The PlayMate System

Nick Hawes[1], Jeremy Wyatt[1], Aaron Sloman[1], Mohan Sridharan[1], Marek Kopicki[1], Somboon Hongeng[1], Ian Calvert[1], Geert-Jan Kruijff[2], Henrik Jacobsson[2], Michael Brenner[3], Danijel Skočaj[4], Alen Vrečko[4], and Nikodem Majer[5]

[1] Intelligent Robotics Laboratory, School of Computer Science, University of Birmingham, Birmingham, UK, {nah,jlw,mzs,axs}@cs.bham.ac.uk
[2] DFKI GmbH, Saarbrücken, Germany, {henrikj,gj}@dfki.de
[3] Albert-Ludwigs-Universität Freiburg, Department of Computer Science, Freiburg, Germany
{brenner,plagem,nebel,burgard}@informatik.uni-freiburg.de
[4] VICOS Lab, Universa v Ljubljani, Slovenia, danijel.skocaj@fri.uni-lj.si
[5] Technische Universität Darmstadt, Darmstadt, Germany, nmajer@mis.informatik.tu-darmstadt.de

1 Introduction

In the PlayMate scenario we were concerned with understanding the problems that an intelligent system must face if it must interact with humans in an object rich environment. In particular while certain problems, such as manipulation, are peripheral aspects of the project. We want to understand how a robot can share a space with a human in which they both manipulate objects, and in which they can talk about those objects. This requires many abilities. The robot must be able to understand and generate references to objects, actions with them, their properties and spatial relationships. It must understand how actions alter the relationships between objects, be able to recognise actions the human performs with objects, and it must be able to learn about the effects of actions on objects, both by discovery and by watching others. If there are several opportunities for action it must choose between a number of potential goals at any one time. Finally, since it is working with a human, the human may change the world while the robot is acting. Thus the robot must meet many of the requirements that we outlined in Chapter 2, and utilise many of the technologies that we described in the remaining chapters. In this chapter we describe how we integrated these technologies to solve some of the tasks that exist in the PlayMate scenario. We will describe both the complete PlayMate system, and the major innovations in the PlayMate. It is important to note that the system we describe is the last of a series of systems. Each of these systems has been used as an experimental platform,

and the lessons we have learned have been used to revise both our approaches to the individual problems, and the architecture employed.

To help us in our exposition we will use a single example that runs through the chapter and shows the various problems that we encounter in building a robot with multiple modes of vision and action. The script for this example is given in Figure 1.

```
Human (H) puts a blue square down on the table
H: ‘‘This is a blue square.’’
Robot (R): ‘‘OK.’’
H puts down a red square to the right of the blue square.
H: ‘‘This is a red square.’’
Robot (R): ‘‘OK.’’
H picks up the red square and puts down a red triangle to the right
of the blue square.
R: ‘‘What is the thing to the right of the blue square?’’
H: ‘‘It is a red triangle.’’
R: ‘‘Ok.’’
H picks up the red triangle and the blue square and puts down a
blue triangle.
H: "What colour is this triangle?"
R: "It is blue."
H picks up the blue triangle and puts down a green triangle on the
right and a red circle on the left.
H: ‘‘What shape is this thing?’’
R: ‘‘Do you mean the green thing or the red thing?’’
H: ‘‘The red thing on the left.’’
R: ‘‘It is a circle.’’
H: ‘‘Put the circle to the right of the triangle.’’
R picks up the circle and puts it to the right of the triangle.
```

Fig. 1. An example script of an interaction between the PlayMate and a human.

The ability to handle this script requires that the robot be able to handle a number of situations. First *it must be able to understand not just the content, but the role of an utterance in the interaction*, when the human says ‘‘`This is a blue square.`’’ the robot must have a way of knowing that it should learn something. Second of all, *the robot must be able to bind information from one modality to that from another*, so that when the red square appears it understands which visual object is being referred to when the human refers to ‘‘`a red square`’’. When it is learning this will require it identify the object to be learned about using information other than the description of the object given as the learning signal. *It must also have ways of finding information that is required by one modality (e.g. language) from another (e.g. vision)*. So that if the human asks a question — ‘‘`What is the thing to the right of the blue square?`’’ — it can produce the answer. Alternatively, since the robot

is required to learn about the relationships between two modalities (vision and language) it must also have the ability to learn mappings between aspects of the two, i.e. to perform cross-modal learning. If the information available to the robot is insufficient for it to be able to perform its task it must acquire it. In other words *the robot must plan clarification processing to resolve ambiguities.* An example of this is when the human asks a question but the referent of the question is unclear — ‘‘`What shape is this thing?`’’ — so that the robot has to ask which thing is being referred to. Also, *if the robot is unsure about something it should be able to raise its own questions, so that it can learn.* An example of this is when the robot realises that it doesn't know the red triangle it is motivated to ask its own question. In understanding the references to objects the robot to map between quantitative and qualitative notions of spatial arrangement. When the human tells the robot to put the circle to the right of the triangle the robot has to map from a qualitative notion to a metric one (it has to put the circle in a particular place). Whereas in order to be able to make the utterance ‘‘`What is the thing to the right of the blue square?`’’ it must perform the mapping in the other direction, from metric to qualitative. Finally, while performing all these activities we desire that the robot perform them in as computationally efficient a manner as possible and make its interaction with the human as comfortable and natural for the human as possible. For us this will mean that we try where possible to have the robot build up its understanding of the scene incrementally. This is challenging when the scene changes frequently.

This script is only one of a large number that the PlayMate can handle, but it illustrates many of the key issues. In the next section (Section 2) we will give a sketch of the components and shared representations that the PlayMate employs. In that section we focus on two of the generic sub-systems that are employed in all activities: the binding system, and the motivation and planning system. This system sketch will provide a framework for understanding the content of Section 3 where we describe the cross-cutting issues and the innovative solutions employed in the PlayMate. These issues include those raised in the discussion of the script above, but we highlight our approach to cross-modal learning, incremental processing, disambiguation and clarification, and mediation between different levels of abstraction.

