Kristiina Jokinen

Adaptation and user expertise modelling in AthosMail

Abstract This article describes the User Model component of AthosMail, a speech-based interactive e-mail application developed in the context of the EU project DUMAS. The focus is on the system’s adaptive capabilities and user expertise modelling, exemplified through the User Model parameters dealing with initiative and explicitness of the system responses. The purpose of the conducted research was to investigate how the users could interact with a system in a more natural way, and the two aspects that mainly influence the system’s interaction capabilities, and thus the naturalness of the dialogue as a whole, are considered to be the dialogue control and the amount of information provided to the user. The User Model produces recommendations of the system’s appropriate reaction depending on the user’s observed competence level, monitored and computed on the basis of the user’s interaction with the system. The article also discusses methods for the evaluation of adaptive user models and presents results from the AthosMail evaluation.

Keywords User modelling · Adaptation · Evaluation · Speech-based human computer interaction · Mobile e-mail applications

1 Introduction

With the increasing complexity and widespread use of electronic devices, the notion of adaptivity has become more important. The ambient nature of applications emphasizes fast and efficient interaction, and the need for natural interaction has also become obvious: humans do not only manipulate electronic devices but also interact with them like with other agents [18]. This kind of intelligent interaction presupposes robust communication capabilities, whereby the system adapts itself to the requirements of the user rather than vice versa. As the system’s internal mechanisms also become too complicated for straightforward description and its component and functional relations too numerous for systematic listing, an ordinary user cannot reasonably be expected to master all the details of the interaction. Systems are also meant to be used in mobile and versatile environment, by various users with different skills, abilities, and requirements, and usability of such systems would increase if their functionality could be tailored according to the varied preferences of different users.

An adaptive system decides on the appropriate action based on its model of the user and the context. The user model incorporates both permanent and temporal user characteristics (e.g., user preferences, long- and short-term interests, language, temporally changing goals, etc.), while the context is the dynamic environment in which the system is used. In interface design, adaptivity is often implemented in a static way: users personalize interfaces by modifying their properties, choosing the most pleasing colour or sound options, and setting up a list of preferences as a personal profile so as to enable preferred filtering and rating of incoming information. This is common in e-commerce and with information providers; for instance, InfoQuest (http://www.inforian.com/quest) offers personalized online information access with the user’s interests stored as keywords, while MyYahoo (http//my.yahoo.com/) provides web portals with the contents customized based on the selected themes according to the user’s preferences.

Dynamic adaptation is realized through the system’s online ability to classify users into certain categories, e.g., on the basis of their navigation choices or a list of specific keywords in their search queries, so that personalized access to information sources can be provided and user profiles updated automatically. For instance, Amalthaea [33] updates the user profile on the basis of the user’s explicit feedback and thus learns about the
user’s interests over time. Besides individual filtering and rating, adaptation can also take place collaboratively. Statistical information of user groups can be used as in recommendation systems [13]: preference information of a group is tracked by comparing the selected items of one user to similar items selected by the other users (collaborative filtering). Adaptive applications of this kind include news filtering ([5], 32) and giving recommendations for web-browsing ([35], books (e.g., http://www.amazon.com), or TV programs ([9],[3]). However, as Brusilovsky and Maybury [4] point out, although adaptation effects can be realized in many ways, due to the fairly rigid context of adaptive hypermedia and web systems (where the user can only navigate and search for information), adaptation in these applications is restricted to three major types: (1) adaptive content selection, whereby the system adaptively selects and prioritizes the relevant items for the user; (2) adaptive navigation support, whereby the system can hide or sort the links while the user navigates from one item to another; and (3) adaptive presentation, whereby the content of a page is presented according to the user preferences.

In dialogue management research, the focus has been on techniques and models that enable users to interact with speech-based dialogue systems. Adaptation has mainly concerned dialogue strategies that allow flexible interaction with the user, and the system’s capability to take initiative depending on the dialogue situation. For instance, Smith [38] observes that it is safer for beginners to be closely guided by the system, while experienced users like to take the initiative, which results in more efficient dialogues in terms of decreased average completion time and decreased average number of utterances. In general, adaptive dialogue capabilities seem to result in more successful dialogue systems. Litman and Pan [31] compared the user-adaptable and non-adaptable versions of an information retrieval system and concluded that the former outperformed the latter. Also Chu-Carroll and Nickerson [8] compared an adaptive dialogue system to two non-adaptive versions of the same system (one with a system initiative and the other with a mixed-initiative dialogue strategy), and found that the adaptive system performed better than the others in terms of user satisfaction, dialogue efficiency (number of utterances), and dialogue quality (speech recognizer performance).

An important question related to adaptivity is who should be in charge of it. Automatic adaptations can be confusing for the user, and the feeling of losing control of the system decreases user satisfaction. The user may, of course, be given an explicit option to change system properties, like the system-initiative dialogue strategy to the user-initiative one, depending on the situation [31]. This makes the system more transparent and apparently also adds to users’ satisfaction as the users can feel they are in control of the system. On the other hand, Jameson and Schwarzkopf [16] considered the question of control in an adaptive recommendation system for the web, and one of their observations was that while letting the user decide the moment of adaptation may enhance the feeling of control, the user may not be interested, i.e., some users do not want to take the initiative.

From the point of view of intelligent and intuitive speech-based interaction, it seems important that the system exhibit the capability to adapt and adjust its behavioural patterns automatically according to the user’s behaviour, without the user explicitly requesting the change. People tend to adapt their behaviour according to the speech partner, which suggests that a system that could adapt itself to various types of users, various situations, and various user actions would make human–computer interaction more natural. For instance, interaction patterns vary according to individual users, and the patterns could thus be learnt by the system in its attempts to adapt to, and anticipate, a particular user’s behaviour [23]. Oppermann [35] investigated adaptivity and adaptability in a spreadsheet application, and ended up proposing a system that suggests adaptations to the user. He observed that the users wanted to influence the timing and content of adaptations, but the controlling of adaptivity was too demanding a task for the user alone, so the system should provide assistance and be helpful in cases where adaptation is possible.

The component that takes care of adaptation in spoken dialogue systems is the User Model (UM). While the Dialogue Manager component deals directly with the system’s conversational capability, the User Model component records the system’s knowledge of the user’s beliefs, knowledge, and characteristics, and allows the system to address specific user properties when planning its responses. For instance, descriptions of physical devices could be built on the descriptive or functional properties of the device, depending on whether the description is intended for a novice or an experienced user [37]. From early on, UM research has focused on the user’s beliefs and knowledge, so as to help the system to provide information that would be appropriate to the user’s level of expertise [6]. New information that is likely to be important and of interest to the user should be presented first and in a form that is likely to be understood and interpreted correctly by the user, in order to support effective and smooth communication.

