Behaviors and Layers in Spoken Dialogue Systems

Pontus Wärnestål
Dept. of Computer Science
Linköping University
ponjo@ida.liu.se

Abstract

The classical view of artificial intelligence (AI) is tied down by a number of problems. These problems indicate that it might be fruitful to approach AI applications, including spoken dialogue systems, in an alternative way. This paper overviews current attempts to apply the idea of layered behavior-based architectures in dialogue system development. It is concluded that there are both theoretical insights and practical advantages to be made with a layered behavior-based approach to dialogue system design.

1 Introduction

The currently dominating view of dialogue systems is inherited from classical artificial intelligence (AI) approaches. This means that most dialogue systems today adheres to the classical AI model depicted in Figure 1. What this model suggests is that intelligence is a matter of transforming perceptual data to a meaningful mental representation of the world inside the agent. This representation is manipulated with one or more cognitive processes so that a set of actions (or plans) is produced. These actions are then translated into some effector(s); such as setting motor speed for robots, or generating a sentence in the case of dialogue systems. Throughout this paper I use the term “classical” to denote this symbol manipulation approach to systems. Whereas the classical AI approach has allowed researchers from both the AI, NLP and Psychology fields to focus on one aspect of “intelligence”, the model as a whole has some critical faults, which are summarized below. Entire books and series of books have been written on this issue, so this paper can only give a glimpse of the problems involved. For more detailed arguments on this issue, see e.g. [16]. It is important to note that this controversy is not only a theoretical one. It is also a matter of practical design work, which becomes increasingly important as we advance the field of dialogue system development. Some of the issues that used

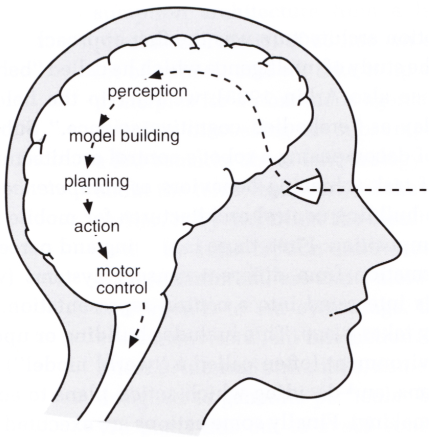

Figure 1: The classical AI model. From Pfeiffer and Scheier [16].

to be more theoretical in nature (e.g. arguing for massive parallelism since this is a "biological" aspect of intelligence found in the human brain), are now becoming more important in engineering dialogue systems as we deal with multimodal continuous feedback and realistic response times etc. Some especially important difficulties with the classical AI program for dialogue systems are:

- Perception, action, and cognition separation fallacy: the sense-think-act cycle is not a correct model in all cases. It is for example known that perception is guided by internal configuration and expectations. The famous McGurk effect is one example of this. Other examples of this include the neurobiological observation that neurons receive as much as 80 percent of their input from *other neurons* and only 20 percent from external stimuli. Furthermore; many actions require constant perceptual feedback for control, without requiring cognitive effort.

- Lack of robustness: Traditional systems are brittle to noise and unexpected input. Natural intelligence is characterized by "graceful degradation" for noisy input and can perform adequately even if certain components are impaired. This is related to the lack of generalization capabilities of classical systems. In real-world situations no two situations are ever the same, so some sort of generalization is required for robust behavior.

- Long processing and delays: Any system functioning in dynamic surroundings must be able to react quickly. Because the sense-plan-act model processes all information centrally we end up with slow monoliths—even for very small domains. In natural language dialogue, this issue is extremely important, since unexpected delays on the order of milliseconds are noticeable and can even change the meaning of the communicative act. “The frame problem” is often brought up in this context, and puts a finger on a disturbing issue. How can a central model or representation of a dynamic world be kept updated? Any proposition my change at any time, which makes it very hard for a central representation and accompanying logical inference engine to function correctly in real-time.

