A Case Study for Integrating Heterogeneous Knowledge Bases for Outdoor Environments

Sebastian Blumenthal[1], Benjamin Brieber[2], Nico Huebel[1],
Fereshta Yazdani[2], Michael Beetz[2] and Herman Bruyninckx[1]

***Abstract*— We present the integration of heterogeneous knowledge bases and reasoning mechanisms in the SHERPA project. SHERPA is about a complex search and rescue scenario that requires planning and reasoning about actions of mixed human-robot teams in space and in time. This has been achieved by integrating different sources of knowledge and data, namely, OpenStreetMap for static map data, the Robot Scene Graph for composing the OpenStreetMap data with dynamic data from the environment and the rescue team, and KnowRob with the SHERPA ontology for abstract reasoning in the application domain. This work explains how these knowledge bases were integrated by model composition and model to model transformations and an application dependent bridge. Design decisions are discussed and an example is given that explains the usage of the heterogeneous knowledge and reasoning methods.**

I. INTRODUCTION

Many applications require various kinds of knowledge originating from different domains and sources. Examples range from applications in manufacturing, like assembly tasks [6] or flexible manufacturing systems [7], over household applications, like making pancakes [8] or folding towels [9], to search and rescue applications, like searching for people [2], [4]. All these application domains have in common that they require knowledge about the task, the environment, the capabilities of the available agents, and the involved objects. For example, for a rescue mission, which is searching for victims, the knowledge about what type of robots are available and which capabilities are required, *e.g.*, flying robots can cross a river while (most) ground robots cannot. Then knowledge about how to split a search area according to the constraints of the available robots and how to create feasible search patterns within these areas is required. For both of these tasks, knowledge about the environments is required, *e.g.*, the course of rivers and where to find bridges, the height of mountain ranges, the course of hiking paths, or which areas are covered by trees. Finally, the mission requires knowledge about the involved objects, *e.g.*, that a human has a heat signature while a rock has none, or that a river is not traversable by a ground robot. This knowledge comes from various sources. Some knowledge, like how to create a search pattern for a given area or how to detect a victim in a thermal image, can be locally available on a robot while other knowledge, like how to split a search area among a set of robots, must be located, where the mission planning takes place. However, for a truly flexible and autonomous system most knowledge must be available at all places but can have different resolutions. *E.g.*, the mission planning must know about the robots' capabilities but does not necessarily have to know which type of hardware or combination of sensors and algorithms are providing these capabilities. For monitoring the execution of the mission, a prediction of a robot's behaviour is necessary, which requires some knowledge about the robot's algorithms. For plan execution in outdoor environments the robot needs to know its own location, the area to navigate and the capability to revise a map based on its sensor inputs and the data coming from Geo Information Systems (GIS). Additionally, a robot needs to know about its role and the roles of its collaborators in a shared task, but is less interested in knowledge about an unrelated tasks. Integrating all these heterogeneous knowledge sources from different domains provides several challenges:

- They have no common encoding. Some ontological knowledge comes in Web Ontology Language (OWL) [10] or in RDF triples [11]. Some knowledge and data is encoded in JSON objects [12] or XML files [13]. For sensor data there exist a vast amount of formats, although for robotics ROS messages[1] are likely among the most common ones. This zoo of knowledge representations prevent the usage of existing ontology or database schema mapping tools that have been recently developed [17] [18]. Finally, much of the knowledge, *e.g.*, about algorithms or software configurations, is not formally encoded at all but only known by the human developers or accessible in human readable documents.
- They have no clean ontological structure, *i.e.*, they are not structured in themselves.
- There is no knowledge available about *how* the various domain specific knowledge sources are connected within a domain and between domains. So connecting facts and terms and mapping them from one source or domain to another mostly has to be done manually.
- Terminology is ambiguous. In some domains the same terms can have different definitions while other different terms can mean the same.

Here we present how several heterogeneous knowledge and data sources have been integrated for the SHERPA

[1]Sebastian Blumenthal, Nico Huebel and Herman Bruynuinckx are with the Robotics Research Group of the KU Leuven Department of Mechanical Engineering, Belgium. The Robotics Research Group is an associated research lab of Flanders Make.