2 System Overview

In the previous section we outlined the kinds of tasks the PlayMate robot can be engaged in, and some of the requirements arising from those. In Chapter 2 we also outlined the requirements that arise generically on the architecture for both the PlayMate and Explorer scenarios. In this section we describe a specific system, built using CAST, which makes much stronger architectural commitments than CAS itself. We in fact built a series of systems to address different aspects of the PlayMate scenario over the four years of CoSy. In this

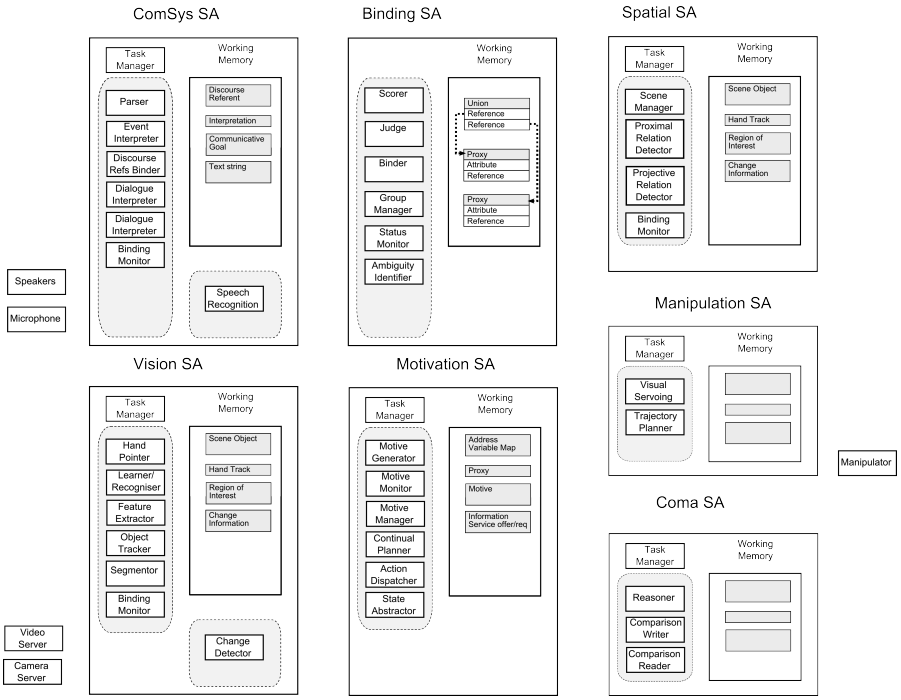

Fig. 2. An overview of the PlayMate System. There are seven sub-architectures, each of which corresponds to a functional and development unit.

section we give an overview of the system structure as it stands at the end of the project. Following this we will describe how the system structure supports principled solutions for a number of possible tasks.

The PlayMate system is an architectural instantiation of the CAS schema. It follows a broadly *functional decomposition* contra much recent work in robotics, but utilises *parallelism* and *incrementality* to build up its representation of the environment efficiently. Thus the PlayMate system makes much stronger architectural commitments than CAS itself.

Following this broadly functional decomposition, there are seven sub-architectures (SAs) in the PlayMate. These are concerned with *visual processing* (Vision SA), *Communication* (ComSys SA), *spatial representation and reasoning* (Spatial SA), *manipulation* (Manipulation SA which includes relevant visual processing), *ontological representations and reasoning* (Coma SA), *binding* of information between modalities (Binding SA), and control of *motivation and planning* (Motivation and Planning SA). In the PlayMate central roles are played by the last three of these sub-archtictures. In the latter part of this chapter we will show how we use these three to control most of the reasoning processes and flow of control, thus provided general control solutions for a variety of potential tasks (question answering, question asking, cross-modal

learning, manipulation, handling ambiguities). We now briefly sketch the components employed in each sub-architecture and the representations that they share in their working memories. The reader should refer to Figure 2 to aid understanding.

2.1 Vision SA

The visual system operates in two ways: it has the ability to analyse static scenes (e.g. categorising objects and their spatial relations), and to analyse dynamic scenes (e.g. recognising actions with objects). When new objects enter the field of view and then cease moving as they are placed on the table the resulting static scene is analysed. A simple *change detector* is used to determine whether the scene is, or has stopped changing. If the scene ceases changing, Regions of Interest in the scene are segmented from the background, and information about them is written to the working memory. It is important to note that all the information is exchanged between components on the working memory, and that most of this is captured in two types of structure. The first of these structures captures information about regions of interest (ROIs) in the image, and the second captures hypotheses about objects that may exist in the world. At the moment in the integrated system we rely on there being a one to one correspondence between ROIs and objects. Clearly this is not always the case, due to occlusion and partial occlusion, and in our work on planning of visual routines we have begun to develop methods to split ROIs into smaller regions that are more likely to correspond to single objects. Given our initial assumption about lack of occlusion the *segmentor* creates, for each ROI, not just a structure representing the ROI, but also one representing the object. These are linked, using the references that CAST allows between items on working memories, and each of is a slot and filler structure that allows components to work in parallel to on different slots. These slots for the object include descriptions of the object's category (by the *categoriser* component), global shape classification and colour classification. The ROI slots include the segmentation of the object within the ROI, and features calculated by the *feature extractor*. This calculates, for example, many local shape and colour features for each region of interest. This split in the level of abstraction is quite useful: ROI structures typically store quantitative local visual features, and object structures capture the more abstract classification of the whole object into a smaller number of qualitatively different classes. This correspondence between *local-quantitative* and *global-qualitative* is part of the process by which we gradually produce abstract representations suitable for planning and high level reasoning. The other key notion from the architectural schema CAS that is displayed here is that the components in the visual system *incrementally refine*, in *parallel*, these *shared representations* of what is in the scene.

When a static scene starts changing again, either because the robot or the human intervenes, dynamic scene analysis starts. In the case of objects, all the objects that have been segmented from the previous static scene analysis

are tracked by the *object tracker*, giving their full pose. In addition we separately track hands that appear with a *hand-tracker*, and analyse not only their position, but also any pointing gestures that they may make. These pointing gestures can be used to manipulate the salience of different objects or areas within the scene. Hand information is stored in a separate *hand* structure on the visual working memory. When objects and hands are being tracked information about their pose is constantly updated on the working memory. Many quantitative spatial features are also extracted from the tracking information that describe the relative pose of hands and objects, and these are used by the *event analyser* to classify simple and complex actions as described in Chapter 7. This allows the system to recognise some of the actions that are performed by the human with an object.