- Another philosophical nightmare is the symbol-grounding problem, which refers to how symbols relate to the real world. Symbols put in a system by a human programmer can never be “true” symbols from the point of view of the system since the mapping between symbol and real-world object is grounded by human cognition/intelligence. Some theorists claim that symbols need to be grounded in terms of the system’s implementation and perceptual/cognitive system (see [16]).

These problems indicate that it might be fruitful to approach dialogue system design in an alternative way. Within the broader AI community the *behavior-based* and *layered approach*has served as an alternative to the classical approach for the past decade. Especially within Robotics, this approach has been very successful and remedies some of the problems of the classical approach outlined above. For a more detailed discussion, see e.g. [16]. However, for higher cognitive functions (e.g. realistic natural language dialogue) than those typically considered within the Robotics community (which a lot of times revolves around avoiding obstacles or finding energy loading stations), the approach has not yet been applied to any wide extent. The reasons for this are unclear, but one issue is scalability (the step from avoiding obstacles to engaging in conversations is quite big). Furthermore, classical NLP has a tradition of dealing with ontological relationships, knowledge sources, logical formalisms, and high-level planning. More low-level types of phenomena in conversation has not received as much interest as e.g. NL interfaces to databases. This paper is an attempt to overview current attempts to apply these ideas to dialogue system development, and discuss the possibilities for a layered approach in dialogue system design. In the reminder of this paper, I will focus on the *layered approach* to behavior-based architectures. There are other ways than layers to implement behavior-based design, and I will touch upon some of these as they arise. However, covering them all is far beyond the scope of this paper.

The rest of the paper is organized as follows: First, the layered approach is described, starting with the general software engineering notion of layers, before narrowing in on layers in the behavior-based AI field in section 2. In section 3, I

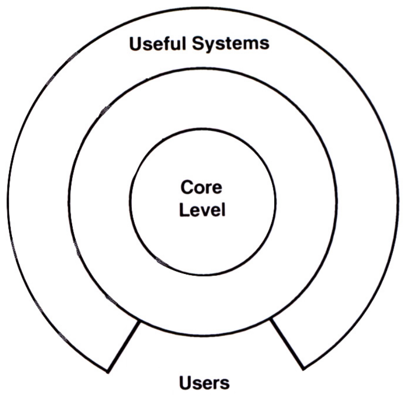

Figure 2: The software engineering view of a layered architecture. From Garlan and Shaw [10].

give an overview of the use of the layered approach in multimodal dialogue systems. Finally, I conclude with a discussion of the material and describe some practical and theoretical implications for modeling dialogue in layered, behavior-based systems in section 4.

2 Layered Approaches

In order to come to terms with the "layered approach" we will first take a software engineering view of what constitutes a layered architecture (section 2.1). Then we focus on layered initiatives within the AI community. A suitable starting point here is the subsumption architecture, which represents the first change from the classical approach towards behavior-based AI (section 2.2). We then deal with some more recent architectures, aiming at resolving higher-level cognitive functions such as natural language and memory in section 2.3.

2.1 The Software Engineering View

A general characteristic of layered systems is that they are organized hierarchically. Each layer provides service or information to the layer above it and serves as a client to the layer below. The connection between layers are defined by protocols, determining how layers will interact [10]. Figure 2 shows the layered architecture style. Outside the world of NLP and Language Engineering (LE), the most widely used application of layers are (a) layered communication protocols such as the

TCP/IP stack, and (b) operating systems, where the kernel runs in one layer, and user space and processes reside in layers above.

