An architecture and applications for speech-based accessibility systems

M. Turunen
J. Hakulinen
K.-J. Räihä
E.-P. Salonen
A. Kainulainen
P. Prusi

Speech can be an efficient and natural means for communication between humans and computers. The development of speech applications requires techniques, methodology, and development tools capable of flexible and adaptive interaction, taking into account the needs of different users and different environments. In this paper, we discuss how the needs of different user groups can be supported by using novel architectural solutions. We present the Jaspis speech application architecture, which introduces a new paradigm for adaptive applications and has been released as open-source software to assist in practical application development. To illustrate how the architecture supports adaptive interaction and accessibility, we present several applications that are based on the Jaspis architecture, including multilingual e-mail systems, timetable systems, and guidance systems.

INTRODUCTION

Accessibility is often mentioned as one of the major motivations for the development of speech applications. For example, there has been much work in speech-based and auditory interfaces to allow visually impaired users to access existing graphical interfaces.[1] In general, multiple modalities have been used to make human-computer interaction accessible for people with disabilities.[2]

In the ideal case, applications should take the needs of different users and usage conditions into account in the first place: interaction should be adapted to each usage situation. The goal of this approach is universal access to services. Current application development architectures tend to lack the flexibility necessary to adapt to a variety of users and usage conditions. In this paper, we present a system architecture capable of supporting the development of accessible interactive systems.

SPEECH SYSTEM ARCHITECTURES

A software architecture defines applications in terms of components and interactions among them.[3] A framework suitable for practical speech applications must provide components for a variety of application requirements, including dialog management, speech recognition, and natural language process-

ing. High-level components (modules) usually contain multiple subcomponents having their own

■ Applications should take the needs of different users and usage conditions into account ■

internal organization and relations. The challenge lies in finding ways for all of these components to be organized and selecting the functionality that the underlying system architecture should offer. This includes the development of principles for interaction and management of information flow among the system components.

When we consider speech architectures from the human-computer interaction point of view, an interesting issue is how the system can be made to support more intuitive and natural speech-based interaction to allow universal access to services. The needs of different user groups vary considerably. Natural interaction requires flexible interaction models supported by the system architecture.[4]

In most speech systems, components are structured in a pipeline fashion; that is, they are executed in a sequential order, although this kind of "pipes-and-filters" model is considered suboptimal for interactive systems.[3] In order to facilitate the development of advanced speech applications, we need more advanced techniques, models, methodology, and tools. In particular, we need system frameworks that support the requirements mentioned because, as with all technical variation, the key issue is the integration of components into a working system.[5]

Earlier work in advanced speech system architectures includes client-server approaches and systems based on agent architectures. Probably the best-known speech-specific client-server architecture is Galaxy-II.[6] The Open Agent Architecture[7] is a general agent architecture that has been used in the construction of many speech applications. These architectures offer the necessary infrastructure components for applications, but they do not support human-computer interaction tasks or adaptation in any particular manner.

A great deal of work has been done in the field of dialog management. Three particularly interesting recent examples include the agenda-based dialog management architecture[8] and its RavenClaw extension,[9] Queen's Communicator,[10] and SesaME.[11] The purpose of these approaches is not to provide a complete speech architecture but instead, a model for dialog management.

The Jaspis architecture addresses many of the same features as the dialog management architectures mentioned but aims for general applicability in a variety of task settings, one of which is dialog management. It introduces a new paradigm for interactive systems that focuses on speech-based applications. In our previous papers[12] we have presented technical and functional aspects of the architecture. In this paper, the architectural principles—and in particular their novel support for adaptive interaction—are described in the context of human-computer interaction. We demonstrate how it is possible to construct highly adaptive systems suitable for different user groups and to support accessibility by using the Jaspis architecture and its principles.

The remainder of the paper is organized as follows. In the next section we introduce architectural foundations for adaptive human-computer interaction. The Jaspis architecture and its novel interaction paradigm based on *agents, managers* and *evaluators* is introduced with examples. After the architecture presentation, we introduce several Jaspis-based applications for various domains. Examples of multilingual e-mail systems, timetable services, and pervasive computing applications are given. Special focus is given to interaction-level issues, such as error management, help, and guidance. In the next section we report experiences and results from user-centered design, "Wizard of Oz"** studies, and evaluations of Jaspis applications. Accessibility issues, such as those raised in design sessions with users who are visually impaired, are discussed. The paper closes with conclusions and discussion.

JASPIS ARCHITECTURE

Jaspis is a general speech-application architecture designed for the challenges of advanced speech applications, especially adaptive and multilingual speech applications. It provides support for human-computer interaction tasks, such as error handling, "Wizard of Oz" studies (i.e., those in which some parts of the system are simulated with a human

operator), and corpora collection. While Jaspis is a general conceptual architecture, it is also a concrete framework which provides components for application development. In this section, we present the principles of the Jaspis architecture, focusing on human-computer interaction tasks, in particular on dialog management, output generation, and input management tasks. In addition, application development aspects are briefly discussed.

Architecture requirements

In order to support more flexible interaction, we have identified requirements for speech application architectures. First, speech applications need *adaptive interaction methods* in all system modules. For example, outputs and inputs should be tailored to the language of the users, and dialog management should adapt to the situation at hand. Second, systems should be *modular* because modular components support reusability, are easy to maintain and extend, can be distributed efficiently, and make adaptivity easier to achieve. The other requirements concern *collaborative and iterative* application design and development, the need for an *extensible and practical* infrastructure for application development, and support for standards. These principles are motivated by the technology, human-computer interaction, and application development viewpoints, all of which should be taken into account. For a more comprehensive description of the Jaspis architecture requirements see Reference 13.

Architecture overview

In order to support the architectural requirements mentioned, the Jaspis architecture uses a *modular and distributed system structure, an adaptive interaction coordination model* and *a shared system context*. These form the basic infrastructure on which other features and components of the system are based.

Figure 1 presents a typical Jaspis-based system setup. The top-level structure of the system is based on *managers*, which are connected to the *central manager* with a star topology. Communication between components is organized according to the client-server paradigm. Local subsystems are located inside the system modules. The interaction coordination model of the Jaspis architecture is based on the agents-managers-evaluators paradigm. *Agents* are interaction components which implement different interaction techniques, such as speech output presentations and dialog decisions. *Evaluators* are used to evaluate different aspects of the agents, in order to determine how suitable the agents are for different tasks. *Managers* are used to coordinate these components. All information in Jaspis-based systems is stored in the shared information storage. All components of the system may access the content of the information storage by using the information manager. These are the key features enabling architecture-level adaptation.