In addition to the ability to describe regions of the scene, objects in the scene, and actions performed with those objects, the visual system also permits learning of object properties. The *learner/recogniser* learns correlations between the low level features of associated with ROIs (e.g. colour histograms) and abstract classifications associated with objects (e.g. the ball being red). To learn this requires a feedback signal. There are many possible feedback signals, but in our case we utilise the information from language. This cross-modal learning is described in more detail in Chapter 7, and also in Section 3.1 below.

2.2 Communication SA

The Communication sub-architecture (or ComSys) has a number of components concernd with understanding and generation of natural language utterances. The main discussion of the techniques employed occurs in Chapter 8, but it is worth highlighting some points here. First a *speech recogniser* is used to convert speech to a text string. But even at this early stage the entities that exist as recorded on the binding sub-architecture are used to bias the likelihoods of recognising certain words. This is an important example of how tentative binding can be used to assist early processing of information by cutting down the space of hypotheses. This process of using binding to limit interpretations is also carried out by the *parser*. This uses a combinatorial categorial grammar with semantics expressed in a modal logic. As described in Chapter 8, this produces logical forms that correspond to semantic interpretations of an utterance. In the ComSys we have the ability to perform interpretation incrementally, i.e. as the elements of the text string arrive from recognition. Whether parsing is performed incrementally or not, the possible bindings of the discourse referents in the possible semantic interpretations cut down the interpretations that can be considered. Each utterance is interpreted as it arrives, and both the content of the utterance and its intent are modelled. The resulting interpretations of all the utterances in the dialogue to date are grouped together in what we call a *packed logical form*. We think about this as a forest of trees, where each tree corresponds to the logical form from a single

utterance. The dialogue model includes a packed logical form, information on all the discourse referents, a representation of events as well static scenes, and contextual information. All of this information is communicated to the binding and motivation sub-architectures by the ComSys binding monitor. This creates proxies representing both the indexical and intentional content of the utterances in the dialogue. In rough terms the indexical content (information about entities in the world) is used by the binder to link with information from other modalities, particularly vision. Meanwhile the intentional content (information about the purpose of the utterance) is used by the motivation and planning sub-architecture to raise goals for activity elsewhere in the system. The ComSys is able to signal to the overall system whether information is intended by the speaker to be new information, e.g. in an utterance that provides a description of an object to be learned. This is useful when it comes to cross-modal learning between language and vision.

2.3 Manipulation SA

Because our primary integration goal in the PlayMate scenario was to understand the integration and architectural issues some of our components replicate the state of the art, rather than each pushing forward the boundaries of each sub-field. This is true for manipulation where we utilise standard approaches to grasping and path planning for the manipulator. The manipulation is given the identity of objects it must manipulate, and their desired target locations. It plans trajectories through space to a suitable location to commence grasping by using a probabilistic road map planner with optimisation of the planned trajectories to minimise energy costs and path length. Having reached the location to commence grasping a local reactive controller utilises information from a visual analysis of the area in front of the manipulator. The manipulator has a camera on the hand, and this provides a high-resolution image of the objects to be grasped. Because of the mechanical restrictions imposed by two fingered manipulators (we use a jaw gripper) we have restricted our objects to always have graspable surfaces in the form of handles, or to objects that naturally afford grasping (e.g. upright oblong boxes of a suitable width). Given these objects however the visual analysis must still correctly identify the graspable surface from within the image. This analysis is performed using edge detection and anytime perceptual grouping methods reported by previously by Zillich [1]. These analyse the image and find candidate graspable top surfaces. The surface that best matches the hypotheses from the visual sub-system about the location and size of the object is chosen, and a power grasp is executed on the object. This approach is simple, but effective.

2.4 Spatial SA

The robot may receive information about the spatial configuration of objects from either language or vision. From language we have a representation that

is essentially qualitative. Utterances that the PlayMate might typically be expected to handle include several different types of spatial references, such as *projective* references, e.g. `"The box to the left of the red ball"`, `"The square in front of the triangle"` or *proximal* ones, e.g. `"The triangle near the red thing"`. Both of these types of reference are essentially qualitative. The visual system can also produce a representation of spatial relations, but this is essentially metric. The spatial sub-architecture captures both metric and qualitative representations of space, which it derives from visual input. How the mapping from these metric to qualitative representations is performed is described in Section 3.3. But note here that this process of abstraction is central to the representational problems the PlayMate faces. In addition that any solution to it that is suitable for supporting communication must also take account of the context sensitive way in which humans use spatial references. The spatial SA has a component called the *scene manager* which monitors changes in the information on the visual working memory. When a new object appears there it creates a record called a *location structure* of the corresponding metric location on its own working memory. These location structures point back to the visual information from which they were derived, creating the kind of *processing trail* that was discussed in Chapter 2. Because, during a dynamic scene, the position of objects can change quickly the actual location information recorded in the spatial working memory is only recomputed when the scene ceases changing. The set of the locations that hold during a period of stasis are grouped by reference in a *scene structure*. This grouping is also performed by the scene manager, which creates an ordered stack of scenes. This means that the spatial system has memory, which is necessary to enable the system to reason about the changes induced by previous trains of action. The spatial system also builds a qualitative representation of the spatial relations between scene locations. Two components search for *proximal* relations and *projective* relations between these locations, and create *scene relations* which are also referenced by the corresponding scene structure. These relations are detected in a context sensitive manner using potential field models that are described in detail in Section 3.3. The qualitative spatial description thus derived is related to descriptions from language by the binding sub-architecture. Our approach to binding is described in the next sub-section.

2.5 Binding

We have already described the binding problem in Chapter 2. We described how CAS allows two levels of solution to the binding problem. Within sub-architectures the results of processing by components are bound by design time decisions about the structure of the representations that reside on the working memory. We referred to this as *implicit binding*. However, as explained in Chapter 2 this is not enough in many cases. If in the PlayMate scenario there are two objects on the table, and we refer to `"the blue circle"` we

need to relate structures on the communication working memory to those on the visual working memory. We cannot use implicit binding to relate these two structures. This is because those structures will have been created at different times. Also because there are many ways of referring to an object, and many ways of interpreting a reference — especially ambiguous references, which may require accommodation or reinterpretation of the scene — the binding of structures from across modalities must be done at run time, and done in context sensitive way that produces an interpretation of the scene that is consistent across both modalities and objects.