In the EU project DUMAS [19], the main goal was to develop a prototype interactive e-mail reading system, AthosMail, with a special User Model component that would enable the user to have more natural and flexible speech-based interactions with the system. Natural and flexible interaction refers to interaction, whereby the users may express themselves using natural language expressions, and the system provides responses that take the particular user’s needs and expertise level into account. The DUMAS UM component thus records the user’s actions and estimates the user’s competence levels, giving recommendations to other system components on the appropriate way of responding.
The focus of this article is on the AthosMail system's capability to support natural interaction and to produce responses that are appropriately tailored according to the user's perceived competence levels. Special emphasis is put on the evaluation methods for adaptive systems, and the experience acquired from the AthosMail evaluation is discussed. The article is organized as follows. Section 2 presents the AthosMail system and its User Model component. It also introduces the system functionality which the users need to familiarize themselves with as well as the specific user expertise areas under investigation. In Sect. 3, adaptation in AthosMail is discussed in detail. The three-level user expertise model and the concepts of DASEX (dialogue act specific explicitness) and INIT (dialogue initiative), which represent the system's view of the user's expertise, are defined. The parameters for calculating the online and offline values of these variables are also described, and examples are given of their effect in the realization of system responses. Section 4 concentrates on the evaluation and evaluation methods of an adaptive system. Issues dealing with adaptation are discussed, with special reference to user expertise modelling, as well as problems in developing methods for adaptive system evaluation in general. The test setups used in the evaluation of the AthosMail system are presented, while the results and experience gathered from the AthosMail evaluation are examined in Sect. 5. Conclusion and future directions are considered in Sect. 6.

2 User modelling in AthosMail

2.1 User model components

According to Benyon and Murray [2], adaptivity provides an answer to many usability problems. However, they also acknowledge that adaptivity is not a shortcut to better interfaces and emphasize the importance of a disciplined task analysis in order to find out whether and where adaptive functionality is needed. They outline a methodology for adaptivity design and suggest guidelines such as: (a) adapt the system where the users are least adaptable, (b) adapt to features which have the largest impact on interaction, and (c) adapt to the needs of intermittent and discretionary users.

In AthosMail, the User Model component draws on previous work, especially on adaptive dialogue management and on research dealing with initiative strategies and response explicitness. This is related to the second point in Benyon and Murray’s guidelines: dialogue features that contribute most to natural and malleable interaction are generally regarded as those dealing with the system’s ability to take initiative and to produce responses with varying degrees of explicitness. Also project-specific requirements of how user modelling should function in the system architecture were taken into account in the design process. Functional and technical requirements of the UM component were obtained by interviewing the partners who were asked about their views and expectations concerning user modelling in the planned AthosMail system. Especially, the need for adaptation was discussed, and the system’s adaptability to different types of users was emphasized (the project considered busy mobile users and visually impaired users). The first version of the User Model was designed on the basis of these guidelines, and further development continued via iterative design later in the project.

The following sub-components were considered relevant to AthosMail [25]:

1. Cooperativity Model. This component provides the system with recommendations concerning the explicitness level of the system utterances. Recommendations are based on the user’s competence level and the level of dialogue control exerted by the system depending on speech recognition success. If a dialogue problem seems to originate from limited user expertise, the user will be given more explicit guidance, and if the fault is in speech recognizer or in language understanding, the system assumes a more active role and takes more initiative. The two variables involved are dialogue act specific explicitness (DASEX) and initiative (INIT).

2. Message Prioritization. This component sorts out incoming messages so that the messages that the user most likely finds interesting and important are at the beginning of the list (ViewList). The importance of a message is calculated on the basis of user actions. For instance, if the user has always deleted messages from a certain sender without reading them first, it is likely that the sender’s messages are not important to the user. The importance of the entire message is a weighted sum of the scores of the message features, which include sender, received group, subject, keywords, and topics and are given a score from $-1$ to $+1$ separately.

3. Goal Guessing. This allows the system to make educated guesses about the user’s goals and behavioural patterns in the interaction situation. The main benefit of goal guessing is to help dialogue management to decide what to do when there is uncertainty of the user action because of bad speech recognition. The goal suggestions can also be used to help, e.g., online tutoring systems to provide more relevant guidance to the user.

4. Message Categorization. This component allows the system to compare the incoming messages to the existing ones and to cluster messages according to their topical similarity with existing messages. The content-based categorization make use of the Random Indexing vector space methodology [26, 28], which accumulates semantic representations of words based on co-occurrence statistics.

5. User Preferences. These are fixed properties of the user dealing with aspects like the preferred speaking
The top-level AthosMail architecture is presented in Fig. 1. It consists of an online and offline component, the former being responsible for the online interaction with the user via speech recognition and synthesis, and the latter performing operations on the mailbox. Similarly to the main system, the User Model component also consists of an online and offline part. The first is included in the online Spoken Dialogue System, while the second is included in the offline e-mail processing component. In particular, the Cooperativity Component has an online and offline part (see Fig. 2), while all the other UM sub-components work offline (this is because they need access to the entire mailbox, and in practice this would have been impossible to implement online). Also the Goal Guessing component has online and offline versions which, however, were not included in the final integrated system, as they had originally been developed for more elaborated tasks than the simple functionality by the final AthosMail system could offer.

The UM components produce recommendations which concern the appropriate level of explicitness in the responses (DASEX), whether the system should assume dialogue control (INIT), and how the message list is best presented to the user when she logs in (ViewList). The Dialogue Manager uses the recommendations to decide on the next action and produces an utterance plan which can be realized in different forms and languages. Details of the framework are reported in Turunen et al. [40].

2.2 System functionality

The functionality of the test prototype comprises three main functions: the user can navigate in the mailbox, read messages, and delete messages. Moreover, the user has the option of asking the system to repeat its previous utterance and can also explicitly ask for help. For ease of navigation, the Message Priority Component produces different “views” of the mailbox content, classifying the messages according to the message features: sender, subject, and topic. The user can move from one view to the next with commands such as “next”, “previous”, or “first view”. Within a particular view, the user can navigate among the messages in the same way, by saying “next”, “fourth message”, or “last message”, and so on. The user may also read messages by selecting a message and then saying “read (the message)”, or by referring to a message by saying, e.g., “read the third message”. Deletion is handled in a similar fashion, by the user saying “delete (the message)”. If the user’s command entails destructive actions, such as cancelling the previous command or deleting a message, the system asks for confirmation. Explicit confirmation is always needed regardless of the user’s familiarity with the system. Although some users might want to opt out this choice, it was motivated by practical consideration concerning the present-day speech recognizers: to make sure that the user command is correctly interpreted, an explicit confirmation is the safest option for irreversible actions. AthosMail may also ask for clarifications, if the speech recognition is deemed unreliable in general.

2.3 Areas of user expertise

The user expertise needed for AthosMail is divided into three main areas, so as to facilitate a more controlled analysis. The three main areas are as follows:

1. Knowledge of system functionality: does the user know what functionality is included in the system?
2. Correct formulation of commands: does the user know which words and phrases the system is capable of understanding?
3. Situational knowledge: does the user know which functionality is available at which point in the dialogue?

