International Journal of Advanced Robotic Systems

OPEN ACCESS
ARTICLE

Concept and Functional Structure of a Service Robot

Regular Paper

Luis A. Pineda[1]*, Arturo Rodríguez[1], Gibran Fuentes[1], Caleb Rascon[1] and Ivan V. Meza[1]

1 National Autonomous University of Mexico, Mexico
*Corresponding author(s) E-mail: lpineda@unam.mx

Received 13 June 2014; Accepted 28 November 2014

DOI: 10.5772/60026

Abstract

In this paper, we present a concept of service robot and a framework for its functional specification and implementation. The present discussion is grounded in Newell's system levels hierarchy which suggests organizing robotics research in three different layers, corresponding to Marr's computational, algorithmic and implementation levels, as follows: (1) the service robot proper, which is the subject of the present paper, (2) perception and action algorithms, and (3) the systems programming level. The concept of a service robot is articulated in practice through the introduction of a conceptual model for particular service robots; this consists of the specification of a set of basic robotic behaviours and a number of mechanisms for assembling such behaviours during the execution of complex tasks. The model involves an explicit representation of the task structure, allowing for deliberative reasoning and task management. The model also permits distinguishing between a robot's competence and performance, along the lines of Chomsky's corresponding distinction. We illustrate how this model can be realized in practice with two composition modes that we call *static* and *dynamic*; these are illustrated with the *Restaurant Test* and the *General Purpose Service Robot Test* of the RoboCup@Home competition, respectively. The present framework and methodology has been implemented in the robot Golem-II+, which is also described. The paper is concluded with an overall reflection upon the present concept of a service robot and its associated functional specifications, and the potential impact of such a conceptual model in the study, development and application of service robots in general.

Keywords Concept of a service robot, Service robots system levels, Conceptual model of service robots, Robotics' task structure, Practical tasks, Structure of a general purpose service robot, RoboCup@Home competition, The Golem-II+ robot

1. Introduction

Current practice in service robot research should be placed in the context of particular research groups: the background and interests of the group and the group's members, the group's research network, the explicit agenda of short-, medium- and long-term goals, the preferred tools and methodologies, and the group's practice and experience. It is also necessary to consider whether the focus is academic research and advancing the state of the art, or whether it is to develop human resources and/or the development of commercial applications. In addition, it is also important to consider how the effort is directed and how productivity is assessed. One has to see, for instance, if the aim is producing robotics devices and algorithms or else fully operational service robots, publishing journal and

conference papers, patents and utility designs or research corpora. Academic productivity can also be assessed in terms of the doctoral and masters dissertations produced within the context of the group, and also through demos, formal evaluations and competitions, such as RoboCup@Home[1]. These dimensions define a very large space in which research efforts can be placed, and although the stated purpose may be to develop 'service robots', different groups may be doing very different things.

In practice, most service robots research and development groups are focused on particular specialities, like navigation [1-5], manipulation [6-10], vision [11-15], robot planning and coordination [16-22], audio [23-25], control and signal processing [26, 27], operating systems [28, 29], speech and language processing [30, 31], machine learning [32-37], human-robot interaction [38-41] and artificial intelligence [42], among others, that constitute the service robot's supporting or enabling technologies. A detailed review and systematic comparison of the different approaches in the specialities mentioned - and possibly others - is a huge task to be undertaken that is beyond the scope of this paper. However, what we would like to highlight at this point is that groups are strong in those specific disciplines, but they simply integrate other technologies needed in the construction of the robot as a whole, and the development of the robot itself is mostly a question of "implementation". From this perspective, the robot "emerges" as a side effect of the various functionalities but it is not the proper object of research, and somehow paradoxically it merely provides a context for making progress in the supporting technologies. The state of the field is hence not coherent, makes communication and interaction between groups difficult, fosters highly unbalanced development and, although there may be a large number of contributions to the supporting disciplines, current practice prevents the clear development of the service robots as an objective in itself.

The present state of the field is due, at least in part, to the lack of a clear and explicit concept of a service robot that is shared by the community and - consequently - of guidelines on how to articulate such a concept in particular research efforts. This paper is concerned with the analysis of such a concept and its impact on the development of service robots in general. In Section 2, we place the problem in the context of system levels, and argue that there is a service robot level corresponding to the knowledge level in Newell's system levels [43] as well as to the computational theory in Marr's system levels hierarchy [44]. We argue that a higher-level specification of the function of a service robot provides context and coherence to the enabling technologies, producing a virtuous cycle that fosters progress in the field of service robots, and also promotes advances in enabling technologies directed specifically to service robots, and hence progress in the discipline as a whole. We also propose placing a ceiling on the tasks that can be performed by machines with current technology and adopt the practical task and domain-independent hypotheses for service robots (as presented in [45]), after the corresponding hypotheses for dialogue systems [46], and pose that the conceptual model for a service robot needs refer to such a notion and hypotheses.

In Section 3, we discuss how the concept of a service robot can be articulated in practice through the explicit definition of a specific conceptual model for particular service robots. This consists of a highly abstract specification of what the robot does from the point of view of human users in terms of a set of behaviours at the level of the task and a behaviours composition mechanism. The conceptual model also permits the introduction of an explicit notion of a *task structure*. We also discuss two kinds of composition mechanisms that we call *static* and *dynamic*. The former is appropriate for situations in which the task structure can be known in advance through analysis. Instances of this kind of task are the standard scripted tests of the RoboCup@Home competition. The dynamic composition mechanism, which is used when the robot has to interpret and perform arbitrary commands and the structure of the task is not available *a priori*, is discussed in Section 4.

The abstract specification with an instantiation of the conceptual model is illustrated in Section 5. For this, we use the notion of a dialogue model for the specification of behaviours, the SitLog programming language for the specification and interpretation of the task structure [45] and the IOCA architecture [47]. The static and dynamic composition modes are exemplified with the *Restaurant Test* and *Endurance General Purpose Service Robot Test* (otherwise known as the *EGPSR Test*) of RoboCup@Home. The conceptual model and these applications have been implemented in the robot Golem-II+, which is described in Section 6.

The main contribution of the present paper is in the articulation of a conceptual model for a service robot in which the specification of generic tasks is stated at a functional level that is oriented to the human user. This level consists of the specification of the robot's competence and is distinguished from the algorithmic and implementation levels that are commonly the focus of robotics research and which determine the robot's performance. The paper is concluded in Section 7, with an overall reflection on the framework and methodology for the development of service robots in diverse application domains and the impact of the conceptual model in the field as a whole.

2. Concept of a service robot

Service robots are the product of implementation efforts and 'emerge' from the integration of diverse technologies, which are in turn supported by system software and

1 http://www.robocup.org/

utilities of different sorts. Questions about the design and implementation of perception and action algorithms can be stated explicitly in terms of specific functionalities and constraints, and the resulting devices can be assessed in relation to such specifications. However, the question of what is that we do when we design and build a service robot is somehow more difficult to answer. To grasp this point, we draw an analogy between service robots research and the design and construction of auto-mobiles of the standard sort. The car comes from the integration of a number of enabling structures and systems, mainly the body shell, the engine, the transmission system, the suspension system, the steering system, the breaks and the electrical equipment. Each of these technologies is the product of a research and development field with its own questions, practices and traditions; however, the car as a unit has a functional definition which consists of transporting people with some range of specific needs (e.g., sport cars, family cars, etc.), and this specification determines and regulates the specification for the particular structure and systems enabling such functionality. Hence, the evolution of auto-mobile technology can be seen as the product of a virtuous cycle between the function and the enabling technologies.

