Using Logic to Handle Conflicts between System, Component, and Infrastructure Goals in Complex Robotic Architectures

Paul Schermerhorn and Matthias Scheutz
Human-Robot Interaction Laboratory
Cognitive Science Program
Indiana University
Bloomington, IN 47406, USA
{pscherme,mscheutz}@indiana.edu

***Abstract*— Complex robots with many interacting components in their control architectures are subject to component failures from which neither the control architecture nor the implementing infrastructure can recover. Moreover, the operating conditions for these components might be at odds with goals the robot might have adopted (e.g., through external commands or in the course of the execution of the current task).**

We argue that the best (if not the only) way to resolve any difficulties that arise from the different requirements at the agent, component and infrastructure levels is to use a common formal logical goal representation for all three layers. We discuss how these representations can be integrated into a complex robotic architecture and demonstrate in an experimental evaluation on a robot how the architecture can recover from a failure situation that it would not have been able to handle without explicit multi-level unified goal representations and their associated monitoring and reasoning processes.

I. INTRODUCTION

Robotic architectures have become increasingly complex in a variety of application domains, including robots for search and rescue missions, for elder care and other assistive domains, or even various kinds of social robots. These architectures are typically composed of many heterogeneous functional components, ranging from components for sensing (e.g., vision, speech processing, etc.) and actuating (e.g., navigation, manipulation, etc.), to components for natural language interactions (e.g., parsers, semantic engines, text planners, etc.), to planning (e.g., navigation and task planners) and reasoning components (problem solvers, etc.).

Managing all these components to produce coordinated robot behavior is already a challenge in and of itself, which has been addressed in several ways, both by the robotic architecture community (e.g., explicitly managing and tracking the robot's goals [17]) and the infrastructure community (e.g., guaranteeing a robust execution environment [11], [13]). While these two approaches have been concerned with handling *goals of the robot* (e.g., as assigned by a human operator) and *goals of the infrastructure* (e.g., providing load balancing across multiple hosts), they are typically not combined. More importantly, there is a third set of usually only implicitly given goals, the *goals of each component* to "maintain its conditions of normal operation" (e.g., that the sound levels be low enough for a speech recognizer to be able to recognize words), which interacts with the other two sets of goals in important ways. For example, component goals that are not met can lead to failures of robot goals (e.g., speech not being understandable and the robot failing to execute commands) and infrastructure goals (e.g., a parser using up all the CPU time for exploring meaningless parse trees causing the infrastructure's load balancing goal to fail).

We believe that it is critical for the long-term reliable operation of complex robots to better understand the possible interactions among the three sets of goals and to provide mechanisms that will be able to represent and resolve possible conflicts among those goals as they occur, to the extent possible. Our approach is to use a formal language to represent goal conditions for the three types of goals and use logical inference to detect goal inconsistencies. The idea is to use logical representations as the "common currency" throughout the system for the various different goal types. This will not only allow the robot to reason about various types of requirements at different system levels in a unified way. In the future, this will also allow for better communication between humans and robots, and easier ways for tracking and resolving faults.

The paper proceeds as follows: we start by motivating the need for explicit goal representation at the different levels in Section II. Then we discuss the utility of common logical representations in Section III and describe the architecture which we used to integrate the new goal representation in Section IV. Next, in Section V, we provide the details on the task that was used for the evaluation of the utility of system-wide logical representations and a detailed trace through critical phases of the experimental run on the robot that demonstrates how the various goal representations interact. Section VI then discusses the implications of the evaluation, and Section VII contains conclusions and directions for future work.