The first area is the most basic one, as it seeks to answer the question of whether the user knows what kind of tasks the system can be used for. This can be addressed by giving the user detailed instructions about the system and its functionality, but, as is often the case, the actual use of the system may be bewildering (especially for users not familiar with interactive speech
applications), and online help is desirable: the system should provide helpful information concerning the possibilities that are available to the user. The other two areas deal with more detailed knowledge: whether the user knows how to talk to the system, and whether the user knows when to input particular commands, i.e., whether the user knows the logic behind the system functionality and how to take turns in the situation.

It must be noticed that the user may have a good knowledge of the commands and their formulations, yet suffer from frustration as the system does not react as expected, e.g., because the user issues the command at a wrong time or the command is not correctly recognized. When this kind of situational error occurs, it would of course be useful for the user to receive meta-level explanations of what went wrong. From the system point of view, however, such errors are impossible to detect except indirectly, like when the user repeats the same command or produces a slightly different variation of it (or changes the tone of voice and starts to use abusive language). A fine-grained linguistic analysis of the user utterance, combined with intelligent reasoning about the topic and task, would thus be necessary to make the link to previously used utterances to detect

![Diagram of functional relationships between offline and online parameters of the Cooperativity Model](image-url)
problems in the interaction flow. However, as it turns out, an expert user’s reaction to speech recognizer problems resembles a novice user’s random trials of acceptable command formulations. Both repeat various formulations of the same command, and it is next to impossible to distinguish a user utterance which is nonsense and thus misrecognized by the system from an utterance which is correct but signals dissatisfaction because of the system’s earlier misinterpretation and “illogical” behaviour. In AthosMail, an independent speech recognizer problem detector was also developed as part of the online User Model, although in practice the Cooperativity Model used speech recognizer scores as a near estimate of the input quality, and tacitly assumed that the users’ expertise in formulating commands increased simultaneously with their improved ability to speak in a way that the speech recognizer would understand.

The implementation of the UM component focused on the expertise area I, while the other two areas were left for external observations during the evaluation of the system. In particular, the goal was to study if the system responses, designed according to the assumed user skill levels, could enhance usability of the system as they provide tailored assistance to the user through adaptation in the interactive situation. An interesting future research question is if the three expertise areas can be formalized so as to support adaptation in the most beneficial way in practical interaction systems.

### 3 Modelling adaptation and user expertise

In this section, the AthosMail user expertise model and its effect on system behaviour are described. The main idea is that while novice users need guidance, it would be inefficient and annoying for experienced users to be forced to listen to the same instructions every time they use the system. As the user grows more familiar with the system and its possible commands, explicitness in the system replies can be decreased, i.e., the system needs to include less information about what the user can do at each dialogue point. Adaptation to the user’s increased familiarity with the system is thus expected to contribute to natural and enjoyable interaction in a positive way.

#### 3.1 Cooperativity Model

Being able to decide when to switch from guiding a novice to facilitating an expert requires the system to be able to keep track of the user’s expertise level. Jokinen and Kanto [20] argued in favour of a three-level model to take care of the varied skill levels of the users (novice, competent, expert), mainly because users can seldom be categorized only into clear novice and clear expert users. On the other hand, a more fine-grained user classification was considered too detailed for a practical interaction system and thus not useful.

In the AthosMail system, the job of detecting the user’s expertise level and producing recommendations for the appropriate system responses is performed by the Cooperativity Model. This Model can be seen as a continuation of the refinement mechanism whereby the system’s communicative goal is filtered through communicative obligations [17]. Obligations refer to the constraints and requirements that are imposed on the communicating partners due to their engagement in the interaction in the first place. They are seen as signs of rational behaviour and akin to Gricean Maxims [14], but differ from them in that cooperation is regarded in a wider context than just exchanging information: it concerns both cognitive and ethical aspects of communication as well as trust [1]. While obligations can be expressed as explicit rules according to which the system’s utterances are generated [17, 12], the Cooperativity Model produces explicitness and dialogue control values which function as recommendations according to which the Dialogue Manager tailors its responses.

The Cooperativity Model consists of two modules: the online and offline modules (see Fig. 2), following the ideas proposed by Yankelovich [43] and Chu-Carroll [7]. The offline module keeps track of relevant dialogue history and, based on that, calculates offline values for the two variables involved in specifying recommendations to the Dialogue Manager: DASEX and INIT. At the beginning of each session, the previous explicitness and initiative values are passed from the offline module to the online module and placed as the initial values for the recommendations. In case of a totally new user, the system assumes that the user is a novice, and the highest control values are used. Although the users may be familiar with similar commands in other systems, the current model takes into account only the user’s experience with AthosMail. This is based on the consideration that the user’s expertise with respect to AthosMail may not be comprehensive, if it is extrapolated from the user’s earlier experience with similar systems rather than from their actual experience with the system. Of course, in reality there will be interference from the user’s earlier experience to the current usage.

The online module modifies the initial “default” values according to the changing circumstances in the current dialogue. The dialogue events that take place during a session are also recorded and relayed to the offline component at the end of the session. This way, the online module can react to specific situations at runtime, whereas the offline module can track long-term developments behind the local variations in the overall learning curve of the user’s experience with the system. Accordingly, the offline parameters change more slowly in order to round off coincidental fluctuation, while the online module reacts rather quickly to the user’s actions, so that the user’s adaptation to the system functionality can be addressed immediately at runtime. It must be noticed that the changes in the DASEX values themselves are not logged by the offline component, only the events that have affected them during the session.
Moreover, the Cooperativity Model monitors and records the user’s actions specifically on each possible system act. Thus the system can provide help adapted to the user’s familiarity with individual acts, not just on the user’s expertise in general.

The online and offline modules use somewhat different input parameters due to their different roles in the system functionality. The parameters and parameter weighting are in part based on those used by Walker et al. [41] and Krahmer et al. [30] in their studies. The online parameters deal with the number of help requests (HLP), timeouts (TIM), interruptions (INT), repetitions of system acts (system act invoked), and repetition of the information content of the user request (info dump). The online module also includes a dialogue problem detector that monitors speech recognizer errors and contributes to assigning a value to the system’s initiative variable. The offline parameters, on the other hand, include the number of help requests (OHLP), timeouts (OTIM), and speech recognition problems (misunderstandings, OMIS), and the module weighs them according to their frequency and recency in order to calculate the default control value. In addition, the module tracks the number of sessions that the user has had with the system (NSES) and then calculates the user’s general expertise level (GEX). From GEX and the dialogue act specific experience (DASE), which is based on the number of sessions during which the particular system dialogue act has been invoked, the default dialogue act specific explicitness (DDASEX) is calculated. The default value represents the user’s assumed skill level as it has been developed through the user’s particular interactions with the system so far. The details of the parameters and their value calculation can be found in Jokinen and Kanto [20].