The relation between the design object and the enabling technologies can be thought of in terms of the notion of system levels. This notion is familiar to the philosophy of science, in which more specific sciences 'reduce' to more general ones, like chemistry which reduces to physics. Each discipline has its own focused phenomena, which are described by a set of general laws and a specialized vocabulary of theoretical terms, but at a particular level of abstraction that is relevant to the phenomena of interest. The reduction proper involves a mapping from the laws and theoretical terms between the corresponding theories. These notions have been applied to computing systems by Newell [43]. Newell's paper was motivated by the lack of a clear or common understanding of what knowledge was at the time, and yet there was a very significant effort devoted to the construction of knowledge-based systems, very much like the current situation in service robots research.

System levels in Newell's sense involve independent layers with a well-defined input, output and transfer function, which can be thought of as systems in themselves or else can be used in the construction of higher levels. Newell's levels for computational systems are the physical level, the device level, the electronic circuit level, the logic circuit level, the transfer-register level (i.e., computer architecture), the symbol level (i.e., programming languages) and, on top of this hierarchy, the knowledge level. An important distinction introduced by Newell was that all levels but the knowledge level reduce to the next down in the hierarchy, in the sense that a computer program written in a programming language can be mapped down into the computer architecture, or a logical circuit can be mapped directly into its implementation in an electronic circuit. The knowledge level, for its part, cannot be reduced. For this, the knowledge level is not only at the top of the systems level hierarchy but it also has a particular quality that makes it altogether different from all the other system levels. Function is stated at the knowledge level and it stands apart from all supporting or enabling technologies; for this reason, we think of the car as something that emerges from its constituent parts but which cannot be reduced to them. We can pose the same distinction for the service robot and think of a functional specification of what the robot does from the point of view of people, and this specification can be distinguished from the robot's mechanisms and systems.

An alternative to Newell's system levels, although somehow from a different perspective, was introduced by Marr [44], who distinguished between three different levels that he called the *computational theory* level, the *representation and algorithmic* level and the *implementation* level; the first corresponds to the system's functional specification and addresses the *what* questions in a very general and abstract way; the second addresses the *how* questions involving symbolic representations and specific algorithms, and the third (lower) level addresses questions related to the details of the computer's architecture and hardware devices. We propose that in the present context the first corresponds to the service robot functional level, the second to the enabling technologies level, and the third to the middleware (e.g., operating systems and communications, software agents, hardware drivers, etc.), which is the level that is addressed by system programming. The correspondence between Newell's and Marr's levels and the present proposal is illustrated in Table 1.

	Newell's Levels	**Marr's Levels**	**Proposed Levels**
Emerge	Knowledge	Computational Theory	Functional Specification
Reduce	Symbol	Representation and Algorithmic	Robotic Algorithms
		Implementation	System Programming

Table 1. Correspondence between Newell's, Marr's and Service Robot's System Levels

Newell's, Marr's and the present notion of system levels should be distinguished from actual computer architectures, like subsumption architectures [47], cognitive architectures [48] or layered architectures [18], as these are mostly orthogonal notions. We also need to consider the limitations of current technology in relation to open tasks that can be performed by people in natural environments. It is clear that human higher mental functions, like language, vision and memory, and also intentional motor behaviour, like walking or grasping objects, for instance, are much more complex than the functions that can be

performed by the most sophisticated current machines, and that a full understanding of these functionalities is far removed from our current state of knowledge. Hence, it is necessary to place a reasonable limit on the things we can do with service robots. For this, we adopt the practical dialogues and domain-independent hypotheses suggested by Allen for dialogue systems [46], and pose the corresponding *practical task* and *domain-independent* hypotheses for service robots, as follows: 1) The structure of practical tasks, although complex, is significantly simpler than open human tasks, and 2) within the genre of practical tasks, the structure of the task and task management are independent of task domains and the particular task being performed, as has been elaborated upon in [45]. Practical tasks are then restricted to scenarios with a well-defined context, composed by a number of agents with their corresponding goals, expectations and intentions, referring to a particular spatial and temporal situation where the service robot can be expected to achieve its goals and satisfy the human-user consistently. Current task scenarios or "tests" in the RoboCup@Home competition are toy instances of practical tasks and should be thought of as proof-of-concept scenarios for the corresponding real-world scenarios in which we can expect service robots to perform in the future. The domain-independent hypothesis, in turn, states that it is indeed possible to produce machines that can be directed to different domains and tasks with the same generic structure and mechanisms. Accordingly, we restrict the subject matter of service robot research and the conceptual model of service robots to the study of the behaviours and composition mechanisms subject to the practical tasks and domain-independent hypotheses.

3. Conceptual model and task structure

We proceed now to discuss how the concept of a service robot described above can be articulated in practice. For this, we abstract over hardware devices and their associated algorithms, and focus on the functionalities that these provide from the point of view of the human-user. We refer to each basic functionality in this set as a *behaviour* and to the execution of a behaviour as a *basic task*, such that complex tasks result from the combination of basic tasks, giving rise to the notion of a *task structure*. Whenever the basic tasks and the composition mechanism have an explicit specification, we say that the service robot has a *representation* of the task structure. This representation may include knowledge of a number of situations that the robot needs to go through during the execution of a task, and each situation can be construed in terms of the expectations of the robot in that situation, the actions that the robot needs to perform whenever a particular expectation is met in the situation, and the situation that results from performing such an action (or what to do in case no expectation is met). The task structure can be *static* or *dynamic*, depending on whether it can be known and specified in advance or else whether it instead unfolds along the way, respectively.

There is a very large range of possibilities regarding the selection and specification of behaviours and composition mechanisms, with their corresponding properties, and we here pose that the conceptual model of a particular service robot consists of this particular choice. Hence, the field of service robots can be construed in terms of the study of behaviours and composition mechanisms with their theoretical properties and empirical validation. In a sense, all service robots have a conceptual model; however, the more explicit this is, the better the properties of the robot are understood and capitalized upon by its designers and users.

Furthermore, the catalogue of the robot's abilities and its composition mechanisms define the robot's competence and permit us to ask explicitly what it is that the robot can in principle do. However, abstract specifications can be implemented with different algorithms and computational devices, and physical robots with the same conceptual model may perform differently with different implementations. In this regard, Chomsky's distinction between linguistic competence and performance [49] can be applied to the field of service robots research: while "the task grammar" defined by a set of basic behaviours and composition mechanisms states the robot's competence, the actual robot's performance depends upon the particular choice of enabling technologies, system software and physical devices.

3.1 Task specification language

The explicit representation and interpretation of the task structure requires a specialized programming language so that final applications can be developed and tested with in reasonable time and with reasonable effort. Such a language should have enough expressive power to state a basic and composite task in a declarative way, and should also be rich enough to allow for the expression of content and control information; it should also have abstraction capabilities to express complex behaviours in a simple way. The design and implementation of such a specification and programming languages is another area of service robots research, and significant efforts in this regard are already apparent ([45, 50-55]).