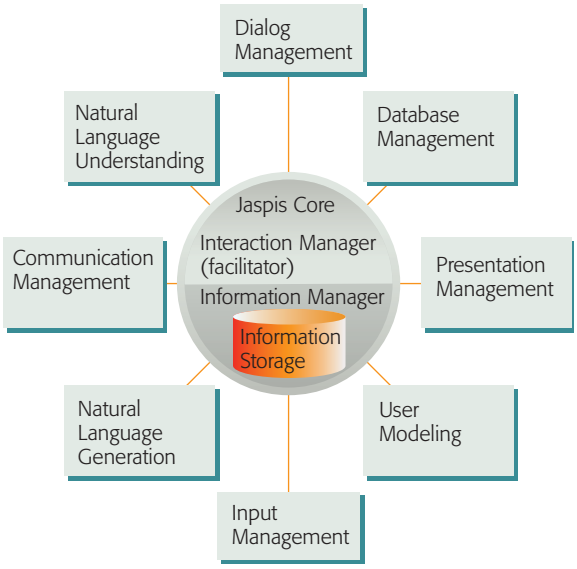

Figure 1
Overall system structure for Jaspis applications

Shared information management

Information management is a crucial element of adaptive, modular, and distributed applications. The repository approach (i.e., using "blackboards" and databases) provides several advantages for adaptive and distributed applications. The term "blackboards," in the context of speech applications, refers to shared information resources, or specifically, a shared knowledge base. Most importantly, the repository approach allows the use of shared information by all system components. The main drawback of this approach is the lack of control.[7] In Jaspis, the coordination and control are performed by a separate component (the interaction manager) to achieve architecture-level coordination.

The Jaspis information management architecture consists of four layers. In this way, the actual

storage (the *information storage*), the application interface (the *information manager*), and the communication interfaces (the *information access protocol* and *communication protocols*) are separated to maximize flexibility. In this section we focus on information storing and application layers, omitting the communication layers.

The information storage holds all the shared system data, that is, the shared system context. The Jaspis architecture assumes that individual components do not store any high-level information inside them, but instead use the information storage for that purpose. This makes the interaction components *stateless*, and the system is able to adapt to each interaction by choosing proper components for each situation. To make this possible, the system assumes that every component updates its knowledge from the information storage when activated and writes modified information back to the information storage when deactivated. In the ideal case, the shared data should be represented at a conceptual level such that it can be used by other components as well. This is one of the main features used to adapt the systems for different users.

The reference implementation of the information storage uses XML (Extensible Markup Language) for its internal information representation. The content or the structure of information inside the information storage is not defined by the system architecture because this is specific to the particular application and domain. The definition of the shared knowledge is an important phase of the application development process.

The programming interface for the information storage is straightforward: it takes XML requests and produces XML results. The information storage offers only the minimal set of operations needed to manipulate its contents. In addition to the shared information storage, direct information exchange between certain I/O components is supported for efficiency reasons. Most notably, the raw audio streams should be passed between components in a cost-efficient way to minimize system overhead and delays.

The information manager provides an application interface to the information storage. It uses the information access protocol to access the information storage and provides a programming interface for other components to access the shared system context. For example, system inputs and outputs may be modified by their own set of methods. Application developers may write new, application-specific methods when needed.

Flexible interaction management

The interaction management model of Jaspis is focused on the key design principles of the architecture: adaptivity and modularity. Interaction management in this context means both the overall coordination of system components and the coordination of those components that implement interaction techniques to be used in human-computer interaction tasks. In practice, this means input, output, and dialog management components in their various forms.

Interaction techniques are implemented by agents, which are software components specialized for certain tasks. Evaluators are used to make selections among different agents. Managers are used to coordinate agents and evaluators. Components specialized for related tasks are organized into modules. An overview of the interaction management model is presented in ***Figure 2***.

As illustrated in the figure, each system module contains one local manager and several agents and evaluators. It is up to the local managers to decide which agents are used in different situations. Instead of centralizing this decision (by assigning it to the managers), evaluators are used to evaluate agents and their suitability for different tasks. Thus, there is no central component which makes these decisions. This makes it possible to construct highly adaptive and modular systems because all functionality is divided into specialized components that have no predefined execution order and relations among them. The principles governing how managers, agents, and evaluators are used are presented next in more detail.

Managers: coordination

The interaction manager is a central component in Jaspis-based systems. It manages other components and is responsible for the overall coordination of the interaction. The interaction manager is similar to some central components found in other speech architectures. Such components include the hub in the Galaxy-II architecture[6] and the Facilitator in the Open Agent Architecture.[7] Similar components can

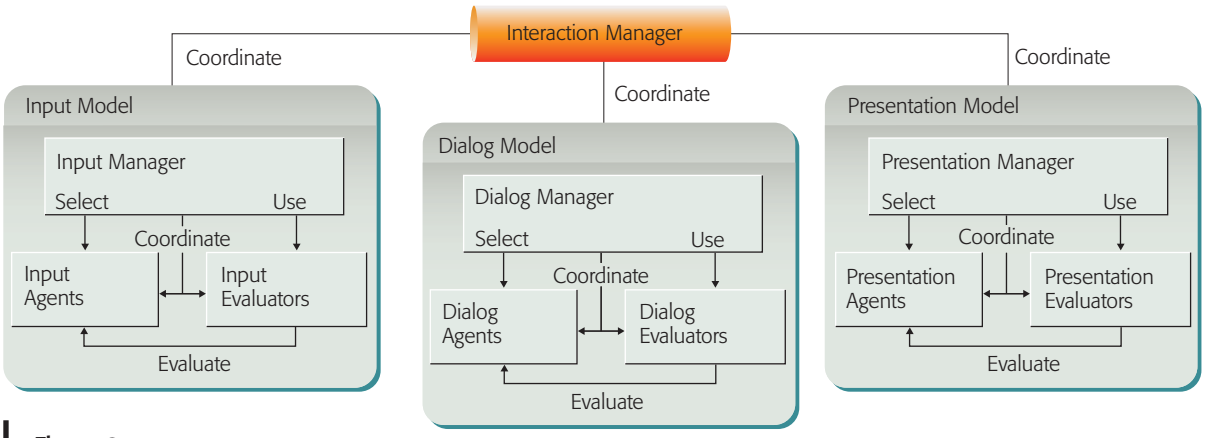

Figure 2
General interaction management architecture

also be found in other distributed architectures. The main differences include the division of labor between the information manager and the interaction manager and the fact that Jaspis is a layered (hierarchical) architecture.

