# 10

# The Explorer System

Kristoffer Sjöö[1], Hendrik Zender[2], Patric Jensfelt[1], Geert-Jan M. Kruijff[2], Andrzej Pronobis[1], Nick Hawes[3], and Michael Brenner[4]

[1] Royal Institute of Technology (KTH), Centre for Autonomous Systems, Stockholm, Sweden
{krsj,patric,pronobis}@csc.kth.se
[2] DFKI GmbH, Saarbrücken, Germany
{zender,gj}@dfki.de
[3] Intelligent Robotics Lab, School of Computer Science, University of Birmingham, Birmingham, UK
nah@cs.bham.ac.uk
[4] Albert-Ludwigs-Universität Freiburg, Department of Computer Science, Freiburg, Germany
brenner@informatik.uni-freiburg.de

## 10.1 Introduction

In the Explorer scenario we deal with the problems of modeling space, acting in this space and reasoning about it. Comparing with the motivating example in Section 1.3 the Explorer scenario focuses around issues related to the second bullet in the example. The setting is that of Fido moving around in an initially unknown (Fido was just unpacked from the box), large scale (it is a whole house so the sensors do not perceive all there is from one spot), environment inhabited by humans (the owners of Fido and possible visitors). These humans can be both users and bystanders. The version of Fido that we work with in the Explorer scenario can move around but interaction with the environment is limited to non-physical interaction such as "talking". The main sensors of the system are a laser scanner and a camera mounted on a pan-tilt enabling Fido to look around by turning its "neck". Figure 10.1 shows a typical situation from the Explorer scenario.

The construction of spatial models from sensor data in the context of mobile robotics has been studied extensively in the literature. Simultaneous localization and mapping (SLAM) is by now a mature technology and it is not primarily in this field that the Explorer makes contributions. The kind of maps typically created by SLAM are, as discussed in Chapter 5, focused on providing the robot with means to localize and determine how to move from one place to another. While this is still required by the robot in the Explorer scenario, it is not the primary focus. Besides the obvious challenges in

H.I. Christensen et al. (Eds.): Cognitive Systems, COSMOS 8, pp. 395–421.
springerlink.com 

**Fig. 10.1.** The user shows the Explorer robot where the living room is. The robot's visualization of a similar situation can be seen on the right-hand side.

creating an integrated system, one of the motivations for the Explorer, as with the PlayMate, is to study the problems that occur when an intelligent robot must interact with humans in a rich and complex environment. In this case we focus on the design of models of space that are able to facilitate such interactions. In order to do this, the representation used by the robot must support the anchoring of spatial concepts shared between the robot and humans. Such spatial concepts may not be needed by and may not, for that matter, be available to a robot acting on its own. However, when communicating with a human, they play a key role in generating a shared understanding of space. For example, instead of the robot talking about an area delimited by a certain polygon it is more natural to talk about a certain room.

In the Explorer scenario spatial models are built using input from sensors such as laser scanners and cameras but equally importantly also based on human input. It is this combination that enables the creation of a spatial model that can support low level tasks such as navigation, as well as interaction. Even combined, the inputs only provide a partial description of the world. By combining this knowledge with a reasoning system and a common sense ontology, further information can be inferred to make the description of the world more complete. Unlike the PlayMate system, all the information that is needed to build the spatial models are not available to it sensors at all times. The Explorer need to move around, i.e. explorer space, to gather information and integrate this into the spatial models. Two main modes for this exploration of space have been investigated within the Explorer scenario. In the first mode the robot explores space together with a user in a home tour fashion. That is, the user shows the robot around their shared environment (Fido needs to know where things are and what stuff is called in his new home). This is what we call the *Human Augmented Mapping* paradigm. The second mode is fully autonomous exploration where the robot moves with the purpose of

covering space. In practice the two modes would both be used interchangeably to get the best trade-off between autonomy, shared representation and speed.

Another important aspect of the Explorer scenario is the ability to perform tasks autonomously. If robots like Fido is ever going to take the steps from toys to utilities they need to do something for us besides providing entertainment. Since the Explorer system does not have manipulation skills tasks are somewhat limited. The focus in the Explorer is not on performing a particular task to perfection, but rather acting within a flexible framework that alleviates the need for scripting and hardwiring. We want to investigate two problems within this context: what information must be exchanged by different parts of the system to make this possible, and how the current state of the world should be represented during such exchanges.

One particular interaction which encompasses a lot of the aforementioned issues is giving the robot the ability to talk about space. This interaction raises questions such as: how can we design models that allow the robot and human to talk about where things are, and how do we link the dialogue and the mapping systems?

### 10.1.1 Related Work

There are a number of systems that permit a robot to interact with humans in their environment. Rhino [1] and Robox [2] are robots that work as tour guides in museums. Both robots rely on accurate metric representations of the environment, and both have quite limited communicative capabilities. In the Explorer the communication with humans and reasoning about space are central elements. Examples of robots with more elaborate dialogue capabilities are RoboVie [3], BIRON [4], GODOT [5], WITAS [6] and MEL [7]. BIRON is endowed with a system that integrates spoken dialogue and visual localization capabilities on a robotic platform. This system differs from ours in the degree to which conceptual spatial knowledge and linguistic meaning are grounded in, and contribute to, situation awareness. In contrast, in our system, information from dialogue and situated contexts can be combined during processing of utterances [8]. Furthermore, whereas RoboVie and BIRON use finite state machines to model dialogue behavior, we combine information states [9], like GODOT; together with a task-oriented perspective, as WITAS or MEL. One more things that the sets the Explorer system aside from the above system is that the integratation mechanisms themselves are as important or maybe more important than the performance of the end product. That is, we want to study how to integrate a large set of components in a cognitve system in a flexible and scalable way rather then creating a system that can perform certain tasks well.

### 10.1.2 Outline

The outline of the rest of this chapter is as follows. In Section 10.2 we give an overview of the Explorer system, focusing mainly on outlining the differences to the PlayMate instantiation presented in the previous chapter. In

Section 10.3 we describe how the spatial model is acquired, how it can be used for conceptual reasoning and finally how cross-modal knowledge is represented and shared across the system. Section 10.4 details the specifics of planning in the Explorer domain. Finally, in Section 10.5, we describe in detail an example task and how different parts of the system contribute to the completion of that task. We present some conclusions in Section 10.6.

## 10.2 System Overview

This section gives an overview of the system structure used in the Explorer scenario. Figure 10.2 shows the subarchitectures (SAs) used and some of the more important data structures published in the working memories of the different SAs. Comparing with Figure 9.4, which describes the instantiation of the PlayMate scenario, we see that the scenarios share four SAs: ComSys SA for *communication* with the user, Binding SA for *binding* of information between modalities, Motivation and Planning SA for *motivation and planning* and the Conceptual map SA, for *ontological representations and reasoning*.

The SAs that have been removed from the PlayMate scenario are Manipulation SA, Spatial SA and Vision SA. There is no manipulation in the explorer scenario which eliminates the need for a dedicated SA. The Spatial SA in its current form deals with spatial relations in a tabletop scene; in the Explorer scenario the environment is large-scale and the Spatial SA is replaced by the Navigation SA, which handles *motion control* and the three lowest levels of the *spatial model* (see Chapter 5), and the Place SA which provides capabilities for *place recognition and categorization*.The Conceptual Mapping SA takes a more prominent role in the Explorer scenario than it does in the PlayMate scenario. Here it is used to represent large-scale space at an abstract level, allowing for natural language expressions to be related to places in the world – and their respective representations in the robot's lower level maps. The requirements on the visual processing system also differ considerably between the PlayMate and the Explorer. The camera is mobile in the Explorer which violates some of the assumptions in the Vision SA. The Object SA, dealing with *object detection* reuses some of the components from the Vision SA.