3.2 Task structure and behaviours

Task structure and behaviours naturally define two layers of functionality, as illustrated in Figure 1. The upper layer stands for a composition of behaviours that constitute the task structure of a particular application, and the lower layer corresponds to the set of basic behaviours constituting the robot's native capabilities. For instance, the *Restaurant Test* of RoboCup@Home is an application defined in a scenario including a number of shelves where different kinds of objects are placed, like food, snacks and drinks, and also a number of tables where particular objects should be delivered. The test involves a recognition round

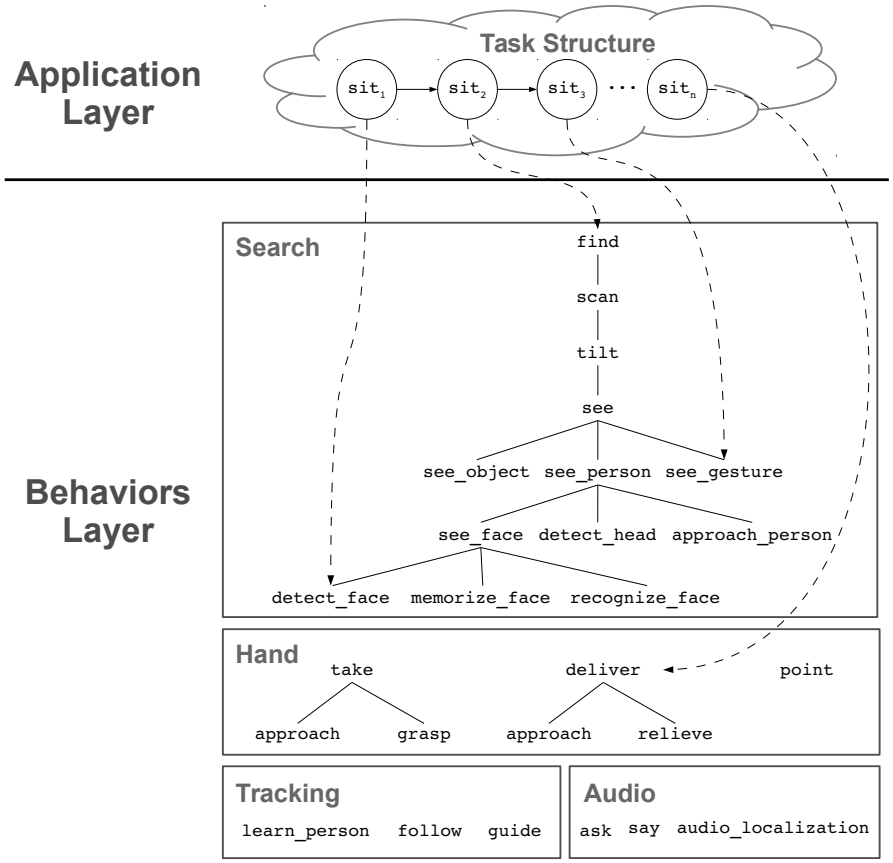

Figure 1. Application and Behaviours Levels

in which a human guide shows the robot where the shelves and tables are, so that the robot is able to build a map of the spatial situation; when this stage of the test is completed, the robot is asked to get a number of specific objects and take them to specific designated locations. An order might be *take the orange drink and the chips to table one, and the soup to table two*. The task structure consists of the situations that the robot needs to visit in order to build the map, as well as to take and deliver the order with its corresponding subtasks. As the stages that the robot needs to go through in this scenario are known in advance, the task structure is static.

Basic behaviours in turn are structured objects defined in terms of other behaviours, as illustrated by the hierarchies of **`find`**, **`take`** and **`deliver`**. The behaviour **`find`**, for instance, has as its input arguments a list of persons, a gesture or a list of objects to be found, a discrete search path and the specification of direction of observations (horizontal and vertical) that must be made at each path location, and returns the identifier of the target or list of targets found with their corresponding positions and poses, and some additional control information including the status of the last observation made in the search process. As can be seen in Figure 1, **`find`** is a composite behaviour that uses **`scan`** at each path position; **scan** in turn is a behaviour that moves the neck horizontally and uses the **`tilt`** behaviour at each scanning orientation, which in turn moves the neck vertically and makes an observation at each tilt position. The observations are properly controlled by the **`see`** behaviour, which takes into account whether the object sought is an object or a person, and so on.

However, independently of its internal structure, a behaviour is also an atomic unit that can be a part of the task structure directly. This is illustrated by the directed dotted lines connecting situations of the task structure in the upper layer with behaviours. From the perspective of the task structure, the behaviours **`detect_face`**, **`find`**, **`see_gesture`** and **`deliver`**, for instance, can be used on-demand according to the application requirements. Furthermore, the specification abstracts over the actual algorithms and implementation considerations, and hence the abstract functional requirement is met.

Finally, the behaviours in the bottom layer of Figure 1 are grouped in four main areas of functionality, which are *search*, *hand*, *tracking* and *audio*. This choice determines in part a particular conceptual model, but other sets of behaviours can be defined too, each giving rise to a different model.

3.3 Task management

The explicit representation of the task structure also allow the use of deliberative resources, knowledge-bases and task management strategies dynamically with the

execution of a task. The task structure may also involve knowledge of constraints, like the time allowed to complete the task and the scores for achieving total and partial goals, as well as knowledge of the robot's own physical resources, like the number of hands and their state (i.e., holding or free) along the execution of the task, and the locations and distances between the places of the scenario, which may be collected dynamically. Additional *a priori* knowledge may be available, such as the time it takes to accomplish a particular basic task and the probability of completing it successfully.

An explicit representation of the task structure also permits the identification of deliberative points where diagnosis, planning and decision-making may be particularly relevant. A particular deliberative situation occurs, for instance, once the robot has received an order in the *Restaurant Test* outlined above; at this point, the robot should induce a number of action plans with different partial and total goals - with their corresponding values - and decide upon a course of action. Once the decision is made, the robot should embark in the execution of the selected plan properly. There may be other decision points where current progress should be assessed, alternative plans could be induced, and a new decision as to how to proceed should be made.

We also need to consider that the scenarios in which service robots are expected to perform are very noisy and that a large number of contingencies may arise along the way, so handling the time and other constraints is crucial in achieving the goals; hence, explicit task management is needed to supervise the process and make decisions along the way. The explicit representation of the task structure makes it possible to define such task management processes along with the deliberative inferences required to support it.

Task management is also relevant to handling faults due to either external or internal contingencies that may arise in the execution of behaviours. For instance, [56] present an analysis of fault diagnosis within a logical framework using naive physics and an ontology for hypothesis generation and fault prevention. A more general kind of fault occurs when the robot becomes out of context due to a mismatch between its expectations and the events in the world. In this latter situation, the robot may get back into context through an abductive inference in relation to a common sense theory about the states and actions that take place in the environment - like the home - including causal rules involving states and actions. Task management in this setting may be construed as a pipeline process involving fault detection, the formulation of a fault hypothesis through abduction in relation to the causal theory of home dynamics, and hypothesis ranking and the identification of possible courses of action through planning and decision-making. This pipeline is required, for instance, to handle dynamic scenarios where the robot cannot accomplish an explicit goal due to changes in the environment, as in the execution of commands Type 3 of the General Purpose Service Robot of the RoboCup@Home competition, as discussed below.

4. Dynamic task structure

There are scenarios in which the structure of the task is not available in advance and must be defined and executed dynamically. An instance of this situation is the *EGPSR Test* of the RoboCup@Home competition. In this test, the robot is expected to accomplish composite tasks stated through natural language commands. The commands can be of Type 1, 2 or 3 according to their complexity, and the robot must be able to infer the intentions expressed by the human user and perform the corresponding actions. In the following discussion, we distinguish the overt natural language expression through which the order is expressed, which we refer to as the *command*, from the act intended by the user or the intention proper, which we refer to as the speech act, as well as from the actual **behaviour** that has to be performed by the robot to satisfy the intention or speech act. Examples of these three types of commands are as follows:

Type 1: *Go to the kitchen, take the coke and leave the apartment*

Type 2: *Carry a snack to a table*

Type 3:

`Situation`: There is nobody in the living room, but there are people in the kitchen. The robot starts in the kitchen.

