS;needs"/>
      <owl:onClass rdf:resource="&SS;SemanticStream"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
        owl:qualifiedCardinality >
    </owl:Restriction >
  </rdfs:subClassOf>
  <rdfs:comment
    >It is a special type of an inference unit that as input might only have
    sensors.</rdfs:comment>
  <rdfs:comment>A sensor.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SmallerOrEqualTo -->
<owl:Class rdf:about="&SS;SmallerOrEqualTo">
  <rdfs:subClassOf rdf:resource="&SS;Condition"/>
  <rdfs:subClassOf rdf:resource="&SS;EqualTo"/>
  <rdfs:subClassOf rdf:resource="&SS;SmallerThen"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SmallerThen -->
<owl:Class rdf:about="&SS;SmallerThen">
  <rdfs:subClassOf rdf:resource="&SS;Condition"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Speed -->
<owl:Class rdf:about="&SS;Speed">
  <rdfs:subClassOf rdf:resource="&SS;SemanticProperty"/>
  <rdfs:comment
    >Speed e.g. of a vehicle.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Task -->
<owl:Class rdf:about="&SS;Task">
  <rdfs:subClassOf rdf:resource="&SS;SemanticStream"/>
  <rdfs:comment
    >A Task is a specific kind of Semantic Stream which is not an input into
    any Semantic Service; in other words, it is a final output stream
    which provides the information specified in a user's query.</rdfs
    :comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Time -->
<owl:Class rdf:about="&SS;Time">
  <rdfs:subClassOf rdf:resource="&SS;SemanticProperty"/>
  <rdfs:comment
    >Time when an event occurred.</rdfs:comment>
</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing -->

```

```

<owl:Class rdf:about="&owl;Thing"/>
<!--
//////////////////////////////////////

//
// Individuals
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/SS.owl#100km/hSpeedLimitBreach_SemanticProperty
-->
<owl:Thing rdf:about="&SS;100km/hSpeedLimitBreach_SemanticProperty">
  <rdf:type rdf:resource="&SS;Speed"/>
  <hasCondition rdf:resource="&SS;ValueGreaterThan100_Condition"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/SS.owl#BreakBeamA_SemanticStream -->
<owl:Thing rdf:about="&SS;BreakBeamA_SemanticStream">
  <rdf:type rdf:resource="&SS;SemanticStream"/>
  <hasEventType rdf:resource="&SS;ObjectDetection_EventType"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/SS.owl#BreakBeamA_Sensor -->
<owl:Thing rdf:about="&SS;BreakBeamA_Sensor">
  <rdf:type rdf:resource="&SS;Sensor"/>
  <creates rdf:resource="&SS;BreakBeamA_SemanticStream"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/SS.owl#BreakBeamB_SemanticStream -->
<owl:Thing rdf:about="&SS;BreakBeamB_SemanticStream">
  <rdf:type rdf:resource="&SS;SemanticStream"/>
  <hasEventType rdf:resource="&SS;ObjectDetection_EventType"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/SS.owl#BreakBeamB_Sensor -->
<Sensor rdf:about="&SS;BreakBeamB_Sensor">
  <rdf:type rdf:resource="&owl;Thing"/>
  <creates rdf:resource="&SS;BreakBeamB_SemanticStream"/>
</Sensor>
<!-- http://www.owl-ontologies.com/SS.owl#CarBreaching100km/
hSpeedLimitDetection_SemanticService -->
<owl:Thing rdf:about="&SS;CarBreaching100km/hSpeedLimitDetection_SemanticService">
  <rdf:type rdf:resource="&SS;SemanticService"/>
  <needs rdf:resource="&SS;BreakBeamA_SemanticStream"/>
  <needs rdf:resource="&SS;BreakBeamB_SemanticStream"/>
  <creates rdf:resource="&SS;CarBreaching100km/hSpeedLimit_Task"/>
</owl:Thing>

```

```

<!-- http://www.owl-ontologies.com/SS.owl#CarBreaching100km/hSpeedLimit_Task -->
<Task rdf:about="&SS;CarBreaching100km/hSpeedLimit_Task">
  <rdf:type rdf:resource="&owl;Thing"/>
  <hasSemanticProperty rdf:resource="&SS;100km/hSpeedLimitBreach_SemanticProperty"/>
  <hasEventType rdf:resource="&SS;CarDetection_EventType"/>
</Task>
<!-- http://www.owl-ontologies.com/SS.owl#CarDetection_EventType -->
<owl:Thing rdf:about="&SS;CarDetection_EventType">
  <rdf:type rdf:resource="&SS;Object"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/SS.owl#ObjectDetection_EventType -->
<Object rdf:about="&SS;ObjectDetection_EventType">
  <rdf:type rdf:resource="&owl;Thing"/>
</Object>
<!-- http://www.owl-ontologies.com/SS.owl#ValueGreaterThan100_Condition -->
<Condition rdf:about="&SS;ValueGreaterThan100_Condition">
  <rdf:type rdf:resource="&owl;Thing"/>
  <conditionValue rdf:datatype="&xsd;double">100.0</conditionValue>
</Condition>
<!--
//////////////////////////////////////

//
// General axioms
//
//////////////////////////////////////

-->
<rdf:Description>
  <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
  <owl:members rdf:parseType="Collection">
    <rdf:Description rdf:about="&SS;Condition"/>
    <rdf:Description rdf:about="&SS;EventType"/>
    <rdf:Description rdf:about="&SS;SemanticProperty"/>
    <rdf:Description rdf:about="&SS;SemanticService"/>
    <rdf:Description rdf:about="&SS;SemanticStream"/>
  </owl:members>
</rdf:Description>
<rdf:Description>
  <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
  <owl:members rdf:parseType="Collection">
    <rdf:Description rdf:about="&SS;Location"/>
    <rdf:Description rdf:about="&SS;Speed"/>
  </owl:members>
</rdf:Description>

```



```

        <rdf:Description rdf:about="&SS;Time"/>
    </owl:members>
</rdf:Description>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->

```

Listing A.2: The SS TR ontology in RDF/XML format

A.3 GL TR Ontology

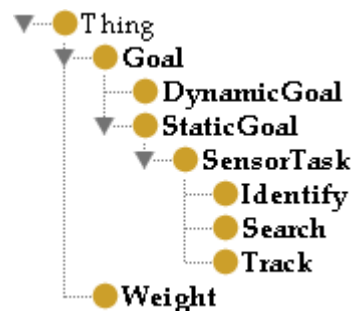


Figure A.3: GL TR class hierarchy

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY GoalLattice "http://www.owl-ontologies.com/GoalLattice.owl#" >
  <!ENTITY GoalLattice3 "http://www.owl-ontologies.com/GoalLattice.owl#0.0" >
  <!ENTITY GoalLattice2 "http://www.owl-ontologies.com/GoalLattice.owl#1.0" >
  <!ENTITY GoalLattice4 "http://www.owl-ontologies.com/GoalLattice.owl#0.5" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/GoalLattice.owl#"
  xml:base="http://www.owl-ontologies.com/GLTRontology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:GoalLattice="http://www.owl-ontologies.com/GoalLattice.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:GoalLattice3="&GoalLattice;0.0"
  xmlns:GoalLattice2="&GoalLattice;1.0"

```

```

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:GoalLattice4="&GoalLattice;0.5"
xmlns:owl="http://www.w3.org/2002/07/owl#">
<owl:Ontology rdf:about="">
  <rdfs:comment
    >Created by Konrad Borowiecki 2010</rdfs:comment>
</owl:Ontology>
<!--
//////////////////////////////////////

//
// Object Properties
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/GoalLattice.owl#contributesToSensorTask -->
<owl:ObjectProperty rdf:about="&GoalLattice;contributesToSensorTask">
  <rdfs:domain rdf:resource="&GoalLattice;DynamicGoal"/>
  <rdfs:range rdf:resource="&GoalLattice;SensorTask"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#hasSubGoal -->
<owl:ObjectProperty rdf:about="&GoalLattice;hasSubGoal">
  <rdfs:range rdf:resource="&GoalLattice;Goal"/>
  <rdfs:domain rdf:resource="&GoalLattice;Goal"/>
</owl:ObjectProperty>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#hasWeight -->
<owl:ObjectProperty rdf:about="&GoalLattice;hasWeight">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="&GoalLattice;Goal"/>
  <rdfs:range rdf:resource="&GoalLattice;Weight"/>
</owl:ObjectProperty>
<!--
//////////////////////////////////////

//
// Classes
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/GoalLattice.owl#DynamicGoal -->
<owl:Class rdf:about="&GoalLattice;DynamicGoal">

```

```

<rdfs:subClassOf>
  <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="&GoalLattice;Goal"/>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&GoalLattice;hasSubGoal"/>
        <owl:onClass rdf:resource="&GoalLattice;Goal"/>
        <owl:qualifiedCardinality rdf:datatype="&xsd;
          nonNegativeInteger">0</owl:qualifiedCardinality>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="&GoalLattice;contributesToSensorTask"/>
    <owl:someValuesFrom rdf:resource="&GoalLattice;SensorTask"/>
  </owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith rdf:resource="&GoalLattice;StaticGoal"/>
<rdfs:comment rdf:datatype="&xsd:string">
  >Dynamic goals that can be instantiated during mission run. And only they
  are measurable.</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#Goal -->
<owl:Class rdf:about="&GoalLattice;Goal">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&GoalLattice;hasWeight"/>
      <owl:onClass rdf:resource="&GoalLattice;Weight"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">1</
        owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="&GoalLattice;Weight"/>
  <rdfs:comment rdf:datatype="&xsd:string">
    >Goals are divided into two sub classes dynamic and static goals. It is
    assumed that only dynamic goals can be measured, and static goals are
    representing only a strategical/tactical breakdown of goals which
    leads to the success of a mission (the top most goal).</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#Identify -->
<owl:Class rdf:about="&GoalLattice;Identify">
  <rdfs:subClassOf rdf:resource="&GoalLattice;SensorTask"/>