In the remainder of this section we will briefly outline the composition of those SAs in the Explorer that do not exist in the PlayMate.

### 10.2.1 Navigation SA

The Navigation SA hosts the three lowest levels of the spatial model, i.e., the metric map, produced by the SLAM Process; the navigation graph, and the topological map; both of which are produced by the NavGraphProcess. The metric map, represented as a line map, and the navigation graph are published in the working memory for other components to use. The SLAM Process also updates one structure for the current metric position and one for the current topological position of the robot. Each node is assigned an `AreaID` representing

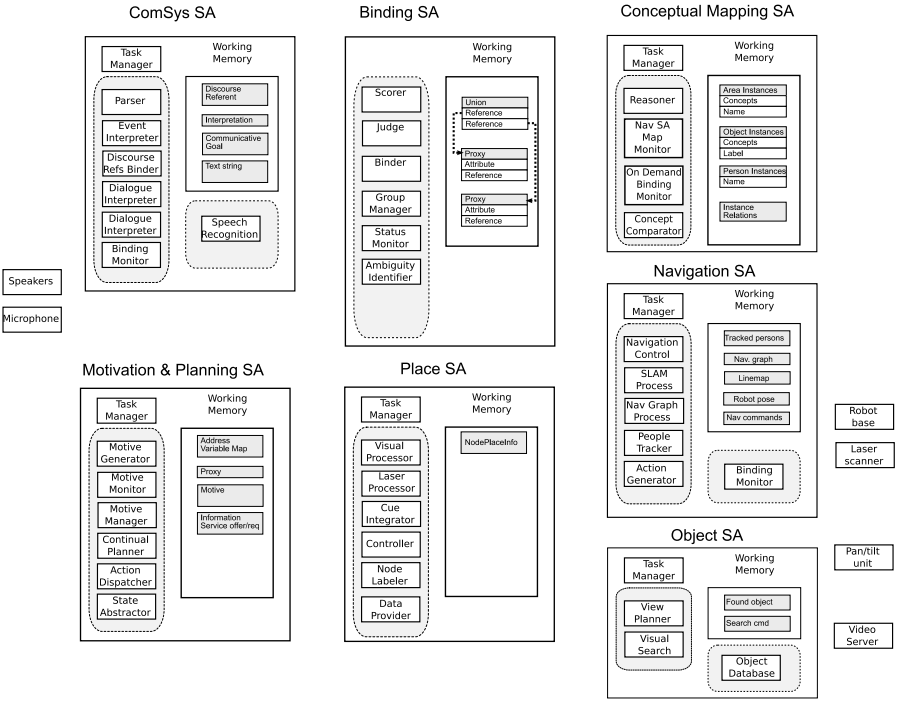


**Fig. 10.2.** An overview of the Explorer System. There are 7 sub-architectures, each of which corresponds to a functional and development unit.

the topological area it belongs to. Some nodes are gateway nodes (doorways), and are connected to nodes with differing `AreaID`. This implicitly defines the topological map: All navigation nodes with the same `AreaID`, taken together, correspond to one node (area) in the topological map, and each gateway node corresponds to an edge connecting two different areas.

*Detecting and Tracking People*

Detecting humans and keeping track of them is one of the key capabilities of a robot that aspires to interact with humans. In the Explorer system we use relatively simple means to realize this and make the assumption that there are not too many people close to the robot. For people detection we use a method similar to that used in [10, 11] which detects motion using laser scanner data. A new person is hypothesized when motion occurs far enough from any existing person. Each new person hypothesis is given a unique `PersonID`. Association between detected motion and a hypothesis is based on nearest neighbor matching. Tracking is accomplished by associating detected motion to hypothesis and using these as measurements in a Kalman filter that estimates the position and velocity of each person. Once a person disappears from the field of view of the sensor the hypothesis is removed. If the same

person reappears they will be given a new `PersonID` since the system is not able to identify individual people. Using the camera in combination with the pan-tilt for actively acquiring and using the appearance of the person could possibly deal with the problem.

*Motion Control*

Mobility is one of the most significant differences between the Explorer and the PlayMate. The Navigation Control module is based on the Nearness Diagram method [12] and executes the low level "go-to" commands. The target location is defined based on the current task which might be to follow a person, move to a specific point in space, etc. This module uses the navigation graph for the purpose of path planning by finding a path from the current robot position via the closest node in the graph, through the graph to the node closest to the goal and finally to the goal location.

### 10.2.2 Object SA

The Object SA collects the components involved in the finding of objects. It consists of a module for view planning and one for visual search. The view planning component creates a plan for which nodes to visit, in what order and in what direction to look given the assumption that objects can be found in places where the metric map registers obstacles. The visual search can be performed using a pan-tilt-zoom camera where an attention mechanism gradually guides the robot to zoom in closer and closer on object hypotheses where finally a SIFT based method is used for recognition. In the absence of a camera with zoom, the SIFT based matching algorithm can be used directly.

The Object SA can be used in two modes: one to perform active object search in the current region, which engages the above-mentioned modules, and another where the images are continuously processed to detect objects. In both cases the objects that are found are published on the working memory. This is detected by the Navigation SA, whereupon it in turn extends the spatial model with the new objects. This then propagates the information onwards to the Conceptual Mapping SA.

### 10.2.3 Place SA

The Place SA is responsible for assigning nodes and areas to one of predefined semantic place categories (e.g. an office, a corridor etc.) as described in Chapter 5. It gathers sensory data from a laser scanner and a camera and processes them in parallel using dedicated visual and laser processing components. The results for both sensors are integrated by a cue integration component, which provides beliefs about a place category for the current viewpoint. This information is, in turn, integrated temporally and spatially in a component responsible for assigning place labels to the nodes and areas.

Each area is initially categorized as unknown until the Place SA delivers a reliable classification result. The place categorization information is pulled from Place SA by the Navigation SA whenever the current node is changed in the navigation graph. The interaction with the Navigation SA is carried out via the structure `NodePlaceInfo` which contains the established place category for an area (and a node).

The area categorization provides important information when reasoning about space. As an example, object search is expensive and the place categorization can help to speed it up by allowing for a more selective search for objects. If the task is to populate the map with objects and it is known that the current area is a kitchen the search can be focused on typical kitchen objects.

### 10.2.4 Conceptual Mapping SA

The subarchitecture for Conceptual Mapping maintains a symbolic representation of space suitable for situated action and interaction. It represents spatial areas, objects in the environment, and abstract properties of persons in a combined A-Box and T-Box reasoning framework. Section 5.6 gives details on the underlying knowledge representation and knowledge processing principles.

For our implementation, we use the Jena reasoning framework[1] with its built-in OWL reasoning and rule inference facilities. Internally, Jena stores the facts of the A-Box and the T-Box of the ontology reasoner as RDF triples.[2] The knowledge base can be queried through SPARQL queries.[3] This reasoning framework is wrapped inside a CAST component, the *Reasoner*, which handles all necessary communication with the reasoning framework through SPARQL queries. There are a number of other components that mainly manage the interaction with other subarchitectures, namely the *Nav SA Map Monitor*, the *On Demand Binding Monitor* and the *Concept Comparator*. In Section 10.3.2 we will describe the run-time behaviour of the individual components of this subarchitecture.