`Command`: *Go to the living room and introduce yourself*

Commands of Type 1 are constituted by a sequence of basic commands, where each command expresses a basic intention or speech act, which in turn corresponds to a basic behaviour or sequence of behaviours that can be performed by the robot directly. The interpretation of this type of command requires no inference or problem solving, but the parsing inference involved in mapping the command to its corresponding speech act, and also in mapping the speech act to its corresponding behaviour or sequence of behaviours. For instance, the speech act move can be expressed by a number of different expressions like *move to*, *go to*, *navigate to*, *leave* or *exit*. This particular speech act has one argument, which is the place or position where the robot is ordered to move, and although there may be a number of ways to express the command, the actual speech act is the same, and there is a basic behaviour or sequence of behaviours associated with each kind of speech act.

Commands of Type 2 are underspecified orders that need to be determined either through linguistic interaction with the human user or through conceptual inference (e.g., querying a conceptual taxonomy in the robot's knowledge-base), or else by a combination of these two strategies. In the present example, the command states that a snack must be carried to a table, but it does not specify which snack or which table. In order to accomplish this order, the robot

may ask the user to specify such information (i.e., solve the task through linguistic interaction); alternatively, the robot may find a particular object through vision, query whether it is a snack in its knowledge-base, and take it to any table whose location might also be stored in the knowledge-base (i.e., solve the task through conceptual inference); a third strategy might consist of finding the snack and asking the user for the table (i.e., combining interaction and inference). In any case, once the references are determined, the Type 2 command is reduced to a Type 1 command, which can be executed directly as before.

Commands of Type 3 also involve error detection and explicit task management. In the present example, the robot will go to the living room but will need to realize that nobody is there and execute a task management action, which might be to search for people in other locations. Error detection and task management are essential for robust behaviour, as robots need to be able to cope with a large number of contingencies that can appear during the execution of a task.

The higher the type of command, the higher the parsing effort; but assuming a robust and comprehensive parsing strategy, the increase in the difficulty of the three types of commands depends mostly on the need to engage in linguistic or visual interaction supported by inference, and also on the need to be aware of whether the actions performed are successful and to carry on with the appropriate actions. This may require explicit task management, involving diagnosis through abduction, planning and decision-making, considering any constraints and *a priori* knowledge, as discussed above for the task structure in general.

Another consideration is that commands, speech acts and behaviours do not necessarily correspond univocally; although this may be the case for particular commands, this is not the case in general, and several commands may correspond to the same speech act, and a speech act may require the execution of several behaviours. In addition, the same command may correspond to different speech acts, and the context may be essential for resolving the ambiguity. A study of the sentence generator for the *EGPSR Test* (RoboCup@Home's test) shows that there are only 11 speech acts that need to be understood by the robot; these are as follows: `move`, `find`, `say`, `ask`, `get`, `memorize`, `recognize`, `follow`, `point`, `retrieve` and `carry`. The set of relations between linguistic forms and speech acts for the *EGPSR Test* is summarized in the following specification, where the expressions in italics to the left of => are the verbs heading the possible command expressions, and the word to the right stands for the name of the intended speech act. As can be seen, some speech acts may be stated in several ways, like move, and one surface form, bring, has two different interpretations.

1. *move to*, *go to*, *navigate to*, *leave*, *exit* => `move`
2. *find*, *detect*, *identify* => `find`
3. *introduce*, *tell me something about you* => `say`
4. *ask* => `ask`
5. *get*, *take*, *grasp*, *fetch* => `get`
6. *memorize* => `memorize`
7. *recognize* => `recognize`
8. *follow* => `follow`
9. *point*, *show* => `point`
10. *retrieve*, *bring* => `retrieve`
11. *carry*, *bring* => `carry`

The relation between speech acts and the corresponding behaviours is not one-to-one either. In this example, there are 11 speech acts but 25 behaviours, as shown in Figure 1; there are also speech acts that need to be assembled out of several behaviours, and there are also behaviours that do not correspond with a unique speech act. Indeed, the different kinds of relations between speech acts and behaviours mostly determine the type of command.

In addition to the behaviours, the *EGPSR Test* requires the definition of a dynamic composition mechanism through which complex behaviours are assembled out of basic ones. For this, we use the symbol ==>, which has the speech act as its left and the behaviour specification as its right. The mechanism is defined in relation to the command types as follows:

Commands of Type 1 state the basic case and relate a fully determined speech act to a particular behaviour, for instance:

```
move ==> move.
get ==> find, grasp.
```

In addition, a command may be interpreted in terms of behaviours and commands, as follows:

```
retrieve ==> move, get, move, deliver.
carry ==> get, move, deliver.
```

This shows that there is a feedback cycle between the linguistic (i.e., speech acts) and the behavioural component giving rise to complex behaviours, but with a simple and well-structured interpretation regime.

Commands of Type 2 involve the specification of the arguments of the speech act by means of conceptual inference, and possibly some linguistic or visual interaction strategy, for instance:

```
carry(Class, Table) ==> query_kb(Class, Object),
                        ask(Table),
                        carry(Object, Table).
```

where **`query_kb`** is an additional internal behaviour through which the robot's conceptual knowledge-base can be queried. Additional action schema of this kind can be defined, thereby enriching the robots set of strategies, which can be used non-deterministically.

Commands of Type 3 add explicit task management to the behaviour so that appropriate actions can be taken in case performance errors occur, or in case the task cannot be executed in the actual scenario. To handle this type of command, a status argument must be included in the specification of all behaviours, for instance:

```
retrieve(status) ==> move(status_move1),
                     get(status_get),
                     move(status_move2),
                     deliver(status_deliver).

carry(status) ==> get(status_get),
                  move(status_move),
                  deliver(status_deliver).
```

The error handling and task management processes check the status of each behaviour and decide whether to proceed with the task or take an appropriate task management action.

Commands of Type 1 and Type 2 should proceed straightforwardly according to the rules of the competition, as the status argument of the behaviours must always be 'ok'; however, this argument in commands of Type 3 may have a different value, in which case the system must engage in a task management process involving diagnosis, planning and decision-making; this is, deliberative behaviour, in order to proceed with the task. The status arguments are defined for behaviours but not for speech acts, as these are the interpretation of the intentions expressed by the users, and hence the specification of the intended behaviour. In the present case, the robot ought to be able to make a diagnosis as to why there are no people in the room and come to the plausible conclusion that they have moved to another room, make a plan for visiting other rooms, and execute the *find* behaviour recurrently until the task is accomplished. In some scenarios, the system may alternatively engage in a clarification protocol to decide what to do if there are people to talk to; but in general, both interaction and deliberative abilities should be available.

Task management involving deliberative behaviour is, of course, quite complex and the conceptual model of the robot should include the required supporting inferential and knowledge-base resources, but once again the particular algorithms and implementation strategies are not part of the functional specification. In particular, the mapping from natural language statements to the corresponding speech acts is not a part of the conceptual model. This facility should of course be available, but speech and language processing are enabling technologies - as any other - and there are several possible strategies and implementations; hence, no particular mechanism should be assumed in the conceptual model.