According to Garlan and Shaw [10, page 11], layered systems support the following properties:

1. design based on different levels of abstraction (i.e. supporting a divide-and-conquer strategy of partitioning complex problems into incremental and manageable steps)

2. enhancement (since each layer only interacts with *at most* the surrounding two layers, changes in one layer affect at most two other layers)

3. reuse (i.e. different implementations of a layer can be used interchangeably, allowing for designs of reusable layer interface standards in a framework)

Layers are of course not a silver bullet. There are some disadvantages one must be aware of, and not all kinds of systems are suitable for the layered approach. *Performance* considerations need to be taken, even if a system can be structured in layers. That is, a system may require close coupling between low-level and high-level layers due to computational performance, such as large search tasks etc. This may thus violate the *enhancement* property above. *Design* considerations are also important, since some systems conceptually are hard to model in this manner. Finding the right level of abstraction for the different layers is a non-trivial design issue. For example, communications layers sometimes need to bridge several layers—an unhealthy sign since this violates both the *enhancement* and the *reuse* advantages mentioned above. It may even violate the *design* property depending on the complexity of the problem the system aims to solve.

2.2 The Subsumption Architecture

The subsumption architecture was proposed by Rodney Brooks in 1986 [3], but was made famous in two papers from 1991 [4, 5], and is the prime example of the behavior-based view that contrasts against the classical view of intelligence. Brooks' papers are written from an engineering perspective, and deals with fundamental robotics such as moving and avoiding obstacles. High-level cognition in the systems is not considered, and the approach is only validated for simplistic behavior. Despite of this, the approach is still of high interest and has been the subject of general discussions on cognitive theory.

The most interesting part of the architecture—and underlying design philosophy—is that it is entirely based on sensor-motor couplings in autonomous *layers*, without a central representation or planning mechanism as classical AI systems would employ. Each layer corresponds to a *behavior* (such as moving forward, avoiding

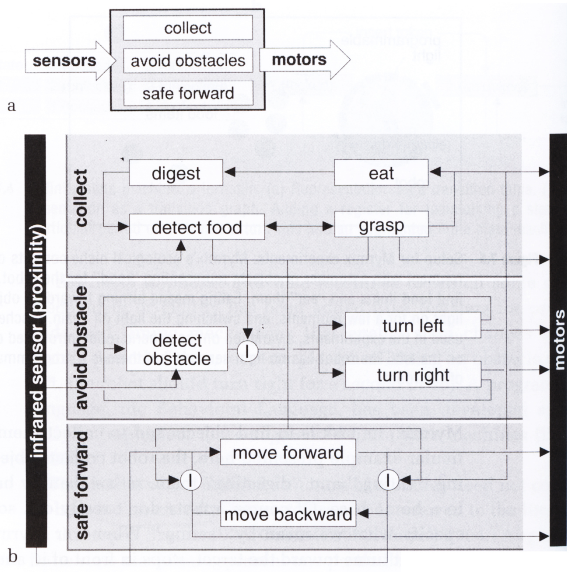

Figure 3: An example subsumption architecture. From Pfeiffer and Scheier [16].

obstacles, detect food, etc.). In other words, the layered style is implemented since behaviors are organized hierarchically. This means that higher-level behaviors can *subsume* lower-level layers by inhibition or modification. Figure 3 shows an instantiation of the subsumption architecture. Another crucial point is that *each* layer reacts to outside stimuli. There is hence no single, central flow of information in the way a classical AI system would normally be built. The architecture thus supports *concurrency*, since information is processed in parallel by all layers. This also means that each layer can run on its own, since it reacts directly on outside stimuli. (However, a robot running only on e.g. a “detect food”-behavior would not be able to avoid obstacles if such a layer were missing!)

The subsumption architecture is able to respond quickly to changes in the environment. Since each layer runs in parallel with the others, the system as a whole does not execute behaviors based on out-dated and irrelevant information. Another important aspect of the subsumption architecture—as well as other behavior-based approaches—is the reliance of *emergent behavior*. That is, the observed behavior of the complete system is not put there by the hand of any programmer. Rather, it is by invoking simple, local rules that the system exhibit complex behavior. One common example of emergence is flocking behavior where agents in a flock follow very simple rules such as staying close to neighbors and facing the same direction as the closest neighbor, but without colliding. The result of these three simple rules is surprisingly coherent and dynamic flocking behavior as seen in swarms of insect, or in fish schools.