### 10.2.5 The Robot Platforms

The Explorer system was developed and tested on two similar mobile robot platforms, Minnie from KTH and Robone from DFKI, seen in Figure 10.3. Both platforms are equipped with a SICK laser scanner, Minnie with an LMS 200 and Robone with an LMS 291. The laser scanner is the main sensor for the Navigation SA. Both robots also have a pan-tilt unit with a camera. Though Robone has a Videre stereo camera setup, only one of the cameras was used in the scenario.

[1] `http://jena.sourceforge.net`
[2] `http://www.w3.org/RDF`
[3] `http://www.w3.org/TR/rdf-sparql-query`


**Fig. 10.3.** The mobile platforms on which the Explorer scenario was developed and tested. Left: Minnie from KTH, Right: Robone from DFKI.

## 10.3 Spatial Modeling and Reasoning

In this section we will look closer at spatial modelling, spatial reasoning and maintaining relations between entities in the spatial model.

### 10.3.1 Map Acquisition

When building spatial models for human-robot interaction it is natural to adopt an interactive scheme for the acquisition process as well. We use the paradigm of *Human-Augmented Mapping (HAM)* [13, 14].

#### Human Augmented Mapping

The acquisition process using HAM can be described as a guided tour scenario. The user takes the robot on a tour of the environment and provides labels for areas and objects of importance. Wizard-of-Oz studies have investigated the interaction between human and robot in HAM [15, 16]. The findings concern, for example, the type of dialogue typically used and strategies to introduce new locations.

In a typical HAM scenario, the user walks up to the robot and initiates the mapping process with a command like “follow me!”. The robot continuously tracks the position of the user and follows them through the environment. As the robot moves, the spatial model is built from the sensor data.

One important aspect of our implementation of HAM is that the interaction is not master/slave-like, with the user initiating all interactions. The robot is also able to engage in, for example, clarification dialogues when it

encounters contradictory or ambiguous information. In [17] we describe in detail a HAM case study. In the example illustrated there, the robot detects a false door when passing by a table and a trash bin placed close together (see Figure 10.4). This makes the robot segment space into a new area. However, as it moves on it finds itself reentering a part of space classified as belonging to the previous area. This should not be possible without passing a door which leads to a contradiction; the robot can then ask if it really passed through a door recently.

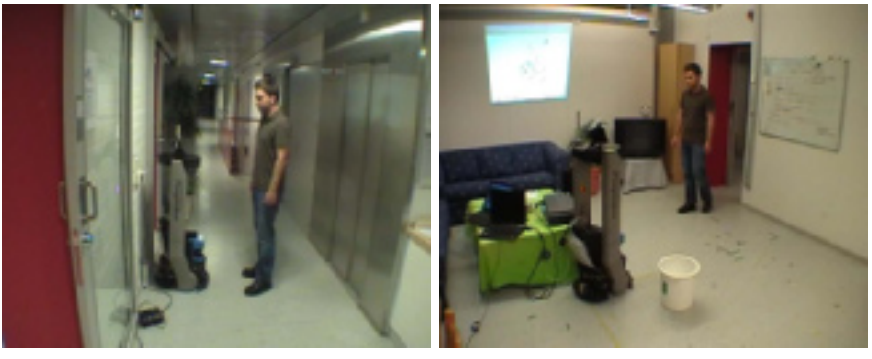

**Fig. 10.4.** Left: The user activates the robot at its recharging station. Right: The robot passes through an opening between a table and a trash bin interpreting the narrow opening as a door.

Throughout the HAM session, the user can query the spatial knowledge of the robot. The robot will be able to provide more and more precise descriptions of space as more information comes in. Although the HAM paradigm is very useful for acquiring a shared representation for spatial knowledge, it is rather time-consuming for the user. It is therefore natural that the user walks through the environment relatively quickly and introduces the main features. The robot could then revisit the environment later when not assigned a specific task and extend the model with more detected objects, better covered space, etc.

### Autonomous Exploration

In addition to the HAM scheme for map acquisition, the Explorer also possesses the ability to do autonomous exploration. The Explorer uses a frontier-based strategy [18] for autonomous exploration. Briefly put, the algorithm maintains a representation of the world where space is classified as `FREE`, `OCCUPIED` and `UNKNOWN`. The frontiers are defined as the borders between `FREE` and `UNKNOWN` space.

Exploration is considered complete when there are no more reachable frontiers. Exploration can be configured to be confined to one area. In this case, the exploration frontiers are considered unreachable if they require passing a door, i.e., changing area. When detecting a door, the robot will back up into the area from which it came and select a new frontier to explore.

Because the way the navigation graph is built requires the robot to move, exploration is not only about having the sensor see all parts of the room, but the robot needs to move there as well. The rationale behind requiring the robot to travel a path to make it part of the navigable space, i.e. a part of the navigation graph, is that some obstacles may not be detected by a sensor such as a laser scanner. Upward-facing IR-sensors for detecting tables are examples of sensors that could detect obstacles which the laser cannot see. Hence, in order to force the robot to actually visit all the space physically, we limit the range of the sensor for the purposes of updating the occupancy grid, which is used to define the frontiers, to 2m. This way, the robot will move across a large part of a room, even if it is able to see all parts of it from the door.

### 10.3.2 Acquiring the Conceptual Map

Upon start-up, the system comes with a rich conceptual ontology consisting of taxonomies of indoor area types, of commonly found objects, and of different spatio-topological relations that can hold between area instances and object instances. Moreover, the ontology contains a concept for persons and a relation that denotes ownership. This conceptual knowledge is held in the reasoner's T-Box. In case the system is started with a blank map, the A-Box of the reasoner is empty. Otherwise, it will contain area, person and object instances, and their relations, as contained in the loaded map. It is worth mentioning that the exact positions of persons are not represented in the conceptual map, just their ownership relations, which hold irrespectively of their current whereabouts.

As the robot learns about the world (either through interaction with the user, or through its autonomous map acquisition skills), this knowledge is added to the A-Box of the reasoner. To this end, the subarchitecture for Conceptual Mapping contains a component that constantly monitors the working memory of the Navigation subarchitecture, the *Nav SA Map Monitor* (see Figure 10.2). Whenever the Navigation subarchitecture identifies a new topological area, it creates a working memory entry for it. The working memory entries for areas on the *Navigation Working Memory* include a field that contains the most specific area category that can be autonomously extracted from sensory data. Initially, the working memory entries for areas will contain the neutral category "area". As soon as the Place subarchitecture for place categorization has reliably determined a more specific category (e.g. "corridor", or "office" etc.), it will overwrite the corresponding working memory entry for that area. The *Nav SA Map Monitor* is notified whenever an area working memory entry is created or an existing one is modified. In these cases, a

```
(area0 rdf:type Corridor),
(area1 rdf:type Library),
(area2 rdf:type Office),
(area3 rdf:type Office),
(...)
(nick rdf:type Person),
(nick name Nick), (nick owns area2),
(...)
(obj1 rdf:type Book),
(obj2 rdf:type Mug),
(...)
```

**Fig. 10.5.** RDF triples in the A-Box of the conceptual map (namespace URIs omitted)

new instance of the area's category is created in the reasoner, or a given one is modified respectively. A similar information flow is implemented for visually detected objects. Figure 10.5 shows a part of the A-Box in the Explorer example scenario.