In summary, the specification of the *EGPSR* highlights that, in addition to the predefined static composition mechanism illustrated in Section 3, there are scenarios where such *a priori* knowledge is not available and tasks have to be composed dynamically by the interpretation of commands in terms of their corresponding speech acts, and the mapping of these latter objects into their corresponding behaviours in the lower layer in Figure 1 directly. The rules in this section define such a dynamic mapping for the *EGPSR* test in the RoboCup@Home competition.

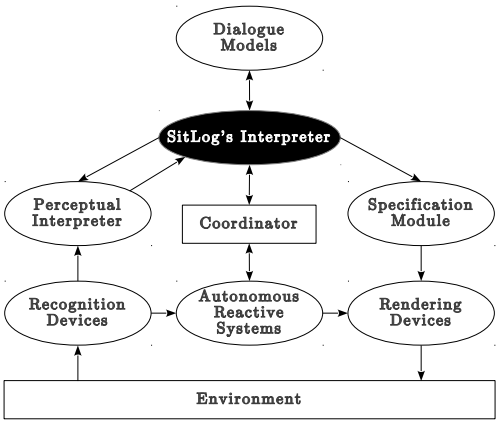

Figure 2. Interaction-oriented Cognitive Architecture (IOCA)

A final remark may be made that applications whose task structure can be defined in advance through analysis - like the standard tests of the RoboCup@Home competition - can also be thought of as a behaviour schema that could be specified and performed as a sequence of behaviours that can be interpreted and executed by dynamic composition mechanisms, as illustrated here for the *EGPSR Test*. If this requirement is met, particular tests (like Cocktail Party, Clean It Up, Emergency Situation, etc.) can be specified as scripted behaviours that can be interpreted directly, and these tests should not require a programming effort.

5. An instantiation of the conceptual model

In this section, we briefly describe a particular instantiation of the conceptual model. The central aspect is the definition of a machine for the declarative specification and interpretation of the task structure of final applications, and an interaction-oriented cognitive architecture *IOCA* [47] to relate such machinery with the perception and action algorithms and the system's software. The task structure is represented through abstract interaction protocols called *dialogue models* (or *DM*s), which in turn are specified and interpreted through the SitLog programming language [45], whose interpreter is the central component of *IOCA*, as illustrated in Figure 2.

IOCA is a cognitive architecture with three layers directed to 1) reactive, 2) interpretation and action specification, and 3) representation and inference levels, from the bottom to the top respectively. In this respect it differs from subsumption architectures, like brooks [45], which reject representations, and also from multi-robot coordination architectures, which place a planning layer at the top as its main deliberative behaviour [18], whose discussion

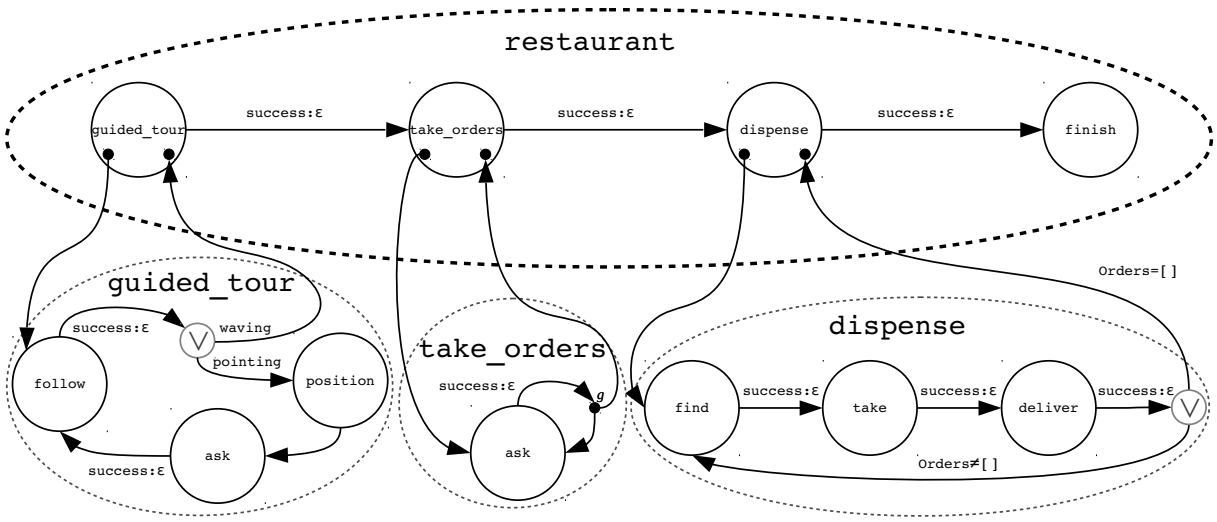

Figure 3. Static Task Structure

belongs to the algorithmic level of analysis advanced in the present paper.

SitLog's interpreter is written in Prolog, and SitLog's programs follow closely Prolog's notation. Each dialogue model consists of a set of situations or information states, and a dialogue model has a diagrammatic representation as a graph of situations. A situation in turn consists of a set of expectation and action pairs in addition to the situation that is reached when an expectation is met (i.e., when an expected state or event in the world is acknowledged through perceptual interpretation) and its associated action is performed, in addition to other content and control information. Situations are represented through a list of attribute-value pairs, as shown above.

```
[
    id ==> ID,
    type ==> Type,
    prog ==> Local_Prog,
     in_arg ==> In_Arg,
    out_arg ==> Out_Arg,
    embedded_dm ==> Embedded_Dialogue_Model,
    arcs ==>[
                    Expect1:Action1 => Next_Sit1,
                    Expect2:Action2 => Next_Sit2,
                            ...
                    Expectn:Actionn => Next_Sitn
                ],
    diag_mod ==> Diag_Mod_ID
]
```

The symbols at the left of ==> are the attribute names, and the symbols at the right stand for their corresponding values, which can be variables or expressions through which the expectations, actions, next situations and control information are expressed. Each situation has its ID, type and input and output arguments; the type indicates the kind of modality that is involved in the perceptual act through which expectations are acknowledged (e.g., vision, language, etc.). The attribute prog has as its value a local program which is executed unconditionally when the situation is reached. If the type of situation is *recursive*, there is an embedded DM which is executed within the scope of the situation, and the system as a whole implements a recursive transition network which is unfolded along the execution of the task. The attribute arcs, in particular, has as its value the list of *expectation:action* pairs of the situation, where the operator => relates each of these pairs with the next situation. DMs and situations can have arguments which are called by reference, and the diag_mod attribute permits the binding of output arguments with local information computed during the situation's interpretation.

There are three main kinds of dialogue models standing for the static task structure: 1) the task structure of final applications, 2) the set of generic behaviours in the conceptual model, and 3) the recovery protocols that are evoked when the flow of interaction is interrupted due to a mismatch between the robot's expectations at the situation and the states and events in the world. DMs of Kind 1 are developed by final application programmers, who rely only on the set of behaviours defined in the conceptual model and the specification and programming facilities provided by SitLog. Dialogue Models of Kind 2, on the other hand, are a part of the specification of the conceptual model proper, and are developed by the robot's production team. Behaviours have an internal structure which is codified in SitLog, and also an external aspect which depends upon the perception algorithms that support the interpretation of external information as well as upon the action algorithms that render the concrete actions performed by the robot through its actual physical devices. Lastly, the recovery protocols can be specific to final applications and hence developed by application programmers, or alternatively generic recovery protocols that enrich the conceptual model and are developed by the robot's production team.