Coordination is one of the main issues in agent-based systems.[7,14] In order to coordinate interaction in a flexible way, it is better to use decentralized control logic; managers and their components may be more autonomous, and the system may adapt to interaction when needed. This is crucial in applications that are targeted for various user groups, and in particular, when applications need to be modified afterwards, as is the case in many accessibility systems which need to be customized for different users. In Jaspis, local managers are quite independent. The interaction manager does not know the internal structure of modules, nor does it coordinate interaction inside modules. Layered systems are easier to maintain, and they can be more efficient. This is because local managers need to coordinate only their own components. Evaluations in particular are more efficient when only those agents that belong to the same module are evaluated.

Jaspis uses a "round robin" interaction management approach, in which one manager is active at a time. The interaction manager enables the managers to take turns, based on a prioritized list. When the option to become active is offered to a manager, it will check if it is able to handle the situation by using its own reasoning algorithm or by consulting evaluators.

In order to support application development, the architecture includes a general manager class that can be customized for different purposes. The general manager asks evaluators to assign a score for each agent in the module. If there is an agent that receives a score above zero, the manager becomes active and activates the agent with the highest score. In the basic algorithm, managers multiply evaluation scores, but more elaborate methods may be used as well.[13]

In addition to the default behavior presented, local managers are able to extend this functionality, or in general, implement their own functionality. For example, some of the Jaspis reference implementation managers (e.g., the presentation manager and the communication manager) have extended the basic functionality. This is because they have different needs when using agents. The agents–managers–evaluators paradigm is used in all system modules, but in various ways.

Agents: interaction tasks

Agents in the Jaspis architecture are software components that handle various interaction tasks. Agents are often used to handle tasks that are fairly complex. For example, in the WITAS (Wallenberg Laboratory for Research on Information Technology and Autonomous Systems) multimodal dialog system, there are six agents.[15] Jaspis has been designed with *compact agents* in mind. Because agents are meant to handle single, well-defined tasks, they are fairly small, and typical systems contain many of them. For example, in the AthosMail application (presented in the following section), there are over

50 basic dialog agents alone (e.g., agents for reading e-mail messages, navigating among folders and messages, and providing context-sensitive help).

Typical agents in Jaspis-based systems are *specialized for certain tasks*, which may be domain-specific or general interaction management tasks. Agents may implement interaction techniques for tasks such as generation of welcome prompts or handling of speech inputs. Some of the agents may be application-independent, while some may be closely tied to the application domain. Application-independent behavior is usually preferred so that components may be used across applications. Usually, application-specific components are encapsulated in their own agents, and inheritance is used to maximize runtime efficiency.[16–18]

Specialized agents make it possible to implement modular, reusable, and extensible interaction components that are easy to implement and maintain. For example, we have constructed general interaction techniques, such as error correction methods, to take care of error situations in speech applications.[19] Another example is that of tutoring agents, which provide guidance for the users in a real context.[20]

In addition to the collaborative approach, where different agents implement different functionality, multiple agents may be specialized for the same tasks with varying behavior. In other words, agents may provide alternative solutions to the same problems. Based on our experiences, we have concluded that different user groups prefer different types of interaction (described in the following sections in more detail). Using modular agents, we can support different interaction strategies within an application and adapt the interaction dynamically to the user and the situation. For example, we may have different agents to take care of speech outputs for various user groups and languages. We have constructed specialized presentation agents for different languages and with varying verbosity levels without modifying the original agents.

Jaspis agents are *stateless*, as stated previously, but they are *autonomous* in the sense that they know when they are able to act. Each agent has a self-evaluation method that checks the current situation (for example, the dialog context) and gives a local estimate that defines how well it can handle the situation at hand. The self-evaluation method is only one part of the agent selection process. All dependencies among agents, including the overall suitability of competing agents for different situations, are modeled using evaluators.

Evaluators: system-level adaptation

Evaluators are the key concept in making applications adaptive and interaction flexible. They determine which agents should be selected for different interaction tasks. It is up to evaluators to compare different agents and their capabilities and assign an evaluation score for every agent in any given situation. Like agents, evaluators are specialized for certain tasks. In practice, this means that different evaluators evaluate different aspects of agents from different system viewpoints. For example, an evaluator may use the dialog history to determine which dialog strategy should be used (i.e., which kind of dialog agent should be selected), and another evaluator may use the user model to select verbose or brief system output formats.

When Jaspis evaluators are employed at the human-computer interaction level, they can monitor interactions and give guidance as to how an interaction is progressing and how it should continue. One example is dialog strategy: if the interaction is not going smoothly when the dialog strategy is based on user initiative, a specialized dialog evaluator may give better scores for dialog agents based on system initiative. Similarly, if the user has problems, presentation evaluators may prefer presentation agents that use more detailed and helpful prompts. Concrete examples are presented in the following sections.

Evaluation is applied in the system modules. When one of the agents inside a module is to be selected, each evaluator in the module assigns a score to every agent in the module. These scores are then multiplied by the local manager. The manager assigns the final score, a suitability factor, to every agent (see Reference 13 for a more comprehensive description). It is noteworthy that there is no single evaluator, nor any single component in general, which selects agents for each situation; instead, the selection is always both dynamic and distributed. This makes it possible to keep the program control and interaction flow highly dynamic and adaptive on the architectural level. This is a major improve-

ment to the "black box" type of adaptivity that most systems offer (see, for example, Reference 21).

Evaluators may use different strategies and techniques to evaluate agents. They may use information such as the current dialog context, user model, and interaction history. Presentation evaluators, for example, may use dialog history to determine (e.g., using heuristic rules) which confirmation agent is most suitable for the current dialog state. Other evaluators may use methods from machine learning, for example, to evaluate agents and their usefulness in a given context.

Evaluators know the properties of the agents, which are expressed in the form of attributes. Attributes are presented in XML-based configuration files and can be configured with domain-specific parameters. In more complex systems, the parameters can be learned automatically. To be generic, the architecture does not force application developers to follow any predefined approach to this process. In conventional dialog systems, all dialog decisions are made by the dialog manager. Jaspis evaluators allow more flexible interaction management to take place because the reasoning is both distributed and dynamic. Evaluators may nevertheless be used for more global evaluation. This can be done with special evaluators to monitor interactions and to favor consistent interaction. For example, the evaluators may favor agents which use a language similar to that used in previous dialogs for inputs and outputs and which follow the same dialog strategy.

