Planning and Acting with an Integrated Sense of Space

N. Hawes[1] and **H. Zender**[2] and **K. Sjöö**[3] and **M. Brenner**[4] and **G.J.M. Kruijff**[2] and **P. Jensfelt**[3]

[1]Intelligent Robotics Lab, School of Computer Science, University of Birmingham, UK

[2]Language Technology Lab, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany

[3]Centre for Autonomous Systems, Royal Institute of Technology (KTH), Stockholm, Sweden

[4]Institute for Computer Science, Albert-Ludwigs-Universität, Freiburg, Germany

[1]nah@cs.bham.ac.uk, [2]{zender, gj}@dfki.de, [3]{krsj,patric}@csc.kth.se, [4]brenner@informatik.uni-freiburg.de

Abstract

The paper describes PECAS, an architecture for intelligent systems, and its application in the Explorer, an interactive mobile robot. PECAS is a new architectural combination of information fusion and continual planning. PECAS plans, integrates and monitors the asynchronous flow of information between multiple concurrent systems. Information fusion provides a suitable intermediary to robustly couple the various reactive and deliberative forms of processing used concurrently in the Explorer. The Explorer instantiates PECAS around a hybrid spatial model combining SLAM, visual search, and conceptual inference. This paper describes the elements of this model, and demonstrates on an implemented scenario how PECAS provides means for flexible control.

1 Introduction

Recently there has been an enormous increase in R&D for domestic robot assistants. Moving beyond the Roomba, more complex robot "gophers" are envisioned, to assist in performing more demanding tasks in human environments. To achieve this vision, the study of integrated robotic systems that fulfill many different requirements is necessary.

Research on individual aspects of such systems has yielded impressive robots, e.g. the museum guides Rhino [Burgard *et al.*, 2000] and Robox [Siegwart and et al., 2003], or the in-store assistant ShopBot [Gross *et al.*, 2008]. Other robots, like RoboVie [Ishiguro *et al.*, 2001], Mel [Sidner *et al.*, 2004], BIRON [Peltason *et al.*, 2009], or our CoSy systems [Hawes *et al.*, 2007; Kruijff *et al.*, 2007] provide capabilities for the robot to interact with a human using spoken dialogue. As impressive as they are, all these systems lack the wide range of capabilities needed by a versatile robotic assistant. Producing such a system by integrating the results of specialized subfields such as control, perception, reasoning, and dialogue remains a major challenge to AI and robotics.

If we wish to build a mobile robotic system that is able to act in a real environment and interact with human users we must overcome several challenges. From a system perspective, one of the major challenges lies in producing a single intelligent system from a combination of heterogeneous specialized modules, e.g. vision, natural language processing, hardware control etc. Ideally this must be done in a general-purpose, extensible and flexible way, with the absolute minimum of hardwired behaviors. This both allows solutions to be reused in different systems (allowing an understanding of the design trade-offs to be obtained), and for the same system to be altered over time as requirements change. Additionally, taking account of the "human in the loop" poses the challenge of relating robot-centric representations to human-centric conceptualizations, such as the understanding of large-scale space [Kuipers, 1977].

In this paper we present PECAS (see Section 2), our novel approach to integrating multiple competences into a single robotic system. PECAS allows us to address many of the previously described problems in an architectural way, providing an approach that is ultimately resuable in other robots and domains. For a general-purpose architecture to be deployed it must be *instantiated* with task-specific content. Section 3 presents the Explorer system, our instantiation of PECAS in an interactive mobile robot. Following this we use the Explorer instantiation to present examples of PECAS as a control system (in a general sense). Section 4 presents a complete system run from our implementation, demonstrating how the flow of information and control passes between low and high levels in our system. Section 5 discusses control in PECAS in general, and the strengths and weaknesses of our approach.

2 The PECAS Architecture

Our recent work on intelligent robotics has led to the development of the PlayMate/Explorer CoSy Architecture Sub-Schema (**PECAS**). PECAS is an information-processing architecture suitable for situated intelligent behavior [Hawes *et al.*, 2009]. The architecture is designed to meet the requirements of scenarios featuring situated dialogue coupled with table-top manipulation (the **P**layMate focus [Hawes *et al.*, 2007]) or mobility in large-scale space (the **E**xplorer focus [Zender *et al.*, 2008]). It is based on the CoSy Architecture Schema (**CAS**), which structures systems into *subarchitectures* (SAs) which cluster *processing components* around *shared working memories* [Hawes *et al.*, 2007]. In PECAS, SAs group components by function (e.g., vision, communication, or navigation). All these SAs are active in parallel, typically combining reactive and deliberative forms of processing, and all operating on SA-specific representations (as

is necessary for robust and efficient task-specific processing). These disparate representations are unified, or *bound*, by a *subarchitecture for binding* (binding SA), which performs abstraction and cross-modal information fusion on the information from the other SAs [Jacobsson *et al.*, 2008]. PECAS makes it possible to use the multiple capabilities provided by a system's SAs to perform many different user-specified tasks. In order to give the robots a generic and extensible way to deal with such tasks, we treat the computation and coordination of overall (intentional) system behavior as a *planning* problem. The use of planning gives the robot a high degree of autonomy: complex goal-driven behaviors need not be hard-coded, but can be flexibly planned and executed by the robot at run-time. The robot can autonomously adapt its plans to changing situations using *continual planning* and is therefore well suited to dynamic environments. Relying on automated planning means that tasks for the robot need to be posed as goals for a planner, and behavior to achieve these goals must be encoded as actions that the planner can process. The following sections expand upon these ideas.