II. MOTIVATION

Consider a scenario in which a robot is a member of a security team tasked with patrolling a remote area of a facility. The robot is equipped with an array of sensors to detect intruders, is able to move around its assigned area to investigate potential break-ins, and can communicate with a human operator in the central monitoring station. Among the robot's behaviors is an automatic motion-based trigger to emit a loud alarm siren whenever it detects what seems

with very high probability to be an intruder. During one of its patrols, the robot senses motion where there is no authorized presence and enables its siren. It then contacts the remote operator to inform her of the alarm and request instructions. The operator instructs the robot to perform an auditory sensory sweep of the area to detect the direction in which the intruder has retreated. However, due to the continuing siren, no other sounds are audible to the robot. Moreover, the robot's control system is distributed across multiple onboard and offboard computers, including the speech production and recognition units, which are run on a computer in the control room that has a microphone and a speaker. The operator, therefore, hears only the "clean" verbal output from the robot, without the background noise from the siren.

The problem in the above scenario is, of course, that the human operator does not know (or recall) that the siren is wailing, and the robot is simple-mindedly following its directives without regard to how some actions will affect the functionality of other components. While it might be possible to track down the problem (e.g., with a series of targeted queries on the part of the operator), it is also possible that the problem will go undetected (i.e., that the absence of audible evidence will lead the operator to conclude that it was a false alarm). Moreover, without mechanisms in place to specify and communicate information about operational conditions and requirements of different parts of the robot's control system, it may be difficult or impossible to track down a problem like the above.

What is needed is a uniform representation of *conditions of normal operations* (CONO) and their constraints that can be used throughout the system to facilitate the coordination among requirements, actively avoid potential conflicts (when they can be predicted), and facilitate resolution of conflicts when they are unavoidable (including recovery from failures if possible).

III. LOGICAL REPRESENTATIONS AS "COMMON CURRENCY"

In the introduction, we identified three (non-exclusive) sources of (different types of) goals in complex robotic architectures:[1]

- *agent goals*, including "built-in" goals in the architecture (e.g., safety of human operators and self-preservation) as well as dynamic goals arising from changes in the environment (e.g., commands from a human operator)
- *infrastructure goals* of the middleware or software environment, in which the architecture is implemented, including goals for maintaining system reliability or balancing the load across multiple computational resources
- *component goals* of the *functional components* in the architecture (i.e., the software and/or hardware unit that implements the particular component algorithm such as a navigation planner or speech synthesizer)

[1]Some robotic architectures (e.g., behavior-based, [4]) do not represent goals explicitly in the architecture. The discussion of goals and, indeed, the mechanisms described in this paper do not apply to such architectures.

All three types of goals can take the form of "maintenance" and " achievement" goals, where *maintenance goals* typically have CONOs attached that have to be maintained true by the robot (possibly augmented by a set of *pre-conditions*), while *achievement goals* are typically defined in terms of their *post-conditions* that have to be made true by the robot (possibly augmented by a set of initial pre-conditions and operating conditions) while the robot is attempting to achieve the post-conditions. The challenge here is how to best represent these different types of goals so that they can be compared in a unified way and that conflicts among them can be detected and possibly resolved—in most robotic architectures, infrastructure goals are not compared to agent goals (but see [7] for an exception), and component goals are almost never explicitly represented (but see [10] for an example of a robot's being aware of its auditory environment to improve natural language interactions).

A logical representation that is rich enough to express all three types of goals but limited enough to allow for efficient inference (about goal conflicts) is a natural candidate, given the diversity of types and sources of goals (possible candidates are temporal logics and subsets of first-order logic). This requires explicit representation of goal requirements for each functional component in the architecture together with the definition and implementation of monitoring processes that will be able to update those representations (e.g., a component's preconditions need to be checked whenever information is requested from it, and the requesting process needs to be informed if the preconditions are not met).

In the example above, there may be a central *control* or *goal management* component that is responsible for coordinating the actions of the robot, in which case there may already be a representation of the postcondition of activating the siren: **enabled**(*speakers*) (since the goal manager is responsible for triggering that behavior in response to the motion). The auditory tracking component, then, needs to encode its own operational constraints, including ¬**enabled**(*speakers*). When the auditory tracking function is invoked by the human operator, it can then either (a) check with other components in the system to see whether its operational constraints are violated, or (b) perform a check of its own to determine whether it would be possible to detect retreating footsteps given the current background noise. The advantage of the former is that it is a simple check, whereas the latter will detect problems not related to the robot's own actions (e.g., sirens from a passing ambulance).