In the AthosMail system (to be presented in the following sections), the presentation agents are evaluated with five different evaluators. Each evaluator uses specific information to assign a score to each agent. The presentation evaluators use the following information: (1) dialog data, (2) language of the output, (3) user expertise, (4) characteristics of the mailbox, and (5) self-evaluation results of the agents. Evaluation scores from these five evaluators are combined, and the agent with the highest overall score is selected to generate the output.

We next present how the general agents–managers–evaluators paradigm is applied to human-computer interaction tasks, including dialog, presentation, and input management tasks.

Human-computer interaction management

The Jaspis architecture provides several general interaction modules that can be customized for specific tasks and applications. These are input, dialog, presentation, and communication modules. These modules contain the components used to implement interaction techniques for human-computer interaction tasks. All of these modules are based on the general agents–managers–evaluators paradigm but use it in slightly different ways. Developers are free to introduce additional modules when needed (e.g., for user modeling or handling modality-specific issues).

■ Natural interaction requires flexible interaction models supported by the system architecture ■

Dialog and error management

The task of the dialog management module is to update the dialog state, that is, to move the dialog from one state to another. This abstract task representation may be modeled in various ways and should not be restricted to any single dialog control model, such as the finite-state model or the form-based model.

Like other agents, dialog agents are specialized for different dialog tasks or provide alternative solutions for the same dialog tasks. Unlike many other agents, such as presentation agents and input agents, in most cases there is usually only one dialog agent active at any given time. The dialog control model is one way to categorize dialog agents. For example, in the *state-based* dialog control model, each dialog state can correspond to one dialog agent, whereas in the *concept-based* model, each concept may form its own agent. In the *form-based* dialog model, every form field may correspond with a dialog agent. In addition, it is possible to use multiple dialog control models, such as those based on state machines and forms, in the same application. The combination of different approaches is especially useful when subdialogs are implemented with different dialog control models.[17]

Dialog evaluators may be specialized for aspects such as general dialog-level issues, functional

aspects of agents, domain-specific issues, or user preferences. For example, one dialog evaluator monitors the dialog flow and checks that the

■ Information management is a crucial element of adaptive, modular, and distributed applications ■

interface is consistent, ensuring that the dialog management strategy is not changed for every dialog instance. Another evaluator may perform the mapping of dialog tasks to the functionality of agents, while other, more specialized evaluators monitor error situations and the user's need for help and guidance.

Dialog agents and evaluators can be used to implement different dialog management strategies to adapt the dialog to the user. Different dialog management strategies, such as the system-initiative approach and the mixed-initiative approach, have different benefits and drawbacks,[22] which makes it useful to use a multiplicity of them in an adaptive way. For example, we have used tutoring agents to provide system-initiative features to the user-initiative interface.[20] In the next section we present a concrete example of such usage in the context of a multilingual e-mail application.

Similarly, alternative solutions for the same dialog tasks can be implemented by using modular agents and evaluators. Alternative dialog agents can be used to provide adaptivity to the interface in special situations. One example is error management. Errors are unavoidable in speech-based applications, and their proper handling is a challenging task. In the agent-based error management model,[19] the task is divided into seven phases, and the actual error correction techniques are implemented using the Jaspis agents and evaluators. For example, there can be agents and evaluators for error detection (e.g., monitoring of speech-recognition confidence scores and dialog history), actual error correction techniques (such as selections from a list and different kinds of confirmations), and decisions regarding how to continue after the error has been corrected.

The agent-based approach for dialog management is demonstrated in the following dialog example (translated from Finnish) from the Busman timetable system.

AGENT U_1: Welcome to Busman, the Tampere bus information service! How may I help you?
USER: When does the next bus leave from Central Square to Her ehhh correction from Central Square to Hervanta?
AGENT E_1: Sorry, I didn't recognize that. Please speak clearly.
USER: What now?
AGENT H_1: Busman understands questions such as "When does the next bus to Hervanta depart?", ... [help continues]
USER: I'm leaving from Central Square, and I'm going to Hervanta.
AGENT E_2: Sorry, I still didn't recognize that.
AGENT S_1: Please tell me where you are leaving from.
USER: From Central Square.
AGENT G_1: What is your destination?
USER: Hervanta.
AGENT G_2: From Central Square to Hervanta use lines 13, 20, 23 and 30.
AGENT S_2: Do you want to hear timetables?
USER: Yes.
AGENT G_3: From Central Square to Hervanta on weekdays at 11:15 bus in line 23...

In this example, five types of dialog agents are present: user-initiative dialog agents (U_1), system-initiative dialog agents (S_1, S_2), error management agents (E_1, E_2), help agents (H_1) and general dialog agents (G_1, G_2, G_3). All of these agents operate on the basis of the shared form and concept definitions that are stored in the information storage. In the beginning, the dialog evaluators select a user-initiative agent (U_1). Recognition errors occur due to the complex recognition grammars, and the first error management agent (E_1) is selected. After that, the user requests help by using a universal command (its usage will be presented in the following sections), and a help agent (H_1) is selected. After the second recognition error, another error management agent (E_2) is chosen, and the dialog evaluators select a system-initiative dialog agent (S_1) to continue the dialog because of the recent errors (the recognition grammars are changed at the same time to more compact ones by the input agents). The recognizer is performing better, and two generic dialog agents (G_1, G_2) are selected to

complete the task. Finally, the dialog evaluators select a system-initiative dialog agent (S_2) again, and a generic agent (G_3) provides the results. In the Busman application every dialog agent has a floating-point value (*initiative attribute*) to determine its suitability for system-initiative and user-initiative dialog. This will be presented in the section "Bus timetable systems."

Presentation management

The presentation management module is responsible for generating system outputs suitable for the current dialog state. Usually, this is done on the basis of conceptual messages that the dialog management module produces. Presentation agents produce a representation of speech outputs, which are synthesized by the communication management module. They perform natural language generation, add prosodic information to sentences, decide which (synthesized) voices should be used, and add other modality-specific features into outputs.

Real-world metaphors for presentation agents are actors, who perform their roles as efficiently as they can. Different agents have different capabilities, and they are chosen for different roles. Unlike the real world, the choice of agents can be tailored for each listener, that is, for each user. A real-world metaphor for presentation evaluators is a casting agency, which tries to find the most suitable actors to perform the roles in a play. Similarly, presentation evaluators try to find which presentation agents are the most suitable for the presentation of speech outputs. Several evaluators can be used in this process: in the e-mail domain, we have used five evaluators to choose among the presentation agents.