```

```

</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#Search -->
<owl:Class rdf:about="&GoalLattice;Search">
  <rdfs:subClassOf rdf:resource="&GoalLattice;SensorTask"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#SensorTask -->
<owl:Class rdf:about="&GoalLattice;SensorTask">
  <rdfs:subClassOf rdf:resource="&GoalLattice;StaticGoal"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&GoalLattice;hasSubGoal"/>
      <owl:allValuesFrom rdf:resource="&GoalLattice;DynamicGoal"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="&xsd:string"
    >Task that a sensor might perform, GL author divides it into three groups
    (Search, Track, Identify).</rdfs:comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#StaticGoal -->
<owl:Class rdf:about="&GoalLattice;StaticGoal">
  <rdfs:subClassOf rdf:resource="&GoalLattice;Goal"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&GoalLattice;hasSubGoal"/>
      <owl:someValuesFrom rdf:resource="&GoalLattice;Goal"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="&xsd:string"
    >Goals which are representing general intentions of mission, which cannot
    be simply measured by sensor means, e.g. success of a mission,
    gaining of superiority in a region. They are unmeasurable.</rdfs:
    comment>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#Track -->
<owl:Class rdf:about="&GoalLattice;Track">
  <rdfs:subClassOf rdf:resource="&GoalLattice;SensorTask"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#Weight -->
<owl:Class rdf:about="&GoalLattice;Weight">
  <rdfs:comment
    >A double value from 0.0 to 1.0, expressing importance of a goal to the
    top most (main) goal of a task.</rdfs:comment>
</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing -->

```

```

<owl:Class rdf:about="&owl;Thing"/>
<!--
//////////////////////////////////////

//
// Individuals
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/GoalLattice.owl#0.0_Weight -->
<owl:Thing rdf:about="&GoalLattice;0.0_Weight"/>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#0.5_Weight -->
<owl:Thing rdf:about="&GoalLattice;0.5_Weight"/>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#1.0_Weight -->
<owl:Thing rdf:about="&GoalLattice;1.0_Weight"/>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#
    AccidentAreaMonitoring_StaticGoal -->
<StaticGoal rdf:about="&GoalLattice;AccidentAreaMonitoring_StaticGoal">
    <rdf:type rdf:resource="&owl;Thing"/>
    <hasWeight rdf:resource="&GoalLattice;1.0_Weight"/>
    <hasSubGoal rdf:resource="&GoalLattice;HumanSearch_SensorTask"/>
    <hasSubGoal rdf:resource="&GoalLattice;VehicleSearch_SensorTask"/>
</StaticGoal>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#CarSearch_DynamicGoal -->
<owl:Thing rdf:about="&GoalLattice;CarSearch_DynamicGoal">
    <rdf:type rdf:resource="&GoalLattice;DynamicGoal"/>
    <hasWeight rdf:resource="&GoalLattice;0.5_Weight"/>
    <contributesToSensorTask rdf:resource="&GoalLattice;VehicleSearch_SensorTask
        "/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#HumanSearch_SensorTask -->
<owl:Thing rdf:about="&GoalLattice;HumanSearch_SensorTask">
    <rdf:type rdf:resource="&GoalLattice;Search"/>
    <hasWeight rdf:resource="&GoalLattice;0.5_Weight"/>
    <hasSubGoal rdf:resource="&GoalLattice;VictimSearch_DynamicGoal"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#TaskSuccess_StaticGoal -->
<StaticGoal rdf:about="&GoalLattice;TaskSuccess_StaticGoal">
    <rdf:type rdf:resource="&owl;Thing"/>
    <hasWeight rdf:resource="&GoalLattice;1.0_Weight"/>
    <hasSubGoal rdf:resource="&GoalLattice;AccidentAreaMonitoring_StaticGoal"/>
</StaticGoal>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#TruckSearch_DynamicGoal -->

```

```

<DynamicGoal rdf:about="&GoalLattice;TruckSearch_DynamicGoal">
  <rdf:type rdf:resource="&owl;Thing"/>
  <hasWeight rdf:resource="&GoalLattice;0.0_Weight"/>
  <contributesToSensorTask rdf:resource="&GoalLattice;VehicleSearch_SensorTask
    "/>
</DynamicGoal>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#VehicleSearch_SensorTask -->
<owl:Thing rdf:about="&GoalLattice;VehicleSearch_SensorTask">
  <rdf:type rdf:resource="&GoalLattice;Search"/>
  <hasSubGoal rdf:resource="&GoalLattice;CarSearch_DynamicGoal"/>
  <hasSubGoal rdf:resource="&GoalLattice;TruckSearch_DynamicGoal"/>
</owl:Thing>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#VictimSearch_DynamicGoal -->
<DynamicGoal rdf:about="&GoalLattice;VictimSearch_DynamicGoal">
  <rdf:type rdf:resource="&owl;Thing"/>
  <hasWeight rdf:resource="&GoalLattice;0.5_Weight"/>
  <contributesToSensorTask rdf:resource="&GoalLattice;HumanSearch_SensorTask"/>
</DynamicGoal>
<!--
//////////////////////////////////////

//
// General axioms
//
//////////////////////////////////////

-->
<rdf:Description>
  <rdf:type rdf:resource="&owl;AllDifferent"/>
  <owl:distinctMembers rdf:parseType="Collection">
    <rdf:Description rdf:about="&GoalLattice;TaskSuccess_StaticGoal"/>
    <rdf:Description rdf:about="&GoalLattice;AccidentAreaMonitoring_StaticGoal
      "/>
    <rdf:Description rdf:about="&GoalLattice;CarSearch_DynamicGoal"/>
    <rdf:Description rdf:about="&GoalLattice;HumanSearch_SensorTask"/>
    <rdf:Description rdf:about="&GoalLattice;VehicleSearch_SensorTask"/>
    <rdf:Description rdf:about="&GoalLattice;TruckSearch_DynamicGoal"/>
  </owl:distinctMembers>
</rdf:Description>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->

```

Listing A.3: The GL TR ontology in RDF/XML format

A.4 SAM TR Ontology

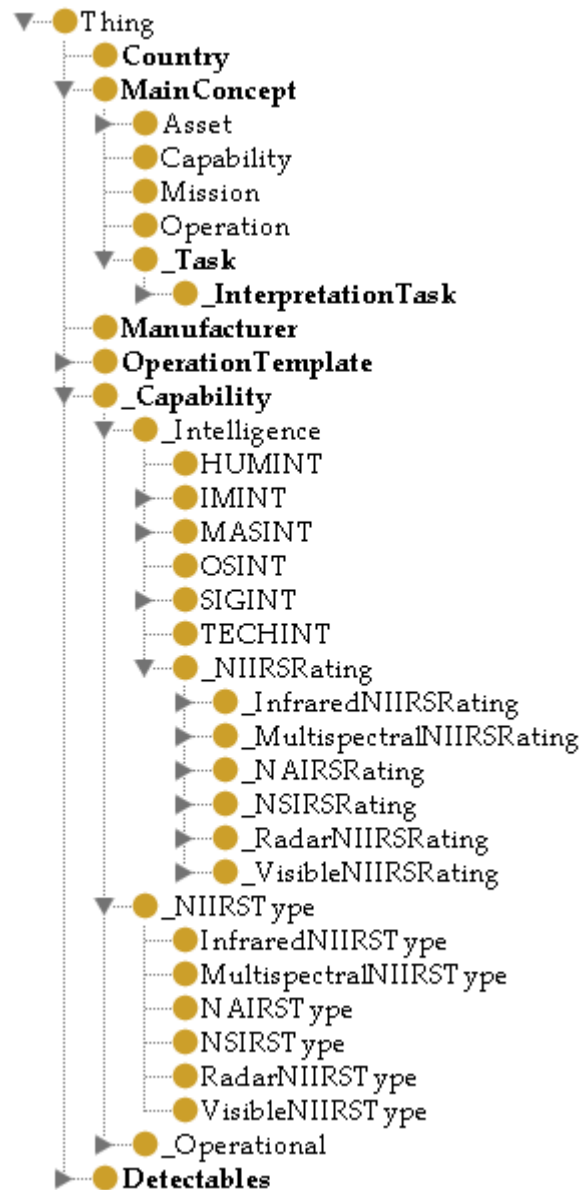


Figure A.4: SAM TR class hierarchy

```

<?xml version="1.0"?>
<!DOCTYPE Ontology [
  <ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <ENTITY istar "http://www.csd.abdn.ac.uk/ita/istar#" >
  <ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >

```

```

<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY SAMTROntology "http://www.semanticweb.org/ontologies/SAMTROntology.owl#"
>
<!ENTITY Detectable "http://www.csd.abdn.ac.uk/research/ita/download/ontology/
Detectable.owl#" >
]>
<Ontology xmlns="http://www.w3.org/2006/12/owl2-xml#"
xml:base="http://www.w3.org/2006/12/owl2-xml#"
xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
xmlns:istar="http://www.csd.abdn.ac.uk/ita/istar#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:SAMTROntology="http://www.semanticweb.org/ontologies/SAMTROntology.owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:Detectable="http://www.csd.abdn.ac.uk/research/ita/download/ontology/
Detectable.owl#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
URI="http://www.owl-ontologies.com/SAMTROntology.owl">
<Import
>http://www.csd.abdn.ac.uk/research/ita/download/ontology/Detectable.owl</
Import>
<Import
>http://www.csd.abdn.ac.uk/ita/istar </Import>
<Annotation annotationURI="&rdfs;comment">
<Constant
>Created by Konrad Borowiecki 2010</Constant>
</Annotation>
<Import
>http://www.csd.abdn.ac.uk/ita/istar </Import>
<Import
>http://www.csd.abdn.ac.uk/research/ita/download/ontology/Detectable.owl</
Import>
<DisjointClasses>
<Class URI="&istar;Country"/>
<Class URI="&Detectable;Detectables"/>
</DisjointClasses>
<DisjointClasses>
<Class URI="&istar;MainConcept"/>
<Class URI="&Detectable;Detectables"/>
</DisjointClasses>
<DisjointClasses>
<Class URI="&istar;Manufacturer"/>
<Class URI="&Detectable;Detectables"/>
</DisjointClasses>