So far we have only described how the Conceptual Mapping Subarchitecture reflects knowledge present elsewhere in the system, e.g. in the Navigation Subarchitecture. However, one of the main roles of this subarchitecture is to infer new or more specific knowledge based on partial information. The Description Logic definitions of the concepts in the T-Box express properties that form necessary and sufficient conditions for being instances of that concept. By combining and reasoning over instances and their relations, the reasoner can infer more specific concepts for those instances. In our current system, the reasoner can infer subconcepts for room instances based on the objects they contain, according to the principles described in Section 5.6.

The other components of the Conceptual Mapping SA, namely the *On Demand Binding Monitor* and the *Concept Comparator*, are used to make the information inside the Conceptual Mapping SA available to other subarchitectures. In the current system, the *On Demand Binding Monitor* registers its competences with the Planning & Motivation Subarchitecture, and upon request, contributes relevant data to the Binder working memory. The competences offered are *spatial reference resolution* and the possibility to provide *typical locations* for objects. We will detail the properties of these competences below. The *Concept Comparator* compares two **Concept** Binding Features according to their taxonomical relation in the T-Box of the *Reasoner*. The comparator will return `true` if the two concepts are ontologically equivalent, or if the concepts are in a taxonomical subsumption relation. It will return `indeterminate` if at least one of the concepts is unknown. Otherwise, i.e. if and only if both concepts exists but are not hierarchically related, the comparison result will be `false`.

### 10.3.3 Cross-Modal Spatial Knowledge Sharing

To enable different modalities to support each other with high-level knowledge, a number of protocols were implemented for the publishing of data on the binder.

#### Current Spatial Context

The spatial context denotes high-level data on the spatial state of the robot and of other entities in its current vicinity. This defines a class of binding proxies that are used by the rest of the system to reason and plan. The following proxies are always present on the binder:

- A *robot* proxy representing the physical robot itself
- An *area* proxy for the area the robot is currently in
- A *position* relation connecting the robot proxy with its area

In addition, the following are represented as appropriate:

- A *person* proxy for each person currently being tracked by the people tracking module
- *area* proxies for each person
- *position* relation proxies between the above
- An *object* proxy for each object belonging to an Area that is being represented
- *position* proxies connecting each object and its corresponding Area
- *Close* relation proxies between the robot and persons that are near to it

#### The Robot Proxy

There is always exactly one robot proxy on the binder. The only feature of this proxy is its **Concept**: `robot`. The proxy is designed to bind to proxies generated by ComSys, representing the listener – the "You" in a dialogue – and does so on the basis of its **Concept** feature (using concept comparators provided by the Conceptual Mapping SA).

#### Area Proxies

As described in Section 5.5.2, the navigation subsystem divides space into areas, based on door nodes. At the level of planning and reasoning, these areas constitute the basic units of spatial location. On the binder, the location of objects, persons and the robot itself is represented by position relation proxies connecting the entity with an area proxy.

Area proxies have the sole binding feature **AreaID**, a number that uniquely identifies the area to the navigation subarchitecture. This feature identifies the proxy as an area proxy, and also provides the information necessary for another subarchitecture to create a navigation command to move to the area in question.

### Object Proxies

An object proxy has the feature **Concept**, describing the particular class of object that it belongs to – for example, `book` or `mug`. The concept string corresponds to the argument that is used to issue a search command to the Object SA; thus, the feature can be used both for binding and for executing a plan.

### Person Proxies

Person proxies store the following binding features:

- **Concept**: always set to `person`
- **Location**: last observed metric position
- **PersonID**: unique identifier

The **Concept** feature provides basic binding control, making sure (through the Conceptual Mapping SA:s comparators) that only other types of person bind to the proxy. The **Location** provides the planner with an exact target for approaching a previously seen person in order to initiate a dialogue. It also helps in binding: as the system is incapable of distinguishing between individuals, it uses position to adjudicate binding. If a newly detected and a previously detected person proxy have locations that are nearer than a certain threshold, they are bound. Obviously, this makes the strong implicit assumption that people are immobile.

### Position Relation Proxies

The position proxy is a relational proxy, denoting the spatial relationship "X is in Y", where X is the proxy at the From side and Y the proxy at the To side of the relation. It has the following features:

- **Label**: always set to `position`
- **OtherSourceID**: set to the ID of the navigation subarchitecture; negated (see below)
- **TemporalFrame**: `PERCEIVED` for directly perceived entities; `ASSERTED` for others

The **OtherSourceID** feature is compared to the **SourceID** feature of other proxies on the binder, as one criterion on whether the proxies should bind or not. Since it is negated, this prevents the relation proxy from binding to other position proxies generated by the navigation subarchitecture.

**TemporalFrame** indicates the currency of the location information. While a person is being tracked, its position is regarded as perceived; if it goes out of perceptual context, yet remains on the binder (due to being bound to another subsystem's proxy), its status is changed to asserted.

This is so that the system can clean up old person proxies: if the asserted proxy binds to a perceived one (such as when the robot returns to a person it has previously had a dialogue with), the newer proxy "takes over" from the older one and the latter can be removed.

Objects' positions are considered perceived indefinitely after they are detected, since object detection is a discrete event, unlike the continuous tracking of persons.

The robot's position is always considered perceived.

### Closeness Relation Proxies

Whenever a person is near to the robot, a relation proxy is created between the two, with the features:

- **Label**: always set to `close`
- **OtherSourceID**: set to the ID of the navigation subarchitecture; negated
- **TemporalFrame**: always `PERCEIVED`

The closeness relation between robot and person is considered a prerequisite for initiating dialogue with the person. That is, if the planner is to plan a dialogue action, it must first ensure that closeness holds, by moving the robot up to the person if necessary.

As with position proxies, the "close" proxy has an **OtherSourceID** feature to prevent binding between the "close" proxies themselves. The relation is also always considered perceived; if a person ceases to be tracked by the robot, it is no longer considered "close" even though it may still be on the binder (with an asserted position).

### Supplementing Non-spatial Context

The preceding paragraphs define the current spatial context of the robot; that is, the entities that are perceptually relevant to it. In addition, there may be spatial proxies that are not part of this context, yet still remain on the binder. This is done in order to supplement other contexts, such as dialogue. Accordingly, any spatial proxies that are bound to proxies from other modalities will not be removed as they go out of spatial context. Any person or object that is thus sustained will also sustain its containing area and the relation between the two. Similarly, sustained area proxies will also sustain their contents in terms of objects.

For example, a person proxy will usually go out of context and will be removed when the person is no longer being tracked (having passed from the sensor scope of the robot). However, if this person proxy was bound to e.g. a ComSys proxy representing the speaker in a conversation, the spatial person proxy will not be removed. Consequently its position relation proxy and the proxy of the area where the person was last seen also have their removal

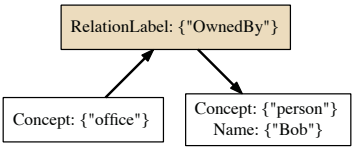


**Fig. 10.6.** Proxy structure generated by ComSys for "Bob's office"

from the binder suspended. This allows e.g. the planner to access cross-modal information relevant to its plan as it executes it or performs re-planning.