2.1 Cross-Modal Binding

Cross-modal binding is an essential process in information-processing architectures which allow multiple task-specialized (i.e., *modal*) representations to exist in parallel. Although many behaviors can be supported within individual modalities, two cases require representations to be shared across the system via binding. First, the system requires a single, unified view of its knowledge in order to plan a behavior that involves more than one modality (e.g., following a command to do something relative to the object or area). Second, binding is required when a subsystem needs information from another one to help it solve a problem (e.g., using visual scene information to guide speech recognition [Lison and Kruijff, 2008]).

Our approach to binding underlies much of the design and implementation of our systems, and so we will reiterate it here (for more details see [Jacobsson *et al.*, 2008]). Each PECAS SA that wishes to contribute information to the shared knowledge of the system must implement a *binding monitor*. This is a specialized processing component which is able to translate from an arbitrary modal representation (e.g., one used for spatial modeling or language processing) into a fixed *amodal* (i.e., behavior neutral) representation. Across a PECAS system the binding monitors provide a parallel abstraction process mapping from multiple, different representations to a single, predicate logic-like representation. Binding monitors deliver their abstracted representations into the binding SA as binding *proxies* and *features*. Features describe the actual abstract content (e.g., color, category, or location) in our amodal language, whilst proxies group multiple features into a single description for a piece of content (such as an object, room, or person), or for relationships between two or more pieces of content. The binding SA collects proxies and then attempts to fuse them into binding *unions*, structures which group multiples proxies into a single, cross-system representation of the same thing. Groupings are determined by feature matching. Figure 1 illustrates this: the SA for navigation (nav SA) and the SA for conceptual mapping and reasoning (coma SA),

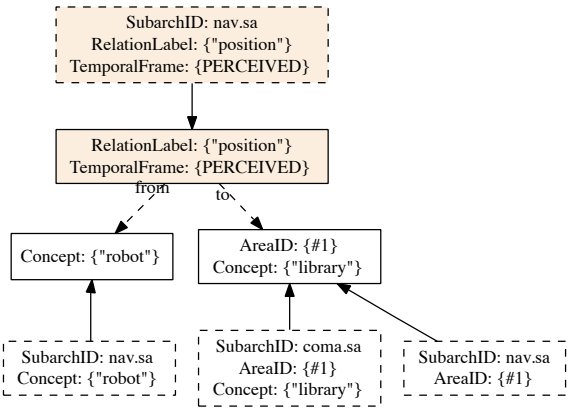


Figure 1: Binding localization and conceptual information: "the robot is in the library." Proxies have dashed borders, unions solid borders. Relation proxies, -unions are colored.

provide their information to the binding SA. Throughout this process links are maintained between all levels of this hierarchy: from modal content, to features and proxies, and then on to unions. These links act like pointers in a programming language, facilitating access to information content regardless of location. Binding thus supports the two identified cases for cross-modal binding: the collection of unions provide a single unified view of system knowledge, and cross-subsystem information exchange is facilitated by linking similarly referring proxies into single union.

2.2 Planning for Action and Processing

For PECAS we assume that we can treat the computation and coordination of overall system behavior as a planning problem. This places the following requirements on PECAS: it must be able to generate a state description to plan with; system-global tasks for the robot need to be posed as goals for a planner; and behavior to achieve these goals must be encoded as actions which can be processed by the planner. In our implementation we use the MAPSIM continual planner and its Multi-Agent Planning Language (MAPL) [Brenner and Nebel, 2009]. In MAPL, we can model beliefs and mutual beliefs of agents as well as operators affecting these, i.e., perceptual and communicative actions. The continual planner actively switches between planning, execution, and monitoring in order to gather missing goal-relevant information as early as possible.

To provide a planning state, the planning SA automatically translates from the unions in the binding SA into MAPL. The planner thus automatically receives a unified view of the system's current knowledge. As we maintain links from unions back to modal content, our planning state, and therefore our plans, remain grounded in representations close to sensors and effectors. In PECAS, planning goals arise as modal *intentional content* which is then abstracted via binding monitors and placed in the planning SA's working memory. From here we use the same translation method as is used on the planning state to produce MAPL goals for the planner.

While the traditional use of planning is achieving goals in the world using physical actions, such direct interpretations of behavior are the exception rather than the rule in cognitive robotics (cf. [Shanahan, 2002]). Here, where infor-

mation is incomplete, uncertain, and distributed throughout subsystems, much of the actions to be performed by the system are to do with processing or moving *information*. Whilst some information processing may be performed continually (e.g., SLAM), much of it is too costly to be performed routinely and should instead be performed only when relevant to the task at hand, i.e., it should be planned based on context.

Underlying our approach to information-processing is the functionally decomposed, concurrently active, structure of PECAS. As each SA is effectively a self-contained processing unit, our design leads naturally to an integration strategy: each SA is treated as a separate agent in a multi-agent planning problem. A crucial feature of this strategy is that each SA's knowledge is separate within the planning state, and can only be reasoned about using epistemic operators (i.e., operators concerned with knowledge). Likewise, goals are often epistemic in nature, e.g., when a human or a SA wants to query the navigation SA for the location of an object.