```



```

<EntityAnnotation >
  <Class URI="&istar;NAIRSType"/>
  <Annotation annotationURI="&rdfs;comment">
    <Constant
      >National Acoustic Interpretability Rating Scale </Constant>
    </Annotation>
  </EntityAnnotation>
<EntityAnnotation >
  <Class URI="&istar;NSIRSType"/>
  <Annotation annotationURI="&rdfs;comment">
    <Constant
      >National Seismic Interpretability Rating Scale </Constant>
    </Annotation>
  </EntityAnnotation>
<DisjointClasses >
  <Class URI="&istar;OperationTemplate"/>
  <Class URI="&Detectable;Detectables"/>
</DisjointClasses >
<DisjointClasses >
  <Class URI="&istar;_Capability"/>
  <Class URI="&Detectable;Detectables"/>
</DisjointClasses >
<SubClassOf>
  <Class URI="&istar;_InterpretationTask"/>
  <ObjectSomeValuesFrom>
    <ObjectProperty URI="&SAMTRontology;hasDetectable"/>
    <Class URI="&Detectable;Detectables"/>
  </ObjectSomeValuesFrom>
</SubClassOf>
<SubClassOf>
  <Class URI="&istar;_Task"/>
  <ObjectAllValuesFrom>
    <ObjectProperty URI="&istar;requiresCapability"/>
    <Class URI="&istar;_Capability"/>
  </ObjectAllValuesFrom>
</SubClassOf>
<ObjectPropertyDomain>
  <ObjectProperty URI="&SAMTRontology;hasDetectable"/>
  <Class URI="&istar;_InterpretationTask"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
  <ObjectProperty URI="&SAMTRontology;hasDetectable"/>
  <Class URI="&Detectable;Detectables"/>
</ObjectPropertyRange>

```

```

<Declaration >
  <ObjectProperty URI="&SAMTROntology; hasDetectable"/>
</Declaration >
<ClassAssertion >
  <Class URI="&Detectable; Car"/>
  <Individual URI="&SAMTROntology; Car_Detectable"/>
</ClassAssertion >
<Declaration >
  <Individual URI="&SAMTROntology; Car_Detectable"/>
</Declaration >
<ClassAssertion >
  <Class URI="&istar; _Detect"/>
  <Individual URI="&SAMTROntology; DetectCar_ InterpretationTask"/>
</ClassAssertion >
<ObjectPropertyAssertion >
  <ObjectProperty URI="&SAMTROntology; hasDetectable"/>
  <Individual URI="&SAMTROntology; DetectCar_ InterpretationTask"/>
  <Individual URI="&SAMTROntology; Car_Detectable"/>
</ObjectPropertyAssertion >
<Declaration >
  <Individual URI="&SAMTROntology; DetectCar_ InterpretationTask"/>
</Declaration >
<ClassAssertion >
  <Class URI="&istar; _Task"/>
  <Individual URI="&SAMTROntology; DetectCar_Task"/>
</ClassAssertion >
<ObjectPropertyAssertion >
  <ObjectProperty URI="&istar; requiresCapability"/>
  <Individual URI="&SAMTROntology; DetectCar_Task"/>
  <Individual URI="&SAMTROntology; IRINT_Capability"/>
</ObjectPropertyAssertion >
<Declaration >
  <Individual URI="&SAMTROntology; DetectCar_Task"/>
</Declaration >
<ClassAssertion >
  <Class URI="&istar; IRINT"/>
  <Individual URI="&SAMTROntology; IRINT_Capability"/>
</ClassAssertion >
<Declaration >
  <Individual URI="&SAMTROntology; IRINT_Capability"/>
</Declaration >
</Ontology >
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->

```

Listing A.4: The SAM TR ontology in RDF/XML format

Ontologies of Hybrid Task Representations

Here we show class hierarchy and ontology for each of the HTR models. We do not present class hierarchy figures for Max SAM, Max SWE, and Max SS since they are just a particular mapping of classes from TRs and GHTR.

B.1 GHTR Ontology

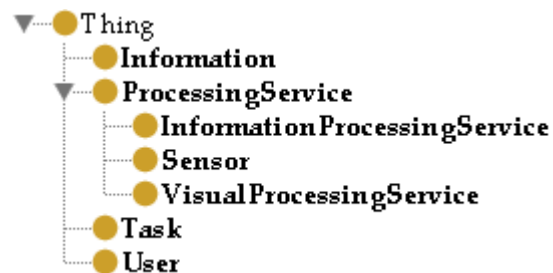


Figure B.1: GHTR class hierarchy

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY GHTRontology "http://www.semanticweb.org/ontologies/GHTRontology.owl#" >

```

```

]>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/GHTROntology.owl#"
  xml:base="http://www.semanticweb.org/ontologies/GHTROntology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:GHTROntology="http://www.semanticweb.org/ontologies/GHTROntology.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
<owl:Ontology rdf:about=""/>
<!--
//////////////////////////////////////

//
// Object Properties
//
//////////////////////////////////////

-->
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#hasSubTask -->
<owl:ObjectProperty rdf:about="#hasSubTask">
  <rdfs:domain rdf:resource="#Task"/>
  <rdfs:range rdf:resource="#Task"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#owns -->
<owl:ObjectProperty rdf:about="#owns">
  <rdfs:range rdf:resource="#Task"/>
  <rdfs:domain rdf:resource="#User"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#provides -->
<owl:ObjectProperty rdf:about="#provides">
  <rdfs:range rdf:resource="#Information"/>
  <rdfs:domain rdf:resource="#ProcessingService"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#requires -->
<owl:ObjectProperty rdf:about="#requires">
  <rdfs:range rdf:resource="#Information"/>
  <rdfs:domain rdf:resource="#Task"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#takes -->
<owl:ObjectProperty rdf:about="#takes">
  <rdfs:range rdf:resource="#Information"/>
  <rdfs:domain rdf:resource="#ProcessingService"/>
</owl:ObjectProperty>

```

```

<!--
////////////////////////////////////
//
// Classes
//
////////////////////////////////////

-->
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Information -->
<owl:Class rdf:about="#Information">
  <rdfs:subClassOf rdf:resource="#owl:Thing"/>
  <rdfs:comment
    >This represents information that is required by a task. It origins from
      sensors. It could also involve some processing.
Since effectively a task might be expressed as a set of needed information these two
  concepts cannot be disjoint. For example, in Semantic Streams they are the same
  thing, i.e. a user expresses his task in the form of the final semantic stream,
  where a stream represents a flow of information.</rdfs:comment>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#
  InformationProcessingService -->
<owl:Class rdf:about="#InformationProcessingService">
  <rdfs:subClassOf rdf:resource="#ProcessingService"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#ProcessingService -->
<owl:Class rdf:about="#ProcessingService">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#takes"/>
      <owl:allValuesFrom rdf:resource="#Information"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#provides"/>
      <owl:onClass rdf:resource="#Information"/>
      <owl:minQualifiedCardinality rdf:datatype="xsd:nonNegativeInteger"
        ">1</owl:minQualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Task"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Sensor -->

```

```

<owl:Class rdf:about="#Sensor">
  <rdfs:subClassOf rdf:resource="#ProcessingService"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#takes"/>
      <owl:onClass rdf:resource="#Information"/>
      <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger">0</
        owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Task -->
<owl:Class rdf:about="#Task">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSubTask"/>
      <owl:allValuesFrom rdf:resource="#Task"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#requires"/>
      <owl:someValuesFrom rdf:resource="#Information"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#User"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#User -->
<owl:Class rdf:about="#User">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#owns"/>
      <owl:someValuesFrom rdf:resource="#Task"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#
  VisualProcessingService -->
<owl:Class rdf:about="#VisualProcessingService">
  <rdfs:subClassOf rdf:resource="#ProcessingService"/>
</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing -->
<owl:Class rdf:about="&owl;Thing"/>
<!--

```

```
//////////////////////////////////////////////////////////////////  
  
//  
// Individuals  
//  
//////////////////////////////////////////////////////////////////  
  
-->  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#CameraFeedInfo -->  
<owl:Thing rdf:about="#CameraFeedInfo">  
  <rdf:type rdf:resource="#Information"/>  
</owl:Thing>  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#CameraSensor -->  
<Sensor rdf:about="#CameraSensor">  
  <rdf:type rdf:resource="#owl:Thing"/>  
  <provides rdf:resource="#CameraFeedInfo"/>  
</Sensor>  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#DetectPersonTask -->  
<Task rdf:about="#DetectPersonTask">  
  <rdf:type rdf:resource="#owl:Thing"/>  
  <requires rdf:resource="#DetectedPersonInfo"/>  
</Task>  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#DetectVehicleTask -->  
<Task rdf:about="#DetectVehicleTask">  
  <rdf:type rdf:resource="#owl:Thing"/>  
  <requires rdf:resource="#DetectedVehicleInfo"/>  
</Task>  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#DetectedPersonInfo -->  
<owl:Thing rdf:about="#DetectedPersonInfo">  
  <rdf:type rdf:resource="#Information"/>  
</owl:Thing>  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#DetectedVehicleInfo  
  -->  
<Information rdf:about="#DetectedVehicleInfo">  
  <rdf:type rdf:resource="#owl:Thing"/>  
</Information>  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#MikeUser -->  
<User rdf:about="#MikeUser">  
  <rdf:type rdf:resource="#owl:Thing"/>  
  <owns rdf:resource="#MonitorBorderTask"/>  
</User>  
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#MonitorBorderTask -->  
<owl:Thing rdf:about="#MonitorBorderTask">  
  <rdf:type rdf:resource="#Task"/>
```



```
<hasSubTask rdf:resource="#DetectPersonTask"/>
<hasSubTask rdf:resource="#DetectVehicleTask"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#
VehicleDetectionInCameraFeedProcServ -->
<owl:Thing rdf:about="#VehicleDetectionInCameraFeedProcServ">
  <rdf:type rdf:resource="#ProcessingService"/>
  <takes rdf:resource="#CameraFeedInfo"/>
  <provides rdf:resource="#DetectedVehicleInfo"/>
</owl:Thing>
<!--
////////////////////////////////////
//
// General axioms
//
////////////////////////////////////
-->
<rdf:Description>
  <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
  <owl:members rdf:parseType="Collection">
    <rdf:Description rdf:about="#Information"/>
    <rdf:Description rdf:about="#ProcessingService"/>
    <rdf:Description rdf:about="#User"/>
  </owl:members>
</rdf:Description>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