[2]Benjamin Brieber, Fereshta Yazdani and Michael Beetz are with the Institute for Artificial Intelligence, University Bremen, Germany. Corresponding author: sebastian.blumenthal@kuleuven.be

[1]http://wiki.ros.org/Messages

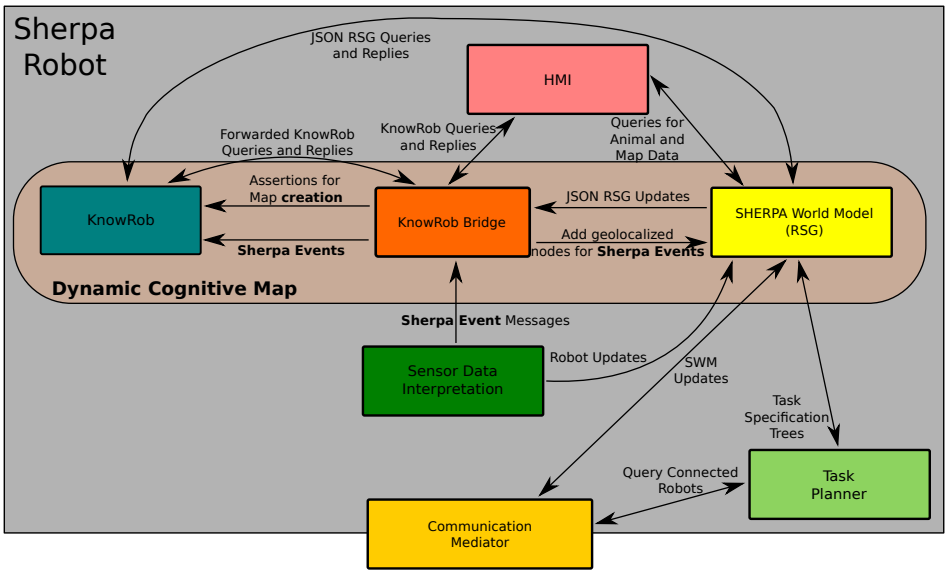

Fig. 1. Simplified architecture of one SHERPA robot with focus on the so called Dynamic Cognitive Map (DCM) and the components it serves. The DCM is the SHERPA terminology for referring to the integrated knowledge bases and their reasoning capabilities. The HMI, planning, and communication components interact with the DCM via the query interface of the Robot Scene Graph (RSG). At start-up the Open Street Map data is loaded into the RSG and at runtime the robots update the RSG with sensor data. Specific events from interpreted sensor data, like a detected victim, are sent to the bridge that makes this knowledge available to the RSG as tagged location and to KnowRob as a semantic event. In a typical SHERPA mission the operator uses the HMI to send voice commands to KnowRob for interpretation. KnowRob uses its knowledge to interpret the context-specific task and accesses the required data from the RSG.

project [2]. The SHERPA project is a search and rescue robotics project that aims at helping a human rescue team to find victims in an alpine environment by adding different types of robots to the team. The heterogeneity of the team, the human-robot interaction, the variety of the tasks during a mission and the potential mission length of several days require reasoning about commands, semantics, data and its history, and spatio-temporal relations.

To achieve this, the following knowledge bases were chosen to be integrated because they are publicly available or based on the authors past efforts.

- The static map based on OpenStreetMap (OSM) [5] data. It is publicly available and contains rich semantic annotations.
- The dynamic and distributed map based on the Robot Scene Graph [3] (RSG). In contrast to environment representation used in comparable search and rescue projects like [4], the RSG is designed as an active spatio-temporal graph database that is decentralized to allow availability of knowledge and data on every robot even in fragile networks.
- KnowRob was extended with the SHERPA ontology to ensure reasoning about indoor as well as outdoor environments.

Integration of these knowledge bases has been achieved by model composition and model to model transformations and an application dependent bridge. The main contribution of this paper is to provide an example how these methods can be applied to integrate heterogeneous knowledge bases in order to perform reasoning. Also the advantages and disadvantages of the chosen methods and design decisions are discussed.

In section II the three chosen knowledge bases are introduced before section III discusses their integration. Then section IV provides an example that explains how the integrated system is used to perform reasoning. Finally, section V contains a discussion and the lessons-learned from the presented effort.

Although SHERPA is a multi-robot scenario, for the sake of brevity and ease of explanation, we will focus here on a single robot example. The architecture is shown in Fig. 1. The presented methods are not limited to the presented use case of the SHERPA project but can also be applied to other robotic applications.

II. KNOWLEDGE BASES

This section gives a short overview of the three different knowledge bases and shows their structures based on ontological, topological and semantical manners.

A. *OpenStreetMap*

OpenStreetMap (OSM) is a community driven effort to collect and publish geographic data used, *e.g.*, for navigation systems. In the SHERPA scenario this data serves as a static environment representation that can be downloaded a-priori to a mission.

The data format stores the static and topological information in graph structure. The core elements are *nodes*, *ways*, and *relations*.

- *Nodes* represents static positions in the map, typically derived from GPS measurements.
- *Ways* denote polygon features in order to represent paths, rivers or buildings.

- *Relations* are used to store general purposes relations between elements, e.g. roads that reference multiple path segments.