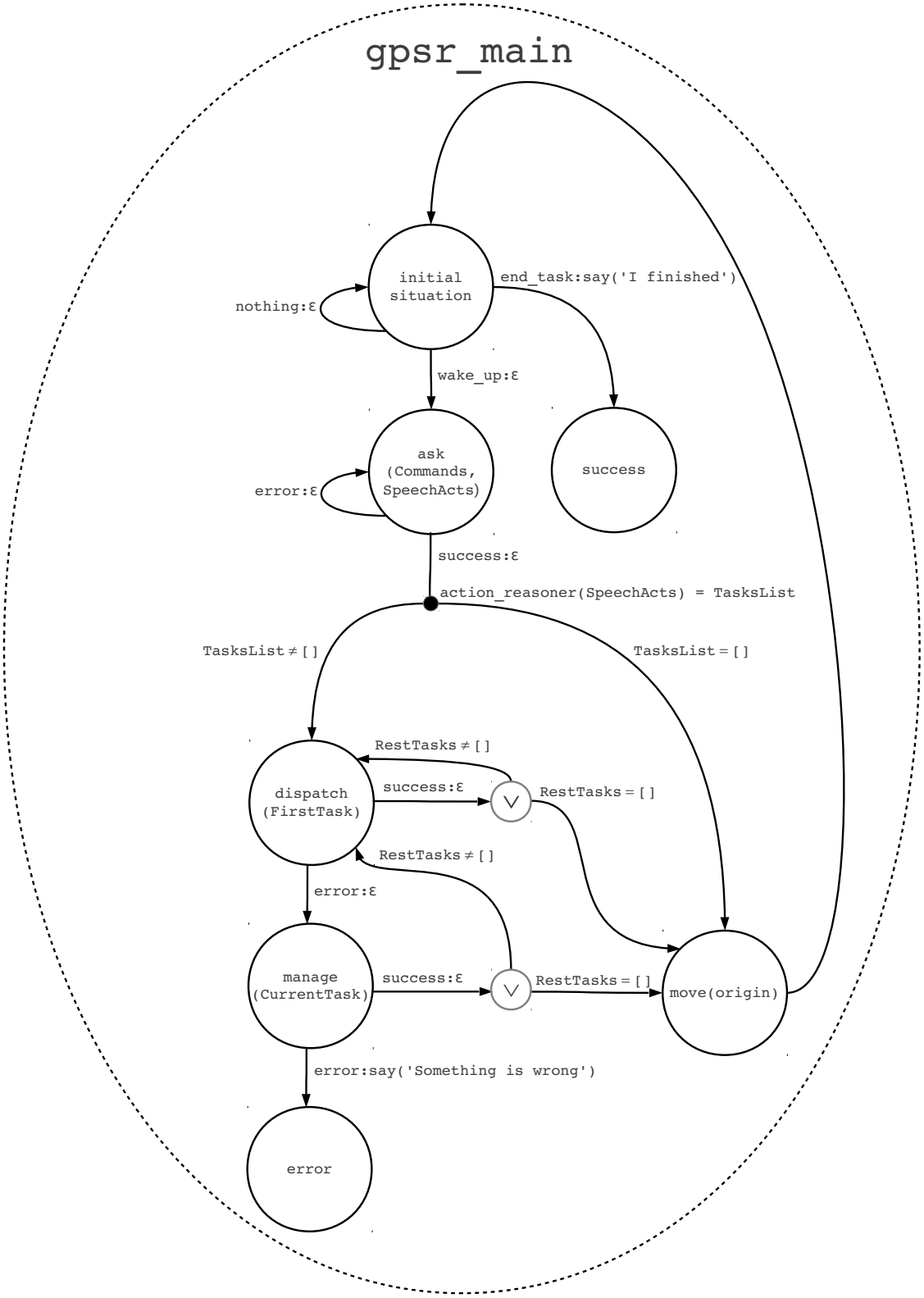

Figure 4. SitLog's specification of a General Purpose Service Robot

5.1 Example of a static task structure: The restaurant test

An instance of an application with a static task structure is the *Restaurant Test* described above in Section 3.2. The task is composed of a main DM and three top-subordinated DMs. The main DM includes three situations in addition to the final one, which is compulsory for all DMs, as shown in Figure 3. In the first, the human guide shows the robot

the locations of the shelves and the tables, and the robot builds a map of the site dynamically; in the second, the robot receives the order and in the third it moves to the shelves where the desired objects are located, according to their classes, and searches for the objects and takes them to their designated tables. Each of the three main situations embeds a subtask, represented by a subordinated DM, and instances of these models are executed recurrently as many times as needed. These subtasks can be executed in a number of ways, and the task needs to be managed dynamically, as explained in Section 3.3. In the scheme, both the main and the subordinated DMs may employ the behaviours in Figure 1 directly.

5.2 Example of a dynamic task structure: The EGPSR test

We illustrate now an instance of the specification and interpretation of a dynamic task structure. For this, we present the DM for the interpretation and execution of the three types of commands of the *EGPSR Test* as described above in Section 4. The central component of this test is a dialogue model that, in addition to managing the start and end protocols of the overall task, listens to the command, interprets it as a list of speech acts, and maps this latter list into the corresponding list of behaviours or tasks, which are executed one at a time. The SitLog's diagram of the dynamic task composition DM is shown in Figure 4. In the initial situation, the robot is waiting to be woken through a speech command or to be told that the test is finished. In case nothing happens, the robot remains in this initial situation (i.e., it is its own next situation). Once the robot is awake, the listen command situation is reached; this is a recursive situation in which the robot asks the user for a command through the **`ask`** behaviour, which also parses the command and returns a list of speech acts in the variable *SpeechActs*. Next, the action's reasoner translates this list into the corresponding list of behaviours, which is codified in the variable *TasksList*. The situation *dispatch* executes the behaviours in this latter list, one at a time. This situation is reached recurrently until the list of behaviours is empty. Otherwise, in case of error, the situation *manage* is reached. This is a recursive situation that handles the task management knowledge available to the robot. This knowledge is codified in a library of error management dialogue models, including at least one for each kind of status. The DMs in this library use behaviours, inferential strategies and interactive protocols to manage the contingency, allowing the robot to continue with the main task. If the task management action succeeds, the dispatch situation is invoked again with the list of the pending behaviours, which are executed one at a time as before; otherwise, the *EGPSR*'s DM is ended in an error state. Finally, the *EGPSR Test* requires the execution of three commands, one of each kind, and the robot needs to remain waiting after the execution of a command (while other robots are performing) before it is called into action again.

The amount of task management knowledge has a very large impact on the overall strength of the service robot and is indispensable in the execution of commands of Type 3, as without this kind of knowledge the robot could not recover from unexpected obstacles or arbitrary changes in the scenario. The current strategies use heuristics and schematic behaviours to deal with expected test scenarios, like the strategies *retrieve(status)* and *carry(status)* described in Section 4; however, to our knowledge, no service robot has been able to successfully complete a command of this type in a competitive setting in the RoboCup Competition as of date. On a deeper level, the resolution of this latter type of command involves common sense knowledge and a more complete specification should be enriched with an explicit set of causal relations that hold in a home environment. Hence, a dynamic diagnostic for unexpected situations should be performed through abductive inferences yielding the best explanation in relation to the given causal theory. Practical tasks should be provided with such types of causal theories of the application domain so that service robots can perform common sense inferences but within the boundaries of a closed domain.