Listing B.1: The GHTR ontology in RDF/XML format

B.2 Max SAM Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY istar "http://www.csd.abdn.ac.uk/ita/istar#" >
  <!ENTITY SS "http://www.owl-ontologies.com/SS.owl#" >
```

```

<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY OGC "http://www.owl-ontologies.com/OGC.owl#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY GoalLattice "http://www.owl-ontologies.com/GoalLattice.owl#" >
<!ENTITY GHTRontology "http://www.semanticweb.org/ontologies/GHTRontology.owl#" >
]>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/MaxSAMHTRontology.owl#"
  xml:base="http://www.semanticweb.org/ontologies/MaxSAMHTRontology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:GHTRontology="http://www.semanticweb.org/ontologies/GHTRontology.owl#"
  xmlns:GoalLattice="http://www.owl-ontologies.com/GoalLattice.owl#"
  xmlns:istar="http://www.csd.abdn.ac.uk/ita/istar#"
  xmlns:OGC="http://www.owl-ontologies.com/OGC.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:SS="http://www.owl-ontologies.com/SS.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://www.owl-ontologies.com/GLTRontology.owl"/>
  <owl:imports rdf:resource="http://www.owl-ontologies.com/SAMTRontology.owl"/>
  <owl:imports rdf:resource="http://www.owl-ontologies.com/SSTRontology.owl"/>
  <owl:imports rdf:resource="http://www.owl-ontologies.com/SWETRontology.owl"/>
  <owl:imports rdf:resource="http://www.semanticweb.org/ontologies/GHTRontology.
    owl"/>
</owl:Ontology>
<!--
//////////////////////////////////////

//
// Classes
//
//////////////////////////////////////

-->
<!-- http://www.csd.abdn.ac.uk/ita/istar#Sensor -->
<owl:Class rdf:about="&istar;Sensor">
  <owl:equivalentClass rdf:resource="&GHTRontology;Sensor"/>
</owl:Class>
<!-- http://www.csd.abdn.ac.uk/ita/istar#_Task -->
<owl:Class rdf:about="&istar;_Task">
  <owl:equivalentClass rdf:resource="&GHTRontology;Task"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#DynamicGoal -->

```

```

<owl:Class rdf:about="&GoalLattice;DynamicGoal">
  <owl:equivalentClass rdf:resource="&GHTROntology;Task"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Sensor -->
<owl:Class rdf:about="&OGC;Sensor">
  <owl:equivalentClass rdf:resource="&GHTROntology;Sensor"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticService -->
<owl:Class rdf:about="&SS;SemanticService">
  <owl:equivalentClass rdf:resource="&GHTROntology;ProcessingService"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticStream -->
<owl:Class rdf:about="&SS;SemanticStream">
  <owl:equivalentClass rdf:resource="&GHTROntology;Information"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Sensor -->
<owl:Class rdf:about="&SS;Sensor">
  <owl:equivalentClass rdf:resource="&GHTROntology;Sensor"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Task -->
<owl:Class rdf:about="&SS;Task">
  <owl:equivalentClass rdf:resource="&GHTROntology;Task"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Information -->
<owl:Class rdf:about="&GHTROntology;Information"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#ProcessingService -->
<owl:Class rdf:about="&GHTROntology;ProcessingService"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Sensor -->
<owl:Class rdf:about="&GHTROntology;Sensor"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Task -->
<owl:Class rdf:about="&GHTROntology;Task"/>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->

```

Listing B.2: The Max SAM HTR ontology in RDF/XML format

B.3 Max SWE Ontology

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <ENTITY owl "http://www.w3.org/2002/07/owl#" >

```

```

<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
<!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
<!ENTITY istar "http://www.csd.abdn.ac.uk/ita/istar#" >
<!ENTITY SS "http://www.owl-ontologies.com/SS.owl#" >
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY OGC "http://www.owl-ontologies.com/OGC.owl#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY GoalLattice "http://www.owl-ontologies.com/GoalLattice.owl#" >
<!ENTITY GHTRontology "http://www.semanticweb.org/ontologies/GHTRontology.owl#" >
]>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/MaxSWEHTRontology.owl#"
  xml:base="http://www.semanticweb.org/ontologies/MaxSWEHTRontology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:GHTRontology="http://www.semanticweb.org/ontologies/GHTRontology.owl#"
  xmlns:GoalLattice="http://www.owl-ontologies.com/GoalLattice.owl#"
  xmlns:istar="http://www.csd.abdn.ac.uk/ita/istar#"
  xmlns:OGC="http://www.owl-ontologies.com/OGC.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:SS="http://www.owl-ontologies.com/SS.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/GLTRontology.owl"/>
    <owl:imports rdf:resource="http://www.owl-ontologies.com/SAMTRontology.owl"/>
    <owl:imports rdf:resource="http://www.owl-ontologies.com/SSTRontology.owl"/>
    <owl:imports rdf:resource="http://www.owl-ontologies.com/SWETRontology.owl"/>
    <owl:imports rdf:resource="http://www.semanticweb.org/ontologies/GHTRontology.
      owl"/>
  </owl:Ontology>
  <!--
  //
  // Classes
  //
  //
  //
  -->
  <!-- http://www.csd.abdn.ac.uk/ita/istar#Sensor -->
  <owl:Class rdf:about="&istar;Sensor">
    <owl:equivalentClass rdf:resource="&GHTRontology;Sensor"/>
  </owl:Class>
  <!-- http://www.csd.abdn.ac.uk/ita/istar#_Task -->

```

```

<owl:Class rdf:about="istar;_Task">
  <owl:equivalentClass rdf:resource="GHTROntology;Task"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#DynamicGoal -->
<owl:Class rdf:about="GoalLattice;DynamicGoal">
  <owl:equivalentClass rdf:resource="GHTROntology;Task"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#ProcessModel -->
<owl:Class rdf:about="OGC;ProcessModel">
  <owl:equivalentClass rdf:resource="GHTROntology;InformationProcessingService
  "/>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Sensor -->
<owl:Class rdf:about="OGC;Sensor">
  <owl:equivalentClass rdf:resource="GHTROntology;Sensor"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticService -->
<owl:Class rdf:about="SS;SemanticService">
  <owl:equivalentClass rdf:resource="GHTROntology;VisualProcessingService"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticStream -->
<owl:Class rdf:about="SS;SemanticStream">
  <owl:equivalentClass rdf:resource="GHTROntology;Information"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Sensor -->
<owl:Class rdf:about="SS;Sensor">
  <owl:equivalentClass rdf:resource="GHTROntology;Sensor"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Task -->
<owl:Class rdf:about="SS;Task">
  <owl:equivalentClass rdf:resource="GHTROntology;Task"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Information -->
<owl:Class rdf:about="GHTROntology;Information"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#
  InformationProcessingService -->
<owl:Class rdf:about="GHTROntology;InformationProcessingService"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Sensor -->
<owl:Class rdf:about="GHTROntology;Sensor"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Task -->
<owl:Class rdf:about="GHTROntology;Task"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#
  VisualProcessingService -->
<owl:Class rdf:about="GHTROntology;VisualProcessingService"/>

```

```
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

Listing B.3: The Max SWE HTR ontology in RDF/XML format

B.4 Max SS Ontology

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY SS "http://www.owl-ontologies.com/SS.owl#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY OGC "http://www.owl-ontologies.com/OGC.owl#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY GoalLattice "http://www.owl-ontologies.com/GoalLattice.owl#" >
  <!ENTITY GHTRontology "http://www.semanticweb.org/ontologies/GHTRontology.owl#" >
]>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/MaxSSHTRontology.owl"
  xml:base="http://www.semanticweb.org/ontologies/MaxSSHTRontology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:GHTRontology="http://www.semanticweb.org/ontologies/GHTRontology.owl#"
  xmlns:GoalLattice="http://www.owl-ontologies.com/GoalLattice.owl#"
  xmlns:OGC="http://www.owl-ontologies.com/OGC.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:SS="http://www.owl-ontologies.com/SS.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/GLTRontology.owl"/>
    <owl:imports rdf:resource="http://www.owl-ontologies.com/SSTRontology.owl"/>
    <owl:imports rdf:resource="http://www.owl-ontologies.com/SWETRontology.owl"/>
    <owl:imports rdf:resource="http://www.semanticweb.org/ontologies/GHTRontology.
      owl"/>
  </owl:Ontology>
  <!--
  //
  //////////////////////////////////////
```

```

// Classes
//
//////////////////////////////////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/GoalLattice.owl#DynamicGoal -->
<owl:Class rdf:about="&GoalLattice;DynamicGoal">
  <owl:equivalentClass rdf:resource="&GHTROntology;Task"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Sensor -->
<owl:Class rdf:about="&OGC;Sensor">
  <owl:equivalentClass rdf:resource="&GHTROntology;Sensor"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticService -->
<owl:Class rdf:about="&SS;SemanticService">
  <owl:equivalentClass rdf:resource="&GHTROntology;ProcessingService"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticStream -->
<owl:Class rdf:about="&SS;SemanticStream">
  <owl:equivalentClass rdf:resource="&GHTROntology;Information"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Sensor -->
<owl:Class rdf:about="&SS;Sensor">
  <owl:equivalentClass rdf:resource="&GHTROntology;Sensor"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#Task -->
<owl:Class rdf:about="&SS;Task">
  <owl:equivalentClass rdf:resource="&GHTROntology;Task"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Information -->
<owl:Class rdf:about="&GHTROntology;Information"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#ProcessingService -->
<owl:Class rdf:about="&GHTROntology;ProcessingService"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Sensor -->
<owl:Class rdf:about="&GHTROntology;Sensor"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Task -->
<owl:Class rdf:about="&GHTROntology;Task"/>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->