With the logical representation mechanisms in place, the problem in the hypothetical scenario will be much simpler to diagnose and remedy. When the auditory tracking component is enabled, it can immediately detect that the speakers are enabled and inform the operator of the conflict. It is then up to the operator to decide whether to instruct the robot to disable the siren or not. Extending the logical checks into the remote components reduces the possibility of a false negative (i.e., the conclusion that there is nothing to track), and allows the operator to decide which is a higher priority: tracking the fugitive or continuing to sound the alarm.

IV. INTEGRATING EXPLICIT MULTI-LEVEL GOAL REPRESENTATION INTO A COMPLEX ROBOTIC ARCHITECTURE

The architecture of choice for the implementation of the goal representation and inference mechanisms above was the *Distributed Integrated Affective Reflective Cognitive* (DIARC) architecture because it already provides explicit goal representations for agent and infrastructure goals.

a) Agent goals: DIARC facilitates HRI by integrating high-level natural language understanding [3], [2], [5] with lower level perceptual and action processing (e.g., vision [1], [15], [12], navigation, behavior coordination [16]). DIARC's integrated NLP (natural language processing) component can take natural language directives and convert them into logical representations of the commands. Specifically, a natural language parser treats English words as lexical items with syntactic annotations from a *combinatorial categorial grammar* and semantic annotations from *temporal and dynamic logics* extended by λ-expressions [5]. These are then mapped onto λ-free temporal and dynamic logic expressions, which represent the goal and action specified in the natural language directive, respectively. This approach of extracting goal and action sequences *incrementally at the same time* has several advantages: (1) it allows for in-line backchannel responses (e.g., to indicate early on failure to understand [3]), (2) it can enable the robot to begin consistency checking of new goals against existing goals, (3) it can give the robot an (at least partial) action sequence to achieve the goals, and may allow initiation of action before the utterance is complete, and (4) it allows the robot to immediately detect and react to (syntactic, semantic, or pragmatic) ambiguities in the directive.

The logical representations extracted from a human instruction are then passed on to the goal management component. DIARC's goal management component supports concurrent progress toward the achievement of multiple goals. The goal manager maintains a database of procedural knowledge in the form of action scripts, indexed by post-conditions achieved. Each top-level goal, such as a command from a human operator, is serviced by an action script interpreter that steps through the script that achieves that goal. When conflicts arise between goals (e.g., when two scripts call for the robot to move in different directions), the script interpreter associated with this highest-priority goal is given precedence and allowed to control the resource. Since natural language goals have already been converted into logical form, the goal manager can simply query its knowledge base to find a suitable sequence of actions with postconditions including the goal state. If an appropriate action is found, the goal manager dispatches the task description to the task management component to be carried out. In cases where no solution can be readily found, the unmodified logical form of the goal is passed to a planner component, which generates a plan sequence and returns it to the goal manager to be stored as procedural knowledge suitable for achieving the new goal, as well as future requests with the same logical form. The system supports two kinds of agent goals: *hard goals* that have to be achieved (e.g., follow the team leader's commands), and *soft goals* that should be achieved if possible (i.e., if the robot detects an opportunity to satisfy the soft goal without violating a hard goal and if its utility $u > 0$).

b) Infrastructure goals: DIARC is implemented in the *Agent Development Environment* (ADE), a distributed multi-agent robotic development infrastructure [8], [14] with mechanisms for distributing architectural components across multiple hosts that has been previously used successfully for developing infrastructure reflection mechanisms for fault detection and error recovery and evaluating them in human-robot interaction experiments [7], [9], [6]. Functional components of DIARC then are implemented as ADE servers (i.e., individual computational processes or "agents") that can interact with each other by virtue of establishing connections among them as required by the architecture. To establish connections, the ADE registry provides a yellow-pages service that allows ADE servers to locate each other without needing to know their physical hosts in the distributed system. Once contact has been made via the registry, ADE servers interact directly, while maintaining periodic contact with the registry to facilitate error detection and recovery.