3.2 Dialogue act specific explicitness and dialogue initiative

Each systems utterance type has three different surface realizations corresponding to the different DASEX values. The higher the value, the more explicit the presentation of system dialogue act, i.e., the more additional information the surface realization will include. The value range is: 1 = taciturn, 2 = normal, 3 = explicit. The system responses are designed based on the core of the highest expertise level responses, so that the user perceives information in the system responses as decreasing: when the user’s expertise rises, the system seems to remove information elements that have become unnecessary, without touching the core. This should contribute to a feeling of consistency and dependability. Explicitness also depends on the previous dialogue topic and new information that is presented to the partner [24]. This kind of context-dependent explicitness was not included in AthosMail, however, since the dialogue model did not keep any specific record of the information presented to the user. On the other hand, Paris [37] argued that the user’s expertise level does not affect only the amount but also the kind of information to be given to the user. This is not explicitly modelled in AthosMail either, but a preliminary attempt in this direction is performed by dividing information elements into “options” and “extra options”, the former including message properties and the latter available system commands.

As for the INIT values, the division of control levels follows the categorization of four initiative modes introduced in Smith and Hipp [39]. The higher is the initiative value, the more the system controls the dialogue. The value range is: 1 = passive (the user has complete dialogue control and the system does not recommend subgoals), 2 = declarative (the system may provide relevant unsolicited information), 3 = guiding (the system makes suggestions about subgoals but is willing to change the course of the dialogue according to the stated user preferences; Smith and Hipp call this suggestive), and 4 = directive (the system has full control and recommends the next subgoal needed to proceed with the task).

Since the effects of explicitness and initiative on the surface generation of utterances often overlap, the effect of the two variables is integrated into a unified Cooperativity Model. Table 1 presents a summary of the utterance level integration. The values for “options” (none, short, long) refer to the amount of unsolicited relevant information that can be given to the users to guide them through a particular dialogue situation, ranging from no additional information to a full-length explanation of what to do. The relevant pieces of additional information must be determined on the basis of the application and domain knowledge; in AthosMail, this kind of information includes message properties (title, sender, time) and instructions about how to speak to the system. The “extra option” values (yes/no) refer to the system providing the user with explicit information of the available commands (giving a full listing of the available commands or related information on how to obtain the list). The “prompts” (open, question, chunk-by-chunk) refer to the system’s possibility to take the initiative. An open prompt refers to the system giving helpful instructions, but leaving the user to decide what to do next. It can be phrased in a passive way by giving the user only minimum information (passive initiative) or in a more active way by giving a helpful suggestion to do something (declarative initiative). A question prompt refers to the system initiative, whereby the system explicitly asks for the necessary information or requests the user to do a necessary action, while the chunk-by-chunk prompt is a sequence of explicit instructions to guide the user through the application task. It must be noticed, however, that in AthosMail the task structure is flat and the user is in command of initiating and completing the application-related tasks (navigation in the mailbox, reading and deleting messages). Thus the application does not require the system to take a guiding or directing initiative to instruct the user through the tasks, as the tasks are fairly straight-
forward and independent of each other. The initiative levels are included in the design for the sake of completeness, so as to allow the UM to be ported also to dialogue systems that deal with more complicated task structures.

In the current implementation of AthosMail, the DASEX recommendation value is used for choosing between different surface realizations. The Finnish version of AthosMail selects an utterance from the predefined set of utterance templates, while the English version uses the original idea of particular generation codes which the planner assembles into a recommended surface form. The following examples, extended from Jokinen et al. [22] and Jokinen and Kanto [20], show the initiative and explicitness value combinations on the surface generation of utterances. Extended from [22]

<table>
<thead>
<tr>
<th>Initiative</th>
<th>Explicitness</th>
<th>(1) Taciturn</th>
<th>(2) Normal</th>
<th>(2) Explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>options: none</td>
<td>extra options: no</td>
<td>prompts: open</td>
<td>extra options: no</td>
</tr>
<tr>
<td></td>
<td>extra options: yes</td>
<td>prompts: open</td>
<td>extra options: yes</td>
<td>prompts: open</td>
</tr>
<tr>
<td>Declarative</td>
<td>options: none</td>
<td>extra options: yes</td>
<td>prompts: open</td>
<td>extra options: yes</td>
</tr>
<tr>
<td>Guiding</td>
<td>options: none</td>
<td>extra options: yes</td>
<td>prompts: short</td>
<td>extra options: yes</td>
</tr>
</tbody>
</table>

**Example 1a:** A speech recognition error (= no interpretation) with passive control of the dialogue:

DASEX = 3:: I'm sorry, I didn't understand. You must speak clearly, but do not over-articulate, and you must speak only after the beep.

DASEX = 2:: I didn't understand. You must speak clearly, and speak only after the beep.

DASEX = 1:: I didn't understand.

**Example 1b:** A speech recognition error (= no interpretation) with declarative control of the dialogue:

DASEX = 3:: I'm sorry, I didn't understand. Please speak clearly, but do not over-articulate, and speak only after the beep. To hear examples of what you can say to the system, please say 'what now'.

DASEX = 2:: I didn't understand. Please speak clearly, and speak only after the beep. To hear examples of what you can say to the system, say 'what now'.

DASEX = 1:: I didn't understand. To hear examples of what you can say to the system, say 'what now'.

**Example 2a:** Basic information about a message chosen from a listing of messages from a particular sender, with the system having passive control of the dialogue:

DASEX = 3:: First message, about “reply: sample file”. You can say ‘read’, if you want to hear the message. You can say ‘tell me more’, if you want to hear a summary and the send date and length of the message.

DASEX = 2:: First message, about “reply: sample file”. You can say ‘tell me more’, if you want more details.

DASEX = 1:: First message, about “reply: sample file”.

**Example 2b:** Basic information about a message chosen from a listing of messages from a particular sender, with the system having declarative control of the dialogue:

DASEX = 3:: First message, about “reply: sample file”. Please say ‘read’, if you want to hear the message, and say ‘tell me more’, if you want to hear a summary and the send date and length of the message. To hear examples of what you can say to the system, please say ‘what now’.

DASEX = 2:: First message, about “reply: sample file”. Say ‘tell me more’, if you want more details. To hear examples of what you can say to the system, say ‘what now’.

DASEX = 1:: First message, about “reply: sample file”. To hear examples of what you can say to the system, say ‘what now’.

In the examples 1a and 1b, the novice user (DASEX = 3) is given additional information about how to try avoiding speech recognition problems as well as how to get the list of possible commands. When the user’s familiarity with the system increases, the extra options are dropped, and finally also specific instructions of how to speak; the taciturn response contains only a notification that the system did not understand the user. In the declarative version, possible commands are hinted at by providing to the user information about how to get a list of commands (extra options = yes), but the user is left in charge of how to continue the dialogue: the system does not take the initiative but only publishes relevant necessary information. Note also that the politeness marker please is used for novice users but dropped in responses for the more experienced ones.
In the second example, explicitness of the system responses is reduced in a similar fashion. Notice that the message number and the title of the message are treated as minimum information that must be given to the user on any chosen message, whereas the information concerning details and the content of the message is treated as additional information that should be especially requested by the user if needed. The optional information thus contains the commands of how to get the information (“tell me more” and “read”) rather than the actual details and the content of the message.