Because we have chosen a largely functional decomposition for the PlayMate systems we have built, explicit binding is largely a cross-modal affair: we only have to perform explicit binding on entities across modalities. This is not to say, however, that binding can occur only after the results of modality specific processing have been completed. In Chapter 8 we described how we use *possible bindings* between entities in the visual scene and discourse referents in the utterances to incrementally interpret the utterances. This use of *incremental binding* means that binding becomes a way of achieving cross-modal inference. This is a powerful feature of the architectural approach taken in CoSy.

However, it would be reasonable to hypothesise that there are several different ways that binding could be performed within a system like the PlayMate. Our approach is carried out using a *central mediator* and *amodal representations* that are *abstractions* from the modality specific representations. This is a very old approach, with a pedigree that in some sense goes all the way back to the Shakey project. An alternative approach would be to perform explicit binding pair-wise across sub-architectures, i.e. to perform *distributed explicit binding* as opposed to our approach of *centralised explicit binding*. This would mean that if entities from N different sub-architectures were all related to one another $N * (N - 1)$ pairwise bindings would be required, giving an overall complexity order of $O(N^2)$. In our scheme N abstractions must be performed and N bindings performed on those abstract representations, giving an overall complexity order of $O(N)$. The relative overall run-time also depends on the relative processing cost of the processes of abstraction and binding on abstract representations, versus that of the processes of binding on non-abstracted representations. In the PlayMate we have found that the abstraction process is time consuming to design, but quick at run-time. Perhaps the run time of binding on non-abstract representations can also be low (it may be performed rapidly by using a content based memory for example). A second challenge for distributed explicit binding is that bindings must be made consistent, and that this must be done incrementally across all the sub-architectures involved. This would make an interesting topic for future work.

We now describe how our binding process works for the case of some simple steps in our example. The discussion should be followed with reference to Figure 3. It is important to note that to simplify the discussion we treat

the steps of the process as if they were synchronous. In fact all the processing runs asynchronously. First recall the script:

```
Human (H) puts a blue square down on the table
H: ‘‘This is a blue square.’’
```

This simple example requires that the robot understand that the sole object in view is the object being referred to. In one sense to bind these two information items (the scene object in the visual working memory and the discourse referent in the communication system working memory) is trivial, the robot must simply associate the only visual object there is with the only discourse referent there is: `"blue square"`. But to do the binding in a way that is general requires that the robot understand that binding them is permissible because of what is known about them. In this first step, since the robot does not yet understand from visual input alone that the object is blue and square, the binding must occur purely on the basis that since they are both structures represent physical things they could be the same. Since no other binding is possible they are bound.

Figure 3 shows this process. As described in Chapter 2, within each sub-architecture a binding monitor component looks for new entities in that sub-architecture's working memory. When a new information structure is created, the monitor writes an abstract amodal representation of that structure's contents to the working memory of the binder. This abstract representation is called a proxy. A binding monitor may write several proxies to the binder to capture different aspects of the information. In our example, after time $t = 1$ when the*scene object* is created in the visual working memory, the visual binding monitor creates three proxies. The first proxy represents the physical object, the second proxy the location of this object, and the third proxy the relation between the first two proxies — in this case the simple fact that the object is at the location. Each proxy is essentially a bag of features, which are attribute-value pairs. In our example, the first proxy has the feature that it's a type of physical thing, and the second that it is a location. Note that the abstraction process into these amodal descriptions of the objects is entirely decided by the binding monitor.

Also by time $t = 2$ a separate entity has been created for the spatial location in the spatial sub-architecture. By time $t = 3$ a proxy for this has also been created on the binding working memory. Recall that the purpose of the binder is to decide which proxies are related to which other proxies. If proxies are related it groups them into a set called a *union*. The precise algorithm for matching is described in Chapter 2, but in the basic idea is that if enough common features of proxies have values that match, and if there are no features in common that mismatch, the proxies are bound into a union. To restrict binding the binder also assumes that different proxies from the same sub-architecture can't be bound. In other words the binding process trusts that the modality specific sub-systems generate the right proxies.

In our example, since the spatial proxy with a location and the visual proxy representing that location match (the locations are the same) they are bound into the same union. Since the spatial information here was derived from vision this seems redundant. In fact it is not since we could equally derive spatial information from language, and the spatial sub-system does not tell the binder the origin of it's representations. Thus the binder is blind to the processing trails that exist from the proxies back to the source data.

During this processing the human has made the utterance: `"This is a blue square."`. Parsing leads to a logical representation of the semantic content of the utterance, together with a record of the discourse referent. The binding monitor for the communication sub-architecture creates a proxy from the discourse referent, as seen at time $t = 4$. This proxy contains information that the referent is blue, square, and that it is a physical thing. Since the only feature this proxy has in common with the visual proxy for the object is *type*, and the types match (they are both physical things) the proxies are bound into the same union. So if one modality provides a richer description of an entity than another modality this does not prevent binding. In our example the binding process ends here.

One key design decision is the abstract set of features and values employed for binding. These form our core amodal language. In our case, not only are these features chosen carefully, but they are related in a type network or ontology. This type network allows matching of entities lower in the type hierarchy with entities higher up. Thus if the visual system sees a toy cow on the table, but the human refers to it as an animal the type network allows matching. This kind of reasoning about types and their relations is performed by the Coma sub-architecture. This is mostly used in the Explorer scenario, so we do not discuss it here, but it allows what we call *ontology mediated binding*. This is another powerful aspect of our approach.

We have now covered the basic notions of binding as they are realised in the PlayMate system. We have introduced the ideas of *implicit*, *explicit*, *incremental* and *ontology mediated* binding. We have described in detail how *explicit binding* occurs. We have emphasised that it relies on the ability to abstract from modality specific representations into a common amodal representation. Our implementation of this binding approach is also asynchronous, and does not depend on information arriving from different modalities in any particular order. This is important when trying to integrate representations that change at different rates. As mentioned in Chapter 2, an important aspect of the abstract features we have chosen for the PlayMate is that they are temporally stable. We now turn to the other central system: the motivation and planning sub-architecture.