Each above listed element can have a list of associated key value pairs to further define the meaning of the elements. The list of valid tags[2] is maintained by a collaborative community effort. For the SHERPA mission only a subset is relevant, mainly tags for paths, forests, rivers and buildings are important. This list can be seen as an ontology relevant for the mission. Though, as an ontology it is not deep as it has only one level of sub classing.

B. Robot Scene Graph

While the OSM provides a priori data about the environment of a SHERPA mission, the Robot Scene Graph (RSG) extends the OSM data with Search And Rescue (SAR) related semantics by combining static and dynamic data about the environment, mission, and the team members and sharing that information among all members during the mission.

The RSG is a distributed world model that is able to answer queries like which objects exist in the environment, where they are, and when they have been at a particular place. This capability is essential to enable reasoning about the world during a rescue mission. However, while most world models are passive repositories for knowledge, the RSG is an active component, that can also run services on its data and take decisions based on monitoring functions. The monitoring capability is also used to synchronize multiple world models by sending local changes to the other peers.

As different robots can have quite different requirements on how to represent the world, the RSG must be a flexible data structure. The design approach for the RSG is to use a graph data structure that is *composable*. The nodes of this graph have the meaning of, *e.g.*, objects or measurements, while edges represent relations between them. Here, *composable* means that domain specific aspects like geometry, topology or semantic tags are composed of the graph primitives. Some examples are explained below.

A rather common **geometric** relation is a relative pose between two objects. The so called *Transform* is a relation in the RSG to represent a relative position and orientation between exactly two nodes in a time series format. In contrast to a static map like OSM, this temporal cache enables to track where objects have been. The cache is configured to be limited in time to account for limited memory resources of the robots. A *Transform* has an indication to distinguish between different representation types like WGS84 for GPS data or Cartesian poses deduced from sensor measurements. Queries for poses between any two objects, connected by a series *Transforms*, are possible and take the representation types into account.

A **topological** relation like a path or a search area is stored as an edge in the RSG. It is denoted as *Connection* and uses an ordered set of nodes to represent way points or polygon edges. Every *Connection* has a valid time interval indicated by a start and end time stamp. In particular, in a rescue mission a path might not exist any more due to an avalanche or rock slide. So it can be invalidated by setting the end time stamp to point in time before its destruction has occurred. Since poses of nodes referenced by a *Connection* are expressed via *Transform* relations, they can be updated. This is in particular relevant when a human rescuer decides to enlarge or shrink an existing search area for the robots.

Semantic tags are represented by *Attributes*. Every node or relation in the RSG can have *Attributes*. A single *Attribute* can be used to store a symbol of an ontology. Since in most rescue missions multiple ontologies are involved, an indication which *Attribute* belongs to which ontology is required. In the SHERPA project such a semantic context is represented by a prefix for the ontology separated from the semantic tag by a colon. This can be regraded as a form of name space for the semantic tags. A SHERPA mission uses among others an `osm` prefix for OSM related data, `gis` for generic GIS data or `sherpa` for mission specific information like a detected victim. An extensible list of relevant name spaces and *Attributes* can be found in the software documentation mentioned in Section V.

An important decision for the SHERPA project was to separate images and 3D shape data from its metadata. Because of the communication constraints and the bandwidth required to update large data between the RSGs on different robots, larger data files like Digital Elevation Maps (DEMs), point clouds, or images are not directly added to the RSG. Instead, only their metadata is added, which includes information like the location of the data, the described area or location and time stamp of creation.

In order to access relevant information about the world, it is possible to send **queries** to the RSG. One kind of queries are *Create*, *Read*, *Update* and *Delete* (CRUD) operations on the individual nodes and relations. This is comparable to the interfaces of general purpose databases. However, more complex queries require a lot of such operations to achieve the desired result. Therefore, another kind of queries trigger the *function block query* mechanism of the RSG. A function block can be seen as an active service for specific queries. This is comparable to a *stored procedure*, that is common for databases. It is dynamically loaded into an instance of the RSG and allows to move aggregated queries to the data. An example used for a SHERPA mission is a function block that retrieves a trajectory of a robot within a given start and end time.

Queries that require reasoning on ontologies are beyond the scope of RSG in a SHERPA mission since the KnowRob system is better suited for such operations.