Finally, the effect of dialogue control or initiative (INIT) on the surface realizations is exemplified in Table 2, taken from Jokinen et al. [22]. In the example, the system provides taciturn responses for an expert user. Detailed parameter calculations are substituted with verbal descriptions.

### 4 Evaluation of adaptive systems

Adaptation in spoken dialogue systems is usually aimed at dealing with communication disfluencies and facilitating more natural interaction. The assessment of adaptation as well as of its appropriateness in a given task setting is not straightforward, and several authors have pointed out the difficulties pertaining to evaluating adaptive systems. For instance, Höök [15] observed that comparing an adaptive system to a non-adaptive one is usually difficult, if not next to impossible, because of the complexity involved in adaptation (when to adapt, etc.).

Moreover, the non-adaptive system may not be optimally designed for the task: if adaptivity is meant to be an inherent part of the system, the non-adaptive system version is incomplete and thus a valid comparison between the two versions is not even possible. Oppermann [35] also points out the complexity of adaptivity and introduces an interactive design cycle whereby the system properties and adaptive criteria are gradually fine-tuned and set. Paramythis et al. [36] concluded that adaptations should not be viewed as monoliths but as processes and, for evaluation purposes, these processes must be decomposed in a systematic manner.

In the evaluation of AthosMail, the recommendations of previous research were followed. For instance, system performance was incrementally revised through expert evaluations, and several user tests were also conducted, designed to address the different aspects of adaptation. In this section, the different tests are described, and the evaluation results are discussed in Sect. 5.

#### 4.1 Evaluation methodology

The evaluation of the Cooperative Model as a whole had two aims:

1. To verify the accuracy of the model, i.e., to check how well the model corresponds to the real world characteristics it is designed to model.
2. To test the effect of the model on AthosMail usability, i.e., to test how the model affects the overall system usability.

<table>
<thead>
<tr>
<th>Utterances (U = user, S = system)</th>
<th>Relevant parameter values</th>
</tr>
</thead>
<tbody>
<tr>
<td>U: Read Tim’s message. S: There are no messages from John. You have messages from Abdul, Tim, Jaana, and Horst.</td>
<td>ASR output = “read John’s message”</td>
</tr>
<tr>
<td></td>
<td>The system misheard the senders’s name. Since there are no messages from John in the current mail folder, the system suspects an ASR error and raises the INT value by one (to declarative).</td>
</tr>
<tr>
<td>U: Read Tim’s message. S: I didn’t understand. The senders in the current folder Inbox are Abdul, Tim, Jaana, and Horst. Whose message would you like to read?</td>
<td>ASR output = “read John’s message”</td>
</tr>
<tr>
<td></td>
<td>The system hears the same thing again, which is a further clue (although tiny) that the ASR conditions are poor. The speech recognizer also reports a low confidence value. Since a problem was suspected in the previous round as well, there is enough evidence for the system to raise the dialogue control (INIT) level again, to three (guiding). The system starts to guide the dialogue more closely, and prompts with a question. The DASEX value also increases from taciturn to normal.</td>
</tr>
<tr>
<td>U: Tim’s message please. S: I didn’t understand. There are no messages from John. You can refer to a message by its number, its sender’s names, and its relative position in the message list. Do you want to choose the message by sender or by number? U: Sender S: The senders in the current folder Inbox are Abdul, Tim, Jaana, and Horst. Please say the sender’s name whose message you would like to hear. U: Tim. S: A message from Tim about What where. Hi Bat, did you tape the Beckett yesterday? I missed it, darn. Tim. End of message.</td>
<td>ASR output = “John’s message first”</td>
</tr>
<tr>
<td></td>
<td>The system can’t make sense of the ASR output. INIT is raised to the highest level (directive), and the system walks the user forwards step by step.</td>
</tr>
<tr>
<td></td>
<td>ASR output = “sender”</td>
</tr>
<tr>
<td></td>
<td>The message is played out to the user. The command is completed successfully. If no further ASR problems are detected, the INIT value starts to drop.</td>
</tr>
</tbody>
</table>

4 Evaluation of adaptive systems

Table 2 An example dialogue demonstrating the effect of control and explicitness values on utterance generation. Adapted from [22]
The verification process concerns only the parameter side of the model, not its effect on system behaviour (i.e., on the surface generation of system utterances or the system functionality). Usability testing, on the contrary, exploits the parameters as well as their effects on system behaviour as implemented in the update algorithm.

To verify the model of user expertise, the task was decomposed into three aspects, for which three separate metrics were developed and used:

1. Analysis of user errors as an indication of limited expertise in system functionality
2. A skill test as an indication of expertise in command formulation
3. Subjective evaluation by the users concerning their own judgments of their level of expertise

The main difficulty in the AthosMail evaluation was the objective classification of expertise levels. It seems reasonable to assume that novices make more expertise-related errors than experts, and it is equally clear that when taking a skill test, the performance of a novice is conceivably lower than that of an expert, assuming of course that there are matters that the users should and can learn during the tests. However, the exact location of the scale points poses problems: although the novice level is clear enough, it is more difficult to determine what is required from the user to qualify as an expert. Even an expert cannot be expected to perform flawlessly every time, so a certain amount of errors should be allowed. Moreover, expertise levels may not be associated with measurable objective performance skills, but the scale should be set for each individual separately. The number and type of user errors seems to be a matter of design rather than of scientific inquiry, and the evaluation thus entails a measure of arbitrariness related to the practical purposes of the expert level.

The location of the competent middle point of the scale is also problematic: a competent user is not a novice anymore but not yet an expert, so its position on the scale can only be set relative to the extremes. There seem to be no specific observables connected to the middle point, because it is basically only an abstract transition state in the user’s development from novice to expert. On the other hand, the middle point seems to be an expertise class that the users can most easily identify themselves with: they have some knowledge of the system and its usage but are not confident enough to say that they fully understand the system’s functionality. Furthermore, the user expertise scale is not necessarily linear, and the individual user’s expertise can vary depending on the frequency and time they have spent using the system. Although this issue was not taken explicitly into account in the evaluation setups, the actual design of the Cooperativity Model addresses it: the model calculates both online and offline DASEX values, and the offline values are meant to balance rapid learning during a single session with possible forgetting effects in between the sessions.

Concerning subjective evaluations, the user responses must be treated with care, since the users tend to over- or underestimate their skills, depending on their personalities. In the field of spoken dialogue systems, however, subjective evaluation methods can prove useful, in particular when combined with other more objective information about the situation, such as the system speed, interaction length, or the number of errors. Besides providing valuable information about the variation of the expertise levels, they are also important for usability studies. After all, the users’ own judgment of their expertise level is associated with their measurable performance and should be taken into account when interpreting the objective evaluation values and comparing interactive systems. For instance, in the PARADISE framework, user satisfaction is a function of task success and dialogue quality [42], and the weights for each factor in the model are trained with respect to the user satisfaction ratings, obtained from the user’s subjective evaluation of the dialogues via a questionnaire. Möller [34] emphasizes that the evaluation should also take into account the perceived qualities of the system that increase the user’s comfort and ease of communication, besides the observable performance metrics.