To realize internal and external information exchange each SA can use two epistemic actions, *tell-value* and *ask-value*, coupled with two facts about SAs, *produce* and *consume*. The actions provide and request information respectively. The facts describe which SAs can produce and consume which predicates (i.e., where certain types of information can come from and should go). For example, if a human teacher tells our robot that "this is the kitchen," this gives rise to the motivation that all SAs which consume room knowledge (e.g., coma SA described in the next section) should know the type of the room in question. This may lead to a plan in which the SA for situated dialogue (comsys SA) uses a tell-value action to give the coma SA this information.

Using this design, planning of information-processing becomes a matter of planning for epistemic goals in a multi-agent system. This gives the robot more autonomy in deciding on the task-specific information flow through its subsystems. But there is another assumption underlying this design: whilst the binding SA is used to share information throughout the architecture, not all information in the system can or should be shared this way. Some of it is unavailable because it is modality specific, and even cross-modal knowledge is often irrelevant to the task at hand. If all information was shared this would overwhelm the system with (currently) irrelevant information (e.g., lists of all the people, rooms, objects, object categories etc. that parts of the system know about). Thus, in order to restrict the knowledge the planner gives "attention" to without losing important information, it needs to be able to *extend* its planning state on-the-fly, i.e., during the continual planning process. In PECAS state extension can be done using ask-value and tell-value actions, and results in a process we call *task-driven state generation*.

3 The Explorer Instantiation

The binding and planning SAs described above are system and scenario independent. We now discuss the Explorer-specific SAs to describe concrete functionality and how this relates to system control. All SAs have been implemented in CAST (an open-source toolkit implementing the CAS schema) and tested on an ActivMedia PeopleBot. Figure 2 shows all SAs used in the Explorer PECAS instantiation. Most components used in the different SAs have been discussed in detail in earlier work (references provided below).

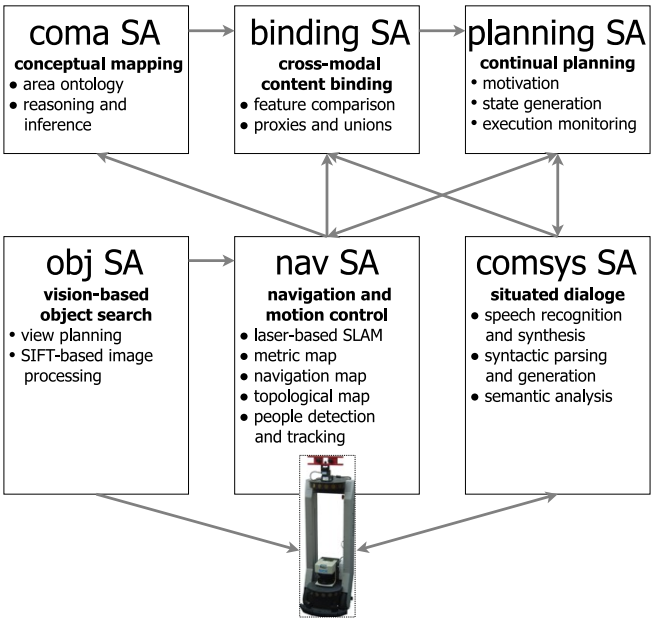


Figure 2: The Explorer Architecture

For a mobile robotic system that is supposed to act and interact in large-scale space, an appropriate spatial model is key. The Explorer maintains a multi-layered conceptual spatial map of its environment [Zender *et al.*, 2008]. It serves as a long-term spatial memory of large-scale space. Its individual layers represent large-scale space at different levels of abstraction, including low-level metric maps for robot motion control, a navigation graph and a topological abstraction used for high-level path planning, and a conceptual representation suitable for symbolic reasoning and situated dialogue with a human. In the Explorer, different SAs represent the individual map layers. For the details on human-augmented map acquisition see [Kruijff *et al.*, 2007].

nav SA The SA for navigation and spatial mapping hosts the three lowest levels of the spatial model (metric map, navigation map, and topological layer). For low-level, metric mapping and localization the nav SA contains a module for laser-based SLAM. The nodes and edges of the *navigation map* represent the connectivity of visited places, anchored in the metric map through x-y-coordinates. Topological areas, corresponding roughly to rooms in human terms, are sets of navigation nodes. This level of abstraction in turn feeds into the conceptual map layer that is part of the coma SA.

The nav SA contains a module for laser-based people detection and tracking [Zender *et al.*, 2007]. The nav SA binding monitor maintains the robot's current spatial position and all detected people, as proxies and relations on the binding SA. The smallest spatial units thus represented are areas. This provides the planner with a sufficiently stable and continuous description of the robot's state. The planning SA can pose *move* commands to the nav SA. 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. Move commands are executed by a navigation control module, which

performs path planning on the level of the navigation graph, but automatically handles low-level obstacle avoidance and local motion control.

