

# A Model for Flexible Interoperability between Dialogue Management and Domain Reasoning for Conversational Spoken Dialogue Systems

Jaakko Hakulinen<sup>1</sup>, Markku Turunen<sup>1</sup>, Cameron Smith<sup>2</sup>, Marc Cavazza<sup>2</sup>, Daniel Charlton<sup>2</sup>

<sup>1</sup>Speech-based and Pervasive Interaction Group, Tampere Unit for Computer Human Interaction, University of Tampere, Finland

<sup>2</sup>School of Computing, University of Teesside, Middlesbrough, United Kingdom  
{mturunen, jh}@cs.uta.fi, {c.g.smith, m.o.cavazza, d.charlton}@tees.ac.uk

**Abstract.** When spoken dialogue systems move beyond task oriented dialogues, their growing complexity calls for more modular structures. In the Health and Fitness Companion dialogue system, we have separated cognitive modelling, i.e., reasoning and learning about the domain on a higher level, from dialogue management, i.e., managing interaction level phenomena such as turn taking and low level error management. This separation, utilizing a dialogue plan to communicate information from a cognitive model to the dialogue management, enables the two components to contribute to the structure of the dialogue, error correction, etc. from their own perspectives without containing overlapping logic.

## 1 INTRODUCTION

Within the EU-funded COMPANIONS-project [1] we are developing a conversational Health and Fitness Companion (H&F) that develops long-lasting relationships with its users to support their healthy living and eating habits. The Companion plans each day together with its user, and suggests healthy activities, such as walking to work. During exercises and in the evening at home, the user is able to report back to the Companion on the day, and get more advice and support. A mobile version of the Companion can be taken to physical exercises while the main system is found at home as a physically embodied conversational agent in the form of a Nabaztag rabbit, as seen in .

Instead of helping with a single well-defined task, the Health and Fitness Companion aims at building a long-term relationship with its user and providing support on a daily basis. Use of spoken dialogue and physical embodiment are designed to support the creation of relations between the user and the system. Such a relationship is likely to encourage the user to make the commitment a lifestyle change requires [2].

Such a long lasting relationship is in contrast to most existing spoken dialogue systems, which are task oriented, designed to solve well-defined tasks, such as making a booking or providing timetable information. However, in addition to the various Companions being created in our project, there are various other systems which have been titled domain-oriented dialogues [3]. These systems differ from traditional task-oriented systems since interaction with the systems, typically modelled as a



Figure 1: Interaction with the Companions home system using Nabztag rabbit.

conversation with a virtual character, can be the main motivation for the interaction.

In the domain-oriented applications, dialogue management easily becomes complex. For example, the Health and Fitness Companion must eventually be able to model interaction history on a longer timescale than just the usual dialogue history of a single session. As the user and system become more familiar with each other, this must also be reflected in the small details of interaction, such as how questions are phrased. Furthermore, the system must include a model capable of reasoning about the domain and learn from the user and his or her actions in order to provide meaningful interaction, such as to provide reasonable guidance on the user's health and progress as the user's condition alters over time. The resulting system can quickly become too complex to implement and maintain. While natural language generation needs to be more dynamic and natural language understanding may, for example, need to handle more references, dialogue management in particular receives new tasks. Increased complexity of the domain is accompanied by the need to maintain the user-system relationship and model the long interaction history.

Dialogue management is usually based on state transition networks, form filling, or some other clearly defined model. Clearly, these models tend to reach their limits in applications such as the Health and Fitness Companion. Separating domain specific processing to the back-end makes it possible to manage dialogue with the existing manageable methods. It is common to separate domain specific processing, such as database queries, into a back-end component. Many researchers have worked with separating generic dialogue management processes from the domain specific processes. Example solutions include shells [4] and object-oriented programming methods [5;6]. On the other hand, simple back-end interfaces, e.g., SQL queries, can be included as configuration parameters [7].