4.2 Experimental setups

Prior to usability tests, a preliminary qualitative expert evaluation was conducted. Five usability experts were asked to assess the designed system prompts from the point of view of their clarity, consistency, and appropriateness for the assumed expertise levels. This provided some important insights into the design of system utterances in relation to user expertise (see [22, 20]). Some ambiguous terms and wordings in the system utterances were then changed to more accurate and appropriate ones according to the given suggestions (e.g., politeness expressions like “I’m sorry” were removed from the utterances intended to be responses for the expert users), and the program code was modified to reflect the desired extensions accordingly (e.g., to be able to ask a list of users that have sent mail).

The main evaluation consisted of two separate user tests. In the first user test, a total of 24 users used the prototype system for four sessions spanning 2 days. The experiment took place at two different sites, involving 11 individuals with background in computer science and 13 individuals with background in linguistics. The former were, on average, more familiar with computers, but none in either site were experts of speech systems. A questionnaire concerning the users’ views of the system and their own performances was carried out after each of the four sessions. In the second user test, a group of five visually impaired people used the prototype system for six sessions spanning 3 days. A skill test was conducted once a day for each participant, while the users’
subjective evaluations of their expertise in different areas were again collected after every session, i.e., twice a day.

In both tests, a short training session with another speech application was included at the beginning of the first session in order to allow the users to familiarize themselves with synthesized speech and speech recognizers. An outline of AthosMail functionality was also handed to the users, and they were allowed to consult it when interacting with the system.

The evaluations consisted of common usability tests where the users were requested to complete a task using the system, and their impressions as well as real behaviour were logged by the system. Each task was associated with a mailbox, the content of which varied in size and message order so as to provide variation in terms of task difficulty. Each task resulted in a separate dialogue in a separate phone call, producing 96 dialogues in total. A sample task is reported below (translated from the original Finnish form):

You are waiting for feedback on your report from Raimo Tuunanen. Find out whether he has sent feedback on your report or not. If he has, find out if he accepted the report without modifications.

In all of the tasks you can mark the messages you find essential, by saying “mark the message”. By so doing, the system knows that this kind of message is important and it affects, e.g., how the messages are categorised. How did Raimo Tuunanen comment your report?

The tasks were designed to encourage the users to exploit different functionalities of the system, and the completion of tasks was both motivated and controlled by requiring the users to answer some questions about the mailbox content (such as “How did Raimo Tuunanen comment your report?”). The users were also encouraged to mark the messages, so as to provide independent, self-initiated, and interaction-based support for the completion of the task and help for the automatic analysis of the results. It was unfortunate, however, that the users did not use the marking facility of the messages, maybe because the task was too simple to warrant specific marking of the messages or because the users simply did not remember to use the function as it was not explicitly requested in the task instructions.

4.3 Skill test

The skill test was designed to serve as the second benchmark for the DASEX recommendations. It comprised of a few functionality-related quiz questions, conducted after every test session. The idea was to have the users answer the same set of quiz questions after every session (or at the end of every testing day), without giving them the right answers in between, and see if the percentage of correct answers increases. The test had two questions, each containing several parts.

The first question included trials for the user’s expertise in system functionality and correct formulation of the commands (expertise areas I and II), and was as follows:

Which of the following commands the system understands and is capable of executing, if random speech recognition errors are not taken into account?

At this point, all users already knew what speech recognizer errors were. The question was followed by a list of 14 commands, of which 6 were flawless, 4 dealt with a function that was not included in the system (such as dictation), and 5 referred to existing functionality but were incorrectly formulated (e.g., using the word “destroy” instead of “delete”). The user’s task was to mark the command as correct or leave it unmarked.

The second question was aimed at evaluating the situational knowledge (expertise area III), i.e., whether the users knew what kind of commands were at their disposal at a given dialogue situation and was as follows:

You have listened to the first message in the group and the system has just said ‘end of message’. You want to hear the body of the next message right away. Which of the following commands fulfill(s) your wish immediately, without intermediate stages?

The question was followed by six commands, of which two were precisely correct in the given situation, whereas the others were incorrect in this situation but would have been correct in others. Like in the first question, the user’s task was to mark the command as correct or leave it unmarked.

4.4 Users’ self-evaluation

The third metrics was a user questionnaire where the users were asked to provide their opinion about their level of expertise. Two versions were employed. In the first user test, the request was simply please estimate your own level of expertise in using AthosMail. This was asked after every session, and there were three levels to choose from: novice, middle level, and expert. In the second user test, a more detailed version was used and the users were asked to evaluate themselves separately for the three areas of expertise (as outlined in Sect. 2.3). The answers were given by choosing one of the five options listed under each question:

1. Do you know what functions are included in AthosMail?
   absolutely yes | mostly yes | main points only | not very well | absolutely not
2. Do you know what you can say next in different situations?
   always | usually | variably | usually not | never
3. Do you know what phrasings AthosMail is capable of understanding?
   absolutely yes | mostly yes | main points only | not very well | absolutely not

5 Results

5.1 Analysis of user errors

A categorization and manual annotation of user errors in all sessions was conducted. The analysis of the data revealed a relatively clear set of common miscommunications. Some were related to speech recognizer problems, slips of the tongue, or other basically random occurrences. Others could be traced to a lack of user expertise. Of a tentative set of six error types, the following four were retained, as they fall into three broad groups related to the areas of user expertise discussed in Sect. 2.3.

1. Problems with system functionality (user expertise I):
   a. Non-existent functionality: the user attempted to invoke a function that does not exist.
2. Problems with command formulation (user expertise II):
   a. Incorrect formulation: the user attempted to invoke a proper function but the formulation was incorrect.
3. Problems with turn-taking (user expertise III):
   a. Speaking before start signal or after end signal: the user started to speak before the start signal or after the end signal. The error was not logged if the end signal came in the middle of a user utterance.
   b. Overlapping speech: the user started to speak while the system was still speaking.

The two other error types were discarded: unnecessary repetition and incorrect reference. The former refers to cases where the user repeated the request before the system had reacted, and it was usually connected to situations where the system appeared abnormally sluggish to respond. The repetition was not thus regarded as an indication of the user’s expertise level, but rather a natural reaction to an unnaturally long response time. The latter error type refers to incorrect references to message numbers greater than the total number of messages in the folder and to senders from whom there were no messages. This was also considered too ambiguous to serve as a sign of user expertise.

The first type of user errors is related to the user expertise area I. An expert user presumably knows the limits of the system, whereas a novice may not have such a clear idea about the functionality, even if given a preliminary listing of available features (internalizing the system capabilities in practice is different from merely memorizing written instructions). The second error type, incorrect formulation, is related to the user expertise area II. Most commonly these errors were loose or otherwise malformed references to a message or a message group, probably occurring because of the limited natural language capabilities that the system possessed. The error was logged even if the incorrect command accidentally produced the desired effect.