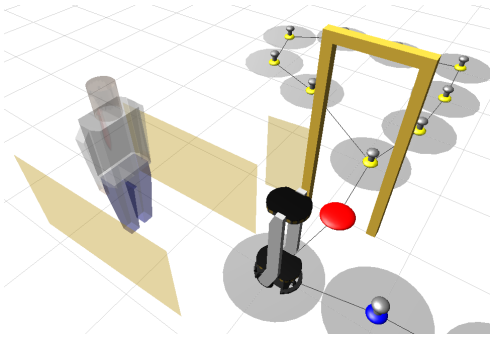
obj SA The SA for vision-based object search contains the components for finding objects using vision. It consists of a module for view planning and one for visual search. The view planning component creates a plan for which navigation nodes to visit, in what order and in what directions to look. Details of the process can be found in [Gálvez López *et al.*, 2008]. The visual search consists of SIFT feature matching directly on acquired images. Objects that are found are published on the obj SA working memory. The nav SA detects this and in turn extends the spatial model with the new objects. This then propagates the information to the coma SA and, if and when necessary, to the binding SA.

coma SA The SA for conceptual mapping and reasoning maintains an abstract symbolic representation of space suitable for situated action and interaction. It represents spatial areas (nav SA), objects in the environment (obj SA), and abstract properties of persons (e.g., ownership relations) in a combined A-Box and T-Box reasoning framework based on an OWL-DL reasoner, which can infer more specific concepts for the area instances [Zender *et al.*, 2008]. The coma SA makes its information available to the binding SA on demand, i.e., whenever planning SA sends an *ask-val* command to the coma SA, it will add its knowledge about spatial entities, especially their most specific concepts. In our system the explicit definitions of area concepts through occurrences of certain objects are also used to raise expectations about typical occurrences of certain objects. If the planning SA needs to know the location of an object that has not been encountered before, it can query the coma SA, which will then provide a *typical* location of the object in question. This is done via special T-Box queries involving the OWL-DL definitions of concepts. An example of this will be discussed in Section 4.

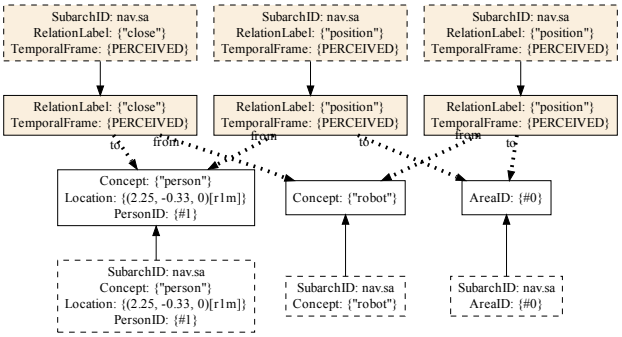
comsys SA The subarchitecture for situated dialogue processing has a number of components concerned with understanding and generation of natural language utterances [Kruijff *et al.*, 2009]. Speech recognition converts audio to possible text strings, which are subsequently parsed. Parsing produces a packed representation of logical forms (LFs) that correspond to possible semantic interpretations of an utterance. Finally, the semantics are interpreted against a model of the dialogue context. Content is connected to discourse referents, being objects and events talked about over the course of an interaction. In the dialogue context model, both the content of the utterance and its intent are modeled. All of this information is communicated to the planning SA and the binding SA through proxies representing the indexical and intentional content of the utterances. In rough terms the indexical content (information about entities in the world) is used by the binding SA to link with information from other modalities. Meanwhile the intentional content (information about the purpose of the utterance) is used by the planning SA to raise goals for activity elsewhere in the system [Kruijff *et al.*, 2009].

4 Example: Finding a book

This section presents a scenario in which a human asks the Explorer to perform a task. It shows how PECAS controls system behavior and information-processing. The example is taken directly from our implemented system, showing system visualizations (with minor post-processing).



(a) Screenshot of the visualization tool



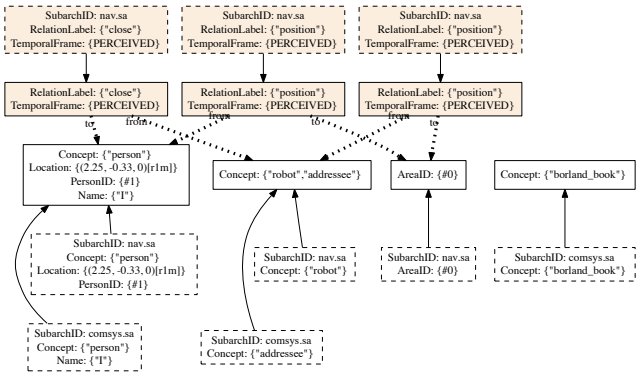
(b) Contents of binding working memory

Figure 3: Initial situation: the user approaches the robot

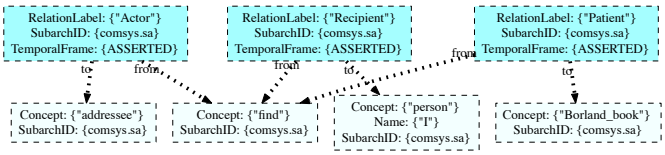
The system starts in the spatial context and binding state visualized in Figure 3: the robot and person are occupying the same area, and the person is close to the robot. The *robot proxy* is provided by the nav SA which it abstracts from its representation of the robot pose. The *person proxy* is provided by the nav SA because a person is being tracked. In addition to these, the nav SA makes available a proxy for the *area* in which one of these proxies occurs, linking them with a *position* relation proxy. Finally, the *close* relation proxy connects the robot proxy to the proxy of the person because the person is geometrically close to the robot. Note that no objects are present, nor are other areas except the current area.