2.6 Motivation and flow of control

The motivation and planning sub-system plays the role of receiving potential system goals from other sub-systems, and then deciding which ones the

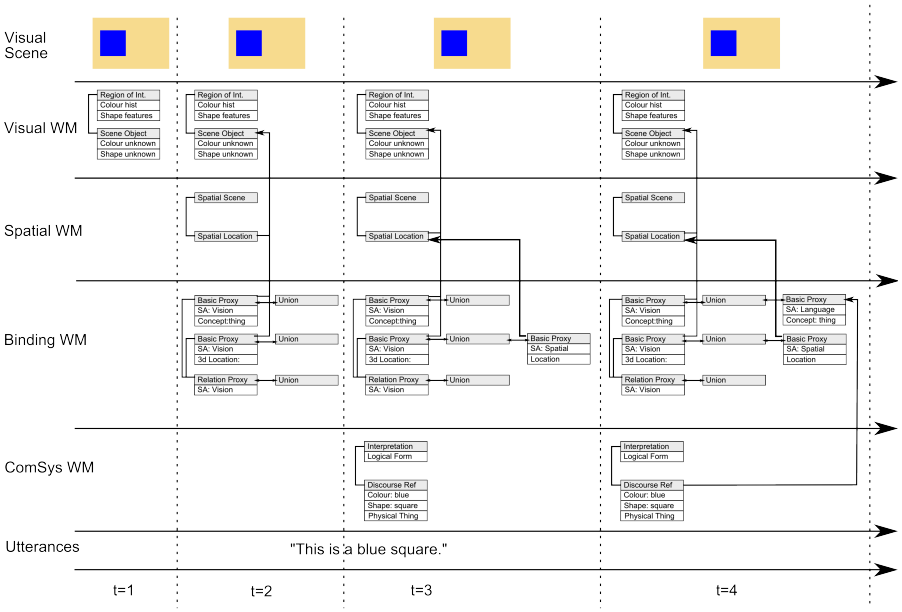

Fig. 3. A timeline for a simple binding between a visual object and an utterance. In a real system run these events will occur asynchronously and thus in a relative order that depends on the timing of the utterance and the placement of the object. For simplicity we have presented the system state here at chosen fixed points. Arrowed links indicate one way references, while non-arrowed connections indicate two way references between representations. The rows represent the working memory contents for each sub-architecture listed on the left.

whole system should pursue (motive management) and how (planning). We wanted to achieve this in as flexible and general purpose a way as possible. For overall system behaviour the planning occurs at an abstract level, using representations that are essentially those employed by the binder. Within other sub-architectures (manipulation, communication and vision) domain specific planners fill out the details of individual steps of the high-level plan.

The motivation sub-system has a *monitor* component. This is a different to monitors elsewhere in that it essentially keeps an up to date local record of entities on the binding working memory that can be used for planning. It watches the proxies and unions that appear on the binding working memory and keeps a list of them - grouped by union - with pointers to the proxies themselves. This list is referred to as the *Address-Variable Map* (AVM). The AVM is used by the planner to access information about the world via the binder.

New system goals are adopted in a three-stage process. First a sub-system such as the communication system or the visual system posts a proxy to the motivation sub-architecture working memory. These proxies contain informa-

tion relevant to deciding what kind of system goal might be raised. This allows sub-systems to raise potential system goals in parallel and asynchronously. In the second stage the *motive generator* component then reads each proxy as it arrives, and decides whether or not to generate a *motive structure* on the working memory. In the third stage the motive structure is picked up by a *motive manager* and a decision is made about whether to turn that motive into a system goal. If so then the motive will be turned into a goal posted to the motivation working memory, which will make reference to variables listed in the address variable map. This three-stage process allows local sub-systems to raise potential system goals, for the system to filter these and to maintain a resulting set of motives, and then to choose which of these motives will be acted upon now. This allows the system to switch goals if urgent new motives arrive, and not to forget old motives that were of low priority, but which can be acted on later. Motives can therefore be thought of as very similar to desires in a BDI framework.

This system goal, posted by the motive manager, will in turn be picked up by the planner. Before planning can begin the planner asks for the information from the proxies (referred to by the AVM) in a form suitable for planning. This translation process is carried out by a *state generator* component. Following the planning process an *action dispatcher* sends action requests to different sub-architectures, and checks that the sub-architectures claim they have been executed. The planner, being continual, also includes an execution monitor which checks the achievement of the plan steps via the *state generator*, the AVM and the proxies referenced by the entries in it.

To illustrate the process of raising and acting on motives we return to our example. In the following sections we will show how the binding and motivation approaches described above provide a general way to tackle both tutor driven learning, self-driven learning, clarification, and following instructions or answering of human posed questions.

3 Key innovations

In the following sections we will go through the steps of our example, using this as a way to describe the commonalities in how the PlayMate handles different types of activity at a system level. The example script requires that the PlayMate be able to handle cross-modal learning, question answering, question asking for clarification, and mediation between qualitative and metric representations of space. Since the component technologies have been described in detail in various chapters we will not focus on the details of those here, but on how the system level flow of information allows the right sub-systems to be active at the right time.

3.1 Cross Modal learning

In Chapter 7 we described methods for cross modal learning at an algorithmic level. In particular we described a continuous learning algorithm that is used for learning maps between vision and language. Specifically it learns the correlations between the abstract qualities of objects (colour and shape names) or relations between objects (names for spatial relations), and features calculated directly from low-level image properties (such as colour histograms or local shape features). The framework described in Chapter 7 allows for the learning to be either tutor driven or tutor supervised. To recap, in the first approach the tutor drives the learning process by choosing the descriptions of the objects that provide the qualitative labels. In tutor supervised learning, the learner raises queries about the visual objects presented in order to obtain qualitative descriptions. In our example the first exchange is tutor driven:

```
Human (H) puts a blue square down on the table
H: ‘‘This is a blue square.’’
```

Whereas the second exchange is an example of tutor supervised learning:

```
H picks up the red square and puts down a red triangle to the
right of the blue square.
R: ‘‘What is the thing to the right of the blue square?’’
H: ‘‘It is a red triangle.’’
R: ‘‘Ok.’’
```