C. KnowRob and SHERPA Ontology

The data acquisition in outdoor environments is enormous. Structuring and reasoning about the available information coming from sensors by the various robots is challenging but desirable for this mission. To reason about available capability, components, and actions of the SHERPA team members,

[2] `http://wiki.openstreetmap.org/wiki/Map_Features`

we are using KnowRob [1]. KnowRob is a knowledge processing framework for robots, which provides features that are particularly essential for autonomous robot control, e.g. for reasoning about tasks, managing uncertainty, and fast inference. By extending KnowRob for the SHERPA mission the robots are able to process high-level information, *e.g.*, to reason if a particular robot has the specific capabilities for executing a given (sub-)task. The extension to KnowRob was made by integrating a rich knowledge base including different ontologies. These ontologies contain among other parts robot actions, robot components, robot capabilities and include also reasoning mechanisms for creating hypotheses and inferences based on available data sets.

Instead of having only *is-a* relations inside of a class hierarchy, KnowRob allows to define classes by restrictions. This means that class declarations can be implicit, based on a set of rules specified in OWL classes. This makes class definitions very flexible while the set of classes that are valid for an entity are not defined beforehand, but can be generated during execution time. Furthermore, this allows **data-driven class definitions** for entities. For the SHERPA mission, an ontology containing classes with meaning for the execution of SAR tasks has been created[3]. Examples include dangerous roads, rivers, bridges as well as additional region properties such as shape of paths and path traversability for specific robots. With regard to the performing robot and its capabilities, KnowRob is able to answer queries about such properties by deriving the corresponding classes from GIS data e.g. if a robot is capable to pass a specific path.

Another aspect of KnowRob are **computables**. Computables describe rule-definitions, which generate data on demand. They are used to extract data from other sources like the RSG, execute computation intensive methods, or answer specific complex queries. A SHERPA relevant example for such a complex query is the computation of virtual areas, which have specific meaning in the mission context, like potentially dangerous or non-traversable terrain.

III. INTEGRATION OF KNOWLEDGE BASES AND REASONING

The goal of the integrated knowledge bases is to enable reasoning about the environment in the context of a SHERPA mission. We denote it as Dynamic Cognitive Map (DCM). It is a knowledge base composed of OpenStreetMap, Robot Scene Graph and KnowRob as illustrated in Fig. 2 that can be queried at run-time. Note, the circles do not overlap completely to indicate only the application relevant aspects are integrated. The remainder of this section explains how this integration is achieved.

A. Integration between OpenStreetMap and Robot Scene Graph

Since OSM provides static map data, the domain specific data and the dynamic data of the mission needs to be composed with that map data. This is done by extracting

[3]https://www.dropbox.com/s/f8s2ek79dzbdl1v/Deliverable5.2.pdf?dl=0

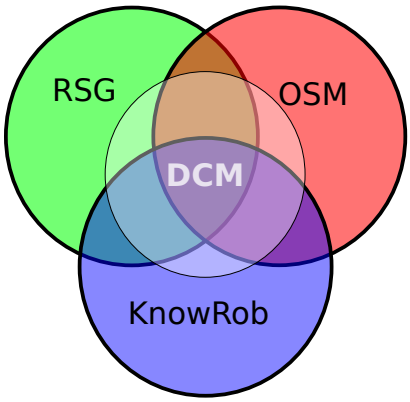

Fig. 2. The knowledge bases that are integrated to form the Dynamic Cognitive Map (DCM).

the OSM about the mission area and making it available in the RSG. So a model to model transform of OSM models to RSG models is required. Both representations are graph based. Thus, nodes and relations can be mapped as follows:

- A single OSM node is mapped onto a RSG node and a *Transform* relative to a common origin node that represents a coordinate reference system in WGS84 representation. This design choice decouples the pose of an object (*Transform*) from the object representation (node) since a pose is not a property of an object but a relation between the object and a coordinate reference system. A benefit is that it also allows multiple *Transforms* connected with a node, *e.g.*, originating from different coordinate reference systems added by different robots.
- All OSM tags are converted to *Attributes* of the created RSG node with an `osm` prefix to indicate from which knowledge base they originate. Also the OSM specific IDs will be preserved in an Attribute `osm:node_id` to enable potential updates of the OSM nodes.
- OSM ways and relations are stored as *Connections* referring to nodes. Analogously to the transformation of OSM nodes, the IDs and tags of OSM ways and relations are preserved. In contrast to OSM, the RSG allows to dynamically update *Connections*, *e.g.*, when the target area was changed by the user. In this case a new pose value is inserted into the temporal cache of one or more *Transforms*.

The a priori mission map based on OSM is composed with dynamic information like the poses of the robots by storing all world model related data in the RSG. Since the RSG is synchronized between the robots, this enables a coherent access to world model information for all team members of a SHERPA mission.