Next, the human approaches the robot and says "find me the Borland book". The comsys SA interprets this utterance, presenting the elements of its interpretation to the rest of the system as proxies. Figure 4a shows the results. The Explorer itself (the recipient of the order) is represented by a proxy with Concept `addressee`, which binds to the robot proxy already present. The word "me" refers to the speaker, and generates a "person" proxy identified by the Name feature `I`. The expression referring to the book is given by a "Borland_book" proxy, not yet bound to any other proxies.

The comsys SA can determine the intention of this utterance, and separates the intentional elements of the interpretation from the aforementioned descriptive proxies. This intentional content is written to planning SA as a proxy structure with links back to the binder. The structure of this *motive* can be seen in Figure 4b. Planning SA, detecting a new mo-



(a) State of binding SA



(b) Representation of intentional content (as a *motive*)

```
Objects:                    Facts:
(area_id_0 - area-id)       (area-id gensym1 : area_id_0)
(gensym0 - robot)           (area-name area_id_0 : gensym1)
(gensym1 - area-name)       (perceived-pos gensym0 : area_id_0)
(gensym4 - person)          (perceived-pos gensym4 : area_id_0)
(gensym6 - movable)         (close gensym4 gensym0 : true)
```

(c) Planning state after processing the intentional content

Figure 4: State after the user has uttered the command "Find me the Borland Book."

tive, begins the process of creating a plan to fulfill it. First, it converts the information on the binder (Figure 4a) to the MAPL representation in Figure 4c. In this process unions become objects and predicates in the planning state. E.g., as the person union is related by a position relation union to an area union, this will be expressed to the planner as `(perceived-pos gensym4 : area_0)`, where `gensym4` is an auto-generated planning symbol referring to the person, and `area_0` refers to the area. The planner similarly converts the motive from Figure 4b into a MAPL goal `(K gensym4 (perceived-pos gensym6))`. This can be read as the Explorer having the goal of the the person knowing the position of the book. We use this interpretation of the command "Find me...", as the robot does not have the ability to grasp objects.

Given this state and goal, the planner creates a plan:

```
L1: (negotiate_plan gensym0 coma_sa)
L2: (tell_val_asserted-pos
        coma_sa gensym0 gensym6)
L3: (find_a gensym0 gensym6 gensym0)
L4: (tell_val_perceived-pos
        gensym0 gensym4 gensym6)
```

This plan states that the Explorer must find the location of the book (`L3`), then report this location to the person (`L4`). Before it does this it must negotiate with the coma SA (as each subarchitecture is treated as a separate agent) to provide a location where it might be able to find the book (`L1`,`L2`). The reasoning behind this plan is that Explorer must provide the person with a perceived location for the book (as is specified in the goal), and, having not seen it recently, the only way to obtain a perceived location is via its object search functionality. To perform an object search the system must have both an object to search for (the book in this case) and an area to search. *Typical* positions of objects (as opposed to their *perceived* positions) are stored in the ontology in coma SA. Rather than make all of this knowledge available via binding by default (a choice which would add many extra and redundant facts to the planning state), typical positions are offered by coma SA using a `produce` fact (see Section 2.2). This allows the planner to query coma SA for typical positions when it requires them. One advantage of this on-demand state generation is that the comsys SA could also be used to provide the same knowledge (and would be if the book was not found initially). In the above plan, the planner makes use of this by

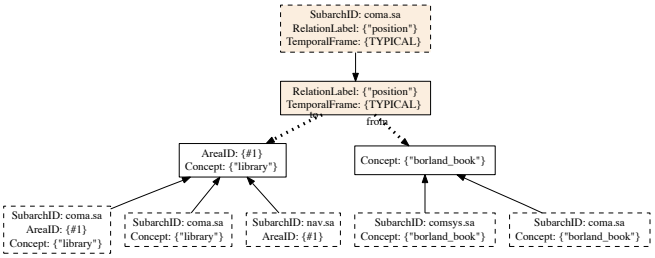


Figure 5: Hypothetical position of the Borland book.

getting the coma SA to `tell-val` the typical position of the Borland book to the binding SA. This yields the information that, as it is a book, it would typically be found in a library. Along with this, the coma SA also volunteers the specific information it has on libraries: an area exists in its map that is a library. This is illustrated in Figure 5.

Given this hypothesis for the book's location, MAPSIM uses a replanning step to expand the initial plan to include steps to move the robot to the library, search there for the book, then move back and report to the user. The updated plan and planning state are now as follows:

```
Objects:
(area_id_0 - area-id)   (area_id_1 - area-id)
(gensym0 - robot)       (gensym1 - area-name)
(gensym4 - person)   (gensym6 - borland_book)
(gensym6 - movable)     (gensym7 - area-name)

Facts:
(area-id gensym1 : area_id_0)
(area-id gensym6 : area_id_1)
(area-name area_id_0 : gensym1)
(area-name area_id_1 : gensym7)
(asserted-pos gensym6 : gensym7)
(perceived-pos gensym0 : area_id_0)
(remembered-pos gensym4 : gensym1)

Plan:
L1: (move gensym0 area_id_1 area_id_0)
L2: (object-search-in-room
        gensym0 gensym6 area_id_1)
L3: (approach-person
        gensym0 gensym4 area_id_0)
L4: (tell_val_perceived-pos gensym0
        gensym4 gensym6)
```