Similar to the goal manager component, the ADE infrastructure uses logical forms to explicitly keep track of actual and desired system states and logical inference to determine operations on the infrastructure to achieve desired configurations (e.g., an ADE server that went down can be restarted, or a server can be moved to another host to achieve better load balancing). These mechanisms are implemented in the ADE registry, which has access to all ADE servers.

c) Component goals: Since DIARC already provides agent and infrastructure goals in logical form, only component-level goal representations had to be added together with the processes necessary to update them. This can, however, be a very complex task depending on the components and the nature of the processes they implement. To make this task tractable, we only added representations and update processes to those components that we are going to use in the proof-of-concept evaluation. Since our chosen evaluation scenario requires visual processing, we have augmented the DIARC vision component to include logical representations of its required operating conditions, specifically, that there be sufficient light in the environment for the camera to be able to deliver reasonable images. A monitoring process was thus added to the server that periodically checks images coming from the camera for their overall brightness values, and when the overall brightness falls below a "darkness threshold" the vision server's operational condition **light-level**(*OK*) will be changed to ¬**light-level**(*OK*), which, in turn, will trigger the goal request for **on**(*light*)—whether or not this request results in an action to turn lights on will then depend on the other goals the agent might have, as we illustrate next in our evaluation.

V. EVALUATION

For the evaluation, we chose a scenario that has become increasingly important in recent robotic efforts: an *urban*

search and rescue (USAR) task. USAR has several attractive aspects for our evaluation: it usually requires a remote human operator to interact with the robot (given that robots are used instead of humans for the exploration of potentially dangerous environments to be able to reduce the risk to human rescue personnel); as such, it requires the robot to be very explicit in its interactions with the human (i.e., about its perceptions, its current position in the environment, etc.) who often only has limited situational awareness and might not be able to fully comprehend the robot's situation; and lastly, it requires the robot to be able to autonomously respond to task contingencies as the operator will not be able to step in if things go wrong.

The hardware platform was a Pioneer P3-AT equipped with several sensors, including a SICK laser range-finder for navigation and obstacle detection, two cameras for visual processing (an infrared camera, and a normal-light camera), and a wireless microphone for speech input. In addition to the mobile robot base, the robot was equipped with multiple other effectors, including a light effector and speakers for speech production. The setup includes the two camera types so that the robot can be used for conducting searches under any lighting conditions; the light effector is included so that the robot can use the natural-light camera even in low light conditions (e.g., to detect particular colors).

We chose a scenario in a "dark indoor environment" where the robot was supposed to search for wounded human victims that might be hiding in rooms. The robot starts out at the beginning of a hallway that has several open doors to the left leading to rooms that might contain victims. The robot has no map of the environment, hence cannot plan ahead. It also has no information about possible locations of victims other than the *prior knowledge* that wounded people might be located in rooms. The goal to locate wounded people is given as a "soft goal" which the robot should pursue if it can afford it without violating any hard goals (e.g., for this particular soft goal, the robot can schedule perception actions for doorways to be able to detect opportunities for satisfying the soft goal). The only hard goal the robot has in the beginning is to obey the commands of the human operator.

The system configuration for the evaluation experiments required a setup with multiple `ADE` servers implementing critical components in the `DIARC` architecture, including a "Pioneer robot control server" (for controlling the robotic platform), a "laser range-finder sensing server" (for detecting doors and avoiding obstacles), two "vision servers" (one with an infrared camera and one with a regular light camera), a "light effector server" to be able to actuate a flash light onboard the robot, a "goal manager server" (to manage the different kinds of goal representations), as well as various servers for natural language processing, jointly referred to as the "NLP servers".