B. Integration between Robot Scene Graph and KnowRob

The architecture of the integration is shown in Fig. 3. KnowRob is connected to the RSG with (i) a bridge to transform primitives from the RSG data model into assertions that are added to the knowledge base of Knowrob and (ii) a

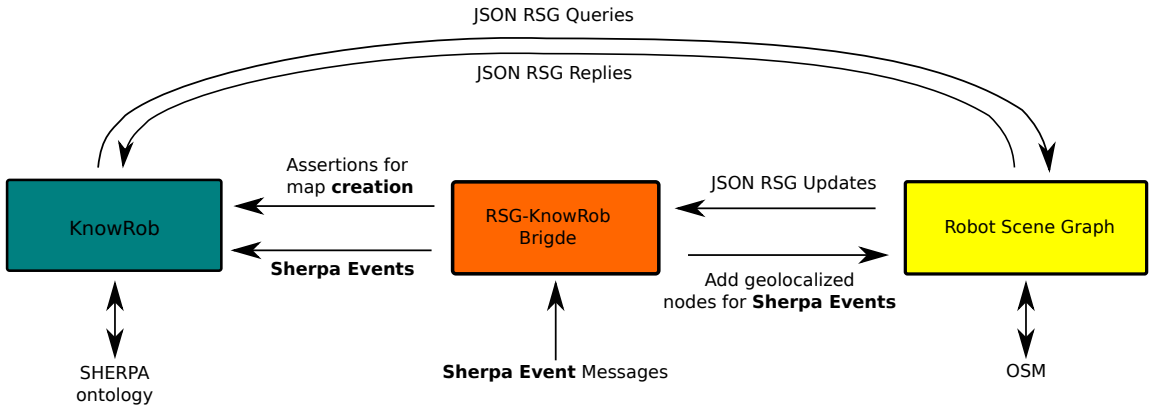

Fig. 3. Communication between the software components realizing the integration between the RSG and KnowRob.

query channel that allows to retrieve and modify information in the RSG as described in II-B. The bridge component serves two purposes: (a) it handles incoming events that are relevant for a SHERPA mission and (b) it translates RSG Updates into assertions.

In the first case, monitor components conforming to the Observer Software Pattern [16] are deployed in the bridge to monitor incoming data and messages in order to create events that the RSG, KnowRob, or both need to react to, e.g., a message that a victim was found. In the second case, RSG updates are sent from the RSG to the bridge whenever a relevant change occurred, *e.g.*, when a new object is added to the map. Then a msg containing the update encoded in a JSON string is sent from the RSG to the bridge, where it triggers the creation of a new assertion in KnowRob. The JSON-Prolog interface of KnowRob transmits the assertions. These interfaces are used to store the raw data and to map it to KnowRob's own class interpretation for further reasoning.

For that purpose two sub-ontologies were designed in KnowRob to represent the GIS data inside of KnowRob. The first one describes GIS data types that are directly asserted into the *prolog* data base. The second one describes additional knowledge that is inferred by the use of restrictions. This ontology describes the knowledge relevant for the SHERPA alpine rescue mission and can be replaced for other outdoor robotic applications. Additional ontologies can be added during runtime if they are needed.

The GIS data from the RSG can be directly asserted in KnowRob. This data is represented as classes of type node, way, connection, shape and transformation. At this stage all that contains information about the semantic connection and nodes is still hidden inside of the key-value pairs. These key value pairs describe the type of node, as well as additional information that can be exploited on demand.

The second ontology allows to extract semantic information from the raw data. This ontology defines new classes based on restrictions to the key-value pairs in the raw data. A path for the rover is defined as equivalent to a class that has a valid value in the *highway*-key, and does not have a key *blocked* with the value *true*, which might be asserted by the perception system.

This representation allows that a robot, like the rover, can query the knowledge base for a path that is traversable by it. This knowledge has not been explicitly defined, but is generated by KnowRob based on a combination of asserted data.

The following example shows a simplified version of the definition for a safe way. Such definitions can become quite complex and are constructed based on smaller sub rules.

```
<owl:Class rdf:about="#GIS-SafeWay">
 <owl:unionOf rdf:parseType="Collection">
  <owl:Restriction>
   <owl:onProperty rdf:resource="#GisProperty"/>
   <owl:allValuesFrom>
    <rdfs:subClassOf rdf:resource="#OSMWay"/>
    <rdfs:subClassOf rdf:resource="#Safe"/>
     ...
   </owl:allValuesFrom>
  </owl:Restriction>
 </owl:unionOf>
</owl:Class>
```

This allows to specify rules about traversability, dangerous areas and other information for a specific robot. This is possible by the fusion of multiple data sources like robot descriptions as well as height-, shape- and sensor-data. Such a rule can check the slope and the terrain of each connection in the path to generate a hypotheses about whether the path is traversable or not. A set of such definitions allow to generate a set of rules like this:

```
path_traversable_by(Robot, Path):-
  owl_individual_of(Path, 'Gis-SafeWay'),
  maximum_slope_for_robot(Robot, Slope),
  path_with_nodes(Path, Connections),
  forall(member(C, Connections),
    within_slope(C, Slope)).
```

This rule checks if the path is of the type *SafeWay*. If this is the case, it will try to validate that the maximum slope the robot can handle is bigger that the actual slope of each sub-connection.