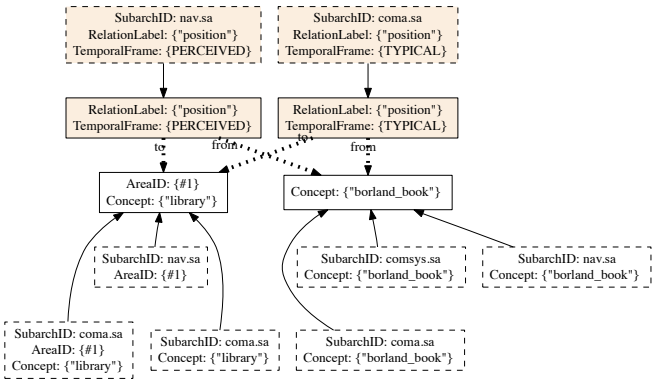


Figure 6: Perceived location of the book

In the above, `gensym7` is the binding union of the library. Using the **AreaID** feature from the this union, the planner issues a command to the nav SA which moves the robot to the library (fulfilling step `L1`). As with all other steps in the plan (including the information-processing ones), the results of this action are checked by MAPSIM to determine whether it has completed successfully or whether replanning is required. This check is performed by inspecting the planning state and comparing it to the expected state. This means that all actions must have effects that are visible on the binding SA (for subsequent translation). Once the check has passed for `L1` (confirming the robot has arrived in the library), the planner issues an object search command to the object SA. The Explorer searches the room as described previously. Once the object is found, the nav 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, which is connected to the room's proxy by a new position proxy. This position proxy has a `PERCEIVED` temporal frame.

The new proxies generated by object search bind to the existing complex (pictured in Figure 5), resulting in the structure in Figure 6. This binding provides the original comsys SA book proxy with a perceived position (in addition to its typical one). With this knowledge in the planning state (i.e., the effect of `L2` is verified, which satisfies one precondition of `L4`), the planner is able to trigger the remaining steps in the plan: moving to the user and reporting the perceived position. A move command is send to the nav SA referencing the **Location** feature from the person proxy. Once close to this location, a `tell-val` is sent to the comsys SA to communicate the book's location to the person. A content generation component in the comsys SA uses the contents of the binder (see Figure 6) to generate the utterance "the Borland book is in the library", thus completing the plan and satisfying the original goal (that the person knows the position of the book).

5 Discussion

The preceding sections illustrate how our architectural ideas come together to create a control system for intelligent behavior. Although the surface form of the scenario does not present much in the way of novel interactions, the PECAS architecture and the multi-level spatial representation provide a novel system-level approach with a number of important features. Including the binding SA in the architecture allows multiple modalities to collaborate on problems that a single modality in isolation would not be able to solve. E.g., in the Explorer the comsys SA initially provides a description of an object based on natural language input, conceptual mapping then extends this description, and vision finally completes it. Whilst other systems may include elements of cross-modal fusion, we have taken the additional, novel step of using the results of fusion to provide input to a continual planner. This allows the behavior of multiple modalities to be marshalled in pursuit of system goals in a general, extensible manner. Using continual planning PECAS achieves this in a way that is responsive to external change and certain types of failure. In PECAS all this is true both of actions that have physical effects, and of internal, information-processing actions.

Both the theory and implementation of the Explorer system and PECAS are works in progress, so we can identify many areas that need further study, or currently limit our approach. E.g., while the the use of MAPSIM provides many of the strengths of our work, planning occurs at quite a high level of abstraction. This consequently also applies to interactions between subarchitectures, and between the Explorer and the world. Whilst this has some advantages (e.g., subsystems are free to interpret commands in modality specific ways, something discussed in more detail below), it may be a hindrance to more closely coupled interactions between behaviors, such as positioning the robot to see an object that it is trying to pick up. Also, actions used in the PECAS architecture must have effects that are visible at the level of binding proxies, which may not hold for actions that have effects in a single SA. Our system also relies on many translations between formalisms. Whilst our structured support for this (via binding) is a clear strength, in practice the translations can become somewhat arbitrary and hard to maintain.

5.1 Approaches to Control

The HYCAS workshop aims to investigate issues of hybrid control in autonomous systems, so what lessons can we learn from the work presented here? In all PECAS systems we have a number of control patterns working in parallel. At the lowest level we can reasonably discuss in terms of architecture, components typically run in one of two modes. Either they perform continuous processing which provides a stream of data to working memory, or they wait for a particular event which triggers some processing (which may or may not result in a change to working memory). In this case events may either be external to the system (e.g., a sensor, such as a microphone, being triggered), or internal (where an event describes a change to working memory contents). Within a SA these types of processing behaviors happen concurrently (with SAs also working in parallel to each other). Control of SA-level processing is typically constrained at design-time, when components are set to listen to particular types of events. At run-time these events, and mediated access to the information they describe, provide implicit synchronization during processing. Thus PECAS does not provide explicit control strategies within SAs (although a few common control strategies tend to be reused).