In addition to retaining spatial proxies that are bound to other modalities' proxies, it is sometimes necessary to conversely supplement other modalities' proxies by creating spatial proxies that will bind with them. For example, when ComSys registers the mention of an area that is not currently part of the spatial context, the Conceptual Mapping SA may resolve this mention to a known area instance (see below). It will then provide a proxy of its own, containing an **AreaID** feature of the known area; detecting this, the Navigation SA will in turn create a proxy for that area (provided it hasn't already got one on the binder).

As in the case of the direct spatial context, an area that is put on the binder in this manner will be accompanied by proxies for all objects known to be in that area, as well as position relations linking the objects to the area.

### Situated Resolution of Referring Expressions

When the user gives the robot an order that involves a reference to a location, ComSys will generate proxies that reflect the given verbal description. Figure 10.6 shows an example of such a proxy structure. The planner, however, can only send a navigation command to the Navigation SA that contains a goal location that is specified in terms of an `Area-ID`.

The *On Demand Binding Monitor* of the Conceptual Mapping SA offers the competence to resolve such a structural description of a location to an **AreaID**. Whenever the planner needs to resolve an **AreaID** feature for a proxy structure, it will send a request for resolution to the *On Demand Binding Monitor*. This component will then try to find an instance in the conceptual map that matches the structural description represented by the proxies. Internally, the proxy structure is translated into a SPARQL query to the A-Box of the *Reasoner*. Figure 10.7 shows an example of a SPARQL representation for "Bob's office".

The results of that query are then transformed into proxies and put on the binder. The proxies generated by the *On Demand Binding Monitor* of the Conceptual Mapping SA will contain the most specific concepts as **Concept** features, any other additional information stored in the A-Box, such as name information, and most importantly the `Area-ID`.

```
SELECT ?x0 ?x1  WHERE {
     ?x0 rdf:type Office. ?x1 owns ?x0.
     ?x1 rdf:type Person. ?x1 name 'Bob'.
     ?x1 owns ?x0.
}
```

**Fig. 10.7.** SPARQL query for "Bob's office"

On the binder the original ComSys proxies and their counterparts from the Conceptual Mapping SA are bound to a common union. The *On Demand Binding Monitor* then reports `task done` back to the planner, which then can continue to try to find a plan that satisfies the user's command.

### Providing Default Assumptions to the Planner

Sometimes the user gives the robot a task that involves an object whose current location is unknown to the robot. In such a case, the system can make use of the conceptual knowledge represented in the Conceptual Mapping SA. As mentioned earlier, the concepts in the ontology are defined through necessary and sufficient conditions, mostly involving the existence of certain objects in certain places. Here we make use of this encoded implicit knowledge to form assumptions about where certain objects can typically be found. For example, the concept of a library is defined as follows:

```
Class(Library complete Room
  restriction(hasObject valuesFrom(LibraryObject) minCardinality(5))
)

SubClassOf(Library restriction(hasObject someValuesFrom(Book)))
```

This means that `Library` is equivalent to the anonymous concept that fulfills the necessary and sufficient properties of being a subconcept of `Room` and containing at least 5 `LibraryObject` instances. Moreover, a `Library` has the necessary condition of containing `Book` instances. The latter one, however, is not a defining (i.e. necessary and sufficient) property because books can also be found elsewhere.

In any case, such a definition allows the system to form a hypothesis that `Book`s and other `LibraryObject`s can be expected in a `Library`. Consequently the *On Demand Binding Monitor* registers its competence to provide typical locations to the Planning & Motivation SA.

When queried by the planner to provide a typical location for a given object, a SPARQL-query to the T-Box of the *Reasoner* is constructed and executed. Figure 10.8 shows a query for the typical location of books.

If a typical location is found, the *On Demand Binding Monitor* creates a proxy for the location that contains its most specific concepts. In our example this would be `Library`. It also creates a proxy for the concept in question (`Book`

```
SELECT DISTINCT ?defaultLoc WHERE {
     ?defaultLoc rdfs:subClassOf ?blankNode.
     ?defaultLoc rdfs:subClassOf oe:Area.
     FILTER (!isBlank(?defaultLoc)).
     FILTER (?defaultLoc != owl:Nothing).
     FILTER (isBlank(?blankNode)).
     ?blankNode owl:someValuesFrom ?defaultObjClass.
     Book rdfs:subClassOf ?defaultObjClass.
}
```

**Fig. 10.8.** SPARQL query for the typical locations of books

in our example), and a Location relation between them with the additional restriction that it has a **TemporalFrame** feature with value `TYPICAL`. This denotes that the presence of such an object is not guaranteed to hold at a specific point in time, but can be assumed to be typically the case. It is then up to the planner to use this information to execute an action, e.g., a visual search in such a place where the object in question is likely to be found, in order to instantiate this "typical" knowledge with perceived information.

## 10.4 Planning

The Motivation and Planning Subarchitecture is general-purpose, and has therefore been reused (with different configuration information) in both the Explorer and PlayMate systems (see Chapter 9 for a more detailed description). The data the subarchitecture reasons about, however, is highly domain-specific. In the Explorer scenario, this includes *goals* referring to spatial concepts, *beliefs* about object positions and spatial relations, and *actions* to perceive and manipulate the environment of a mobile robot. The goals, beliefs and actions are represented in the symbolic planning language MAPL (see Chapter 6). The symbols used are maintained by the Address-Variable Map (AVM, see Chapter 2) such that they can be mapped back to binding proxies and, consequently, to component-specific representations later during plan execution.

In the Explorer scenarios described in this chapter, the main motivation for the robot to act is usually provided by an extrinsic source, i.e. a human user gives the robot a task. The ComSys generates appropriate proxies on the Motive working memory which are then translated into a planning goal using the AVM.

Planning starts with the creation of an initial planning state from the contents of the current state of the binder. The task of the planner is then to determine a sequence of actions whose execution from this state will achieve the goal. Interestingly, the planning problems arising in the Explorer scenario are characterised by a large degree of uncertainty and incompleteness in knowledge: the map may still be incomplete, object locations might be unknown and the observability of the environment is severely limited. As explained in

Chapter 6, the planner actively tries to reduce such gaps in the robot's knowledge by using a *continual* planning approach: the planner repeatedly switches from planning to execution in order to gather additional knowledge. Based on the information gained, the planner first revises its current state and, subsequently, its plan. It can then execute this plan further (possibly switching back to planning later again) until a goal has been achieved.

A plan consists of both external (physical) actions and internal (i.e. processing) actions. Physical actions are often scenario-specific. For the Explorer they include:

- *Follow person*: Track and follow a human
- *Approach person*: Move to the proximity of a person
- *Gain attention*: Attract the attention of a human (e.g. say "Excuse me")
- *Move*: Move to a given area in the map
- *Object search*: Perform object search for a given object
- *Inform*: Verbalize and transfer information on an object's position to a human that is close-by

Internal actions are mostly concerned with the task-driven extension of the planning state, i.e. the querying of subarchitectures for additional information that may be relevant for the problem at hand. In the current Explorer scenario, it is mainly the Conceptual Mapping Subarchitecture that is queried for default information, e.g. about where books are usually found. While this kind of information could be provided to the planner in the initial state, this would lead to an information overload: there is just too much information that the different parts of the system could provide to the planner – often at high processing costs, yet relevant only to few tasks (e.g. visual information that has to be extracted from camera data). Thus, instead of generating all this potentially irrelevant data in advance, the continual planner will determine possibly relevant sources of information on its own as part of the initial planning phases in the continual planning process. When these information gathering actions have been executed, a new plan is generated in the next planning phase that exploits the new information.