From a methodological perspective, the subsumption architecture supports evolutionary reuse. The underlying philosophy for this claim is that once a layer (i.e. behavior) has been built (and debugged/tested), it can be reused without change. The parallel from evolutionary biology is that once "nature" has found a solution (such as the design of the mammalian eye) it is reused as is over and over again. Even though this claim may seem a bit stretched, it still implies an evolutionary aspect of software development. The subsumption architecture can thus be viewed as a natural platform for evolutionary software development.

The subsumption architecture has to my knowledge not been used for higher-level cognitive functions such as natural language interaction and dialogue. Indeed, Thórisson [19] points out that behavior-based architectures such as the subsumption architecture, often are specified at a very low level in the sensor-action control modules. This makes it hard—if not impossible—to build large systems in this way.

2.3 Other Layered Approaches

Many researchers have proposed other behavior-based architectures since Brooks. Among them, Maes' competence modules and spreading activation network [14] is well-known, and interesting to mention, since it is behavior-based, but *not* a layered architecture. This architecture differs from subsumption since there is no hierarchical control. Instead, there are a number of perception, action, and goal nodes that are connected to each other. The nodes specify how much linked nodes must activate before firing. This allows for both goal-initiated (top) as well as perception-initiated (bottom) activation.

There are several other behavior-based and layered approaches around (cf. SOAR [15] and ACT-R [1]), but they will not be dealt with in any detail in this paper. In summary, they all perform well in terms of fast action selection and some of them allow various forms of learning. However, Thórisson [19, page 2] notes that these types of architectures often lack methods to deal with external and internal time-constraints and planning. Furthermore, some of them have a reputation of having huge overheads in terms of learning to program and implement them[1].

In retrospect, the main long-standing contribution of the behavior-based AI community is that there now is an almost universal agreement that some types of intelligence is best modeled in this way. But in order to come to terms with the kind of intelligence required for natural and realistic spoken dialogue systems

[1]Bryson [6] even reports that programming SOAR is "teaching by brain surgery" (page 53).

something else is probably needed. In particular, this seems to be true from an engineering and design point of view. Note Thórisson's argument above; and the fact that Steels' non-hierarchical, massively parallel approach [18] was abandoned for *engineering reasons* since it was not practical for each behavior to model all other behaviors, even though the theory seemed sound from a biological perspective [6].

In light of this, it is understandable that a lot of hybrid, layered systems have flourished since. A traditional way to implement such an architecture is through a three-layer solution, which includes behavioral, executive, and planning layers, e.g. [17, 6, 2]. The behavioral layer is reactive, much like purely behavior-based solutions; the executive layer often consists of precoded plans which are selected more or less reactively; and the planning layer is a traditional deep planner that belongs to the classical AI tradition. The top-level planner often takes into account the middle layer's representations, instead of acting directly on the behavioral primitives in the base layer.

3 Layered Dialogue Systems

The layered and behavior-based approach to dialogue system construction is very limited. This ties into the observation that the notion of layers can mean different things. Even though the general—software engineering—notion of layers have been used in several architectures adhering to the classical paradigm, implementations of a more dynamic behavior-based layer approach are rare.

This section briefly describes two such approaches.

3.1 Reactive, Process Control and Content Layers

The Ymir system [19] is a hybrid architecture that consists of three layers as shown in Figure 4. It is focused on communication and models speech, intonation, body language and facial gesture. The three layers are: reactive, process control, and content. Each layer contain perception and decision modules. Response times are very important in natural language communication, as a pause or delay can change the entire meaning of an utterance depending on context. The reactive layer has very short response times (in the 150–500 ms range) in order to provide realistic gaze fixation or blinking etc. Processing time increases in the higher layers. The top-most layer—the content layer—even have infinite response times. The content layer consists of topic information in knowledge bases. The output of the layers is processed by a fourth main component, the Action Scheduler, which prioritizes and morphs action requests from the three layers.