At a systems level the challenge is to decide when to engage in one type of learning or another. In addition the system must distinguish between when learning activity is required, versus another kind of activity (question answering, physical action). To do this we use the motivation system described above. Sub-systems raise potential system wide activities, and the motivation sub-system chooses between them. Learning also raises a second problem for our broad approach. When I give a description intended for learning, how does the learner know which object is being referred to? In the section on binding above (Section 2.5) we described how we handle this, binding only on common features, and where necessary using the ontology to handle binding across the type hierarchy. To understand how the motivation and binding solutions work together to enable cross-modal learning at a systems level we return to our simple example of `"This is a blue square"`. To aid the discussion Figure 4 shows a timeline that begins where the time-line from Figure 3 ends. As described in the section on motivation, when the utterance is processed a proxy telling the system about the intent of the utterance is posted on to the working memory of the motivation sub-architecture (time $t = 1$). This proxy contains the information that the ComSys has identified as new. The *motive generator* has a number of templates for creating motives from proxies posted by other sub-systems. In this case it raises a motive and posts it onto the motivation working memory ($t = 2$). Motives are essentially possible system

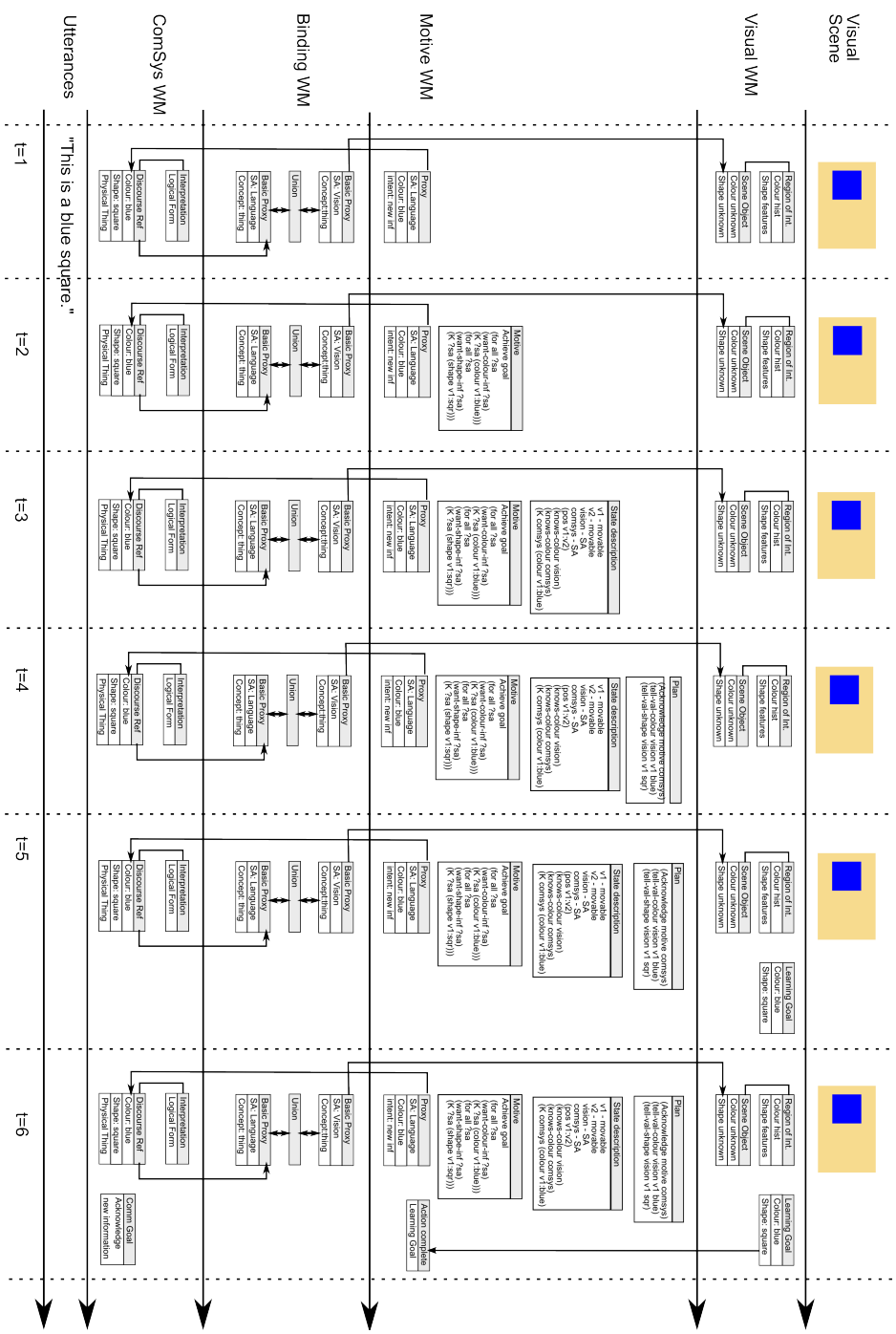

Fig. 4. The extended timeline for learning activity for the `"This is a blue square."` example.

goals, expressed in the language of the planner. The goal arising from the statement `"This is a blue square"` is:

```
(for all ?sa (want-colour-info ?sa) (K ?sa (colour v1:blue))
and
(for all ?sa (want-shape-info ?sa) (K ?sa (shape v1:square)))
```

where `?sa` refers to any sub-architecture, and `K` is a modal operator for believes, such that `(K agent fact)` means that some agent believes some fact. As described in Chapter 6, our planning language allows us to make both epistemic states into system goals. In addition we model the architecture as a multi-agent system in which the sub-architectures are agents. In order to take advantage of this all the other sub-architectures register with the motivation and planning sub-architecture which kinds of beliefs they are concerned with. These *information service offers/wants* are stored during the entire run of the system on the motivation sub-architecture working memory. To support cross-modal learning both the ComSys and Vision sub-architectures register that they can offer information about colour and shape. They also register that they want to be told information about colour and shape that may be provided by other sub-systems. In the case of cross modal learning the goal created essentially says that all agents that want information of the type that has just arrived should believe this new information. If the motive is adopted by the *motive manager* the goal is adopted by the *planner*. It requests an abstract description of the state suitable for planning, which is in turn supplied by the *state generator* and posted onto the working memory ($t = 3$). This state includes information about the general service offers and wants of different sub-architectures. It also represents the specific beliefs that sub-architectures have. As mentioned previously, the objective of binding is partly to provide stable entities to support planning. The state generator essentially uses the abstract state descriptions used by binding, and re-represents them in the language of the planning domain. A plan is produced ($t = 4$) and its steps are executed one by one ($t = 5, t = 6$). After each plan step has been executed the continual planner we employ performs execution checking, again using the abstracted state description provided via *binding* and the *state generator*. The plan in this case is very simple:

```
(acknowledge motive comsys)
(tell-val colour vision v1 blue)
(tell-val shape vision v1 square)
```