In this work, we present a model where the domain specific module is more than just a simple interface and includes active processing of domain information, reasoning, learning, or other complex processes. We call such a component a cognitive model. While the task of a dialogue manager is to maintain and update the dialogue state, the cognitive model reasons using domain level knowledge. Separation of the two is not trivial. For example, managing dialogue level phenomena, such as error handling and basic input processing, are tasks clearly in the area of dialogue modelling. However, cognitive modelling can help error handling, by spotting input that seems suspicious based on domain level information, and input parsing, by providing information on potential discussion topics a user may be switching to. The solution we have devised is to have the cognitive model guide the dialogue management with a dialogue plan while the dialogue management provides parsed user inputs back to the cognitive model. In addition to connecting the cognitive model and the dialogue manager within the home Companion, the same solution is used for the communication between the home Companion and mobile Companion.

Next, we look at the software architecture and knowledge representation of the Health and Fitness Companion. This is accomplished by examining the system's actions as it processes a sample dialogue. Discussion of the benefits of our approach and how we seek to extend this work follows.

## 2. DIALOGUE MANAGEMENT AND COGNITIVE MODELING

There is great consensus that components of a dialogue system can be split into at least three parts: an input module, which receives user input and parses it into a logical form, dialogue management, which maintains and updates the dialogue state based on user input and generates output requests, and an output module, which generates natural language output to user based on the requests. In the case of the H&F, we have also separated a cognitive model (CM) from the dialogue manager (DM).

The DM takes care of only dialogue level phenomena, such as confirmations, initiative and turn taking. This leaves reasoning about the domain, deciding what information to collect from the user and what suggestions to give, to the CM. The CM contains all higher-level reasoning, or what can be considered the higher-level cognitive processes of the Companion, thus the name.

Figure 2 provides an overview of the system architecture from the point of view of information stored and passed between components. In the following, we look at the communication between the cognitive model and the dialogue manager by examining how the system processes a sample dialogue

(Example 1). This covers generation of an initial activity model, parsing of user input and the consequent regeneration of the activity model. We then take a closer look at some of functions carried out by the dialogue management. Further details on the cognitive model and on the technical side of the architecture, such as the support of Nabaztag rabbits, can be found in dedicated papers [8;9;10].

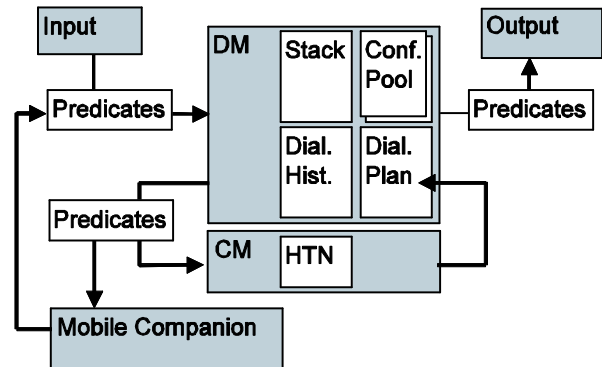


Figure 2: Overview of H&F components focusing on the dialogue manager (DM) and the cognitive model (CM).

### Generating an Initial Activity Model

From the perspective of the CM, the purpose of the user's dialogue with the home Companion is to help the user plan their day at work. This is accomplished by constructing an activity model for the day which consists of normal daily activities such as travelling to work, getting lunch and dinner, and participating in leisure activities after working hours. This is an iterative process where, starting with default knowledge and information on a user's routine preferences, the activity model is gradually refined and agreed with the user. The Companion makes suggestions to the user which aim to encourage a healthy lifestyle while incorporating constraints and preferences gathered from conversation with the user. Dialogue plans are used to guide the conversation with the user so that all the necessary information for constructing the activity model is covered. Currently there are two main phases handled by the CM with corresponding dialogue plans for each. The planning phase, generally undertaken at the start of each day, involves agreeing a plan for what activities the user will do during that day. The reporting phase, generally undertaken at the end of each day, involves getting the user to report back on the actual activities carried out.

*System: Why don't you cycle to work today?*

*User: Not today, it's raining.*

*System: Okay. Why don't you take the train to work?*

*User: No, I would prefer to take the bus today.*

*System: That's fine. Will you be taking a packed lunch to work?*

*User: No, I'm having lunch in a restaurant with some colleagues.*

*System: Alright. So, it is raining, you are going to take the bus to work and you are eating lunch in a restaurant, are they all correct?*

*User: Yes, that's right.*

Example 1: A Sample Dialogue

The central technology of the CM is a Hierarchical Task Network (HTN) planner [11] which is used to generate both the activity model itself and the dialogue plans which support this. A HTN planner works by taking a high-level task and decomposing this task into sub-tasks until a series of primitive tasks are reached which can be carried out sequentially to achieve the high-level task. Modelling the domain using HTNs is thus very useful as it corresponds well with both an activity model, with a general task of the user undertaking their working day moving down to the individual activities carried out by the user, and a dialogue plan, with a high-level view of what should be discussed decomposing down into the individual items for discussion.