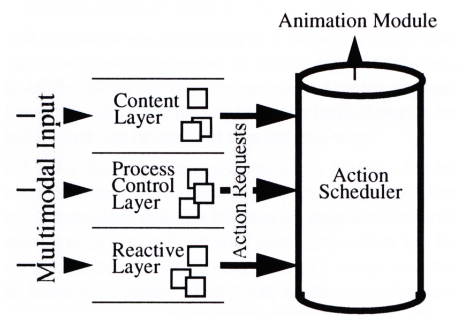

Figure 4: Principal layers and components of the Ymir architecture. From [19].

Implementations of Ymir display a nice set of characteristics that are not usually found in traditional dialogue systems:

- there is a non-rigid interruptible quality of the behaviors[2].
- gestures and body language (including facial gestures) are integrated with the communication content, without artificial communication protocols.
- behaviors run concurrently at the expected times and without unnatural delays.
- miscommunication and speech overlaps are handled in a “natural” way, by using stops and restarts.

In summary, the Ymir model is inspired by the behavior-based philosophy—even if it does not go “all the way” as it relies on high-level planning—and this renders it with some of the expected qualities that classical systems seem to have a hard time to conquer. Recently, the Ymir architecture has been enhanced with a more elaborate turn-taking model [20]. This work addresses the fact that each participant in an ordinary dialogue may take 2–3 communication decisions every second. One interesting aspect of this work is that it assumes no protocol, or implementation of turn-taking *rules*.

3.2 Content and Interaction Layers

Another example of a multi-layer approach to spoken dialogue systems is presented by Lemon et al. [13]. This work is inspired by (a) communication layers as pre-

[2]Note that this is different from barge-in, which is a general way to just skip the reminder of a system utterance.

sented by Clark [7], and (b) robot architectures as outlined in the behavior-based AI field. As they put it [13, page 169]:

> We view the process of natural interaction with a dialogue participant as analogous to the interaction with a dynamic environment: dialogue phenomena arise which need to be negotiated (as a new obstacle must be avoided by a robot).

The proposal is a two-layer architecture that separates interaction-level phenomena from content and context management and conversation planning.

The main benefit is that the interaction layer makes use of low-level signals to provide more natural and robust interactions between dialogue systems and human participants in a way that the content layer is not able to do solely. This is supported by a number of findings in psycholinguistic literature that stresses the importance of asynchronous interaction-level processing in realistic and natural dialogue.

Examples of interaction-level phenomena handled by systems implementing this architecture include: realistic timing and turn-taking, immediate grounding and thus more realistic (continuous) feedback, barge-in management, and NP selection (anaphora management).

Unfortunately, no user evaluation data is available yet, but the approach nevertheless seems promising for end-users since it addresses issues that are handled badly with most other dialogue systems.

4 Discussion

4.1 The Layer Terminology

During the course of this work, I have realized that layers mean very different things for different people. The traditional software engineering view of layers differs a bit from e.g. Brooks' notion of a layer. Software engineers may for example view the user interface as the top layer (final abstraction of a low-level operating system or hardware communication protocols), whereas a "user interface" in robotics is constituted by the sensors (and motors) of each layer. In fact, one of the claims *against* using layers in the view of Garlan and Shaw [10] is the fact that users may want to directly interact at different levels (layers) of abstraction. If the user interface is viewed as the top layer, such a design is only feasible if we bridge several layers. On the other hand, in subsumption layers this claim sort of falls between chairs, since each layer interacts directly with the environment. I draw one conclusion from this: an architecture is not easy to separate from a design methodology. In fact, I would go as far as saying that an architecture *is* a manifestation of design knowledge and practice. Thus, the notion of a layer in one "mind-set" (e.g. communication protocols or operating system kernels) is based on the context and problem inherent of exactly that mind-set (and not in the context of artificial intelligence). Therefore, it may be hazardous to focus on the term *layer* as opposed to the underlying design philosophy that can roughly be summarized in behavior-based design.