If the operator needs the information whether the *rover* is endowed with the required capabilities in order to pass a generated *path*, she can query such rules from the outside with a call like:

```
capability_on_robot(chain_wheels,rover),
path_traversable_by(rover, path).
```

Other useful operations on the data are boolean operations on geometric shapes. The system can generate sub-shapes with the operations *intersection*, *union* and *difference*. These operations can be used to generate the set of all areas that are traversable and have a *landuse* that allows the robots to pass it or to generate a set of areas that have high probability to find a victim and are reachable by the robot. These operations use *computables* and depend on additional external programs that perform the actual mathematical operation. The actual operation is executed with a call like this:

```
shape_difference_of(targetArea,
                    scannedArea,
                    ToBeScanned).
```

All shapes, points and ways that are generated in this way can be sent back to the RSG and there they can be used by actual path-planners, other reasoners or the human operator for additional operations. The RSG ensures that this data is distributed between all agents, even if they do not have a running KnowRob instance. The following section provides a more detailed example on the rover.

IV. EXAMPLE

In order to illustrate the functionalities from the previous chapter, a more detailed example of common *Search and Rescue* task is shown. All code examples have been simplified to illustrate the functionality of the system.

The human operator wants the robot to execute a scan mission. He marks an area on his map to specify the search region `area_scan`, and delegates the task to the robot. The rover generates two main actions for this purpose. The first action is the navigation to the area of interest and the second one the scan of the desired area.

The robot's reasoning system relies on previously asserted knowledge about the environment when a new mission starts. The rover has access to Digital Elevation Maps (DEM) generated by laser scan data from other drones, since the meta data for each DEM is stored in the RSG and distributed among all team members. The OSM data is loaded into the RSG on start up as well.

Before the rover starts the actual navigation, it can check whether or not the target position lies within an area that is traversable for the robot. The robot first validates if the target `p_target` and starting points `p_start` are inside of a traversable area. For this reason it sends the query below:

```
sub_area_traversable_by(rover,
                        Area_A,
                        Area_Sub),
inside_of_area(p_start,Area_Sub),
inside_of_area(p_1,Area_A),
inside_of_area(p_2,Area_B),
path_traversable_by(rover, Path),
path_connects(Path, p_1, p_2).
```

If a valid path is returned it can be used by the low-level components to navigate to the goal. In addition to that the path is added to the RSG to visualize it to the human operator or to be used by other robots. The path is added to RSG exactly as a path from OSM, by adding the respective nodes and *Transforms* for waypoints, and relating them by a *Connection*. In order to better distinguish the different sources of knowledge a `kr` prefix is used for the *Attributes* that originate from *KnowRob*.

For the following scan action, the rover needs to segment the `scan_area` into valid sub areas `sub_scan_area_X` that are traversable by the robot.

```
area_by_slope(slope, AreaTraversable),
shape_intersection_of(area_scan,
                      AreaTraversable,
                      Sub_area_scan).
```

The robot can now use this information to scan all areas that it is able to traverse. All gathered knowledge about the scanned areas, traversable and untraversable terrain etc. is added to the RSG. The monitoring capability of the RSG propagates such changes immediately to the other robots to make it as fast as possible available for them as well.

V. DISCUSSION AND CONCLUSIONS

This paper presents a case study for integrating heterogeneous knowledge bases used in a search and rescue robotics project.

OpenStreetMap (OSM) has been used as a source for static map data and map related semantics and topological knowledge. OSM is free, extensible and composable, has a good coverage in most areas, and there are many extensions and tools already available for it. Therefore, it can be recommended as a starting point for robotic applications that require maps.

The RSG contains a *composition* of world model related data and knowledge originating from different knowledge bases. It integrates the relevant OSM related data by transforming the OSM data model into its own. Therefore, the OSM data is available for its reasoning a querying methods. In addition, since semantic tags are preserved, they can also be exploited for reasoning within the RSG or KnowRob. Of course these advantages did not come for free. An analysis of both model structures was required, and since they are both conforming to the same metamodel (they are both graph structures), a model to model transformation was defined and encoded. This conforms to model to model transformations using so called triple graph grammars [14], [15].