Our worked example thus starts with the generation of an initial activity model. This makes use of the default information on what activities are available along with any stored information on a user's habitual preferences (that is, those activities routinely carried out by the user) to construct the healthiest possible activity model. Dialogue plans are then generated from this initial activity model to direct the course of the conversation: an instantiation plan (corresponding to the planning phase) and a reporting plan (corresponding to the reporting phase). The individual plan items are represented as predicates by the HTN planner and encoded into XML to be sent to the DM. They cover a range of system responses including suggestions, queries and confirmations. An instantiation plan (Example 2) for the worked example would therefore include items suggesting cycling to work, (Suggest-Travel-Method Cycling-Travel Home Work), and taking a packed lunch to work, (Suggest-Meal-Source Eat-Packed-Meal Lunch), as these are the healthiest options which fit with the user's routine.

```
<plan>
<plan-name>Generate-Task-Model-Questions</plan-name>
<plan-item>
  <action>QUERY-PLANNED-ACTIVITY</action>
</plan-item>
<plan-item>
  <action>SUGGEST-TRAVEL-METHOD</action>
  <param>CYCLING-TRAVEL</param>
  <param>HOME</param>
  <param>WORK</param>
</plan-item>
```

...

Example 2: Start of the Instantiation Dialogue Plan.

### Parsing User Input

The DM receives the dialogue plan and uses this in conversation with the user. Each plan item contains all the information necessary to discuss that dialogue item both in terms of generating the system response and categorising the user's reply. The DM maintains a dialogue stack. It contains the currently active input and topic for each item. New items are pushed into the stack when the current topic changes before the previous topic is finished. This can occur when the user initiates a new topic before providing enough information to complete the previous topic. When a dialogue topic is finished, the DM checks the stack to see if it should return back to an unfinished topic. New topics are taken from the plan only when the stack is empty. The DM waits until a topic is complete before passing this information back to the CM. On occasion a user input may

contain predicates which are missing a parameter. In such cases the DM adds clarification questions to complete the predicates before sending them to the CM.

Speech inputs are recognized using the Loquendo ASR engine with rule based grammars. The current size of the grammars is about 1400 individual words. There is a separate, dedicated grammar for each topic and special grammars for confirmations, etc. The grammars contain Semantic Interpretation for Speech Recognition (SISR) tags, which return items compatible with the predicate presentation. These can be full predicates, partial predicates or lists of both of these.

To parse input to a predicate, the items are unified to form full predicates. Input parsing knows the entire universe of legal predicates (values of some predicate parameters are left open). The unification to complete predicates starts by matching to predicates which match the last system output. For example, if the output was a suggestion, acceptance and rejection predicates with the same parameters are tested. If that does not result in a parse, next all predicates of the current topic are used. Each plan item belongs to a topic group which is used when parsing the input. Finally, the entire predicate space is used as necessary. This model could be further elaborated by using also the dialogue history and the plans to provide intermediate steps. However, at the moment that has not been necessary.

Currently, the system contains simple rule based dialogue act tagging. Mark-up used for this is a subset of DAMSL tags. At the moment the dialogue acts are not necessary as the predicate logic already dictates the dialogue act and input parsing does not need the dialogue act information.

In our example dialogue (Example 1), the user responds to the suggestion of cycling with "Not today, it's raining". The ASR returns the semantic information #REJECT-TASK, #WEATHER BAD for this (Example 3). The first of these is unified with the question predicate (Suggest-Travel-Method Cycling-Travel Home Work) to generate the response (Reject-Task Cycling-Travel Home Work). The second is already the full predicate (Weather Bad). This is then returned back to the CM.