4.2 Practical Issues: Language Engineering

Design-wise, the layered approach (with Brooks' evolutionary twist) implies some issues that fit nicely with the *construction* of knowledge-intensive systems such as dialogue systems. By starting out by iteratively designing complete behavior modules that can then incrementally be added to a complete system seems like a sound way to build dialogue systems efficiently. Such evolutionary methodology has been found to be suitable for dialogue system development in general [11, 9].

This ties in with the idea of *software reuse*, that has long been an issue for software engineering [8]. Within Language Technology research groups, this issue has been neglected due to several reasons. For one, the groups are usually rather small with interests spanning large areas. Secondly, personnel are not usually software engineers. This means that the field is theory-heavy, with an unbalanced practice (the practical aspect is even sometimes dismissed as "implementation details"). However, by acknowledging LE as a complementing field of Language Technology, to the more theory-dense NLP field, we may be able to incorporate things that at first glance seems like "implementation details", but that might turn out to be highly interesting stuff—such as the qualities of Ymir shows. Furthermore, we could expect to get both *reusable* and *robust* modules and systems in this way, which is an important point from both an engineering as well as a usability perspective.

With today's computers it is also possible to design for and implement concurrency, as most programming languages have easy-to-use thread APIs.

In summary: Ranging from the overall methodology (evolutionary, agile development) to advantages of software reuse, and all the way to low-level details such as concurrent programming ease-of-use, the practicalities seem to satisfy the LE part of language technology.

4.3 Theoretical Issues: Towards a Theory of Cognition

The theoretical part is—not surprisingly—trickier to resolve. Pfeiffer and Scheier [16] and the proponents of embodied cognitive science are convinced that the behavior-based (and related areas) are necessary in order to advance the task of explaining cognition and intelligence.

Whereas the hybrid approach sketched in section 2.3, and exemplified by the architectures mentioned in section 3, seems to be a happy marriage between the best of two worlds, one of the central themes from the behavior-based camp—namely *emergence*—is lost in a hybrid architecture simply because it lies in the very nature of emergence *not* to provide any handle or interface for the top layer to operate on. It is an open question what this means for a theory of cognition and AI. On the one hand a lot of work is put into traditional planners, but on the other it is hard to neglect the power and potential importance of emergent behavior in any cognitive theory. This is thus a crucial area of AI research with, ultimately, implications for natural language studies—and if NLP researchers are not carrying out the work; who will?

Communication management and interaction-level phenomena seem to benefit from the hybrid approaches examined above. As we strive to build conversational interfaces and address usability issues such as ease-of-use and natural qualities, there is a need to take dynamic interaction qualities such as timing and continuous feedback into account. These qualities seem to be handled nicely.

Until now, the mainstream in dialogue system construction has relied on planning and more AI-flavored approaches, which have been successful to a certain extent, and we should obviously try to use and reuse as much of these theories as possible.

Can we then expect more realistic behavior of spoken dialogue systems? I believe that a layered approach, that could separate content from interaction and let different layers handle each of them, provides a positive answer to this question. Indeed, the kind of problems that low-level behavior-based layering seems to be particularly suitable for, are in fact related to the problems classical AI thus far has had trouble coping with (see section 1).

I also believe we can expect to gain *theoretical insight* in dialogue interaction and communication by focusing on interaction and communicative layers in combination with the traditional approach. But by addressing interaction-level phenomena with behavior-based layering we may also achieve higher *acceptance for practical dialogue system use*, since it would seem like issues such as better feedback management would increase usability and naturalness of dialogue system interaction.

Leaving the hybrid approach for a moment, one of the interesting issues of approaching dialogue system design more radically, is that of scalability: The purely behavior-based, layered approach is known to work for low-level specifications of real-world robots. But how well does it scale? How big is the step to language and dialogue? It is an open question, and there are many skeptics, e.g. [12].