The present case studies for both the static and dynamic tasks are illustrations of how the conceptual model is implemented; however, the same conceptual model has been used in the implementation of the rest of the RoboCup@Home tests, providing additional support for the generality of the present approach.

6. Description of the robot Golem-II+

Golem-II+ is our in-house service robot, presented in Figure 5. It is based on a PeopleBot model, with several major enhancements carried out in-house, both software- and hardware-wise. In Table 2, a brief summary is presented of its hardware and the software libraries used.

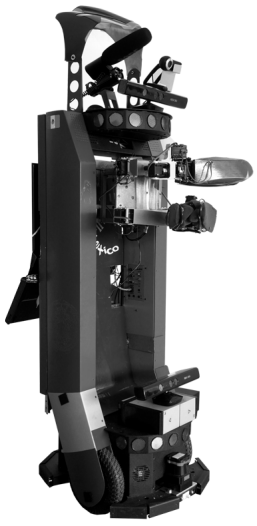

Figure 5. The Golem-II+ Service Robot

Module	Hardware	Software Libraries
Dialogue manager	–	SitLog, Prolog
Knowledge-base	–	Prolog
Vision	Microsoft Kinect, Point Grey Flea Camera	SVS, OpenCV, PCL, and OpenNI
Navigation	Mobile Robots Inc. PeopleBot Platform, Hokuyo UTM-30LX Laser	Player
Voice recognition	RODE VideoMic, M-Audio Fast Track Interface	JACK, PocketSphinx
Voice synthesizer	Infinity 3.5-Inch Two-Way Built-in Loudspeakers	PulseAudio, Festival TTS
Object Manipulation	In-house Built Robotic Arm and Gripper	Dynamixel RoboPlus
Camera/Mic. Movement	In-house Built Robotic Neck	Dynamixel RoboPlus

Table 2. Software Libraries used by the IOCA Modules and the Hardware of Golem-II+

7. Conclusions

In this paper, we have introduced and discussed a concept of a service robot. A service robot is an entity that is able to perform a number of basic behaviours and compose them in the execution of complex tasks. This concept is articulated in practice through the definition of an explicit conceptual model for particular service robots. We have discussed how the conceptual model and framework can be applied in general, and we have also illustrated the model with a particular implementation in the robot Golem-II+ with the IOCA architecture and the SitLog programming language, which have been developed within the context of the Golem Project[2]. Video demonstrations of the examples discussed in this paper, as well as the actual SitLog code, can be accessed at *http://golem.iimas.unam.mx/servicerobot*.

As an overall reflection, the present framework permits us to conceive of service robots research as a discipline of its own, consisting of the study and functional specification of useful behaviours from the point of view of human users and the ways in which these can be combined in the composition of complex tasks, including the design and implementation of specification languages for robot tasks. We propose that a clear demarcation between research into service robots and research into their enabling technologies and system software facilitates communication and collaboration, as individual researchers and groups will have a clearer idea of what the focus and system level are in relation to which their efforts are made. We also suggest that this demarcation of labour will foster a virtuous cycle between service robot research and enabling technologies, yielding progress in the field as a whole.

More generally, the definition of the conceptual model in terms of behaviours and composition mechanisms not only provides for a clear demarcation of application and robot development teams, but also determines the set of enabling technologies that are required to support a given set of behaviours. This latter specification is also functional, and the robot's design team must decide on particular algorithms and system design considerations. There may be a large range of technologies and implementations to choose from and a given selection will have an impact on the robot's performance but not on its competence, as this is determined by conceptual model. A final aspect of the design as a whole is the system software required to support the robot's architecture, which is the backbone of the robot. However, this is also an implementation decision that will have an impact on the robot's performance but not on its competence.

Finally, the conceptual model and a declarative language to state the task structure permit us to demarcate clearly the activities related to the development of the robot proper from the activities related to specifying the structure of a particular task or programming a particular application. In current practice, these two activities are heavily interwoven, as developers of enabling technologies and system programmers need to take into account specific aspects of particular applications. Moreover, application developers need to work with algorithms and system programming, making rather difficult the development and testing of final applications. Like cars, if a robot is offered to the general public, the focus of the robot-maker should be what the robot can do in principle and how well it performs in practical tasks (e.g., how reliable and efficient it is) so that final applications can be easily specified and developed, and robots can be used in practice by human users.

8. Acknowledgements

We would like to thank the support of the members of the Golem Group who participated in the development of the Golem-II+ robot as follows: Lisset Salinas, Mauricio Reyes, Hernando Ortega, Varinia Estrada, Mario Peña, Joel Durán, Albert Orozco and Sebastián Chimal. We also acknowledge the support of grants CONACYT's 178673, ICYTDF-209/12 and PAPIIT-UNAM's IN-107513.

9. . References

[1] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

2 http://golem.iimas.unam.mx/

[2] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404 vol.2, Apr 1991.

[3] J. Minguez and L. Montano. Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. *Robotics and Automation, IEEE Transactions on*, 20(1):45–59, Feb 2004.

[4] J.W. Durham and F. Bullo. Smooth Nearness-Diagram Navigation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 690–695, Sept 2008.

[5] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *Robotics, IEEE Transactions on*, 23(1):34–46, Feb 2007.

[6] J. Stuckler, D. Holz, and S. Behnke. Robocup@home: Demonstrating everyday manipulation skills in robocup@home. *IEEE Robotics Automation Magazine*, 19(2):34–42, 2012.

[7] Sachin Chitta, E. Gil Jones, Matei Ciocarlie, and Kaijen Hsiao. Mobile Manipulation in Unstructured Environments: Perception, Planning, and Execution. *IEEE Robotics and Automation Magazine*, 19(2): 58–71, 2012.

[8] Siddhartha Srinivasa, Dmitry Berenson, Maya Cakmak, Alvaro Collet Romea, Mehmet Dogar, Anca Dragan, Ross Alan Knepper, Tim D Niemueller, Kyle Strabala, J Michael Vandeweghe, and Julius Ziegler. Herb 2.0: Lessons learned from developing a mobile manipulator for the home. *Proceedings of the IEEE*, 100(8):1–19, 2012.

[9] Charles C. Kemp. Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *Proceedings of the IEEE International Conference on Development and Learning*, 2006.

[10] Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. Learning trajectory preferences for manipulators via iterative improvement. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 575–583. 2013.

[11] Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. The MOPED framework: Object Recognition and Pose Estimation for Manipulation. *The International Journal of Robotics Research*, 30:1284–1306, 2011.

[12] A. Pronobis, B. Caputo, P. Jensfelt, and H. I. Christensen. A realistic benchmark for visual indoor place recognition. *Robotics and Autonomous Systems*, 58(1):81–96, 2010.

[13] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *International Journal of Robotics Research*, 32(8):951–970, 2013.

[14] Matthew Johnson and Yiannis Demiris. Perceptual perspective taking and action recognition. *International Journal of Advanced Robotic Systems*, 2(4):301–308, 2005.

[15] Pablo Espinace, Thomas Kollar, Nicholas Roy, and Alvaro Soto. Indoor scene recognition by a mobile robot through adaptive object detection. *Robotics and Autonomous Systems*, 61(9):932—947, 2013.

[16] M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[17] Robert Zlot and Anthony Stentz. Market-based multirobot coordination for complex tasks. *The International Journal of Robotics Research*, 25(1):73–101, 2006.