As described in the preceding sections, the path to high-level control in PECAS comes via SAs exposing modal con-

tent to the rest of the system via the binding system. The process by which this occurs plays a major role in system control. Binding monitors provide abstracted representations of SA-local content. They typically do this based on three different triggers: SA-internal events, SA-external events, and on-demand. The first of these is the most basic case: the generation of a new local representation triggers the SA's binding monitor to generate a proxy. This happens in the Explorer for discourse referents in the comsys SA. The second case, SA-external events, typically provides a way for the existing binding state to influence the generation of further proxies (one of the limited, distributed, forms of attention in PECAS): the monitor listens for both SA-internal and -external events, then, when some particular events co-occur, it generates a proxy from some local content. This happens in the Explorer when the conceptual mapping SA provides proxies in response to proxies generated by other SAs (e.g., when the comsys SA generates a proxy for an object, the coma SA provides additional proxies to bind with it). The final case, on-demand monitor operation, occurs when a binding monitor is explicitly asked (rather than implicitly triggered) to provide information about a particular entity already represented in the binding SA. This approach is used by the system to deliberatively add information to the binding system. This is typically done during the planning process (as an element of on-demand state generation).

The first two of these binding monitor triggers represent additional design-time control decisions within PECAS systems. The designer explicitly chooses which SA and system events should cause information to be shared via binding (and thus added to the planning state for system control). The underlying assumption is that the system will need high-level access to this information regardless of context, and therefore this hard-wired approach is acceptable. The latter case, on-demand triggering, provides a system with explicit control over the information shared between all SAs and used for planning. We expect this approach, whether driven by planning or other mechanisms, to become the dominant approach in future PECAS systems. The alternative (implicit control over the contents of the binding SA) would place the system entirely at the mercy of reactive control, potentially flooding the binding SA with irrelevant or redundant information.

Binding monitors typically provide two types of abstraction: level-of-detail abstraction and temporal abstraction. The former has been taken for the implicit meaning of "abstraction" in the preceding sections: translation of a complex modal representation into a less complex amodal representation. Temporal abstraction is often implicit in level-of-detail abstraction, but it is important to make its presence explicit as it influences our control approach. Changes within SAs typically occur at a rate linked to the rate of change of sensors used for that SA's modality or the processing schemes used to interpret the results of those sensors. E.g., in the nav SA the pose of the robot is updated by SLAM at 5Hz, in the comsys SA elements of the discourse references are incrementally updated during an utterance interpretation (and across multiple utterances if they are reused), and in the obj SA object positions are updated as close to framerate as the system can manage. If the planner, or any other deliberative system, had to take control decisions using information at this level of description from multiple SAs, its decisions would only be valid for that length of time all of these representations remained unchanged (a number limited by the most volatile item of information). This would make system-wide control rather difficult. Binding monitors ameliorate this problem by only propagating *relevant* changes from the SA level to the binding SA. What constitutes a relevant change is both SA- and task-specific, but often relevance is coupled to the potential of the change to significantly alter the global state of the system. Temporal abstraction occurs because significant changes typically do not occur at the same rate as all changes; they often happen much less frequently. This highlights the close coupling between temporal abstraction and level-of-detail abstraction, as the latter defines our global state. This fact is often relied upon in systems which operate on multiple levels of abstraction. In this sense the role binding plays in PECAS can be meaningfully compared with the definition of an *interface layer* in the work of Wood (e.g., [Wood, 1994]). Interface layers are where a designer identifies critical points in the representations used by a system. These are points at which the representations become suitable for particular types of reasoning tasks. The identification of these layers is crucial for system control; they provide a way to match up representations with decision making approaches, e.g., detailed, dynamic representations for reactive control, and more abstract, stable representations for deliberative control. So, to reiterate an important point, unions and proxies (and to some extent the actions used by the planner) represent a stable point in the space of representation used by PECAS systems. Without them we would not be able to use planning (which requires such stability) to control system behavior.

From a control perspective there are two interesting aspects to our use of planning. First, as mentioned previously, we use continual planning: we integrate execution monitoring and replanning into our high-level control system. This provides a form of closed-loop high-level control for our system, where the effects of actions are monitored relative to expectations established by their definitions, and replanning is triggered if these expectations are violated. Second, the planner only has an opaque interface to the actions themselves. Rather than being concerned with how each action is implemented, PECAS only requires that the implementing SA abides by the contract provided by the action definition; otherwise planning and monitoring will fail. This is in contrast to other systems (e.g., 3T [Bonasso *et al.*, 1997]) where high-level control is used to schedule behaviors all the way down to the lowest-level (e.g., skills) too. By adopting a less exacting approach to action execution we allow each SA to interpret the action in a contextually appropriate way. SAs may choose to use one or many components to execute an action and may go through as many intermediate steps as required. This allows a single high-level control action to become a multiple step lower-level action, e.g., when an action results in a dialogue, or a visual search behavior. Of course, this means the planner is unable to directly influence the creation or scheduling of these lower level tasks. This is not a problem in our current domains where actions do not compete for resources across SAs, but in future this could become a problem. Possible

solutions include making the actions available to the planner less coarse but still not providing a one-to-one mapping to SA-internal actions (i.e., giving it tighter control over SA behavior), or annotating actions with resource constraints.

In summary, the overall behavior of a PECAS system, including the Explorer instantiation described in this paper, emerges from the interaction of reactive and deliberative control systems at multiple levels of abstraction. Multiple concurrent components within SAs are controlled implicitly by design-time event-subscription rules, and use CAST's event mechanisms and working memories to synchronism their processing at run-time. Across the system a collection of binding monitors provide an interface at which representations become abstract and stable. This allows a single deliberative control process to interact with the multiple concurrent SAs. It is this interface level which allows a PECAS instantiation to solve some problems with deliberative approaches (e.g., cross-SA coordination) and others with reactive approaches (e.g., within-SA coordination and sensor and effector control) whilst remaining contextually appropriate and responsive to its environment (i.e., no single control strategy ever exclusively takes charge of the entire system). However, this approach currently relies on an external designer fixing the representations either side of the interface level. Whilst this is not necessarily a problem in the short-term, in the future we would like to investigate what properties define a good interface level so that new system designers will not have to make uninformed design decisions.