In the following, we will describe various phases of one experimental run. For each phase, we first describe the events happening in that phase, followed by the agent-level goals *AG*, infrastructure-level goals *IG*, and component-level goals *CG* for that phase (note that for clarity's sake, expressions marked with a $\star$ are simplified representations of the actual internal representations). We also show a snippet of the dialogue exchanges that occurred between the human *H* and the robot *R*, followed by a brief description of the processes occurring in the architecture and any additional clarifying remarks about the phase.

BEGIN: The robot is awaiting orders in a dark hallway.

AG: **obey_commands**(*robot*)
IG: **maintain_health**(*servers*)

PHASE 1: The operator contacts the robot with orders, instantiating two hard goals and one soft goal (as indicated by the "try" keyword).

H: "Go to the end of the hallway. Keep the lights off at all times. And try to report the locations of wounded people."
R: "OK."

AG: **obey_commands**(*robot*),
  at(*end-of-hallway,robot*),
  ¬**on**(*light*),
  looked_for(*victim,room*)*
IG: **maintain_health**(*servers*)

Notes: The robot begins to traverse the hallway, attempting to detect doorways (i.e., opportunities to pursue the soft goal of locating wounded people). The robot detects a doorway and enters the room.

PHASE 2: The robot searches the room and detects a wounded person.

R: "There is a wounded person in the first room on the left."

AG: **obey_commands**(*robot*),
  at(*end-of-hallway,robot*),
  ¬**on**(*light*),
  looked_for(*victim,room*)*
IG: **maintain_health**(*servers*)

Notes: The robot leaves the room.

PHASE 3: The robot continues down the hallway. For demonstration purposes, the IR camera is sabotaged by the experimenter.

R: "I have lost contact with the infrared camera."

AG: **obey_commands**(*robot*),
  at(*end-of-hallway,robot*),
  ¬**on**(*light*),
  looked_for(*victim,room*)*
IG: **maintain_health**(*servers*),
  recovered(*vision_server*)

Notes: The infrastructure notices that a fault occurred and attempts to recover the IR camera server, but is unable to do so.

PHASE 4: The robot continues down the hallway. The

infrastructure attempts to find a substitute for the sabotaged vision server.

R: "Switching to the regular camera."

AG: **obey_commands**(*robot*),
at(*end-of-hallway,robot*),
¬**on**(*light*),
looked_for(*victim,room*)*
IG: **maintain_health**(*servers*),
find_substitute(*vision_server*)

Notes: The normal-light vision server is successfully started and takes the place of the IR vision server in the architecture. This substitution occurred at the infrastructure level without DIARC noticing. Because both vision servers implement the same API, this notification is unnecessary; the registry could swap in the normal-light server without giving a warning and the goal manager would not need to know about the change. The warning is given to facilitate the operator's understanding of the current state.

PHASE 5: The robot detects a second doorway and enters the room to search. When the vision processing command fails, the request to enable the robot's onboard light is submitted to the goal manager, which detects the conflict.

R: "I need to activate a light, but I have a goal to keep lights off. What shall I do?"

CG: **light-level**(*OK*)
AG: **obey_commands**(*robot*),
at(*end-of-hallway,robot*),
¬**on**(*lights*),
looked_for(*victim,room*)*
IG: **maintain_health**(*servers*)

Notes: The required operating conditions of the normal-light camera are different, a fact that is expressed by a logical formula (**lighted**(*room*)).

PHASE 6: The operator gives permission to cancel the lights-off goal.

H: "Um, cancel keep lights off."
R: "OK, goal cancelled."