Multiple presentation agents may be active at a time. First, there may be multiple output requests in information storage, and all of these will be processed. Second, presentation agents may produce several outputs from a single output request. For example, multiple agents may contribute to the output, and a message may be rendered by using multiple modalities. Furthermore, the output generation process can be modular and involve several steps. This could be used when outputs for different user groups are produced. For example, additional information can be added with separate agents for those users who prefer more informative outputs.

Aside from speech outputs, the presentation management module handles the generation of outputs in other modalities. The presentation management module performs *fission*; that is, it decides which modalities should be used for outputs. In this way, it is easy to use different modalities in the system and to take user preferences into account both because dialog management components do not need to be modified when available modalities are changed and because user preferences are learned or given by the user. The presentation management module can be used to support multimodal outputs for different user groups to make single-mode systems more accessible. We have introduced new modalities in this way without the need to modify existing output generation components. Examples of multimodal timetable systems are given in the following sections.

Multilingual outputs can be hard to handle in speech applications. The modular structure of the presentation management module helps to separate language-specific issues into their own agents, and the developers of a new language version are able to translate the system in an iterative way and keep the process manageable. This supports localization, which is one of the key features in making applications more accessible for older users, for example. We used this approach in the case of the Mailman and AthosMail applications, which are presented in the next section. Finally, speech outputs have a strong influence on the user's choice of words, and therefore on the interaction as a whole. For this reason, presentation agents should produce a consistent speech interface and use knowledge from dialog management, user inputs, and the user model in this process.

Input management

Input agents and evaluators are similar to presentation agents and evaluators. They take conceptual input requests from information storage, select appropriate modalities, and add the necessary control information, such as device configuration parameters. The resulting *control messages* are then processed by the communication module. In this way, conceptual input requests, modality selection, and device control are separated from each other.

Input management components take care of the selection and creation of the input vocabulary (recognition grammars or language models in

speech applications). Context-sensitive grammars may be selected or generated in system-initiative dialogs, meta-dialogs, and other situations where users are expected to speak in various ways. Recognition grammars should be consistent with the way that the system speaks to the user. Even in simple cases, such as when synonyms and yes/no dialogs are used, a wrong lexical selection may lead to problems if output and input languages are not consistent.[23] Because the order in which Jaspis managers are executed can be freely customized, application developers may decide how they model topics, speech styles, and lexical selection in different modules, that is, in which order the corresponding modules will be executed.

Generation of control information is modality-specific, and a highly application-specific and iterative process. By selecting input modalities for specialized agents, dialog-level components can be used without modification. It should be noted, however, that modality change is not just a configuration parameter, but instead may involve complex relations between input, output, and dialog modules. Input management components do not perform actual multimodal fusion (i.e., the process in which the results of multiple input modalities, generally multiple information sources, are combined). This is the task of the communication module.

Communication management and application development

Communication, that is, low-level input and output management, differs in several ways from other parts of the architecture, although it still shares the agents–managers–evaluators paradigm. In addition to agents and evaluators, communication management includes a layered and concurrent architecture for handling communication between devices and other external resources. This architecture consists of *devices, clients, connections, servers, handlers*, and *engines*. Devices and engines are I/O-specific components; clients, connections, servers and handlers can be used in other parts of the system for distribution of components.

In order to create intuitive and rich interfaces, we need flexible models for handling interaction.[4] In Jaspis, the communication manager coordinates input and output devices, using I/O agents to interpret and conceptualize inputs (including natural language understanding and multimodal fusion), and I/O evaluators to coordinate devices. This process is carried through in a timely and concurrent manner, and feedback may be provided to the devices while they are still gathering inputs or presenting outputs.

With respect to technology, the Jaspis framework is based on dynamic objects, XML documents, a set of core infrastructure classes, and more than 20 extensions. XML is used in all parts of the architecture, and the system supports several standards and markup languages. For example, it includes support for several synthesis markup languages, and we are currently implementing VoiceXML support for distributed dialogs and various mobile devices. Because Jaspis-based systems are modular and distributed, an important part of practical application development is configuration, which is performed using XML-based configuration files.

The reference implementation of the architecture is released as open-source software (under the LGPL [Lesser General Public License] license) to support practical application development and can be downloaded from its home page (http://www.cs.uta.fi/hci/spi/Jaspis/). Further details about the communication management and application development are outside the scope of this paper, but can be found in Reference 13.

JASPIS APPLICATIONS

In this section, we present several applications using the Jaspis architecture. These applications cover many areas, including information services, multilingual systems, and pervasive computing applications. Many human-computer interaction aspects are involved, and in all of them accessibility issues are taken into account. In particular, we have worked with special user groups to design, iteratively develop, and evaluate these applications.

E-mail applications

The e-mail domain is an especially suitable area for speech applications. E-mail itself is one of the most successful applications in the history of computing and involves many issues that are relevant to other services and to universal access for various users in different settings.

Our research group has been involved in the development of several multilingual (Finnish, English, and Swedish) speech-based e-mail applications. The first version of the Mailman application was developed for a Finnish national research project.[24] In the EU-funded DUMAS project, the existing Mailman application served as a basis for the AthosMail application.[25] AthosMail supports more advanced functionality, including alternative solutions for input processing, dialog management, and output generation, to bring more robustness and adaptivity to the interaction.[17]

These e-mail applications allow the user to access his or her mailbox by using a standard mobile or fixed-line phone. They offer functionality for most e-mail reading tasks. The systems provide both speech input (speech recognition) and DTMF (Dual-Tone Multi-Frequency, also known as "touch-tone") interfaces which may be used in a multimodal fashion. We have worked together with visually impaired users and learned in the user studies that their preferences are quite different from those of other users, especially those of normally sighted novice users. In order to help different users, we have implemented several features that provide adaptive interaction supporting accessibility for various user groups.[26]

Flexible interaction management

In the AthosMail application, two alternative dialog management approaches are used to bring more flexibility to the interaction.[18] All dialog components use the same dialog history, which is represented as a discourse tree in the information storage. In addition, the tutoring agents bring system-initiative features to the user-initiative interface,[20] complementing the main dialog components.