It simply asks the motivation system to tell the ComSys that its request is being dealt with, and then tells the vision system about the colour and shape of the object. This plan has two important features. First it does not decide whether or not the human needs an acknowledgement. That decision is left to a separate dialogue management and planning process. This means that our continual planner is isolated from some aspects of the system's activity:

planning is carried out in a distributed fashion by a variety of specialised planners. The second point is that the discourse referent `blue square` and the visual object were bound explicitly, but the new information to support learning was sent to vision, avoiding the binder. Thus in this case learning occurs mediated centrally, but information exchange occurs between modalities, not via a central world model.

In our second learning exchange recall that the learning is driven by the robot:

```
H picks up the red square and puts down a red triangle to the
right of the blue square.
R: ‘‘What is the thing to the right of the blue square?’’
H: ‘‘It is a red triangle.’’
R: ‘‘Ok.’’
```

This is handled at a system level in a very similar way to the tutor driven learning. When the visual system realises that it is unsure of the qualitative colour and shape labels for the red triangle it posts a proxy to the motive sub-system. Rather than containing new information this proxy is marked as representing a question. This time the motive generator creates a goal that the vision sub-architecture wants information on the colour and shape of a visual object:

```
(K Vision (Colour v1 ?c))
```

This goal is then planned for as previously, and a plan will be produced of the following form:

```
(acknowledge-goal-accepted vision)
(ask-val colour motive comsys v1 ?c)
(ask-val shape motive comsys v1 ?s)
(tell-val colour motive vision v1 ?c))
(tell-val shape motive vision v1 ?s))
```

In this case the planner realises that the ComSys can be asked about the colour and shape of the object and that it can then send this information directly to the vision sub-system to support learning. It is of course perfectly possible that a motive for learning would be raised on behalf of vision while the speaker was making an utterance intended to support learning. In this case the system should be able to catch this by realising that a simpler plan will suffice.

3.2 Clarification and Question answering

We now briefly show how both question answering and clarification requests can be handled using essentially the same mechanisms described above for

tutor driven and tutor supervised learning. Recall that in the dialogue there is the following exchange:

```
H picks up the red triangle and the blue square and puts down
a blue triangle.
H: "What colour is this triangle?"
R: "It is blue."
```

In this case the proxy will be raised by the ComSys, and converted into a goal for the ComSys to know what the colour of the triangle is. In this case the plan will be:

```
(acknowledge-goal-accepted comsys)
(tell-val colour motive comsys v1 blue)
```

Because the visual system has analysed the scene, and confidently classified the object's colour this information has been stored in binding. Thus the planner has access to this. The discourse referent for `"this triangle"` has also been bound to the visual object. This means that the planner doesn't need to ask vision. Essentially the reason for this boils down to the fact that in our current system visual processing is data driven, so information flows directly from vision to the binder. In Chapter 2 we outlined how the visual system could have task driven processing. This would make it much more like language, in that vision would need to be asked, so that the plan would need an extra `ask-val` step as did the one for tutor-supervised learning. Note that both cases are handled by the planning process, rather than by the designer.

So far we have seen that our system handles two kinds of learning, and question answering in essentially the same manner. Finally we see that clarification requests can also be handled in a similar way. In the next step of the dialogue the human places two objects on the table:

```
H picks up the blue triangle and puts down a green triangle
on the right and a red circle on the left.
H: ‘‘What shape is this thing?’’
R: ‘‘Do you mean the green thing or the red thing?’’
H: ‘‘The red thing on the left.’’
R: ‘‘It is a circle.’’
```

In this dialogue the utterance from the human is ambiguous. Perhaps the robot should assume that the most salient, or recently deposited object is the topic of the query. However, lacking such information it must clarify the ambiguous reference. In our case this occurs in the process of the system's attempt to raise a motive to answer the question: ‘‘`What shape is this thing?`’’. There will be three relevant proxies on the binder: one from vision for the green triangle (`v1`), one from vision for the red circle (`v2`), and one from language for the discourse referent for `"thing"` (`v3`). Since there is not enough information from the binder to bind the linguistic proxy to either of the

others the question cannot be directly answered. An important design decision is how this is handled. On the one hand the ambiguity could be written into the planning state, together with actions that have the effect of removing ambiguities. This presents problems in that expressing the possible bindings in terms of planning operators is extremely challenging. An alternative is for the motive generator to perform simple reasoning to establish that the goal will be unachievable without further information. This is the approach we take. Thus when presented with an ambiguity that leads to an unachievable goal the motive system instead posts a goal to resolve the ambiguity. When this is resolved it posts the goal to answer the original question:

```
(K Binder (Colour v3 ?c))
(K ComSys (Shape v3 ?s))
```

The first goal is to enable the binder to get the information it needs to distinguish which object the referent should be bound to. The reasoning to generate the distinguishing features is performed by the binder itself. When `v3` has been bound to one of `v1` or `v2` the next goal of the ComSys knowing the shape of the object can be adopted. The plan for the initial goal in this case will be:

```
(acknowledge-goal-accepted comsys)
(ask-val colour motive comsys v3 ?c)
```

which will cause the ComSys to generate the clarification question: `"Do you mean the green thing or the red thing?"` as a way of obtaining the required colour information to complete the binding of `v3`. When the answer is: `"The red thing on the left."` `v3` and `v2` are bound, and the second goal is planned for:

```
(tell-val shape motive comsys v2 ?s)
```

A comparison of the relationship between these different types of control flows is given in Figure 5.