### Regenerating the Activity Model

The CM takes the predicates returned from the DM and adds them directly to the HTN planner state. If any of the instantiation plan has been agreed with the user, thus returning an accept-task predicate, those parts of the activity model are updated as being 'planned' and will no result in items into the updated dialogue plan. The activity model is then regenerated and those parts not updated (and thus remaining 'unplanned') are replanned. This ensures that items agreed with the user are not revised while items not yet agreed benefit from any additional information gained in conversation with the user.

The HTN planner makes use of semantic information as a heuristic while planning to generate the best plan possible. When there are multiple possible options for a given decomposition of the activity model, each option is given a semantic score based on the semantic tags assigned to it. The semantic score is determined dynamically based on the current state. Any information from the user, such as the fact it is raining, (Weather Bad), and their preference for bus travel, (Pref-For Bus-Travel), added to the state can thus affect the replanning of the activity model. Following the first user response, cycling is explicitly rejected by the user with the poor weather condition given as a

reason. While the explicit rejection of cycling prevents that option from being selected at the next replanning stage, the (Weather Bad) predicate alters the semantic scores such that those options featuring significant amounts of walking are also ignored and (Suggest-Travel-Method Train-Travel Home Work) is selected as the best option. Likewise the preference for bus travel provided with the next utterance ensures that (Suggest-Travel-Method Bus-Travel Home Work) is the next suggestion.

Once the activity model has been regenerated then updated dialogue plans are also generated and passed back to the DM to guide the next round of conversation. The instantiation and reporting plans are essentially mirrored yet asymmetric in that as items are agreed with the user they disappear from the instantiation plan and are added to the reporting plan. (That is, as a suggestion of a particular activity is agreed it is removed from the instantiation plan and an item on the reporting plan is creating querying whether that particular activity has been completed.) As all of the initial activity model must be agreed with the user, the starting instantiation plan will be full of discussion items while the starting reporting plan will be empty. The plans will generally progress towards a mid-point where the instantiation plan has become empty with all the items effectively moving onto the reporting plan. Finally, the reporting plan will be emptied throughout the reporting phase until both plans are empty and the activity model has been completed for the day. However, some new information may result in additional items being created on the plans. For example, if the user announces they are planning on going to the gym an item will be added to the instantiation plan to discuss what method of transport they will use to reach the gym.

*System: (Suggest-Travel-Method Cycling-Travel Home Work)*

*System: Why don't you cycle to work today?*

User: Not today, it's raining.

ASR: #REJECT-TASK, #WEATHER BAD

User: (Reject-Task Cycling-Travel Home Work)

User: (Weather Bad)

*System: (Suggest-Travel-Method Train-Travel Home Work)*

*System: Okay. Why don't you take the train to work?*

User: No, I would prefer to take the bus today.

ASR: #REJECT-TASK, #PREF-FOR BUS-TRAVEL

User: (Reject-Task Train-Travel Home Work)

User: (Pref-For Bus-Travel)

*System: (Suggest-Travel-Method Bus-Travel Home Work)*

*System: [Auto-Accept]*

*System: (Suggest-Meal-Source Eat-Packed-Meal Lunch)*

*System: That's fine. Will you be taking a packed lunch to work?*

User: No, I'm having lunch in a restaurant with some colleagues.

ASR: #REJECT-TASK, #PREF-FOR EAT-AT-RESTAURANT

User: (Reject-Task Eat-Packed-Meal Lunch)

User: (Pref-For Eat-At-Restaurant)

*System: (Suggest-Meal-Source Eat-At-Restaurant Lunch)*

*System: [Auto-Accept]*

*System: [Group Confirmation]*

*System: Alright. So, it is raining, you are going to take the bus to work and you are eating lunch in a restaurant, are they all correct?*

User: Yes, that's right.

ASR: #YES

Example 3: A Sample Dialogue with Predicate Annotation

## Dialogue Management

While the CM focuses on guiding the conversation towards completing the high-level goal of constructing a complete activity model for the user, the DM focuses instead on the interaction level issues of the dialogue itself. While the most important of these is perhaps error management, our dialogue management encompasses a range of functionality also including dialogue-based optimisations, session management and maintaining a dialogue history.