```

Listing B.4: The Max SS HTR ontology in RDF/XML format

B.5 Max SAM Plus Ontology

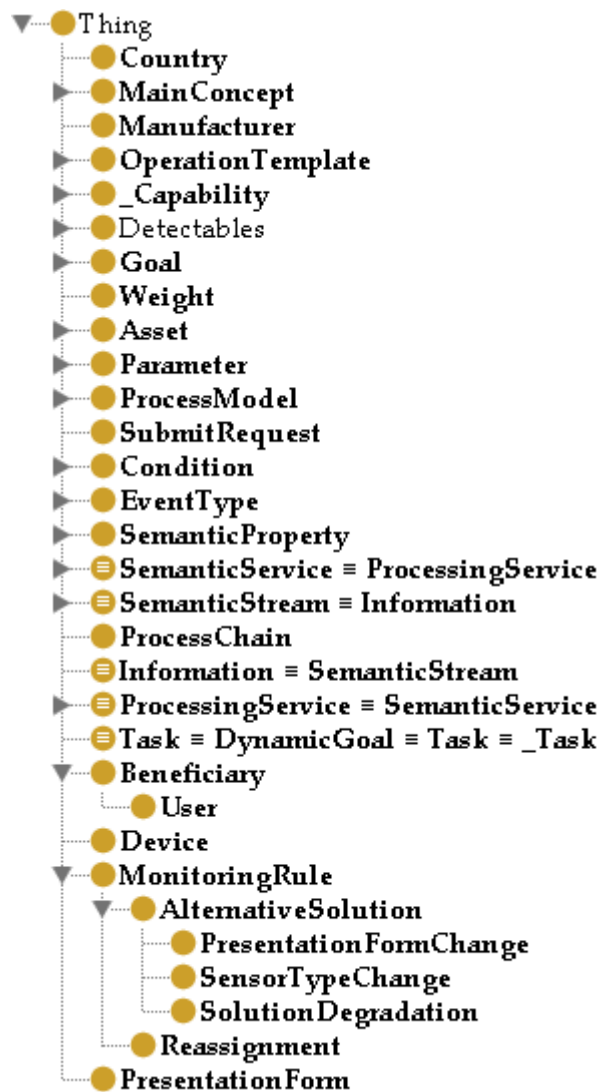


Figure B.2: Max SAM Plus class hierarchy

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY istar "http://www.csd.abdn.ac.uk/ita/istar#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY SS "http://www.owl-ontologies.com/SS.owl#" >
  <!ENTITY OGC "http://www.owl-ontologies.com/OGC.owl#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >

```



```

<!ENTITY GoalLattice "http://www.owl-ontologies.com/GoalLattice.owl#" >
<!ENTITY SWETROntology "http://www.owl-ontologies.com/SWETROntology.owl#" >
<!ENTITY GHTROntology "http://www.semanticweb.org/ontologies/GHTROntology.owl#" >
<!ENTITY MaxSAMPlusHTROntology "http://www.semanticweb.org/ontologies /
      MaxSAMPlusHTROntology.owl#" >
]>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#"
  xml:base="http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:GoalLattice="http://www.owl-ontologies.com/GoalLattice.owl#"
  xmlns:istar="http://www.csd.abdn.ac.uk/ita/istar#"
  xmlns:OGC="http://www.owl-ontologies.com/OGC.owl#"
  xmlns:MaxSAMPlusHTROntology="http://www.semanticweb.org/ontologies /
      MaxSAMPlusHTROntology.owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:SS="http://www.owl-ontologies.com/SS.owl#"
  xmlns:SWETROntology="http://www.owl-ontologies.com/SWETROntology.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:GHTROntology="http://www.semanticweb.org/ontologies/GHTROntology.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://www.semanticweb.org/ontologies /
      MaxSAMHTROntology.owl"/>
</owl:Ontology>
<!--
//////////////////////////////////////

//
// Object Properties
//
//////////////////////////////////////

-->
<!-- http://www.owl-ontologies.com/SS.owl#creates -->
<owl:ObjectProperty rdf:about="&SS;creates"/>
<!-- http://www.owl-ontologies.com/SS.owl#needs -->
<owl:ObjectProperty rdf:about="&SS;needs"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#hasSubTask -->
<owl:ObjectProperty rdf:about="&GHTROntology;hasSubTask"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#provides -->
<owl:ObjectProperty rdf:about="&GHTROntology;provides"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#requires -->
<owl:ObjectProperty rdf:about="&GHTROntology;requires"/>

```

```

<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#
    deliversSolutionTo -->
<owl:ObjectProperty rdf:about="#deliversSolutionTo">
    <rdfs:domain rdf:resource="&GHTROntology;Task"/>
    <rdfs:range rdf:resource="#Beneficiary"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#hasRule -->
<owl:ObjectProperty rdf:about="#hasRule">
    <rdfs:domain rdf:resource="&GHTROntology;Task"/>
    <rdfs:range rdf:resource="#MonitoringRule"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#supports -->
<owl:ObjectProperty rdf:about="#supports">
    <rdfs:domain rdf:resource="#Device"/>
    <rdfs:range rdf:resource="#PresentationForm"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#uses -->
<owl:ObjectProperty rdf:about="#uses">
    <rdfs:domain rdf:resource="#Beneficiary"/>
    <rdfs:range rdf:resource="#Device"/>
</owl:ObjectProperty>
<!--
//
// Classes
//
//////////////////////////////////////////////////////////////////

-->
<!-- http://www.csd.abdn.ac.uk/ita/istar#Country -->
<owl:Class rdf:about="&istar;Country"/>
<!-- http://www.csd.abdn.ac.uk/ita/istar#HDTV -->
<owl:Class rdf:about="&istar;HDTV"/>
<!-- http://www.csd.abdn.ac.uk/ita/istar#MainConcept -->
<owl:Class rdf:about="&istar;MainConcept">
    <owl:disjointWith rdf:resource="#Beneficiary"/>
</owl:Class>
<!-- http://www.csd.abdn.ac.uk/ita/istar#Manufacturer -->
<owl:Class rdf:about="&istar;Manufacturer"/>
<!-- http://www.csd.abdn.ac.uk/ita/istar#OperationTemplate -->
<owl:Class rdf:about="&istar;OperationTemplate"/>
<!-- http://www.csd.abdn.ac.uk/ita/istar#TVCamera -->
<owl:Class rdf:about="&istar;TVCamera"/>

```

```
<!-- http://www.csd.abdn.ac.uk/ita/istar#_Capability -->
<owl:Class rdf:about="&istar;_Capability"/>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#DynamicGoal -->
<owl:Class rdf:about="&GoalLattice;DynamicGoal"/>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#Goal -->
<owl:Class rdf:about="&GoalLattice;Goal">
  <owl:disjointWith rdf:resource="#Beneficiary"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/GoalLattice.owl#Weight -->
<owl:Class rdf:about="&GoalLattice;Weight"/>
<!-- http://www.owl-ontologies.com/OGC.owl#Asset -->
<owl:Class rdf:about="&OGC;Asset">
  <owl:disjointWith rdf:resource="#Beneficiary"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/OGC.owl#Parameter -->
<owl:Class rdf:about="&OGC;Parameter"/>
<!-- http://www.owl-ontologies.com/OGC.owl#ProcessModel -->
<owl:Class rdf:about="&OGC;ProcessModel"/>
<!-- http://www.owl-ontologies.com/OGC.owl#Sensor -->
<owl:Class rdf:about="&OGC;Sensor"/>
<!-- http://www.owl-ontologies.com/OGC.owl#SubmitRequest -->
<owl:Class rdf:about="&OGC;SubmitRequest"/>
<!-- http://www.owl-ontologies.com/SS.owl#Condition -->
<owl:Class rdf:about="&SS;Condition"/>
<!-- http://www.owl-ontologies.com/SS.owl#EventType -->
<owl:Class rdf:about="&SS;EventType"/>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticProperty -->
<owl:Class rdf:about="&SS;SemanticProperty"/>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticService -->
<owl:Class rdf:about="&SS;SemanticService">
  <owl:disjointWith rdf:resource="#Beneficiary"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SS.owl#SemanticStream -->
<owl:Class rdf:about="&SS;SemanticStream">
  <owl:disjointWith rdf:resource="#Beneficiary"/>
</owl:Class>
<!-- http://www.owl-ontologies.com/SWETROntology.owl#ProcessChain -->
<owl:Class rdf:about="&SWETROntology;ProcessChain"/>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#Information -->
<owl:Class rdf:about="&GHTROntology;Information">
  <owl:disjointWith rdf:resource="#Beneficiary"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/GHTROntology.owl#ProcessingService -->
<owl:Class rdf:about="&GHTROntology;ProcessingService">
```

```

    <owl:disjointWith rdf:resource="#Beneficiary"/>
  </owl:Class>
  <!-- http://www.semanticweb.org/ontologies/GHTRontology.owl#Task -->
  <owl:Class rdf:about="&GHTRontology;Task">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#deliversSolutionTo"/>
        <owl:someValuesFrom rdf:resource="#Beneficiary"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasRule"/>
        <owl:someValuesFrom rdf:resource="#MonitoringRule"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Beneficiary"/>
  </owl:Class>
  <!-- http://www.semanticweb.org/ontologies/GHTRontology.owl#User -->
  <owl:Class rdf:about="&GHTRontology;User">
    <rdfs:subClassOf rdf:resource="#Beneficiary"/>
  </owl:Class>
  <!-- http://www.semanticweb.org/ontologies/MaxSAMplusHTRontology.owl#
    AlternativeSolution -->
  <owl:Class rdf:about="&AlternativeSolution">
    <rdfs:subClassOf rdf:resource="#MonitoringRule"/>
  </owl:Class>
  <!-- http://www.semanticweb.org/ontologies/MaxSAMplusHTRontology.owl#Beneficiary
    -->
  <owl:Class rdf:about="&Beneficiary">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#uses"/>
        <owl:someValuesFrom rdf:resource="#Device"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:comment
      >Is anyone (or anything) who benefits from a task's satisfaction. It
        could be the task creator, other users or a service that benefit from
        a solution for a particular task.</rdfs:comment>
  </owl:Class>
  <!-- http://www.semanticweb.org/ontologies/MaxSAMplusHTRontology.owl#Device -->
  <owl:Class rdf:about="&Device">
    <rdfs:subClassOf>

```