3.3 Mediating between qualitative and quantitative representations

In the final step of our example the human gives the robot an instruction to move one of the objects:

```
H: ‘‘Put the circle to the right of the triangle.’’
R picks up the circle and puts it to the right of the
triangle.
```

To act on the kinds of action commands we are interested in, the robot must be able to translate from the qualitative linguistic spatial description of the location to place the object, to both a geometric description of the location that can be used by the manipulation system (i.e. a geometric waypoint

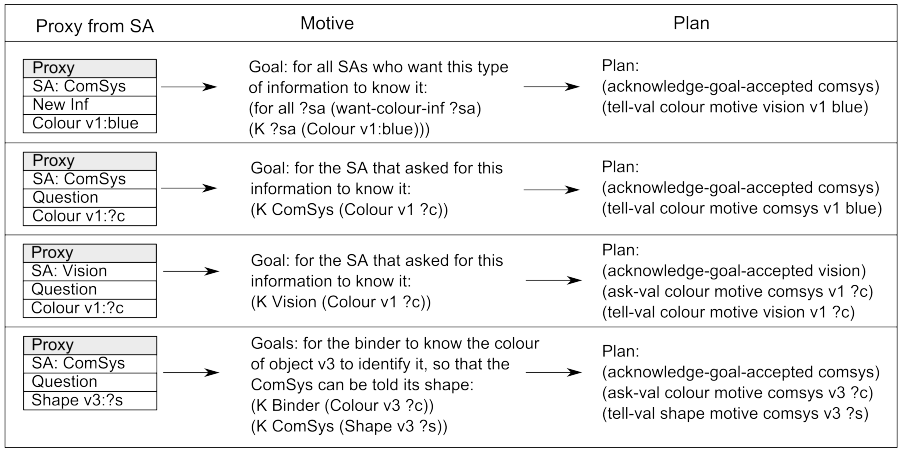

Fig. 5. Question asking, question answering, clarification of ambiguity, tutor driven and tutor supervised learning are all handled in a similar framework. Essentially each begins with a proxy posted on the motivation and planning SA working memory. Then a motive/goal is created, which takes into account the desired knowledge state of the various agents. Finally the planner creates a plan which includes knowledge generating actions. These actions may cause specialised planning processes to be run in other sub-systems.

positioned in the robot's world), and a logical description for the planning domain (i.e. a symbolic representation of this waypoint and its relationships with other waypoints). This translation involves constructing geometric models of the semantics of spatial terms. In our approach we use potential field models to create abstract spatial relationships from metric information in the spatial sub-architecture. These potential field models are very simple, but are able to take into account the context when interpreting spatial relationships. For instance, the notion of the nearness of two objects is mediated by the proximity and salience or size of other objects that may act as landmarks. The potential field models thus let us extract qualitative spatial relations that correspond to the terms as used by humans. In moving from qualitative to metric we simply take the mode of a field with respect to some landmark. This is how we generate actual goal locations for objects when given instructions like the one in the final step of our example.

4 Conclusions and Discussion

In this chapter we have looked at some of the issues involved in building a robot controller for situations where the robot and a human can talk about and manipulate a set of objects in a shared space. Working from requirements we explained how the PlayMate architectural solution provides a specialisation

of CAS. Essentially the PlayMate is a functional instantiation of the CAS architecture schema. This is only one possible instantiation among many. The functional approach, while still deeply unfashionable in parts of robotics has some advantages. In particular it is one way that we can produce abstract enough representations to support planning of system wide activity. While we do embrace an abstract world model we wish to emphasise that the approach here is not that of Shakey or similar. There are a number of design choices that we have embraced and from which we believe others could benefit. These are:

- **Shared Representations**: while systems like Shakey employed a central representation and a serial processing model we have used parallel processing of distributed representations and explicit sharing of representations. Brooks criticised approaches that pass representations as being subject to a "strong as the weakest link" problem. Sharing representations can overcome this weakness because by refining shared representations we can leverage many sources of information to refine our hypotheses and render our conclusions more reliable. This is already a feature of statistical inference in AI, we have taken the same principle and employed it at an architectural level. We have also explored one set of approaches for parallel and incremental refinement. The sharing rather point to point transmission of representations, and the parallel rather than serial refinement make the model very different to that of early representation heavy systems like Shakey.
- **Abstraction**: In our view the ability to coordinate system wide behaviour requires that we have quite abstract, stable representations that are suitable to support symbolic planning. This was another criticism of the Shakey approach: that such abstract representations are hard to come and brittle in the face of change. In the PlayMate we employ ideas like *processing trails* to allow the stable and abstract representations to point back to rapidly changing information without being corrupted by this change. In addition we employ a variety of levels of abstraction: within vision for instance we maintain both low level visual features, and qualitative descriptions of objects, in the spatial system we store both metric and qualitative descriptions of spatial relations. Finally we have looked for ways that we can move in both directions: from metric to qualitative and back again.
- **Binding**: Once the benefits of representations that are distributed across many sub-systems are accepted there is a price to be paid. This is really the binding problem: that it will not always be obvious without some inference the way in which representations from one sub-system are related to those from another. We can think of our approach to explicit binding as being a way of tying together elements across a field of evolving representations rather than as a single monolithic central representation. We have shown in different parts of this book how a cognitive system with such a distributed set of representations can benefit from strategies such as *incre-*

mental binding, *implicit binding* and *ontology mediated binding*. We have employed all these in the PlayMate system to good effect.

- **Processing control:** Once we think about a distributed system composed of many processing agents, we need to address the issue of how that processing is controlled, and how information flows through the system. How can the system produce coherent behaviour from these many parts? Rather than take an approach in which coordination emerges we have pursued one in which it is the result of informed decision making. This idea — that we can model the internal informational effects of processing — is a very powerful one. The idea need not be restricted to planning approaches as here: it is perfectly reasonable to suppose that learning strategies could be employed to acquire some aspects of processing control. In this chapter we have focussed on the way that a variety of different activities can be modelled using planning operators.

Overall in this chapter we have presented, at a systems level, a synthesis of old and new approaches from AI and robotics to building complete cognitive systems. We draw on, and synthesise, several traditions: representational approaches to AI, aspects of parallel processing from robotics, and ideas from statistical machine learning. The problems their synthesis posed resulted in new architectural ideas. Their synthesis in the PlayMate in particular has been an important driver in preventing us from shying away from hard issues we would otherwise have ignored.

References

1. M. Zillich and M. Vincze. Anytimeness avoids parameters in detecting closed convex polygons. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPR Workshops 2008*, pages 1–8. IEEE, 2008.