When the interface is adapted for different users, it is not possible for all functionality to be used by all user groups in all cases (e.g., for technological, economical, or interface reasons). To share components and maintain portability, a common core of functionality is needed. More advanced interaction methods can be built on top of this core, taking into account the specific needs and capabilities of each user group. In the Mailman and AthosMail systems, the different configurations of the system share the same basic functionality. In addition to speech inputs, all functionality can be accessed by the DTMF interface. In small mailboxes, elementary numbers (1–9) are used to select messages and folders. In this way, both the speech and the DTMF interfaces are robust and simple to use. If the user has a large number of messages, more complex recognition grammars are selected by input evaluators to access the messages, and the DTMF interface uses multiple keys to access messages. In addition, we have adapted the recognition grammars to contain the names and keywords

■ In order to coordinate interaction in a flexible way, it is better to use decentralized control logic ■

found in each user's e-mail messages. We have found the DTMF interface to be important for users who are visually impaired, who have usually learned to use it in other applications and often prefer it over the speech recognition interface. However, when we designed DTMF layouts with these users and tested them with sighted novice users, the novice users found them hard to use. We discuss this further in the following sections.

Adaptive system outputs

In the e-mail domain, the main challenge is how to read messages and present other information so that the outputs are both intelligible and pleasant for the user. In the presentation components, different prompts are used for different users (novices, regular users, experts). This is implemented in such a way that each type of prompt (for example, a brief and a verbose version of a welcome prompt) has its own agents, and the appropriate one is selected at runtime, based on dynamically calculated user profiles. Presentation agents for multiple languages are implemented in the same way, thus making it easy to port the applications to different languages. Overall, multilinguality is one of the key issues in e-mail systems.[24] Users have different ways to approach multilingual systems: for example, some of the visually impaired users prefer to use a Finnish synthesizer for English messages as well because they are accustomed to how the synthesizer pronounces English messages. This kind of user preference can be modeled using Jaspis evaluators to dynamically select proper agents for reading the messages.

In order to present large lists of messages efficiently and adapt them for each user, the messages are categorized into meaningful groups by using an algorithm that tries to create both meaningful and easily manageable groups. Information about the messages, such as header information, is used for this purpose. The number of messages within each group is kept within the limits of human capabilities to handle spoken data.[27] This helps the user to get an overview of the mailbox contents even with a large number of messages. By dynamically adapting the groupings for each situation, it is possible to use alternative input methods and strategies, such as the DTMF keys. In particular, users who are visually impaired often prefer these communication methods. In addition, tapering is used to reduce the length of the lists, by including only the meaningful information in them.

■ Usually, application-specific components are encapsulated in their own agents, and inheritance is used to maximize runtime efficiency ■

Help and guidance

In order to provide help and guidance for novice users, AthosMail supports a set of universal commands[28] that adapt to the interaction. With the "tell me more" command, the user can get a detailed presentation of the current dialog situation. For example, when a message is selected, only the main information is spoken. With the "tell me more" command, the user can access additional information, such as length of the message, date, and time. Another universal command is the "what now?" command, which triggers context-sensitive help, in which only relevant guidance is given to the user, taking into account the current dialog situation and history. These features are demonstrated in the following dialog (translated from Finnish):

SYSTEM: Message number one from Esa-Pekka Salonen, subject is "AthosMail review"
USER: Tell me more.
SYSTEM: Message number one from Esa-Pekka Salonen, subject is "AthosMail review, urgent." Message was sent on Wednesday at 14.24. It has 6 lines. Summary of message: "We should prepare for the review. How about meeting tomorrow?"
USER: What now?
SYSTEM: You can make the system read the message by saying "read", or you can select another message by saying, for example, "second message", ...

Another form of guidance is provided by tutoring agents.[20] They introduce the system to the user, monitor how well the user interacts with the system, and provide guidance accordingly. Tutoring components take the initiative in certain situations. They use the dialog history, the user model, and their own tutoring plan in this process. Technically, the Jaspis agents and evaluators make it possible to add the tutoring agents to the system without modifications to other components. Furthermore, the tutoring feature can be turned dynamically on and off. The following dialog (translated from Finnish) gives an example of the tutoring feature:

SYSTEM: Please wait, connecting to your mail server ... Hi, Test user. No new messages. 17 old messages. You have 3 groups. Group one, ...
TUTOR: Hi, I'm your tutor. I'll teach you how to use the system. Next, choose one of the available groups. You can do this by saying, for example, "third group." So, please use the group number you wish.
USER: Go to the third group
TUTOR: Good. Now you are in the third group. Next AthosMail will list messages in the group.
SYSTEM: [the system lists the messages in the selected third group]

Bus timetable systems

In addition to e-mail systems, we have been working with several bus timetable systems. As with the e-mail domain, the timetable domain is practical, provides plenty of research challenges, and the results can be used in other information services. Similar human-operated systems are widely available, and there is a need to replace existing systems with automated systems. Furthermore, with automated systems it is possible to produce new services that are either not possible or not economically viable with human operators. For special user

groups, they may represent the only possibility to access information, as has been noted recently in many areas.

The Busman and Interact systems provide bus route and timetable information for the Tampere and Helsinki areas, two major cities in Finland. The functionality of these systems is similar to other timetable services, such as MALIN.[29] The user may request information such as bus routes (e.g., "Which bus goes to the university hospital?") and timetables (e.g., "When does the next one leave?"). Like the e-mail applications, these systems have a speech interface and can be used with mobile and fixed-line telephones. In contrast, a third timetable system (Stopman) and its ongoing continuation (PUMS), are based more on the system-initiative approach and can be used with a multimodal personal digital assistant (PDA) interface as well. These applications are designed in close collaboration with users who are visually impaired. Currently, we are implementing mobile clients for the Stopman system. Wireless connections, such as GPRS (General Packet Radio Service) or UTMS (Universal Mobile Telecommunications System), provide speech recognition and synthesis services on PDAs and cellular phones for users who are visually impaired. Jaspis components are used both on the server and the client side, and the interaction, such as menu selection, is adapted by the agents and evaluators to the capabilities of each device. In addition to speech and nonspeech audio, we have plans to use touch-sensitive displays. In the Interact system, multimodal extensions such as a graphical touch-screen interface for information kiosk usage and a short-message interface for mobile nonspeech usage were implemented for users who are deaf or hard of hearing.