```

        <owl:Restriction >
            <owl:onProperty rdf:resource="#supports"/>
            <owl:allValuesFrom rdf:resource="#PresentationForm"/>
        </owl:Restriction >
    </rdfs:subClassOf>
    <rdfs:comment
        >Device that a beneficiary uses to consume a task&#39;s solution.</rdfs:
        comment>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
        MonitoringRule -->
    <owl:Class rdf:about="#MonitoringRule">
        <rdfs:subClassOf rdf:resource="#owl:Thing"/>
        <rdfs:comment
            >Rules controlling how a task can/should respond to changes in the
            environment.</rdfs:comment>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
        PresentationForm -->
    <owl:Class rdf:about="#PresentationForm">
        <rdfs:subClassOf rdf:resource="#owl:Thing"/>
        <rdfs:comment
            >It captures how a task&#39;s solution can be represented on a device.</
            rdfs:comment>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
        PresentationFormChange -->
    <owl:Class rdf:about="#PresentationFormChange">
        <rdfs:subClassOf rdf:resource="#AlternativeSolution"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#Reassignment
        -->
    <owl:Class rdf:about="#Reassignment">
        <rdfs:subClassOf rdf:resource="#MonitoringRule"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
        SensorTypeChange -->
    <owl:Class rdf:about="#SensorTypeChange">
        <rdfs:subClassOf rdf:resource="#AlternativeSolution"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
        SolutionDegradation -->
    <owl:Class rdf:about="#SolutionDegradation">
        <rdfs:subClassOf rdf:resource="#AlternativeSolution"/>

```

```

</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing -->
<owl:Class rdf:about="&owl;Thing"/>
<!--
////////////////////////////////////

//
// Individuals
//
////////////////////////////////////

-->
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      BreakBeam1_Sensor -->
<OGC:Sensor rdf:about="#BreakBeam1_Sensor">
  <rdf:type rdf:resource="&owl;Thing"/>
  <SS:creates rdf:resource="#BreakBeam1_Stream"/>
  <GHTRontology:provides rdf:resource="#DetectSpeedingCar_Information"/>
</OGC:Sensor>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      BreakBeam1_Stream -->
<SS:SemanticStream rdf:about="#BreakBeam1_Stream">
  <rdf:type rdf:resource="&owl;Thing"/>
</SS:SemanticStream>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      BreakBeam2_Sensor -->
<OGC:Sensor rdf:about="#BreakBeam2_Sensor">
  <rdf:type rdf:resource="&owl;Thing"/>
  <SS:creates rdf:resource="#BreakBeam2_Stream"/>
  <GHTRontology:provides rdf:resource="#DetectSpeedingCar_Information"/>
</OGC:Sensor>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      BreakBeam2_Stream -->
<SS:SemanticStream rdf:about="#BreakBeam2_Stream">
  <rdf:type rdf:resource="&owl;Thing"/>
</SS:SemanticStream>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      CCTVCameraA_Sensor -->
<istar:TVCamera rdf:about="#CCTVCameraA_Sensor">
  <rdf:type rdf:resource="&OGC;Sensor"/>
  <rdf:type rdf:resource="&owl;Thing"/>
  <SS:creates rdf:resource="#CCTVCameraA_Stream"/>
  <GHTRontology:provides rdf:resource="#Image_Information"/>
</istar:TVCamera>

```

```

<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      CCTVCameraA_Stream -->
<owl:Thing rdf:about="#CCTVCameraA_Stream">
  <rdf:type rdf:resource="&SS;SemanticStream"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      CaptureSpeedingCarImageBreakBeam_Service -->
<SS:SemanticService rdf:about="#CaptureSpeedingCarImageBreakBeam_Service">
  <rdf:type rdf:resource="&owl;Thing"/>
  <SS:needs rdf:resource="#CCTVCameraA_Stream"/>
  <SS:creates rdf:resource="#CaptureSpeedingCarImage_Task"/>
  <SS:needs rdf:resource="#SpeedingCarBreakBeam_Stream"/>
</SS:SemanticService>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      CaptureSpeedingCarImageSpeedCamera_Service -->
<SS:SemanticService rdf:about="#CaptureSpeedingCarImageSpeedCamera_Service">
  <rdf:type rdf:resource="&owl;Thing"/>
  <SS:creates rdf:resource="#CaptureSpeedingCarImage_Task"/>
  <SS:needs rdf:resource="#SpeedingCarSpeedCameraA_Stream"/>
</SS:SemanticService>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      CaptureSpeedingCarImage_Task -->
<owl:Thing rdf:about="#CaptureSpeedingCarImage_Task">
  <rdf:type rdf:resource="&SS;SemanticStream"/>
  <rdf:type rdf:resource="&GHTRontology;Task"/>
  <GHTRontology:hasSubTask rdf:resource="#DetectSpeedingCar_Task"/>
  <GHTRontology:requires rdf:resource="#Image_Information"/>
  <deliversSolutionTo rdf:resource="#PoliceOfficer_Beneficiary"/>
  <hasRule rdf:resource="#ReassignmentRule"/>
  <deliversSolutionTo rdf:resource="#TrafficAnalyst_Beneficiary"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      CarDetails_PresentationForm -->
<PresentationForm rdf:about="#CarDetails_PresentationForm">
  <rdf:type rdf:resource="&owl;Thing"/>
</PresentationForm>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      DetectSpeedingCar_Information -->
<owl:Thing rdf:about="#DetectSpeedingCar_Information">
  <rdf:type rdf:resource="&GHTRontology;Information"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      DetectSpeedingCar_Task -->
<GoalLattice:DynamicGoal rdf:about="#DetectSpeedingCar_Task">

```

```

    <rdf:type rdf:resource="&GHTROntology;Task"/>
    <rdf:type rdf:resource="&owl;Thing"/>
    <GHTROntology:requires rdf:resource="#DetectSpeedingCar_Information"/>
    <hasRule rdf:resource="#SensorTypeChangeRule"/>
</GoalLattice:DynamicGoal>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#
    Image_Information -->
<GHTROntology:Information rdf:about="#Image_Information">
    <rdf:type rdf:resource="&owl;Thing"/>
</GHTROntology:Information>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#
    Image_PresentationForm -->
<owl:Thing rdf:about="#Image_PresentationForm">
    <rdf:type rdf:resource="#PresentationForm"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#
    Location_PresentationForm -->
<owl:Thing rdf:about="#Location_PresentationForm">
    <rdf:type rdf:resource="#PresentationForm"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#PC_Device -->
<owl:Thing rdf:about="#PC_Device">
    <rdf:type rdf:resource="#Device"/>
    <supports rdf:resource="#Statistics_PresentationForm"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#PatrolCarOn-
    Board_Device -->
<owl:Thing rdf:about="#PatrolCarOn-Board_Device">
    <rdf:type rdf:resource="#Device"/>
    <supports rdf:resource="#CarDetails_PresentationForm"/>
    <supports rdf:resource="#Image_PresentationForm"/>
    <supports rdf:resource="#Location_PresentationForm"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#
    PoliceOfficer_Beneficiary -->
<Beneficiary rdf:about="#PoliceOfficer_Beneficiary">
    <rdf:type rdf:resource="&owl;Thing"/>
    <uses rdf:resource="#PatrolCarOn-Board_Device"/>
</Beneficiary>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTROntology.owl#
    ReassignmentRule -->
<owl:Thing rdf:about="#ReassignmentRule">
    <rdf:type rdf:resource="#Reassignment"/>
</owl:Thing>

```



```

<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      SensorTypeChangeRule -->
<owl:Thing rdf:about="#SensorTypeChangeRule">
  <rdf:type rdf:resource="#SensorTypeChange"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      SpeedCameraA_Sensor -->
<owl:Thing rdf:about="#SpeedCameraA_Sensor">
  <rdf:type rdf:resource="&istar;HDTV"/>
  <rdf:type rdf:resource="&OGC;Sensor"/>
  <GHTRontology:provides rdf:resource="#DetectSpeedingCar_Information"/>
  <GHTRontology:provides rdf:resource="#Image_Information"/>
  <SS:creates rdf:resource="#SpeedingCarSpeedCameraA_Stream"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      SpeedingCarBreakBeam_Service -->
<owl:Thing rdf:about="#SpeedingCarBreakBeam_Service">
  <rdf:type rdf:resource="&SS;SemanticService"/>
  <SS:needs rdf:resource="#BreakBeam1_Stream"/>
  <SS:needs rdf:resource="#BreakBeam2_Stream"/>
  <SS:creates rdf:resource="#SpeedingCarBreakBeam_Stream"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      SpeedingCarBreakBeam_Stream -->
<owl:Thing rdf:about="#SpeedingCarBreakBeam_Stream">
  <rdf:type rdf:resource="&SS;SemanticStream"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      SpeedingCarSpeedCameraA_Stream -->
<owl:Thing rdf:about="#SpeedingCarSpeedCameraA_Stream">
  <rdf:type rdf:resource="&SS;SemanticStream"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      Statistics_PresentationForm -->
<owl:Thing rdf:about="#Statistics_PresentationForm">
  <rdf:type rdf:resource="#PresentationForm"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/MaxSAMPlusHTRontology.owl#
      TrafficAnalyst_Beneficiary -->
<owl:Thing rdf:about="#TrafficAnalyst_Beneficiary">
  <rdf:type rdf:resource="#Beneficiary"/>
  <uses rdf:resource="#PC_Device"/>
</owl:Thing>
<!--

```

```

//////////////////////////////////////////////////////////////////
//
// General axioms
//
//////////////////////////////////////////////////////////////////