KnowRob with the SHERPA ontology has been used for abstract reasoning within the task domain. While the underlying knowledge bases of KnowRob and the RSG could potentially also be integrated by a model to model transform (the OWL representation of KnowRob and the

RSG model both represent graph structures), these two components are not just mere databases that answer queries but *active* components. *Active* means the components do not just passively provide knowledge or data but also have functionalities outside of the scope of the integration that are required by the overall system. *E.g.*, the RSG actively keeps the data and knowledge on multiple robots consistent while monitoring incoming data to trigger events or KnowRob can cause a robot to change its task execution based on new facts. Therefore, these two knowledge bases have not been transformed into *one* representation, but only the parts relevant for the application have been mapped onto each other. This has been encoded in a bridge component, which takes care of changing the representation of messages at runtime (see Fig. 3). This architecture allows both, the RSG and KnowRob, to continue their activities that are relevant for other parts of the overall system architecture while also allowing to update each others knowledge and data as well as using each others querying functionalities.

The implementation of this integration effort is not ready for official release, but can be found at `https://github.com/blumenthal/sherpa_world_model_knowrob_bridge`.

From our efforts, we have learned that there are different ways to integrate heterogeneous knowledge bases and data sources and for every application a combination of them is needed. In the following we will give tentative rules when to apply which method.

- If the purpose is to make knowledge and data available but the specific reasoning methods and activities of some source are not required, we have chosen to transform that knowledge into a framework that then uses it for its own reasoning. This requires the analysis of both representations and the creation of a (bidirectional) model to model transformation.
- The integration of active components that have their own specialized reasoning methods and functionalities is more complicated. One possibility is to duplicate all available knowledge by finding transforms and making it available to all components. However, that leads to largely redundant knowledge and data as well as an overhead to keep dynamically changing data consistent. In this use-case typically components do not make extensive use of each others' functionalities so ad-hoc integration is possible but does not scale. If the components only use few well defined functionalities of each other and memory is not a concern but communication between the components is, this is a viable option together with one or several bridges that transform data for all components. On the other end of the spectrum is the integration through flexible query interfaces where every operation is a query and no data is duplicated. This implies bidirectional communication and that the query itself has to comply to a metamodel [20]. Unfortunately, most existing systems offer a static API, rather than such a dynamic interaction. This solution can also become inefficient depending on the overhead and the reply times of the queries. However, these two cases are only the extremes of a continuum, which provides opportunities for application dependant trade-offs.
  In the presented use case, KnowRob provides a fixed, ROS-message based, binary encoded interface for its queries. This is an optimization for efficiency and ease of use in a specific use case but makes difficult to reuse in a different context. On the other hand, the RSG provides a flexible query interface that only defines a message format but not its content. That makes it easier to integrate and extend but requires an application dependant query processing. The bridge does the translation for the RSG message format to the KnowRob call, while the query processing for the communication with the RSG is done within KnowRob.
  We have chosen to duplicate (and thus transform) the map data for KnowRob and the RSG while dynamically changing data like robot positions (including its history, *i.e.*, trajectories) is stored only in the RSG and exchanged by explicitly querying for it when necessary. That allows to exploit the capabilities of both systems. The RSG handles (and answers queries for) dynamically changing data while KnowRob can detect missing information and query for it. KnowRob can then also add the outcome of its reasoning to the RSG via the query interface, *e.g.*, a changed goal position for a robot.
- A general design guideline for system and component level development is to design composable systems, in which each subsystem has exactly one meaning [19] and to keep the composability into a larger system in mind. In the presented use case the RSG and KnowRob can both load and access internal or external functionalities like specialized reasoning methods for geometrical reasoning.
- Bridges usually contain model transformations that are applied at runtime. So they require the same effort as the discussed model to model transformations.

The key issue is the insufficient modelling of software and data in robotics. Often, there are no formal models at all and if there are models, they are meant for people and not for automated reasoning. There are few common meta-models or schemas that would help with model to model transformations, which would help translating the plethora of formats for knowledge and data representations and even for creating bridges to connect legacy software with fixed interfaces. In other communities some solutions are available or under development. A few interesting examples are the linked data community [21] looking for generalized (query) interfaces, the ontology and database communities with their work on database/ontology alignment and merging [18], or the internet of things community [22] with their work on service discovery and sharing of heterogeneous data. However, it is difficult to find the relevant knowledge and then most of these solutions are also not interoperable. However, they can serve as inspiration for robotic solutions.

ACKNOWLEDGMENT

The authors acknowledge the support from the KU Leuven Geconcerteerde Onderzoeks-Acties *Model based intelligent robot systems* and *Global real-time optimal control of autonomous robots and mechatronic systems*, and from the European Union's 7th Framework Programme (FP7/2007–2013) projects *BRICS* (FP7-231940), *ROSETTA* (FP7-230902), *RoboHow.Cog* (FP7-288533) and *SHERPA* (FP7-600958).