Currently the main focus of our error management has been confirmations. User input must be confirmed (at least the most important items) to enable errors to be spotted and corrected and to make users feel confident about the system. Currently, all major user input is confirmed. That is, all user input whose validity cannot be implicitly confirmed based on the validation of another segment of user input. Within our dialogue example, acceptance of bus travel to work, eating lunch in a restaurant and the fact it is raining must all be explicitly confirmed. However, if these are confirmed successfully then the other information can be implied as confirmed. For example, it is pointless to confirm that the user wants to reject cycling travel and that they prefer bus travel (in this specific context) as, if this was not true, they would not have confirmed acceptance of bus travel.

By default, when confirming an input predicate it is placed in the group confirmation pool. Items in the group confirmation pool are confirmed altogether, in a single question, at an appropriate point in the conversation. However, if there is a reason to believe that a predicate is incorrect, it is placed into the immediate confirmation pool where items are confirmed independently of each other. This happens when the user rejects a group confirmation and also when the ASR confidence score of a user input is below a certain threshold.

While error management mainly consists of identifying errors in understanding the user, it is also important to handle incidents where the user may not have understood the Companion. The grammar is able to handle comments such as "I'm sorry, can you say that again?" and issue a repeat request to the DM who will repeat the previous question for the user.

Beyond identifying errors is the ability of the DM to enhance the conversation. Since the DM handles communication with the user as part of an ongoing dialogue and not in terms of individual, independent utterances it has the potential to carry out optimisations by making use of this overall knowledge of the dialogue. One such optimisation is the auto-accept functionality which automatically accepts suggestions if they follow a stated preference by the user. For the CM, suggestions are the result of a range of factors coming together to highlight a particular option as being particularly suitable. However, it can be particularly jarring to the user when the Companion suggests a course of action they have just expressed a preference for as, despite this being a sign the Companion has understood and is acting on the previous statement, it seems to imply the Companion is not listening to the user. The DM is able to identify such a situation and automatically accepts the suggestion on behalf of the user. This is then confirmed with the user to ensure this is what the user actually wanted.

Another aspect of the ongoing dialogue with the user is that of allowing them control over when they talk to the Companion and for how long. While the system is designed around the principle of two conversations with the Companion, one in the morning and one in the evening, this may not always be suitable

for the user. The DM therefore allows conversations to be curtailed by the user, before the Companion would otherwise choose to finish them, and recommenced at a later stage. It should also be noted that the CM further supports this functionality in how it manages the dialogue plans. For example, the user could plan their day up till dinner and then leave the session. When they return home from work (and initiate conversation with the Companion) they would then be asked to report back on the portion of the day they had previously planned and carried out before going onto plan the evening's activities. A final session later in the evening would then allow reporting back on the remaining part of the day.

Finally, the DM maintains a dialogue history which consists of a list of dialogue turns, one seen in Example 4. Each turn has the participant information (user, system), current topic, logical content of the turn (a set of predicates) and, for system turns, actual speech produced or, for user turns, recognition results (n-best list with confidence scores) and dialogue act tagging with DAMSL based tags.

```
<dialogueTurn>
<participant>user</participant>
<topic>
<pred>
<action>SUGGEST-TRAVEL-METHOD</action>
<param>null</param>
<param>HOME</param>
<param>WORK</param>
</pred>
</topic>
<speech>okay I can do that</speech>
<logical>
<pred>
<action>ACCEPT-TASK</action>
<param>CYCLING-TRAVEL</param>
<param>HOME</param>
<param>WORK</param>
<DAct Forward-function="Influence-on-speaker" Backward-
fintion="Agreement" Info-level="Task">
</DAct>
<score>0.631088</score>
</pred>
</logical>
</dialogueTurn>
```

Example 4: Dialogue History Item.

### Mobile Companion

The mobile companion [12] supports exercise activities such as jogging using GPS. The mobile system is linked to the home system so that as a plan for the day is generated with the user, this plan is published in a web service from where the mobile system can download it as it becomes available. The plan consists of the same predicates the CM sends to the DM. From this plan of the day, the mobile companion selects all information pertinent to the exercise activities so, at the appropriate point in the user's day, it can go through these with the user. The results of the exercises are delivered back to the home system which removes the need to report back on them verbally in the evening.