-->
<rdf:Description>
  <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
  <owl:members rdf:parseType="Collection">
    <rdf:Description rdf:about="&istar;Country"/>
    <rdf:Description rdf:about="&istar;Manufacturer"/>
    <rdf:Description rdf:about="&istar;OperationTemplate"/>
    <rdf:Description rdf:about="&istar;_Capability"/>
    <rdf:Description rdf:about="&GoalLattice;Weight"/>
    <rdf:Description rdf:about="&OGC;Parameter"/>
    <rdf:Description rdf:about="&OGC;ProcessModel"/>
    <rdf:Description rdf:about="&OGC;SubmitRequest"/>
    <rdf:Description rdf:about="&SS;Condition"/>
    <rdf:Description rdf:about="&SS;EventType"/>
    <rdf:Description rdf:about="&SS;SemanticProperty"/>
    <rdf:Description rdf:about="&SWETROntology;ProcessChain"/>
    <rdf:Description rdf:about="#Beneficiary"/>
    <rdf:Description rdf:about="#Device"/>
    <rdf:Description rdf:about="#MonitoringRule"/>
    <rdf:Description rdf:about="#PresentationForm"/>
  </owl:members>
</rdf:Description>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->

```

Listing B.5: The Max SAM Plus HTR ontology in RDF/XML format

Bibliography

- [1] P. Andreas. Redrawing the line: Borders and security in the twenty-first century. In *International Security*, pages 78–111. MIT Press, 2003.
- [2] S. Bacharach. New Implementations of OGC Sensor Web Enablement Standards. Technical report, OGC, December 2007. <http://www.sensorsmag.com/sensors/Environmental/New-Implementations-of-OGC-Sensor-Web-Enablement-S/ArticleStandard/Article/detail/480557> (accessed 19.03.2010).
- [3] P. Barnaghi, M. Compton, O. Corcho, R. Garcia Castro, J. Graybeal, A. Herzog, K. Janowicz, H. Neuhaus, A. Nikolov, and K. Page. Semantic Sensor Network XG Final Report. Technical report, W3C Incubator Group, March 2011. <http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/> (accessed 24.11.2011).
- [4] S. Bechhofer, F. van Harmelen, J.A. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein. OWL Web Ontology Language Reference, February 2004. <http://www.w3.org/TR/owl-ref/> (accessed 12.09.2011).
- [5] L. Bengtsson, X. Lu, A. Thorson, R. Garfield, and J. von Schreeb. Improved response to disasters and outbreaks by tracking population movements with mobile phone network data: A post-earthquake geospatial study in haiti. *PLoS Med*, 8(8):e1001083, August 2011.

- [6] F. Bergamaschi, D. Conway-Jones, C. Gibson, and A. Stanford-Clark. A Distributed Test Framework for the Validation of Experimental Algorithms Using Real and Simulated Sensors. In *Proceedings of the 1st International Technology Alliance Conference, 2007*. ITA, 2007.
- [7] M. Boots and A. Robin. OpenGIS Sensor Model Language (SensorML) Implementation Specification. Technical report, Open Geospatial Consortium Inc., July 2007. http://portal.opengeospatial.org/files/?artifact_id=21273 (accessed 23.11.2011).
- [8] K. Borowiecki and A. Preece. An ontological approach to integrating task representations in sensor networks. In *Proc 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010)*. Springer, October 2010.
- [9] M. Botts, G. Percivall, C. Reed, and J. Davidson. OGC Sensor Web Enablement: Overview and High Level Architecture. In S. Nittel, A. Labrinidis, and A. Stefanidis, editors, *GeoSensor Networks*, volume 4540 of *Lecture Notes in Computer Science*, pages 175–190. Springer Berlin / Heidelberg, 2008.
- [10] L. Cabral, J. Dominigue, E. Motta, T. Payne, and F. Hakimput. Approaches to semantic web services: An overview and comparisons. In *The Semantic Web: Research and Applications*, volume 3053/2004, pages 225–239. Springer, 2004.
- [11] X. Chu and R. Buyya. Service oriented sensor web. In Nitaigour P. Mahalik, editor, *Sensor Networks and Configuration*, pages 51–74. Springer Berlin Heidelberg, 2007. 10.1007/3-540-37366-7_3.
- [12] Collaboration between government - the Canadian Space Agency, and industry - MacDonald, Dettwiler and Associates Ltd. (MDA). RADARSAT-2 information web page. <http://www.radarsat2.info/> (accessed 15.04.2010).
- [13] OOSTethys community. OOSTethys ontology. <http://mmisw.org/ont/mmi/20090519T125341/general> (accessed 28.11.2011).

- [14] A. Cowlard, W. Jahn, C. Abecassis-Empis, G. Rein, and J.L. Torero. Sensor assisted fire fighting. *Fire Technology*, 2008.
- [15] S. Cox. Observations and Measurements - XML Implementation. Technical report, Open Geospatial Consortium Inc., March 2011. http://portal.opengeospatial.org/files/?artifact_id=41510 (accessed 23.11.2011).
- [16] G. de Mel, M. Sensoy, W. Vasconcelos, and A. Preece. Flexible resource assignment in sensor networks: A hybrid reasoning approach. In *1st International Workshop on the Semantic Sensor Web (SemSensWeb 2009)*, 2009.
- [17] D. Dudek, Ch. Haas, A. Kuntz, M. Zitterbart, D. Krüger, P. Rothenpieler, D. Pfisterer, and S. Fischer. A wireless sensor network for border surveillance. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, SenSys '09*, pages 303–304, New York, NY, USA, 2009. ACM.
- [18] M. Eid, R. Liscano, and A. El Saddik. A universal ontology for sensor networks data. In *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSAS 2007. IEEE International Conference on*, pages 59–62, June 2007.
- [19] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
- [20] FarmWorks. Agriculture Software Company. <http://www.farmworks.co.uk/index.php> (accessed 17.04.2010).
- [21] D. Ferguson, N. Kalra, and A. Stentz. Replanning with RRTs. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1243–1248, May 2006.
- [22] I. Fermin and T. La Porta. Framework for Sensors System: Data Fusion and Information Dissemination. Technical report, International Technology Alliance, 2008.

- [23] D.E. Gale. *Understanding Urban Unrest: From Reverend King to Rodney King*. Sage Publications, Inc, 2455 Teller Road, Thousand Oaks, CA 91320, US, 1996.
- [24] B.P. Gerkey and M.J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.
- [25] P.W. Gething and A.J. Tatem. Can mobile phone data improve emergency response to natural disasters? *PLoS Med*, 8(8):e1001085, August 2011.
- [26] F. Golatowski, Jan Blumenthal, M. Handy, M. Haase, H. Burchardt, and D. Timmermann. Service-oriented software architecture for sensor networks. In *In Proc. Int. Workshop on Mobile Computing (IMC'03)*, pages 93–98, 2003.
- [27] M. Gomez, A. Preece, and G. de Mel. Towards Semantic Matchmaking in Sensor-Mission Assignment: Analysis of the Missions and Means Framework. Technical report, International Technology Alliance, March 2006.
- [28] M. Gomez, A. Preece, M.P. Johnson, G. de Mel, W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. La Porta, G. Pearson, T. Pham, D. Pizzocaro, and H. Rowaihy. An Ontology-Centric Approach to Sensor-Mission Assignment. In *EKAW 2008*, pages 347–363, 2008.
- [29] M. Goodman, G. Jedlovee, H. Conver, M. Botts, A. Robin, R. Blakeslee, R. Hood, S. Ingenthron, X. Li, M. Maskey, and K. Stephens. The Sensor Management for Applied Research Technologies (SMART) Project. In *Proceedings of the 7th NASA Science Technology Conference (NSTC2007)*. NASA, June 2007.
- [30] S. Havens. OpenGIS Transducer Markup Language (TML) Implementation Specification. Technical report, Open Geospatial Consortium Inc., 7 2007. http://portal.opengeospatial.org/files/?artifact_id=19371 (accessed 23.11.2011).

- [31] K.J. Hintz. Implicit collaboration of sensor systems. In Ivan Kadar, editor, *Signal Processing, Sensor Fusion, and Target Recognition XIII*, volume 5429, pages 89–94. SPIE, April 2004.
- [32] K.J. Hintz and M. Henning. Instantiation of dynamic goals based on situation information in sensor management systems. In *SPIE*, volume 6235, 2006.
- [33] K.J. Hintz and J. Malachowski. Dynamic goal instantiation in goal lattices for sensor management. In Ivan Kadar, editor, *Signal Processing, Sensor Fusion, and Target Recognition XIV*, volume 5809, pages 93–99. SPIE, 2005.
- [34] K.J. Hintz and G.A. McIntyre. Goal lattices for sensor management. In I. Kadar, editor, *Signal Processing, Sensor Fusion, and Target Recognition VIII*, volume 3720, pages 249–255. SPIE, 1999.
- [35] 52 North Initiative. Use of OGC SWE Standards Demonstration. <http://52north.org/maven/project-sites/swe/demos.html> (accessed 25.03.2010).
- [36] Marine Metadata Interoperability. MMI device ontologies working group. <http://marinemetadata.org/community/teams/ontdevices> (accessed 28.11.2011).
- [37] B.W. Johnson and J.M Green. Naval Network-Centric Sensor Resource Management. Technical report, Science Applications International Corporation and Naval Post Graduate School, 2002.
- [38] B. Kanagawa and E.K. Mugisa. Architecture analysis of service oriented architecture. In *Proceedings of the 2007 International Conference on Software Engineering Research & Practice, SERP 2007*, Las Vegas, Nevada, USA, June 2007. CSREA Press.
- [39] J.S. Kinnebrew, G. Biswas, N. Shankaran, D.C. Schmidt, and D. Suri. Integrating Task Allocation, Planning, Scheduling, and Adaptive Resource Management to Support Autonomy in a Global Sensor Web. In *Proceedings of the 7 NASA Science Technology Conference (NSTC2007)*. NASA, June 2007.

- [40] D. Liping. Geospatial Sensor Web and Self-adaptive Earth Predictive System (SEPS). Technical report, NASA, 2006.