CG: **light-level**(*OK*)
AG: **obey_commands**(*robot*),
at(*end-of-hallway,robot*),
looked_for(*victim,room*)*
IG: **maintain_health**(*servers*)

Notes: The robot enables the light and conducts the search, finding no victim. It leaves the room and continues to the rendezvous point, watching for doorways on the way.

The execution trace demonstrates how goals and operational constraints at different levels interact. The goal manager uses a simple first-order resolution-type inference procedure to detect conflicts; this is only possible because all components maintain their goals and constraints in a common (quantifier-free) first-order logical representation. In many cases, the goals persist throughout the run (e.g., the agent-level **obey_commands**(*robot*) or the infrastructure-level **maintain_health**(*servers*)). These represent long-term goals built into the system. A video of the full evaluation can be viewed at `http://hri.cogs.indiana.edu/videos/icra10.wmv`.

VI. DISCUSSION

As we demonstrated, a logical language used to represent dynamic functional constraints in complex robotic architectures can significantly simplify the identification and handling of conflicts among different types of goals in a complex robotic architecture. The common representation not only streamlines the communication of constraint violations (e.g., to the central goal management component), but it also allows for the use of common inference mechanism across the board (e.g., to check for goal consistency or to determine possible ways on how to resolve goal conflicts). However, which way to resolve a conflict if there are multiple options, or whether to resolve it at all when it is detected, is not clear. In the example scenario above, for example, the vision system continuously monitored the light conditions, and could have actively informed the goal manager. Instead, a passive approach was taken in which the vision system gave no indication of the problem until the task manager made a sensing request. This suggests that there are several options on how the system could react to goal inconsistencies, including:

- *Actively monitor and attempt to resolve problems immediately when detected.* In some cases this will have the benefit of saving time at the point when the component is used, for example, the operator query could be issued while the robot is traversing the hallway and the conflict resolved before the task manager makes the request for visual information. Moreover, in cases where conflict cannot be resolved, the system can rule out future actions that depend on the component in question (e.g., bypassing any rooms encountered instead of entering them if the human operator does not give permission to activate the light).
- *Actively monitor, but postpone resolution attempts until it becomes clear that the fault is critical.* This could save a busy operator from having to respond to the request in cases where the fault is unimportant, such as a case in which an added time constraint precludes any further searching and the constraint violation would never cause a problem for the robot's goals. Moreover, the information would be immediately available in case the goal manager (or some other system component) actively seeks information about the present operating conditions.
- *Postpone constraint-checking until the request is made.* In some cases, detecting a problem may be prohibitively expensive, or the probability that the component will be accessed very small, making active monitoring unattractive. This option would not preclude other servers

from making preemptive checks on the components constraints, although it would take longer to respond to such requests.

In the example scenario described above, the policy was to actively monitor but postpone notifying the goal manager (while the operator was informed). Monitoring the light level is relatively inexpensive, but the uncertainty regarding the need for visible-light sensing (i.e., the possibility of dynamic temporal constraints precluding future room searches) made the policy of immediate constraint repair unsuitable. The assumption here is that the operator in a USAR scenario will typically be under significant cognitive load, potentially interacting with multiple robots at different stages of their searches, so a higher emphasis is placed on avoiding unnecessary interactions. In some cases, such as when resolving the problem will be time-consuming, the added request may be deemed appropriate. Consequently, the decision on which strategy to use in response to actual and possible goal inconsistencies will dependent on multiple factors, and isolating these factors seems a worthwhile direction for future work.

VII. CONCLUSION

In this paper, we proposed the use of a common logical representation for agent goals, infrastructure goals, and operating constraints on functional components in the architecture ("component goals"). Encoding constraints of these different types of goals in a logical form allows for the detection and possible resolution of conflicts that could otherwise severely impact if not end a complex robot's operation. We integrated this common representation into the DIARC architecture, implemented in the ADE infrastructure, and demonstrated its use in a urban search and rescue scenario. In the evaluation, the system was able to resolve a potentially fatal conflict between a goal from the human operator and the operational requirements of a system component.