If a subarchitecture provides some behaviour (sensing, acting, reasoning, etc.) that is to be used by the planner, it needs to define two interfaces to this action for the planner:

- A MAPL action in the planning domain including preconditions, parameters and effects
- An action dispatcher, which translates a request from the planner to execute a specific MAPL action, i.e. with all parameters instantiated, into whatever format is required to set the subarchitecture in action.

During plan execution, the planner calls the appropriate action dispatchers to map the MAPL actions into the local representation used by the executing subarchitecture. For example, an action that in a MAPL plan is described as

```
PhysicalAction motmon0: approach-person motmon4 area_id_0
```

```
Human (H) approaches robot (R) who is idling in the corridor.
H: 'Find me the Borland book.'
R: 'OK.'
R turns and moves off, going to the door into the library. It
enters the door.
R moves about the room, turning to face different directions at
each location, searching for the Borland book.
R detects the book.
R moves back out into the corridor and up to H.
R: 'The Borland book is in the library.'
```

**Fig. 10.9.** An example of the Explorer performing an object localization task

contains the planner symbols `motmon0`, `motmon4` and `area_id_0`, which correspond to the robot proxy, the user person proxy, and the area proxy on the binder, respectively. The action dispatcher uses the AVM to find the proxy corresponding to `motmon4`, extract the **Location** feature for the person and issue a low-level movement command to the Navigation SA to move to that location.

## 10.5 Scenario: Find Object

Here, we describe in detail an example of a task performed by the robot, and how the different parts of the system contribute to fulfilling this task.

The robot is ordered by a user to "Find me the Borland book". It moves off to look for the book, visually locates it, and returns to report its findings. The externally apparent features of the task are described in Figure 10.9.

### Initial Binding State

Before the order "Find me the Borland book" is spoken, the following entities are represented on the binder:

- The robot
- The area corresponding to the corridor (with **AreaID** #1)
- A person
- A Position relation connecting robot and area (TemporalFrame `PERCEIVED`)
- A Position relation connecting person and area (TemporalFrame `PERCEIVED`)
- A Close relation connecting robot and person (TemporalFrame `PERCEIVED`)

Note that no objects are present, nor are other areas except the current area. A snapshot from a system run illustrates the initial state (Figure 10.10).

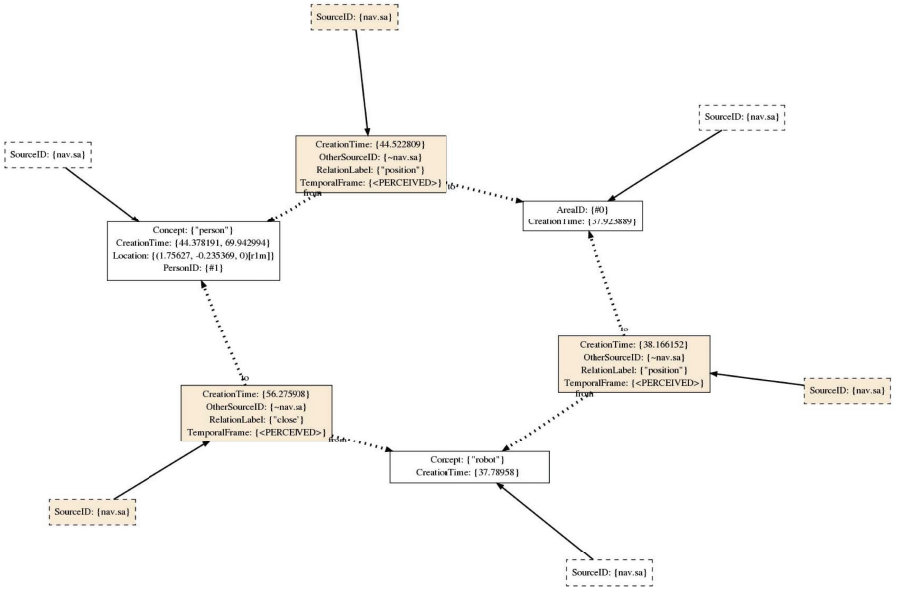


**Fig. 10.10.** Initial binding state

### Processing Utterance

As ComSys receives and interprets the user's phrase "Find me the Borland book", it adds the following proxies to the binder, corresponding to the different parts of the utterance.

- The robot itself, being the recipient of an order, is represented by a proxy with **Concept** `addressee`, which binds to the robot proxy already present.
- The word "me", referring to the speaker, generates a "person" proxy identified by the **Name** feature `I`.
- The expression referring to the book is represented by a "Borland_book" proxy, which is not bound to any other proxies at this point.

### Motive Generation

The phrase "Find me the Borland book" is used to generate the motive that will provide the planner with a goal state, as described in Chapter 2: ComSys writes proxies to Motivation SA working memory corresponding to the semantic interpretation of the command. The motive generator then creates a motive structure, which the motive manager turns into a planning goal.

The planning goal resulting from the above command is an epistemic one, saying, in words, that the user needs to know the position of the Borland

book.[4] In the planning language MAPL (see Chapter 6) this goal is represented as follows:

```
(:goal (K motmon4 (perceived-position motmon6)))
```

where the symbols `motmon4` and `motmon6` represent the user and the book, respectively. These symbols are maintained by the Address-Variable Map (AVM, see Chapter 2) which allows the planner to refer back to their respective source modalities during execution.

**Continual Plan Creation**

Planning starts with the creation of an initial planning state from the contents of the current state of the binder. The planner will determine a sequence of actions whose execution from this state will achieve the goal.

Planning in both the Explorer and the PlayMate system is a continual process, i.e. the plan is revised repeatedly during its execution. Plan revision occurs due to external reasons (exogenous events, unexpected execution results) or because internal state changes enable the planner to fill in details in its plan that have been deliberately postponed before.

The early phases of continual planning in this Explorer scenario can be described as means-end reasoning to determine necessary information for more detailed planning. For example, the planner first forms a very abstract plan which, in words, can be expressed as "determine the possible position of the book by querying some person or subarchitecture who supposedly knows, then verify this information by actually going there and identifying the book. Finally, go back and relate that information to the user." The continual planning process thus first queries the internal Conceptual Map Subarchitecture which provides information about default locations of books. This new information will trigger a plan revision (for details of this process, see Chapter 6) that leads to a new, more detailed, plan that involves concrete movement to the library and a search for the object there. If execution of this plan fails, e.g. because the book is not in the library, this is detected during plan monitoring and leads to another plan revision. The planner will then rely on sources of information other than Conceptual Mapping, e.g. humans (except the user). For the planner, querying a human and querying another subarchitecture about some information are identical. Both behaviours are planned as `ask-val` actions, the only difference to the planner being the addressee and the state variable the query is about. Likewise, providing information to another agent is realised by the same planning operator `tell-val`, regardless of whether this agent is a human or another subarchitecture.

While humans and other external agents are mostly treated the same as internal components, the planner makes one important distinction: external agents will usually be given *acknowledgements* when requests have been accepted and when achieved. To that end, the planner implements the Continual

[4] This is a convenient interpretation of "Find me the Borland book" because the robot cannot pick anything up.