- [41] J. Liu and F. Zhao. Composing semantic services in open sensor-rich environments. *Network, IEEE*, 22(4):44–49, 2008.
- [42] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless Sensor Networks and Applications*, WSNA '02, pages 88–97, New York, NY, USA, 2002. ACM.
- [43] K. Martinez, J.K. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37:50–56, August 2004.
- [44] L.A. Maver, C.D. Erdman, and K. Riehl. National Imagery Interpretability Rating Scale (NIIRS). <http://www.fas.org/irp/imint/niirs.htm> (accessed 15.04.2010).
- [45] S.A. McIlraith, T.C. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.
- [46] A. Na and M. Priest. Sensor Observation Service. Technical report, Open Geospatial Consortium Inc., 10 2007. http://portal.opengeospatial.org/files/?artifact_id=26667 (accessed 23.11.2011).
- [47] G.W. Ng and K.H. Ng. Sensor management - what, why and how. *Information Fusion*, 1(2):67–75, 2000.
- [48] S. Nolfi. Power and the limits of reactive agents. *Neurocomputing*, 42(1-4):119–145, 2002. Evolutionary neural systems.
- [49] University of Alabama in Huntsville (UAH) Visualization-Analysis Sensor Technology (VAST) Team. Sensor Model Language UAH - VAST webpage. <http://vast.uah.edu/SensorML/> (accessed 15.04.2010).

- [50] Open Geospatial Consortium (OGC). OGC Web Service-Phase 5 Demonstration. <http://www.opengeospatial.org/pub/www/ows5/index.html> (accessed 17.04.2010).
- [51] G. Pearson and T. Pham. The challenge of sensor information processing and delivery within network and information science research. In R. Suresh, editor, *Proc. SPIE*, volume 6981, page 5. SPIE, 2008.
- [52] G. Percival and C. Reed. OGC Sensor Web Enablement Standards, September 2006. On-line Magazine ‘Sensors & Transducers’ (S&T e-Digest), Vol. 71, Issue 9, pages 698-706, September 2006.
- [53] D. Perez, M. Nicolau, M. O’Neill, and A. Brabazon. Reactiveness and navigation in computer games: Different needs, different approaches. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 273–280, September 2011.
- [54] R. Perrey and M. Lycett. Service-oriented architecture. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT’03 Workshops)*, SAINT-W ’03, pages 116–119, Washington, DC, USA, 2003. IEEE Computer Society.
- [55] T. Pham, G.H. Cirincione, D. Verma, and G. Pearson. Intelligence, Surveillance, and Reconnaissance Fusion for Coalition Operations. In *2008 11th International Conference on Information Fusion*, pages 1–8. IEEE ICIF, 2008.
- [56] D. Pizzocaro, S. Chalmers, and A. Preece. Sensor Assignment In Virtual Environments Using Constraint Programming. In *Proceedings 27th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 333–338. Springer London, 2006.
- [57] D. Pizzocaro and A. Preece. Towards a taxonomy of task allocation in sensor networks. In *INFOCOM Workshops 2009, IEEE*, pages 1–2, April 2009.

- [58] D. Pizzocaro, A. Preece, F. Chen, T.L. Porta, and A. Bar-Noy. A distributed architecture for heterogeneous multi sensor-task allocation. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–8, June 2011.
- [59] A. Preece, M. Gomez, G. de Mel, W. Vasconcelos, D. Sleeman, S. Colley, G. Pearson, T. Pham, and T. La Porta. Matching Sensors to Missions using a Knowledge-Based Approach. In *SPIE Defense Transformation and Net-Centric Systems 2008 (SPIE Defense and Security Symposium)*. SPIE, 2008.
- [60] A. Preece, M. Gomez, G. de Mel, W. Vasconcelos, D. Sleeman, S. Colley, and T. La Porta. An Ontology-Based Approach to Sensor-Mission Assignment. In *Proceedings of the 1st International Technology Alliance Conference*. ITA, 2007.
- [61] A. Preece, M. Jackson, G. Pearson, and T. Pham. Sensor-Mission Assignment: A Scenario-Driven Walkthrough. Technical report, International Technology Alliance, 2008.
- [62] A. Preece, D. Pizzocaro, K. Borowiecki, G. de Mel, M. Gomez, W. Vasconcelos, A. Bar-Noy, M.P. Johnson, T. La Porta, H. Rowaihy, G. Pearson, and T. Pham. Reasoning and Resource Allocation for Sensor-Mission Assignment in a Coalition Context. In *MILCOM 2008*, 2008.
- [63] A. Preece, D. Pizzocaro, K. Borowiecki, G. de Mel, M. Gomez, W. Vasconcelos, A. Bar-Noy, M.P. Johnson, T. La Porta, H. Rowaihy, G. Pearson, and T. Pham. Sensor Assignment to Missions in a Coalition Context: The SAM Tool. In *Adjunct Proceedings of INFOCOM 2009 demo session*. INFOCOM, April 2009.
- [64] A. Preece, D. Pizzocaro, K. Borowiecki, G. de Mel, W. Vasconcelos, A. Bar-Noy, M.P. Johnson, T. La Porta, and H. Rowaihy. Knowledge-driven agile sensor-mission assignment. In *Proc 3rd Annual Conference of the International Technology Alliance (ACITA 2009)*, page 8. ACITA, 2009.

- [65] Repast. Repast Symphony 2.0 Beta. http://repast.sourceforge.net/repast_simphony.html (accessed 12.09.2011).
- [66] H. Rowaihy, S. Eswaran, M. Johnson, D. Verma, A. Bar-Noy, T. Brown, and T. La Porta. A survey of sensor selection schemes in wireless sensor networks. *Proceedings of SPIE*, 6562(13):65621A–65621A–13, 2007.
- [67] H. Rowaihy, M.P. Johnson, D. Pizzocaro, A. Bar-Noy, L. Kaplan, T. La Porta, and A. Preece. Detection and localization sensor assignment with exact and fuzzy locations. In Bhaskar Krishnamachari, Subhash Suri, Wendi Heinzelman, and Urbashi Mitra, editors, *Distributed Computing in Sensor Systems*, volume 5516 of *Lecture Notes in Computer Science*, pages 28–43. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-02085-8_3.
- [68] S.H. Russ, V. Perepa, S. Leavesly, and B. Webb. Novel low-cost salinity sensor for embedded environmental monitoring. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pages 53–56, March 2010.
- [69] D.J. Russomanno, C.R. Kothari, and O.A. Thomas. Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In Hamid R. Arabnia and Rose Joshua, editors, *The 2005 International Conference on Artificial Intelligence (IC-AI)*, pages 637–643. CSREA Press, 2005.
- [70] J.H. Sheehan, P.H. Deiz, B.E. Bray, B.A. Harris, and A.B.H. Wong. The military missions and means framework. In *Interservice/Industry Training, Simulation, and Education Conference*, pages 655–663, 2003.
- [71] I. Simonis and J. Echterhoff. OGC Sensor Planning Service Implementation Specification. Technical report, Open Geospatial Consortium Inc., 3 2011. http://portal.opengeospatial.org/files/?artifact_id=38478 (accessed 23.11.2011).
- [72] F. Souza and I. Kushchu. Acoustic Sensors on Small Robots for the Urban Environment. In *Proceedings of EURO mGOV 2005*, pages 455–466. Mobile Government Consortium International LLC, July 2005.

- [73] Rapiscan Systems. Rapiscan Eagle M60. http://www.rapiscansystems.com/en/products/cvi/productsrapiscan_eagle_m60/ (accessed 19.04.2012).
- [74] NDI Recognition Systems UK. TALON ANPR for Enforcement, Surveillance and Intelligence Gathering. <http://ndi-rs.com/ukrs/policing/> (accessed 19.04.2012).
- [75] D.C. Verma, G. Cirincione, and T. Pham. Policy enabled interconnection of sensor networks using a message queue infrastructure. In R. Suresh, editor, *Proc. SPIE*, volume 6981, page 698108. SPIE, 2008.
- [76] K. Whitehouse, F. Zhao, and J. Liu. Semantic Streams: a Framework for Declarative Queries and Automatic Data Interpretation. Technical report, Microsoft Research, 2005. Microsoft Research Technical Report MSR-TR-2005-45.
- [77] K. Whitehouse, F. Zhao, and J. Liu. Semantic streams: A framework for composable semantic interpretation of sensor data. In K. Römer, H. Karl, and F. Mattern, editors, *Proceedings of the Third European Workshop, EWSN 2006*, pages 5–20. Springer, February 2006.
- [78] Wikipedia. Description of Precision Agriculture. http://en.wikipedia.org/wiki/Precision_agriculture/ (accessed 17.04.2011).
- [79] Wikipedia. Explosives trace detector. http://en.wikipedia.org/wiki/Explosives_trace_detector/ (accessed 19.04.2012).
- [80] K.J. Witt, J. Stanley, D. Smithbauer, D. Mandl, V. Ly, A. Underbrink, and M. Metheny. Enabling Sensor Webs by utilizing SWAMO for autonomous operations. In *8th NASA Earth Science Technology Conference (ESTC2008)*. NASA, 2008.
- [81] World Wide Web Consortium (W3C). SPARQL query language web page. <http://www.w3.org/TR/rdf-sparql-query/> (accessed 17.04.2010).

-
- [82] J. Wright, C. Gibson, F. Bergamaschi, K. Marcus, T. Pham, R. Pressley, and G. Verma. ITA Sensor Fabric. In *SPIE*, volume 7333, 2009.
- [83] J. Wright, C. Gibson, F. Bergamaschi, K. Marcus, R. Pressley, G. Verma, and G. Whipps. A dynamic infrastructure for interconnecting disparate ISR/ISTAR assets (the ITA sensor fabric). In *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pages 1393–1400, July 2009.
- [84] C.S. Yang and C.G. Kang. Ship detection experiments using RADARSAT/SAR images. In *Geoscience and Remote Sensing Symposium, 2005. IGARSS '05. Proceedings. 2005 IEEE International*, volume 2, pages 1177–1180. IGARSS, July 2005.
- [85] E. Yoshida, K. Yokoi, and P. Gergondet. Online replanning for reactive robot motion: Practical aspects. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5927–5933, October 2010.
- [86] R. Young. How information requirements are specified? Or defining the mission objectives for Sensor Networks. Technical report, International Technology Alliance (ITA), 2008. ITA Technical Note.