[18] Dani Goldberg, Vincent Cicirello, M Bernadine Dias, Reid Simmons, Stephen Smith, and Anthony Stentz. Market-based multi-robot planning in a distributed layered architecture. In *Multi-robot systems: From swarms to intelligent automata: Proceedings from the 2003 international workshop on multi-robot systems*, volume 2, pages 27–38, 2003.

[19] M Bernardine Dias. Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments. PhD thesis, Carnegie Mellon University, 2004.

[20] Kai Zhang and Xiabo Li. Human-robot team coordination that considers human fatigue. *Int J Adv Robot Syst*, 72(3-4):541–558, 2013.

[21] Kai Zhang, Jr. Collins, EmmanuelG., and Adrian Barbu. An efficient stochastic clustering auction for heterogeneous robotic collaborative teams. *Journal of Intelligent Robotic Systems*, 11(91), 2014.

[22] Kai Zhang, Emmanuel G Collins Jr, and Dongqing Shi. Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(2):21, 2012.

[23] Multiple Direction-of-Arrival Estimation for a Mobile Robotic Platform with Small Hardware Setup. In Haeng Kon Kim, Sio-Iong Ao, Mahyar A. Amouzegar, and Burghard B. Rieger, editors, *IAENG Transactions on Engineering Technologies*, volume 247 of *Lecture Notes in Electrical Engineering*, pages 209–223. Springer Netherlands, 2014.

[24] Kazuhiro Nakadai, Toru Takahashi, Hiroshi G. Okuno, Hirofumi Nakajima, Yuji Hasegawa, and Hiroshi Tsujino. Design and Implementation of Robot Audition System 'HARK' — Open Source Software for Listening to Three Simultaneous Speakers. *Advanced Robotics*, 24(5-6):739–761, 2010.

[25] The ManyEars open framework. *Autonomous Robots*, 34(3):217–232, 2013.

[26] DeLiang Wang and Guy J. Brown, editors. *Computational auditory scene analysis: Principles, Algorithms, and Applications*. IEEE Press/Wiley-Interscience, 2006.

[27] E. Martinson and A. Schultz. Robotic Discovery of the Auditory Scene. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 435–440, April 2007.

[28] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 3, 2009.

[29] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *11th International Conference on Advanced Robotics (ICAR 2003)*, pages 317–323, June 2003.

[30] Komei Sugiura, Naoto Iwahashi, Hisashi Kawai, and Satoshi Nakamura. Situated spoken dialogue with robots using active learning. *Advanced Robotics*, 25(17):2207–2232, 2011.

[31] Xiaoping Chen, Jiongkun Xie, Jianmin Ji,, and Zhiqiang Sui. Toward open knowledge enabling for human-robot interaction. *Journal of Human-Robot Interaction*, 1(2):100 – 117, 2012.

[32] Kyuhwa Lee, Yanyu Su, Tae-Kyun Kim, and Yiannis Demiris. A syntactic approach to robot imitation learning using probabilistic activity grammars. *Robotics and Autonomous Systems*, 61(12): 1323–1334, 2013.

[33] M. Do, T. Asfour, and R. Dillmann. Towards a unifying grasp representation for imitation learning on humanoid robots. In *IEEE International Conference on Robotics and Automation*, pages 482–488, 2011.

[34] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5), 2011.

[35] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *International Journal of Robotics Research*, 31(3):360–375, 2012.

[36] Manuel Mühlig, Michael Gienger, and Jochen Steil. Interactive imitation learning of object movement skills. *Autonomous Robots*, 32(2):97–114, 2012.

[37] Sylvain Calinon, Florent D'halluin, Eric L. Sauser, Darwin G. Caldwell, and Auge G. Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics Automation Magazine*, 17(2):44–54, 2010.

[38] R. Stiefelhagen, H.K. Ekenel, C. Fugen, P. Gieselmann, H. Holzapfel, F. Kraft, K. Nickel, Voit, and Alex Waibel. Enabling Multimodal Human #x2013;Robot Interaction for the Karlsruhe Humanoid Robot. *Robotics, IEEE Transactions on*, 23(5):840–851, Oct 2007.

[39] HiroshiG. Okuno, Kazuhiro Nakadai, and Hiroaki Kitano. Social Interaction of Humanoid Robot Based on Audio-Visual Tracking. In Tim Hendtlass and Moonis Ali, editors, *Developments in Applied Artificial Intelligence*, volume 2358 of *Lecture Notes in Computer Science*, pages 725–735. Springer Berlin Heidelberg, 2002.

[40] Maia Garau, Mel Slater, Simon Bee, and Martina Angela Sasse. The Impact of Eye Gaze on Communication Using Humanoid Avatars. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 309–316, New York, NY, USA, 2001. ACM.

[41] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, and Hagita. A Communication Robot in a Shopping Mall. *IEEE Transactions on Robotics*, 26(5):897–913, 2010.

[42] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. online planning for multi-agent systems with bounded communication. *artificial intelligence*, 175(2):487 – 511, 2011.

[43] Allen Newell. The Knowledge Level: Presidential Address. *AI Magazine*, 2(2):1–20, 1981.

[44] David Marr. *Vision*. W. H. Freeman and Company, New York, 1982.

[45] Luis A. Pineda, Lisset Salinas, Ivan Meza, Caleb Rascon, and Gibran Fuentes. SitLog: A Programming Language for Service Robot Tasks. *International Journal of Advanced Robotic Systems*, pages 1–12, 2013.

[46] James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Toward Conversational Human-Computer Interaction. *AI MAGAZINE*, 22(4):27–38, 2001.

[47] Luis A. Pineda, Ivan Meza, Hector Aviles, Carlos Gershenson, Caleb Rascon, Monserrat Alvarado, and Lisset Salinas. IOCA: Interaction-Oriented Cognitive Architecture. *Research in Computing Science*, 54:273–284, 2011.

[48] Rodney Brooks. Intelligence without representation. *Artificial Intelligence*, (47):139–159, 1991.

[49] Noam Chomsky. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press, 1965.

[50] Martin Lotzsch, Max Risler, and Matthias Jüngel. XABSL - A pragmatic approach to behavior engineering. In *Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems*, pages 5124–5129, 2006.

[51] Tim Niemuller, Alexander Ferrein, and Gerhard Lakemeyer. A lua-based behavior engine for

controlling the humanoid robot nao. In *Proceedings of the RoboCup Symposium 2009*, pages 240–251, 2009.

[52] Jonathan Bohren, Radu Bogdan Rusu, E. Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the pr2. In *Proceedings of the International Conference on Robotics and Automation*, pages 5568–5575, 2011.

[53] S. Tousignant, E. Van Wyk, and M. Gini. Xrobots: A flexible language for programming mobile robots based on hierarchical state machines. In *Proceedings of the International Conference on Robotics and Automation*, pages 1773–1778, 2012.

[54] Thijs Jeffry de Haas, Tim Laue, and Thomas Röfer. A scripting-based approach to robot behavior engineering using hierarchical generators. In *Proceedings of the International Conference on Robotics and Automation*, pages 4736–4741, 2012.

[55] Stefan Schiffer, Alexander Ferrein, and Gerhard Lakemeyer Reasoning with qualitative positionalinformation for domestic domains in the situation calculus. *Journal of Intelligent and Robotic Systems*, 66(1–2):273–300, 2012b.

[56] Anastassia Kuestenmacher, Naveed Akhtar, Paul G. Plöger, and Gerhard Lakemeyer. Towards robust task execution for domestic service robots. Journal of Intelligent Robotic Systems, pages 1–29, 2013.