*Collaborative* Planning algorithm presented in Section 6.7. The realisation of acknowledgements is not determined by the planner and is usually realised by the ComSys.

The plan created initially upon receiving the order "Find me the Borland book" consists of the following steps:

1. Acknowledge command acceptance to user
2. Have the Conceptual Mapping SA provide a default position for the book
3. Search the position provided for the book
4. Tell the user the perceived position of the book

### Plan Execution and Revision

The first actions in the plan (acknowledging the new command and querying for default information about the position of the book) are dispatched to the appropriate execution modules (ComSys and Conceptual Mapping SA, respectively). As a result, ComSys produces the utterance "OK" and the Conceptual Mapping SA augments the binder state with the information that the Borland Book, being a book, will typically be found in a library. This is represented by a proxy of **Concept** `borland_book` and a proxy of **Concept** `library`, connected by a relation `position`, with `TYPICAL` status.

The Conceptual Mapping SA also volunteers the specific information it has on libraries; namely, the fact that it knows about an area that is a library, with a specific **AreaID** (#1). It publishes this information in the form of another proxy with **Concept** `library` and **AreaID** #1. On the binder, this information is bound together into a structure whose meaning is "The Borland book will typically be in area #1" (see Figure 10.11).

The addition of this new knowledge to the planning state will lead to the expansion of the assertion `object-search-abstract` which abstracted from the actual position of the book as long as there was no hypothesis for its position. When such a hypothesis exists, this assertion is no longer allowed to be used (see Chapter 6), i.e. the planner is forced to reason in a more detailed fashion with respect to the book position. After replanning, a more concrete plan is produced:

1. Move to area #1
2. Search the current room for the book
3. Get back to user (area #0)
4. Tell the user the perceived position of the book

Note that by now, the planner also is certain that for finding the book the robot must move away from the user. Thus it includes an action for getting back to him later in the plan.

Using the correct **AreaID**, the planner issues a navigation command "Go to area #1" to the Navigation SA, and the robot moves into the library. The planner then proceeds to issue an object search command to the Object SA.

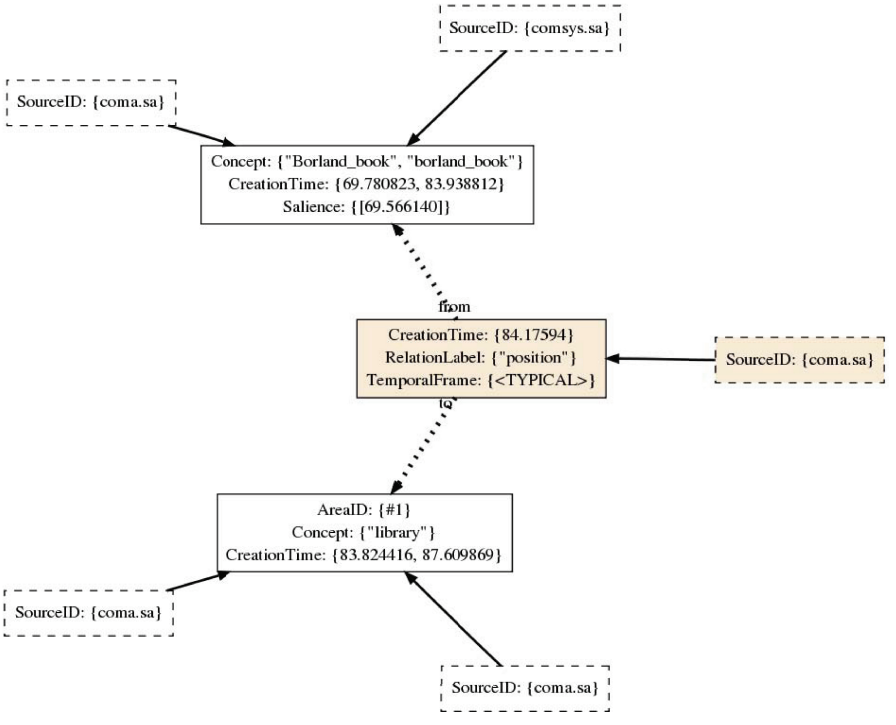


**Fig. 10.11.** Binding representation of hypothetical position of Borland book

The robot searches the room as described in Section 5.7.1. Once the object is found, the Navigation SA adds it to the navigation graph. Since it is part of the current spatial context, it is also exported to the binder in the form of an object proxy, connected to the room's proxy by a new position proxy. This position proxy has `PERCEIVED` temporal frame.

The new proxies bind to the old complex, resulting in the structure in Figure 10.12. Plan monitoring verifies that the search action was successful. Because the perceived book proxy (with source ID `nav.sa`) has been bound to the proxy from ComSys, the Planner can surmise that the position of the former applies also to the latter. Thus, the robot now knows the position of the Borland book that the user mentioned, which means the precondition now holds for "Tell the user the perceived position of the book". Consequently, plan execution goes on to the "Move to the user" action. If object search had failed, this would have invalidated the plan and triggered replanning.

The person proxy on the binder put there by the Navigation SA remains, even though it is no longer being tracked, because it is bound to the user's ComSys proxy (see Figure 10.13). Thus, the planner can read the **Location** feature of the person from the binder, and issue a navigation command to

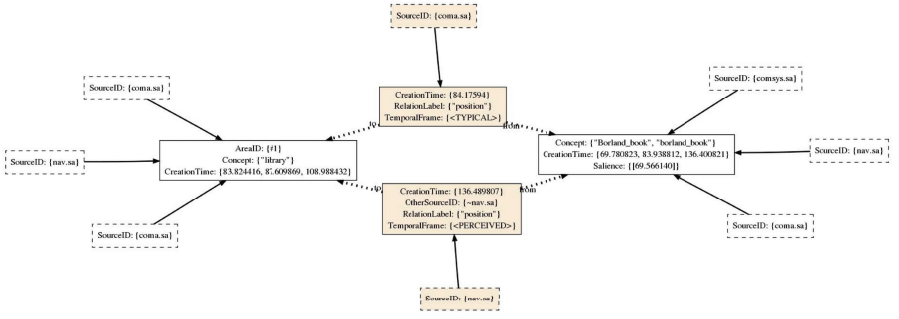


**Fig. 10.12.** Visually confirmed information about the book is added to the binder

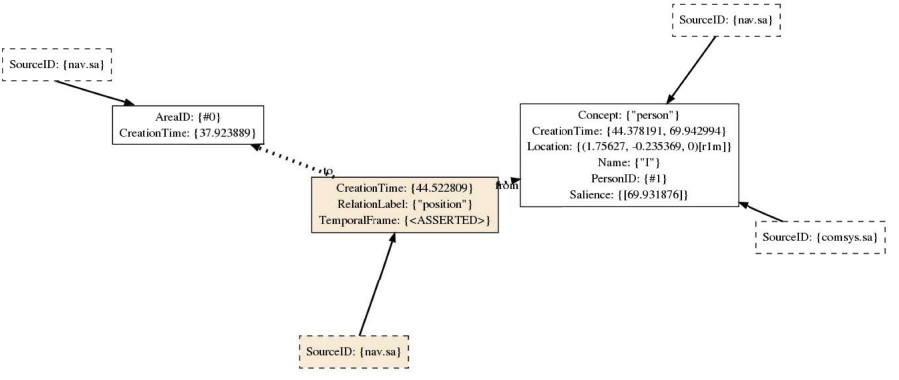


**Fig. 10.13.** Retained binding data about the user

the Navigation SA to go to this location. Once there, it calls upon ComSys to formulate a response to the user's request, using the binder contents (Figure 10.12) to do so.