As for the problems with turn-taking, i.e., speaking before or after the signal or speaking simultaneously with the system, they seem to indicate that the user is not yet accustomed to the limitations of speech recognizers. The errors of overlapping speech occurred in situations where the user started to speak although the system had not yet finished its turn. The current system did not have a barge-in facility, which would, of course, have been helpful. The errors specifically exclude situations where the system started to speak while the user was still speaking, as this was not an indication of the user’s expertise level.

Due to technical constraints, observing the start and end signals is imperative for the operation of the speech recognizer, and therefore of the whole system. This makes the turn-taking errors a prominent error type. The errors also show the power of human communication strategies when interacting with speech-based systems: users are not accustomed to waiting for a particular signal in order to issue their commands, but would intuitively react to the system’s spoken output as quickly as possible. Also here, a barge-in facility would have somewhat alleviated the problematic timing of the commands as encountered by the users.

Among the 24 users, the average number of errors per user is 3.3. The more computer-oriented group made consistently less errors than the linguistically oriented group (1.5 and 5.5 per user, respectively), with errors relating to non-existent functionality and overlapping speech almost totally lacking for first group. Also the length of the dialogues measured in speaker turns varied between the two test groups: the linguistically oriented users had more than 40% longer dialogues than the computer-oriented users: 64 turns and 45 turns, respectively. The reason for these differences is most likely due to the users’ different backgrounds and interests.

Distribution of user errors per session is given in Fig. 3 and the distribution per site in Fig. 4. A more detailed analysis of the various user errors is reported in Jokinen et al. [21].

The most common error is incorrect formulation (IF), indicating that the users did not really master the way they should express their commands. However, some learning may be seen in the fact that the absolute numbers decrease towards session 4. Similarly, the NF error, or the speaker using a non-existing function, shows a slight downward tendency, suggesting accumulation of the user’s familiarity with the system. The elevation in the NF numbers in session 4 may be due to
the fact that this was the last test session: the users felt more confident with the system than in the beginning of the tests, and they wanted to explore and test the limits of the system functionality.

The second most common error in absolute numbers is speaking without noticing the start or end signal (SS). It seems to be an equally common error across all the sessions. One possible explanation is that the four sessions do not allow enough learning for the users to unlearn their natural response patterns: in human–human communication one usually reacts immediately when the partner has finished, often overlapping with the partner’s turn. The requirement that the user needs to wait for a beep before being allowed to speak is counter-intuitive, and to learn this kind of turn-taking mechanism takes more time than the four short test sessions admitted. However, there were also several test users who did not exhibit this error type at all. For them, the unnatural turn-taking mechanism was obviously part of the special setup for speech-based human–computer interaction in general.

5.2 Skill test

There were five participants in the skill test experiment. However, the answers of two test users were incomplete and had to be discarded. The results of the three remaining test users are presented in Figs. 5, 6 and 7. Figure 5 shows the number of markings that the users had considered as correct command formulations and which, in fact, were correct, for each testing day. Figure 6 presents the number of instances where the user has left a correct command unmarked, i.e., thought that the command could not be used when interacting with the system, even though it was, in fact, acceptable. Finally, Fig. 7 shows the number of instances where the user has marked an incorrect command as correct.

The test users S1 and S2 gave the highest number of correct answers after the second testing day, which means that they have come to realize that some of their earlier impressions of the acceptable commands had been incorrect. The same applies to incorrect negatives
(Fig. 6), i.e., the number of instances where the user failed to recognize a correct command decreases, indicating that the user is more experienced with the system commands. The fact that the lowest point was reached after testing day 2, and again increasing after that, suggests that some unpredictable speech recognizer errors may play a role: it is conceivable that a user had used a command once and found it working, but, running into a crippling speech recognizer error on the second try, concluded that the error was in fact in the formulation of the command. As for the test user S3, it appears that this user made up his/her mind at the beginning of the eval-

![Incorrect negatives](image1)

**Fig. 6** The number of incorrect negative answers in the skill tests for each user (S1–S3)

![Incorrect positives](image2)

**Fig. 7** The number of incorrect positive answers in the skill test for each user (S1–S3)

![Userself-evaluation](image3)

**Fig. 8** The users’ average self-evaluation results per session
uation and did not change any of the answers during the next sessions. The number of incorrect positives, however, i.e., of instances where the user marked an incorrect command as correct, seems to decrease steadily for S1, whereas for S3, the number first decreases but then returns to the same level where it began.

5.3 Users’ self-evaluation results

5.3.1 First test

Figure 8 shows the development of average self-evaluation of the 13 linguistically oriented users over the four sessions in the first user experiment. In the figure, ‘novice’ has been counted as 1, ‘competent’ as 2, and ‘expert’ as 3. The users’ impression about their abilities improved from 1.3 at the end of the first session to 1.8 at the end of the fourth, on a scale from 1 to 3. One of the users considered himself to be an expert after the very last session.

5.3.2 Second test

In the second test, where the users were asked more detailed self-evaluation questions, the results do not reveal a similar trend as in the first test. Quite on the contrary, the users’ impression of their expertise in area II declined steadily towards the end of the test and, in area III, the trend was similar except for a small rise near the end.

Figures 9, 10 and 11 show the results from the second, more detailed self-evaluation questionnaire. The self-evaluations pertaining to expertise area I are presented in Fig. 9, expertise area II in Fig. 10, and expertise area III in Fig. 11. The numbers are averaged over sessions and users. Note that in Fig. 8, referring to the first experiment, the values represent the actual expertise levels and the bigger values are better, whereas in Figures 9, 10, 11, the values refer to the scale of 1–5 of the evaluation options (see Sect. 4.4), and smaller values are better representing the user’s confidence of mastering the requested skill. It is likely that the specific questions made the users more aware of their incomplete knowledge of the system and of their limited expertise in using the system, contrary to the single expertise level question in the first experiment, interpreted as an assessment of the user’s overall familiarity with the system, which the users felt had indeed increased through their participation in the tests.

Direct comparison of the user groups in the two tests is not possible. Both tests show, however, the
users’ personal variation in evaluation tasks: most users indicated that some change took place in their expertise levels, but there were also users who chose one expertise level in the first time and stuck to that for the remainder of the test. The tests also clearly demonstrated incompatibility between subjective and objective evaluations. Some users were very modest about their skills and considered themselves inexperienced while making few errors in the actual tests. Others were somewhat too sure of themselves and gave themselves high marks while still struggling with the available functionality and correct formulation of the basic commands.

6 Discussion and future studies

This article has presented the design and evaluation of the adaptive User Model component in the AthosMail e-mail application, as developed in the project DUMAS. One of the goals of the AthosMail user modelling was to facilitate more natural interaction. The experimental setup for investigating the methods and techniques for such a goal was chosen to be one where the system adapts its utterances according to the perceived user expertise level. This kind of adaptive performance is supported by the fact that the users vary in their knowledge and experience with speech-based interactive systems; they need help in learning how to use a new system, and often need additional information about the various commands of the system too.