## 3 DISCUSSION & FUTURE WORK

Key to the interoperability between the DM and CM is the level at which the two are split. The CM's aim is to provide the higher-level cognitive functions of the Companion. Currently those functions offered mainly cover planning and reasoning over the health and fitness domain. The main remit of the CM is thus to handle domain (or encyclopaedic) knowledge, common sense knowledge and task knowledge. The CM therefore also includes those components of the dialogue management which cover the user model and the task model. The CM's focus on domain knowledge allows the DM to remain relatively domain-independent. Conversely, the DM's focus on dealing with the ongoing dialogue allows the CM to abstract itself from dialogue-related issues. This simplifies the implementation of both which is of particular benefit for complex applications such as the Health and Fitness Companion.

As mentioned previously, the separation of the two is non-trivial as information possessed by one part can be of use in processing falling under the remit of the other. However, the interface between the DM and the CM offers great flexibility as each can pass information without a need to understand how this information will be used. For example, if the CM notices an error with a user input based on some domain reasoning, the CM can simply include this information in the dialogue plan without worrying about how it will be resolved. The DM, on processing the dialogue plan, can then handle this error and update the CM appropriately without needing to understand the cause of the error.

Currently the DM generally follows the dialogue plan as set by the CM. However, the DM is not limited to following the plan as given but can instead select items based on the most suitable order for the current dialogue with the user. This is something we look to make greater use of in the future, in particular as we work on introducing small talk.

A Companion should be able to take into account the relative intrusiveness of topics and this is particularly important when dealing with potentially sensitive subjects such as the user's health. The flexibility offered by the dialogue plan can help by allowing the Companion to present items in an order which builds up to more delicate questions and even introduce new discussion items for the sole purpose of building up trust with the user. Likewise the flexibility offered by the dialogue plan allows the CM to use its domain knowledge to advice on appropriate discussion points for small talk without the need to understand when or how to include them in conversation.

This approach to dialogue management has enabled the CM and the DM to be developed in parallel by different groups using different programming languages while maintaining a coherent system. The dialogue system is implemented using Java and the Jaspis framework [13] with jNabServer for Nabaztag connectivity. The cognitive model is implemented in Allegro Common Lisp and integrated into the Jaspis framework.

On an implementation level, the model is independent of the mechanics of either the DM or the CM. The DM can be implemented using state transition networks (a network per plan item), forms (form per item), agent based model, as in our case, or any other suitable method. Similarly, the plan does not tie the CM to any specific implementation.

The H&F is a working system capable of planning, with a user, an office worker's typical day at work and then getting them to report back on the actual activities performed. The

system has undergone user testing which is discussed in detail elsewhere [10]. Our focus now is to expand the system to handle the complexity necessary for a Companion.

One aspect of this complexity is the conversational ability of the Companion. As we seek to increase the conversation beyond a purely functional level towards that of a relationship with the user, so it is desirable for the Companion to be capable of an increased depth in its conversations. This means being capable of discussing things in detail when required, but not overloading the user with unnecessary details.

Currently, individual dialogue plan items are currently classed as distinct topics. We plan to revise this representation such that plan items can be grouped together under topics. Further, topics can then be nested to create a hierarchical structure such that top-level topics offer discussion items at a very general level while lower-level topics offer increasing levels of detail to deal more thoroughly with specific aspects of the higher-level topics. This will allow us to harness the power of the CM by applying various types of knowledge to a situation while making use of the HTN planner to ensure this information is relayed at an appropriate granularity.

Another aspect of this complexity is how the ongoing relationship with the user is managed. The system currently collects information for a single day, treating each day as a standalone entity. We plan to build upon this by aggregating the daily information so it can be used to inform future sessions. The long-term nature of this interaction with the user introduces both possibilities and dangers. Possibilities in that each new interaction is an opportunity for the system to both expand its knowledge of the user and use its knowledge of past interactions to further develop its relationship with the user. Dangers in that insufficient variance and adaptability in the Companion will mean that conversations become stale and indistinguishable. The flexibility of the model can be of use here in ensuring that there is sufficient scope to avoid stagnation in the conversation while helping to prevent a spiralling complexity that makes the system impossible to implement.