In the Busman system, we have experimented with a truly mixed-initiative approach to dialog management, which we have found to be the key aspect in making dialogs work in this domain. In this approach, each dialog agent has an initiative attribute, which indicates its suitability for system-initiative and user-initiative dialogs. In the current implementation, one dialog evaluator is used to give preference to system-initiative agents when the interaction proceeds normally, and to user-initiative agents when errors are present. In this way, the system tries to adapt to the interaction by changing the dialog strategy on the basis of the success of the

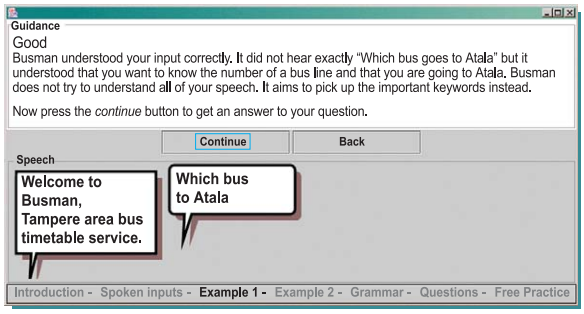

Figure 3
Multimodal tutoring in the Busman system

interaction. An example dialog with this feature was given in the section "Dialog and error management." In addition, the Busman application includes a set of tutoring agents[20] to provide multimodal guidance and assistance in error situations. *Figure 3* illustrates this guidance. It also demonstrates how spoken dialogs can be visualized automatically. In terms of technology, the tutoring is implemented as a set of agents and evaluators providing GUI outputs and inputs, while the Busman agents and evaluators provide speech inputs and outputs. Again, the tutoring component could be included in the system without any modifications to the Busman code. All the information necessary for the tutor agents is already available in the information storage.

Pervasive speech applications

Doorman is a pervasive computing system that uses speech and audio as its main modalities. It has been designed with both sighted users and users who are visually impaired in mind. We have used it to experiment with implementations of speech interfaces in pervasive-computing settings. In this field, many new challenges have been encountered. Experiments from this domain have greatly influenced the development of the new Jaspis-2 architecture,[30] which supports multiple concurrent users and dialogs.

The Doorman system serves staff members and visitors in an office environment. The system controls access to this environment by identifying staff members and helping visitors to find the place or person that they are looking for. The system gives guidance to visitors about how to reach their destinations. Several extensions, such as an auditory awareness information service, have been added to

the system. The system has features similar to the Office Monitor[31] and PER[32] systems.

The Doorman system uses speech recognition and speaker identification to recognize users from their speech. Synthesized speech, nonspeech audio, and pointing gestures are used for multimodal outputs. For visually impaired users, the speech outputs, augmented with nonspeech audio (including auditory icons) contain the necessary information. The presence, location, and state of users can be tracked from different signals and their fusion. People on the move can be followed using infrared and pressure-sensitive EMFi (Electro-Mechanical Film) technologies. Other information (such as keyboard, mouse, or application activity) is received from desktop computers and mobile devices.

■ Tutor-based guidance is a promising approach to making systems accessible for different users ■

The main feature of the system is guidance. The office layout is modeled in such a way that the system is able to generate various alternative guidance descriptions with varying levels of detail. The selection of details is highly dependent on the particular user group process. Route instructions given to users who are visually impaired need to be founded on different criteria from those for sighted users.[33] For example, the visual landmarks that are useful for sighted users can be replaced with auditory landmarks for visually impaired users. This can be understood by comparing the following system outputs:

For sighted users:

> Follow the hallway until you come to a crossroad with a sign "copy machine."

For users who are visually impaired:

> Continue ahead until you can hear the following sound [sound of a copy machine].

Communication with the user is handled through communication points located in strategic places in the facility, such as intersections. This enables the distribution of route instructions in small units, which are easier to understand and memorize, hence easing the cognitive load on the user.[34] These smaller units are presented according to the user's progress along the route. Distributed route instructions are supported by auditory cues, which function as beacons along the route. An auditory cue played from the next communication unit on the route functions as a target which the user can aim at. This serial component in multimodal spatial information is very important for visually impaired users.[33] The user can ask for more detailed information from the system when passing communication units.

In addition to guidance related to navigation, the system is capable of giving added information in the manner of an exhibition guide. The guide gives a tour of the premises, showcasing physical artifacts. The same identification and positioning information enables cross-references to earlier events and personalized content generation with temporally as well as spatially dynamic modality choices. Such choices are helpful for various accessibility problems, for example, as demonstrated in the following system outputs:

For sighted users:

> As you can see, the artist uses the same technique of contrasting colors when choosing materials, as in the three previous sculptures.

For users who are visually impaired:

> If you touch the sculpture, you can feel the contrasting textures of soft wool and unfinished cast bronze, a similar method to that used in the artist's two previous sculptures.

One group of extensions to the Doorman system is applications for supporting awareness and group communication. In these applications, auditory information is presented in a way which helps to keep users aware of events occurring in their surroundings. For example, an application monitors the presence of group members and informs other group members about the activity in the group (e.g., usage of desktop computers or tracking of people using floor sensors). The presentation of the information is done so that auditory icons and other

auditory information indicate important changes in the environment, such as arrival of other people and incoming messages. Through ambient soundscapes (i.e., acoustic environments that surround the user), information is made available in the periphery of the senses.[35] Both continuous and temporal information are presented. Natural sounds, such as walking sounds and birds singing, are used in auditory presentations.[36] From the architectural point of view, creating and controlling such soundscapes is not trivial. One has to map the information to specified qualities of the sound, as well as manage and adjust the whole composition dynamically according to preset rules, in order to assure the continuity and harmony of the presentation.

The guidance and awareness applications can be very useful for users with special needs. For example, navigation in an unfamiliar environment can be a challenging task for users who are visually impaired and cannot rely on visual landmarks. Similarly, a good deal of tacit awareness information can be sensed, fused together, processed, and made available through several modalities and levels of abstraction. Our goal is to provide methods to express meaningful information from the environment to help different users in their everyday tasks. We have addressed this challenge by implementing alternative interaction techniques and applied them in the various applications using the agents–managers–evaluators paradigm. In order to better understand the needs of different users, we plan to install the Doorman system in places where special user groups work and live.

USER STUDIES

We arranged several user experiments at different stages of the development of the Jaspis applications. We have used various representative user groups in these studies, focusing on users who are visually impaired in particular. In addition, user feedback has been received from the public use of the applications. Mailman has been available to the public since 1999, and the bus timetable service Stopman has been in public use since 2003. In these studies, various Jaspis evaluation tools have been used.