The AthosMail User Model consists of several sub-components that support the system’s adaptation in interaction, presentation, and search. The focus has especially been on the Cooperativity Model, which deals with the user’s expertise levels and the system’s tailoring of its responses in accordance with the assumed user expertise. The Cooperativity Model presents a particular kind of adaptive model, where three levels of user expertise are distinguished and monitored on the basis of observable characteristics of the user’s behaviour. The model integrates initiative handling and explicitness by observing various parameters that describe the user’s interaction with the system and computes recommendations to the Dialogue Manager about the appropriate way to respond to the user. The model also aims at capturing a gradual consolidation over time of the user’s experience, as opposed to rapid learning in a single session.

The article also addresses problems concerning evaluation methodology and objective measures for assessing adaptation in speech-based dialogue systems. Since adaptation can only be observed by change through time, it is important that the evaluation setup takes this into account and the users have possibility to use the system for several days. In the AthosMail evaluation, the setups were designed for 2 and 3 days. This turned out to be too short for a reliable assessment of the system’s adaptation capability, although clear development in the user skills could be observed. Further development of the evaluation methods for adaptive systems would thus benefit from user tests with longer duration.

Another point to notice is that the evaluation tasks were designed to be as natural and plausible as possible from the point of view of system usability. This meant that their coverage of the different areas of user expertise (see Sect. 2.3) was less controlled than would have been desirable: in commonly used tasks, the dividing lines between different areas of user expertise become blurred. Concerning the user expertise areas, an ideal task would be one where the users need particular functions at particular times and in particular measures, so that their expertise in the different areas could be easily monitored. Moreover, the users may grow tired of a single test lasting too long and/or of a repetition of similar tests over several days, and this will conceivably affect the results. The evaluation should thus be designed so as to maintain the users’ interest in the task and application in a natural way over time. For instance, the users may be involved in a story or a group scenario where several test users assume fictional characters in the story and use the system to communicate with each other, as was used in the project data collection phase [27]. The reason why a group scenario was not used in the final evaluation of
It is reasonable to believe that the number of expertise-related user errors would decrease as the user’s experience accumulates. The number of user errors can thus be used as an indication of the user’s adaptation to the system and, indirectly also as an independent metric for assessing and comparing the expertise level recommendations produced by the Cooperation Model. Some encouraging results could be found in AthosMail in this respect, although further refinement would be needed. For instance, the most common error type is incorrect formulation, indicating that the users did not really master the way they should express their commands. Some learning in this respect is evident in the fact that the absolute numbers decrease towards the last session. Similarly, the number of errors for using non-existing function shows a slight downward tendency, suggesting accumulation of the user’s familiarity with the system. However, the persistence of user errors even at the later sessions in both user tests indicates that the four sessions on two consecutive days did not leave the users enough time for learning everything they needed to learn.

The amount of user errors provides some indication of the user’s adaptation but there is also a caveat in their straightforward interpretation. As the users gain confidence in their capability to interact with the system through repeated sessions, some users also felt encouraged to start exploring the system limits, and there was an increase in the number of errors they made. These errors are not caused by a sudden degeneration of the user’s expertise level to that of a novice, but by the users’ intentional experimenting with such aspects of the system that they already know may not belong to the core functionality but still produce unexpected interesting responses. Evaluation of the user’s expertise would thus require some control over the user’s own experiments with the system or, alternatively, including educated exploration as a characterization of the expert level user. In AthosMail, the user’s explorations with the system were not a serious problem since the application was rather simple and its limits soon recognized. With more complex systems, however, and with adventurous users, the number of user errors should be accompanied with a higher-level intention analysis of the user’s behaviour. Controlled exploration can be seen as a sign of mastering the system’s basic functionality and as a means by which an expert user learns creative use of the system. This is an essential part of becoming an expert [11], and the system should thus distinguish between errors that originate from the expert user’s intentional deviations from the already mastered correct actions, and errors that stem from the novice user’s ignorance of the appropriate actions. How to best take this kind of expertise behaviour into consideration in user models and adaptive systems is an interesting but challenging research question.

Considering the verification of the Cooperation Model itself, we must remember that there are two separate components: the offline and online components. They represent two independent factors in the usability tests but are difficult to evaluate independently. A usability test with just one of the two components does not seem reasonable, since the overall system functionality would differ significantly from the actual system that is supposed to be tested. A verification test would rely on user reports; and the users might be able to tell whether the default DASEX at the beginning of a session seemed correct, but discerning this from the online component’s workings would probably be impossible. Another important point is the division of the amount and kind of information to be provided to the user, discussed in Sect. 3.2. It will be interesting to reconcile these views in a more general kind of user expertise modelling as well as in the further development of the evaluation methodology.

Finally, although the model is implemented in a rule-based way, thus facing the known problem of rule-based thinking (the number of factors influencing adaptation is too large to enumerate in rules) and apparently forcing counterfeit processing of adaptation (learning takes place gradually, without clear-cut and mutually exclusive expertise levels), the user evaluation shows that the users are generally impressed by the system and find its functionality adequate. Since in practical speech-based interaction systems, the goal of adaptive user modelling is to enable more usable and user-friendly interfaces rather than to investigate adaptation as such, correct functioning of the application is preferable to sophisticated modelling. On the other hand, practical prototyping and evaluation provide researchers with experimental platforms to test insights on how adaptation takes place. This contributes to wider views of the possibilities that adaptation can offer for natural, intuitive human–computer interaction.

In general, the AthosMail Cooperativity Model and the method of observing the user’s skill levels may prove useful when assessing the user’s expertise for the purposes of benchmarking systems related to adaptivity. Previous research on user modelling, concerning various speech-based applications and dialogue systems, has already shown the importance of adaptation in making the human–computer interaction more efficient and enjoyable. Although there are several controversial points concerning the level at which adaptation should take place, who is in control of adaptation, how does adaptation affect system usability, etc., the general tendency in the system design community seems to go towards the computer-as-an-agent metaphor where the system’s ability to adapt to different users has become relevant. The problems dealing with the evaluation of adaptive systems are bound to proliferate when more complex systems are to be considered, but it is still reasonable to conclude that a better understanding of
how adaptation works in interactive situations provides a partial answer to many usability problems, too. It is true that automatic adaptation due to bad design can be confusing and thus decrease user satisfaction and service quality, with increased feeling of lack of control of the system. However, the opposite can also happen: users take it for granted that natural easy-to-use applications take their habits and personal qualities into account. In fact, most users already use various software packages and search engines without any particular interest in enhancing their feeling of control over the actual working of the software: what is crucial is the interface that allows them to complete the designated task efficiently. In these cases, a solid model of the user’s expertise levels and the system’s adaptation capability may prove valuable. After all, we want the computers to make life easier for us.

Acknowledgements The research was carried out within the EU project DUMAS (Dynamic Universal Mobility for Adaptive Speech Interfaces), IST-2000–29452 (http://www.sics.se/dumas), and the financial support of the EU is gratefully acknowledged. The author would also like to express her thanks to all the project participants for their cooperation and inspiring discussions concerning the AthosMail system and her research group at the University of Art and Design for carrying out the implementation and experiments with the user model components. The author would especially like to thank Kari Kanto for his contributions to the design and evaluation of the Cooperativity Model.

References