## 4 CONCLUSIONS

We have described a model to separate cognitive modelling from dialogue management which enables their flexible interoperability. This division, while similar to separation of a back-end from dialogue management, draws the line deeper into the area of interaction management. The cognitive model processes domain-level information and generates dialogue plans. The dialogue manager focuses only on interaction level phenomena, such as initiative and error management, and other meta-communication.

The model for interoperability between the dialogue manager and the cognitive model has provided many benefits for the development of the H&F. It provides great flexibility for both components while at the same time removing the need for either component to implement processing contained in the other. While the dialogue plan generated by the CM provides a base for dialogue management, which, in most cases, is followed, the DM can deviate from it. The DM can handle confirmations as it pleases, add small talk, and process the plan items in any order. Most importantly, all this is possible without including domain specific knowledge. All such information is kept exclusively in the CM.

One example of flexibility is error management; while the actual error correction is the task of the dialogue manager, domain level knowledge can help reveal errors. Using the plan, the cognitive model can provide information to the dialogue manager without knowledge on the details of error management. The model also enables user initiative topic shifts, management of the user-system relationship and other novel issues relevant to domain-oriented dialogue systems.

The model has allowed development of the H&F in parallel by two different groups, resulting in a working system. The H&F is now being expanded to use the benefits of the model to tackle the challenges of creating a long-term Companion.

## 5 ACKNOWLEDGEMENTS

This work was funded by the Companions project ([www.companions-project.org](http://www.companions-project.org)) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-034434.

## REFERENCES

- [2] Wilks, Y., Is There Progress on Talking Sensibly to Machines?, *Science*, 9 Nov 2007.
- [1] Bickmore, T. W., Picard, R. W. Establishing and maintaining long-term human-computer relationships. *ACM Trans. Computer-Human Interaction* Vol. 12, No. 2. (June 2005), pp. 293-327.
- [3] Dybkjaer, L., Bernsen, N. O., Minker, W., Evaluation and usability of multimodal spoken language dialogue systems, *Speech Communication*, 43, 1-2 (June 2004), pp. 33-54.
- [4] Jönsson, A. A Natural Language Shell and Tools for Customizing the Dialogue in Natural Language Interfaces, Research Report, LiTH-IDA-R-91-10, 1991.
- [5] Salonen, E.-P., Hartikainen, M., Turunen, M., Hakulinen J., Funk, J. A. Flexible Dialogue Management Using Distributed and Dynamic Dialogue Control. *Proceedings of ICSLP 2004*. pp. 197-200.
- [6] O'Neill, I. Hanna, P. Liu, X., McTear, M., The Queen's Communicator: An Object-Oriented Dialogue Manager, *Eurospeech 2003*, Geneva, Switzerland (2003), pp. 593-596.
- [7] Pellom, B. Ward, W. Pradhan, S., The CU Communicator: An Architecture for Dialogue Systems, *Proceedings of ICSLP 2000*, Beijing China, November 2000.
- [8] Turunen, M., Hakulinen, J., Smith, C., Charlton, D., Li, Z., Cavazza, M. Physically Embodied Conversational Agents as Health and Fitness Companions. In *Proceedings of Interspeech 2008* (to appear).
- [9] Cavazza, M., Smith, C., Charlton, D., Zhang, L., Turunen, M., Hakulinen, J., A 'Companion' ECA with Planning and Activity Modelling (Short Paper), *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 1281-1284. (2008)
- [10] Smith, C., Cavazza, M., Charlton, D., Zhang, L., Turunen, M., Hakulinen, J., Integrating Planning and Dialogue in a Lifestyle Agent, *Proc. of the 8th Int. Conf. on Intelligent Virtual Agents (IVA 2008)*, LNAI 5208, pp. 146-153, (2008).
- [11] Ghallab, M., Nau, D., and Traverso, P., *Automated Planning: Theory and Practice*. Morgan Kaufmann. (2004)
- [12] Ståhl, O., Gambäck, B., Hansen, P., Turunen, M., and Hakulinen, J. A Mobile Fitness Companion. *4<sup>th</sup> International Workshop on Human-Computer Conversation*, 2008.
- [13] Turunen, M., Hakulinen, J., Riihää, K.-J., Salonen, E.-P., Kainulainen, A., and Prusi, P. An architecture and applications for speech-based accessibility systems. *IBM Systems Journal*, Vol. 44, No 3, 2005, pp. 485-504.