6 Conclusion

We described PECAS, an architecture for intelligent systems. PECAS is a new architectural combination of information fusion and continual planning. Its purpose is to plan, integrate and monitor the asynchronous flow of information between multiple concurrent systems to achieve a task-specific system-wide goal. We used the Explorer instantiation to show how this works out in practice. The Explorer instantiates PECAS around a hybrid spatial model combining SLAM, visual search, and conceptual inference, with the possibility to use spoken dialogue to interact with a human user. We described the elements of this model, and demonstrated using a realistic (and implemented) scenario how PECAS provides a novel approach to control for autonomous systems.

Acknowledgements

Supported by the EU FP7 ICT Cognitive Systems Integrated Project "CogX" (FP7-ICT-215181-CogX) and in part by the Swedish Research Council, contract 621-2006-5420.
For more information see `http://cogx.eu`.

References

[Bonasso *et al.*, 1997] R. P. Bonasso, R. J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack. Experiences with an architecture for intelligent, reactive agents. *J. of Experimental and Theoretical Artificial Intelligence*, 9(2-3):237–256, 1997.

[Brenner and Nebel, 2009] M. Brenner and B. Nebel. Continual planning and acting in dynamic multiagent environments. *Aut. Agents and Multi-Agent Sys.*, 2009. accepted for publication.

[Burgard *et al.*, 2000] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1–2), 2000.

[Gálvez López *et al.*, 2008] D. Gálvez López, K. Sjö, C. Paul, and P. Jensfelt. Hybrid laser and vision based object search and localization. In *ICRA '08*, pages 2636–2643, 2008.

[Gross *et al.*, 2008] H. M. Gross, H. J. Böhme, C. Schröder, S. Müller, A. König, C. Martin, M. Merten, and A. Bley. ShopBot: Progress in developing an interactive mobile shopping assistant for everyday use. In *SMC '08*, pages 3471–3478, 2008.

[Hawes *et al.*, 2007] N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G. J. Kruijff, M. Brenner, G. Berginc, and D. Skočaj. Towards an integrated robot with multiple cognitive functions. In *AAAI '07*, pages 1548–1553, 2007.

[Hawes *et al.*, 2009] N. Hawes, M. Brenner, and K. Sjöö. Planning as an architectural control mechanism. In *HRI '09*, pages 229–230, New York, NY, USA, 2009. ACM.

[Ishiguro *et al.*, 2001] H. Ishiguro, T. Ono, M. Imai, T. Maeda, T. Kanda, and R. Nakatsu. Robovie: an interactive humanoid robot. *Int. J. Industrial Robot*, 28(6):498–503, 2001.

[Jacobsson *et al.*, 2008] H. Jacobsson, N. Hawes, G. J. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. In *HRI '08*, 2008.

[Kruijff *et al.*, 2007] G. J. Kruijff, H. Zender, P. Jensfelt, and H. I. Christensen. Situated dialogue and spatial organization: What, where... and why? *International Journal of Advanced Robotic Systems*, 4(1):125–138, 2007.

[Kruijff *et al.*, 2009] G. J. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, H. Zender, I. Kruijff-Korbayová, and N. Hawes. Situated dialogue processing for human-robot interaction. In H. I. Christensen, G. J. Kruijff, and J. L. Wyatt, editors, *Cognitive Systems*. Springer Verlag, 2009. to appear.

[Kuipers, 1977] B. Kuipers. *Representing Knowledge of Large-scale Space*. PhD thesis, MIT, 1977.

[Lison and Kruijff, 2008] P. Lison and G. J. Kruijff. Salience-driven contextual priming of speech recognition for human-robot interaction. In *ECAI '08*, 2008.

[Peltason *et al.*, 2009] J. Peltason, F. H. K. Siepmann, T. P. Spexard, B. Wrede, M. Hanheide, and E. A. Topp. Mixed-initiative in human augmented mapping. In *ICRA '09*, 2009. to appear.

[Shanahan, 2002] M. Shanahan. A logical account of perception incorporating feedback and expectation. In *KR '02*, pages 3–13, 2002.

[Sidner *et al.*, 2004] C. L. Sidner, C. D. Kidd, C. H. Lee, and N. B. Lesh. Where to look: A study of human-robot engagement. In *IUI '04*, pages 78–84, 2004.

[Siegwart and et al., 2003] R. Siegwart and et al. Robox at expo.02: A large scale installation of personal robots. *Robotics and Autonomous Systems*, 42:203–222, 2003.

[Wood, 1994] S. Wood. *Planning and Decision Making in Dynamic Domains*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994.

[Zender *et al.*, 2007] H. Zender, P. Jensfelt, and G. J. Kruijff. Human- and situation-aware people following. In *RO-MAN '07*, pages 1131–1136, 2007.

[Zender *et al.*, 2008] H. Zender, O. Martínez Mozos, P. Jensfelt, G. J. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008.