## 10.6 Conclusions

In this chapter we have presented one more instantiation of the CAS architecture schema in the form of the Explorer system. The same software framework and many of the components are common with the PlayMate instatiation. While the PlayMate and the Explorer share the underlying framework and many components, the scenarios/tasks are in fact quite different. This provides evidence that CAS/CAST are general tools and can be used for a wide variety of problems.

In this chapter we have also shown how we can share representations across different modules by employing abstraction and various strategies for binding knowledge, that is associating knowledge from one sub-system with that from another. We showed how the system can bind not only with explicitly

perceived knowledge but also with ontological knowledge. This provides a mechanism to, for example, fall back to knowledge about the typical location of objects and thus provide a working hypothesis for a task even when the location of an object is not known.

The abstraction of knowledge also provided the means to move away from the hard-coded coupling between user input and action and use a much more flexible solution using a planner.

With the Explorer system, we also wanted to investigate how to create spatial representations which could bridge the gap between low-level robot control, and the qualitative ways in which humans tend to understand space. We approached this problem from a system point of view, looking at how the combination of different information sources could help to bridge that gap – building up a more comprehensive sense of space.

Following out that approach, we have found that integrating different modalities leads to significant synergies in building up a more complete understanding of the spatial organization of an environment, particularly towards a semantic understanding. Synergetic effects could be observed in information sources complementing each other, and in disambiguating interpretations. These synergies happen over time, and have highlighted an important requirement for spatial knowledge representation and reasoning. Namely, knowledge must not, and cannot, be irrevocable. Spatial reasoning appears to be inherently non-monotonic. The robot needs to be able to retract earlier inferences, to prevent that erroneously acquired or asserted knowledge leads to irrecoverable errors in inferred knowledge.

Synergies only arise when we integrate many components. And that integration brings not only more complete knowledge and more capabilities, it also increases complexity and presents problems due to the fact that the real world is unpredictable to some extent. For example, in a scenario where the robot continuously interacts with a user and is facing her/him most of the time, the information content of the sensor input suffers as the user occupies a large part of the field of view. In our case, the camera was mounted on a pan-tilt unit and could have been used to actively look for objects and build a metric map using visual information while following the user. However, this conflicts with the use of the camera to indicate the focus of attention on the user. As a result, most of the time the camera only sees the user and not the environment. The user's presence not only disturbs the visual object recognition but also influences the performance of the multi-modal place classification. From this practical issue, we can derive again more fundamental issues too. Integration should not only lead to synergy, but also to robustness and plasticity. In forming more complete interpretations, dependencies between modalities should not be static. In situ, a robot should be able to resort to alternative means for perception. And over time, a robot should be able to complete its incomplete observations, e.g. through autonomous exploration.

In addition to such practical issues, the experiments we ran in real environments highlighted new requirements for the system. For example, spatial

referencing needs to be improved in both directions of the communication and using several modalities. This would allow the user to indicate a specific object through, e.g., gesture or gaze direction when saying "This is X".

## References


1. Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S.: Experiences with an interactive museum tour-guide robot. Artificial Intelligence 114(1-2)
2. Siegwart, R., et al.: Robox at expo.02: A large scale installation of personal robots. Robotics and Autonomous Systems 42, 203–222 (2003)
3. Ishiguro, H., Ono, T., Imai, M., Maeda, T., Kanda, T., Nakatsu, R.: Robovie: an interactive humanoid robot. Int. J. Industrial Robotics 28(6), 498–503 (2001)
4. Haasch, A., Hohenner, S., Huewel, S., Kleinehagenbrock, M., Lang, S., Toptsis, I., Fink, G.A., Fritsch, J., Wrede, B., Sagerer, G.: Biron - the Bielefeld robot companion. In: Prassler, E., Lawitzky, G., Fiorini, P., Haegele, M. (eds.) Proc. Int. Workshop on Advances in Service Robotics, pp. 27–32. Fraunhofer IRB Verlag, Stuttgart (2004)
5. Bos, J., Klein, E., Oka, T.: Meaningful conversation with a mobile robot. In: Proceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003), Budapest, Hungary (2003)
6. Lemon, O., Bracy, A., Gruenstein, A., Peters, S.: A multi-modal dialogue system for human-robot conversation. In: Proceedings of the Second Meeting of the North American Chapter of the Association of Computational Linguistics (NAACL 2001), Pittsburg, PA (2001)
7. Sidner, C., Kidd, C., Lee, C., Lesh, N.: Where to look: A study of human-robot engagement. In: Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI), pp. 78–84 (2004)
8. Kruijff, G., Lison, P., Benjamin, T., Jacobsson, H., Hawes, N.: Incremental, multi-level processing for comprehending visually situated dialogue in human-robot interaction. In: Proceedings of the Symposium on Language and Robotics (LANGRO 2007), Aveiro, Portugal (2007), `http://cognitivesystems.org/cosybook/chap10.asp#Kruijff/etal:LANGRO2007`
9. Traum, D., Larsson, S.: The information state approach to dialogue management. In: van Kuppevelt, J., Smith, R. (eds.) Current and New Directions in Discourse and Dialogue. Kluwer Academic Publishers, Dordrecht (2003)
10. Lindström, M., Eklundh, J.-O.: Detecting and tracking moving objects from a mobile platform using a laser range scanner. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), Wailea Maui HI, USA, vol. 3, pp. 1364–1369 (2001)
11. Wang, C.-C., Thorpe, C.: Simultaneous localization and mapping with detection and tracking of moving objects. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002), vol. 3, pp. 2918–2924 (2002)
12. Minguez, J., Montano, L.: Nearness diagram navigation (ND): Collision avoidance in troublesome scenarios. IEEE Transactions on Robotics and Automation 20(1), 45–59 (2004)

13. Topp, E.A., Christensen, H.I.: Topological modelling for human augmented mapping. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), Beijing, China (2006)
14. Kruijff, G.-J.M., Zender, H., Jensfelt, P., Christensen, H.I.: Situated dialogue and understanding spatial organization: Knowing what is where and what you can do there. In: Proc. of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Hatfield, UK, pp. 328–333 (2006),
http://cognitivesystems.org/cosybook/chap10.asp#kruijff/etal:2006-roman
15. Topp, E.A., Hüttenrauch, H., Christensen, H., Severinson Eklundh, K.: Bringing together human and robotic environment representations – a pilot study. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China (2006)
16. Shi, H., Tenbrink, T.: Telling rolland where to go: Hri dialogues on route navigation. In: Workshop on Spatial Language and Dialogue (5th Workshop on Language and Space), Delmenhorst, Germany (2005)
17. Kruijff, G.-J.M., Zender, H., Jensfelt, P., Christensen, H.I.: Situated dialogue and spatial organization: What, where... and why? International Journal of Advanced Robotic Systems, special section on Human and Robot Interactive Communication 4(2),
http://cognitivesystems.org/cosybook/chap10.asp#kruijff/etal:jars
18. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proc. of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, pp. 146–151 (1997)