REFERENCES

[1] Moritz Tenorth and Michael Beetz, "KnowRob – Knowledge Processing for Autonomous Personal Robots", In IEEE/RS International Conference on Intelligent Robots and Systems, pp. 4261-4266, 2009.

[2] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano, A. Sala, N. Tomatis, "The SHERPA project: smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments", In IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), College Station, Texas, USA, 2012.

[3] S. Blumenthal, N. Hochgeschwender, E. Prassler, H. Voos and H. Bruyninckx, "An approach for a distributed world model with QoS-based perception algorithm adaptation." In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015

[4] G. De Cubber, D. Doroftei, D. Serrano, K. Chintamani, R. Sabino, and S. Ourevitch, "EU-ICARUS: Developing assistive robotic tools for search and rescue operations", in IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013.

[5] M. Hakaly and P. Weber, "Openstreetmap: User-generated street maps". in IEEE Pervasive Computing, 2008, 7. Jg., Nr. 4, S. 12-18.

[6] M. Naumann and M. Bengel and A. Verl, "Automatic Generation of Robot Applications Using a Knowledge Integration Framework", in 41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK), 2010.

[7] T.N. Wong and C.W. Leung and K.L. Mak and R.Y.K. Fung, "Dynamic shopfloor scheduling in multi-agent manufacturing systems", in Expert Systems with Applications, Volume 31, Issue 3, October 2006, Pages 486-494.

[8] Beetz, M., Klank, U., Kresse, I., Maldonado, A., Mosenlechner, L., Pangercic, D., Ruhr, T., & Tenorth, M., "Robotic roommates making pancakes", in IEEE-RAS International Conference on Humanoid Robots, 2011

[9] BJ. Maitin-Shepard and M. Cusumano-Towner and J. Lei and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding", in IEEE International Conference on Robotics and Automation (ICRA), 2010.

[10] Antoniou G, van Harmelen F. "Web Ontology Language: OWL". In: Staab S, Studer R (eds). Handbook on ontologies. Berlin; New York: Springer, 2004: pp. 6792.

[11] Richard Cyganiak, David Wood, Markus Lanthaler. "RDF 1.1 Concepts and Abstract Syntax". W3C Recommendation, 25 February 2014. URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/. The latest edition is available at http://www.w3.org/TR/rdf11-concepts/

[12] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, ¡http://www.rfc-editor.org/info/rfc7159¿.

[13] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Franois Yergeau, and John Cowan. "Extensible Markup Language (XML) 1.1 (Second Edition)." W3C Recommendation, 16 August 2006. URL: http://www.w3.org/TR/2006/REC-xml11-20060816. The latest edition is available at https://www.w3.org/TR/xml11/

[14] Ekkart Kindler and Robert Wagner. "Triple graph grammars: Concepts, extensions, implementations, and application scenarios." in Technical Report, University of Paderborn, 2007.

[15] Anthony Anjorin and Erhan Leblebici and Roland Kluge and Andy Schürr and Perdita Stevens. "A Systematic Approach and Guidelines to Developing a Triple Graph Grammar." in Proceedings of the 4th International Workshop on Bidirectional Transformations co-located with Software Technologies: Applications and Foundations CEUR Workshop Proceedings; Vol. 1396, 2015.

[16] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, "Design Patterns: Elements of Reusable Object Oriented Software", Addison Wesley Publishing Company, 1995.

[17] L. Otero-Cerdeira, F. J. Rodriguez-Martinez and A. Gmez-Rodriguez, "A. Ontology matching: A literature review", Expert Systems with Applications, Elsevier, 2015, Pages 949-971

[18] E. Rahm, "Towards large-scale schema and ontology matching" Schema matching and mapping, Springer, 2011, Pages 3-27

[19] Dominick Vanthienen, Markus Klotzbuecher and Herman Bruyninckx. "The 5C-based architectural Composition Pattern: lessons learned from re-developing the iTaSC framework for constraint-based robot programming." JOSER: Journal of Software Engineering for Robotics 5.1 (2014): 17-35.

[20] Huahai He and Ambuj K. Singh, "Graphs-at-a-time: query language and access methods for graph databases." Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008.

[21] Markus Lanthaler and Christian Gtl. "Model your application domain, not your JSON structures." In Proceedings of the 22nd International Conference on World Wide Web (WWW '13 Companion), 2013.

[22] Fan Yang, Nelson Matthys, Rafael Bachiller, Sam Michiels, Wouter Joosen and Danny Hughes. "PnP: plug and play peripherals for the internet of things." In Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15). ACM, New York, NY, USA, , Article 25 , 14 pages, 2015.