System-wide integration of a common logical representation of these constraints not only facilitates a systematic approach to handing conflicts, but it also opens the door to more informed conflict resolution approaches with the potential to further streamline and improve the coordination among heterogeneous components. In future work, we intend to explore questions such as how to deal with *potential* conflicts. It is likely that some problems could be detected early on by examining future execution traces of currently active scripts (in service of currently active goals). This could allow for better action selection, such as when multiple action sequences lead to outcomes that satisfy a given goal, but one or more has the possibility of running afoul of the requirements of a needed component.

Similarly, extending the capabilities of a planner to anticipate and handle component requirements could significantly improve a robot's performance. E.g., a planner could schedule a sensing action to check whether the current operating conditions are outside the acceptable range of critical components and trigger replanning based on the new information; or better yet, generate a conditional plan that includes the sensing action to check the current conditions and includes a branch to handle that contingency, potentially obviating the need for replanning. Using a common logical representation for operational constraints would simplify the application of these techniques, which should lead to quantifiable performance improvements.

REFERENCES

[1] Virgil Andronache and Matthias Scheutz. Design and experimental validation of a minimal adaptive real-time visual motion tracking system for autonomous robots. In *Proceedings of the 2005 International Conference on Artificial Intelligence*, pages 663–669, Las Vegas, NV, June 2005.

[2] Timothy Brick, Paul Schermerhorn, and Matthias Scheutz. Speech and action: Integration of action and language for mobile robots. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1423–1428, San Diego, CA, October/November 2007.

[3] Timothy Brick and Matthias Scheutz. Incremental natural language processing for HRI. In *Proceedings of the Second ACM IEEE International Conference on Human-Robot Interaction*, pages 263–270, Washington D.C., March 2007.

[4] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[5] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the 2009 International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[6] James Kramer and Matthias Scheutz. ADE: A framework for robust complex robotic architectures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4576–4581, Bejing, China, October 2006.

[7] James Kramer and Matthias Scheutz. Reflection and reasoning mechanisms for failure detection and recovery in a distributed robotic architecture for complex robots. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 3699–3704, Rome, Italy, April 2007.

[8] James Kramer and Matthias Scheutz. Robotic development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2):101–132, 2007.

[9] James Kramer, Matthias Scheutz, and Paul Schermerhorn. 'Talk to me!': Enabling communication between robotic architectures and their implementing infrastructures. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3044–3049, San Diego, CA, October/November 2007.

[10] Eric Martinson and Derek Brock. Improving human-robot interaction through adaptation to the auditory scene. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 113–120, New York, NY, USA, 2007. ACM.

[11] N. Melchior and W. Smart. A framework for robust mobile robot systems. In *Proceedings of SPIE: Mobile Robots XVII*, volume 5609, 2004.

[12] Christopher Middendorff and Matthias Scheutz. Real-time evolving swarms for rapid pattern detection and tracking. In *Proceedings of Artificial Life X*, pages 419–425, June 2006.

[13] K. Reichard. Integrating self-health awareness in autonomous systems. *Robotics and Autonomous Systems*, 49(1-2):105–112, November 2004.

[14] Matthias Scheutz. ADE - steps towards a distributed development and runtime environment for complex robotic agent architectures. *Applied Artificial Intelligence*, 20(4-5):275–304, 2006.

[15] Matthias Scheutz. Real-time hierarchical swarms for rapid adaptive multi-level pattern detection and tracking. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium*, pages 234–241, April 2007.

[16] Matthias Scheutz and Virgil Andronache. Architectural mechanisms for dynamic changes of behavior selection strategies in behavior-based systems. *IEEE Transactions of System, Man, and Cybernetics Part B*, 34(6):2377–2395, 2004.

[17] J. Trafton, N. Cassimatis, M. Bugajska, D. Brock, F. Mintz, and A. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man and Cybernetics*, 25(4):460–470, 2005.