In the meantime, NLP and LE practitioners can choose to approach the issues in (at least) four different ways: (1) they can choose to ignore the problems with the classical approach and continue to advance the mainstream classical approach, or (2) they can be pragmatic about it and use a hybrid approach (or whatever works) for any given application problem and not care too much about a unified, coherent cognitive theory; (3) they can be pessimistic about it and abandon the whole of AI research, and focus on building natural language systems that *support* human–human communication, or (4) they can be optimistic and pioneer things that have been avoided thus far in classical AI research.

Whatever view one decides to take, it is clear that natural language researchers at least ought to keep informed about advances made in the behavior-based camp.

References

[1] J. R. Anderson. *Rules of the Mind*. Lawrence Erlbaum Associates, 1993.

[2] R. P. Bonasso, R. J. Firby, E. Gat, David Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 1997.

[3] Rodney A Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, pages 14–23, 1986.

[4] Rodney A. Brooks. Intelligence Without Reason. In *Proceedings of International Joint Conference of Artificial Intelligence'91*, pages 569–595, 1991.

[5] Rodney A. Brooks. Intelligence Without Representation. *Artificial Intelligence Journal*, 47:139–159, 1991.

[6] Joanna Bryson. *Intelligence By Design*. PhD thesis, Massachusetts Institute of Technology, 2001.

[7] Herbert H. Clark. *Using Language*. Cambridge University Press, 1996.

[8] Hamish Cunningham. A definition and short history of language engineering. *Journal of Natural Language Engineering*, 5(1):1–16, 1999.

[9] Lars Degerstedt and Pontus Johansson. Evolutionary Development of Phase-Based Dialogue Systems. In *Proceedings of the 8th Scandinavian Conference on Artificial Intelligence*, pages 59–67, Bergen, Norway, 2003.

[10] David Garlan and Mary Shaw. An Introduction to Software Architecture. *Advances in Software Engineering and Knowledge Engineering, Series on Software Engineering and Knowledge Engineering*, 2:1–39, 1993.

[11] Pontus Johansson, Lars Degerstedt, and Arne Jönsson. Iterative development of an information-providing dialogue system. In *Proceedings of the 7th ERCIM Workshop on User Interfaces for All: Universal Access Special Theme*, pages 29–36, Chantilly, France, 2002.

[12] David Kirsh. Today the earwig, tomorrow man? In Margaret A. Boden, editor, *The Philosophy of Artificial Life*, Oxford Readings in Philosophy, chapter 8, pages 237–261. Oxford University Press, 1996.

[13] Oliver Lemon, Lawrence Cavedon, and Barbara Kelly. Managing dialogue interaction: A multi-layered approach. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 168–177, 2003.

[14] Pattie Maes. How to do the right thing. *Connection Science Journal*, 1(3):291–323, 1989.

[15] Alan Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.

[16] Rolf Pfeiffer and Christian Scheier. *Understanding Intelligence*. The MIT Press, 1999.

[17] Reid Simmons, Trey Smith, M Bernardine Dias, Dani Goldberg, David Hershberger, Anthony (Tony) Stentz, and Robert Michael Zlot. A layered architecture for coordination of mobile robots. In *Multi-Robot Systems: From Swarms to Intelligent Automata, Proceedings from the 2002 NRL Workshop on Multi-Robot Systems*. Kluwer Academic Publishers, May 2002.

[18] Luc Steels. A case study in the behavior-oriented design of autonomous agents. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From Animals to Animats*, pages 445–452. MIT Press, 1994.

[19] Kristinn R. Thórisson. Layered, modular action control for communicative humanoids. In *Computer Animation '97*, pages 134–143, Geneva, Switzerland, June 5–6 1997.

[20] Kristinn R. Thórisson. Natural turn-taking needs no manual: Computational theory and model, from perception to action. In B. Granström, D. House, and I. Karlsson, editors, *Multimodality in Language and Speech Systems*, pages 173–207. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.