User-centered design

To take representative users into account in the development of the Mailman application, we arranged a design session with the Finnish Federation of the Visually Impaired. The purpose of the session was to introduce the components that had been implemented and to design speech inputs and outputs by experimenting with different alternatives. Similarly, we designed bus timetable systems with representative users. We found it useful to conduct early user tests with partially implemented applications. In particular, we managed to set up design sessions with Jaspis tools without any modifications to the application code. This significantly accelerated the process and allowed a wide range of experiments to be conducted interactively.

In order to identify usability problems, we collected feedback by asking specific questions and requesting free-form comments on the usability of the applications. These are good ways of identifying specific problems, such as poor rendering of e-mail contents. Users who are visually impaired, especially those working in the information technology industry, may be considered to be expert users of speech applications, and they are also, in many ways, a very demanding user group. Other expert opinions were gathered from students taking a usability course with no prior experience of speech applications, but with an ability to analyze user interfaces.

The results obtained with representative end users have been used in implementations using Jaspis agents and evaluators in practical applications. For example, we have used and evaluated different DTMF layouts for e-mail applications designed with users who are visually impaired. Similarly, we have implemented new agents to read e-mail messages more fluently on the basis of the comments from users who are visually impaired. With our modular and adaptive approach, these components can be included in applications and selected dynamically when needed. Furthermore, the resulting agents can be used in other similar applications as well. For example, a telephone-based document-reading system for users who are visually impaired has been implemented by other developers on the basis of the Jaspis architecture and the AthosMail application.[37]

Wizard of Oz studies

"Wizard of Oz" (WOZ) experiments, in which some parts of the system are simulated with a human operator, are useful for speech application development as a usability-testing and data-collection method. There are several tools to aid such experi-

ments, such as SUEDE.[38] One particular limitation of such external tools is that they simulate the whole system, and as a result, the software used in the experiment is different from the software to be used in the actual system. In most cases, however, at least a partly implemented system is needed or wanted. Thus, it would be useful to run the actual system in a WOZ mode. This is the purpose of the Jaspis WOZ tools. For building WOZ interfaces, Jaspis includes a generic WOZ GUI (graphical user interface), which can be customized for different purposes by using an XML-based description language. This way, setting up a WOZ GUI requires a definition of the interface, and no programming is necessary.

■ The need for adaptivity and supporting architectures is identified to be one of the key elements in the next generation of speech applications ■

We used the Jaspis tools to set up a WOZ experiment to test the Doorman and AthosMail applications. The usability of the Doorman system was evaluated as part of the development process by performing a WOZ experiment in which speech recognition and speaker recognition were simulated. This was done by using a Jaspis WOZ applet. Otherwise, the guidance system was functional, and there were no modifications to the Doorman components. Aside from evaluating the system functionality and multimodal guiding instructions, we wanted to collect data about how people speak to the system.

With the Doorman WOZ experiment, we were able to find many ways to improve the system. The main findings from the user study concerned the way that the users spoke to the system and the guidance given by the system. For example, the guidance algorithm was modified to use landmarks and relative measures, and the dialog was divided to consist of smaller dialog units. There were several user groups (staff members, students, and visitors), who had very different needs for the interaction. The experiment and the results are described in detail in Reference 39. In the future, we will implement more adaptivity in the guidance to customize it for more user groups and individuals, making the system more accessible.

We also used the WOZ method with a similar setup to test the AthosMail application. We used the WOZ study as a starting point for the AthosMail application design, together with experiments in the usage of the Mailman system.

Evaluation of working applications

We have done several experiments with working versions of Jaspis applications. The e-mail systems have been evaluated with several approaches. In these studies, we have found several interesting phenomena from the user diversity point of view; for example, males and females have quite different expectations and perceptions of the speech interface;[40] tutor-based guidance is a promising approach to making the systems accessible for different users;[41] and sighted and visually impaired users prefer different DTMF layouts. When we tested various DTMF layouts, some of which had been designed together with users who were visually impaired, sighted novice users found the layouts to be randomly assigned to the application functionality. Users who were visually impaired, on the other hand, had a strong mental model (learned from the use of text-based applications and screen readers) concerning how the DTMF keys should be used for navigation in speech interfaces. In addition, this showed that novice users need support and guidance when DTMF interfaces are used. In the case of the Jaspis applications, customized interfaces and their dynamic selection are modeled using agents and evaluators, and we have implemented several context-sensitive features for help and guidance, as discussed earlier.

We have conducted user studies with the working versions of the timetable systems as well. Several problems were found after the tests. Based on the findings, Jaspis agents were used to improve usability and add new functionality to the system. For example, the users' language in the experiments was different from the language in the recorded conversations between human operators and users. In the human-to-human recordings, conversational language was used, but in the human-to-computer interaction, shorter sentences, terser style, and different words and structures overall were used. Most interestingly, in the human-to-computer con-

versations, some people used nongrammatical sentences similar to universal commands[28] and tried to adapt their language to the system. A representative example is "departure: railway station [pause] destination: Arabia." The lesson learned here is that the language model should not be acquired directly from human-to-human conversations because the language used in human-to-computer communication is quite different. We solved this problem by adding a set of new Jaspis agents to take care of those inputs that the original, corpus-based input-interpretation components were not able to handle.

In addition, we concluded that it is more efficient to use a system-initiative dialog strategy as a default. We used this approach in the more recent Stopman application, which has been evaluated by analyzing the calls from the public use of the system and has been found usable in real-life practical situations. In this context as well, it seems that users who are visually impaired have quite different ways to interact with the system. For example, they seem to interrupt the system prompts immediately in the beginning of the call, whereas most users do not interrupt the greeting prompts (however, because the calls are anonymous, we cannot identify the users accurately).

In summary, the tests and experiments with the applications have provided essential information for iterative application development. In particular, we have found the needs of different user groups to be quite different. From the technical perspective, we have found it efficient to work with the Jaspis agents–managers–evaluators paradigm, which has allowed us to improve the accessibility of the systems by implementing alternative components for the same tasks. Jaspis agents have made it possible to efficiently realize results in terms of software components and to construct sophisticated tools to aid the evaluation process.

CONCLUSION

We have presented a number of ways in which the needs of different users can be taken into account with flexible and adaptive architecture solutions. The Jaspis architecture includes the generic agents–managers–evaluators paradigm, which has proven to be successful when adaptive applications and interaction techniques have been built on top of the architecture. When applications are constructed with adaptivity in mind, it is easy to support different user groups, as demonstrated in several Jaspis applications.

Overall, the need for adaptivity and supporting architectures is identified to be one of the key elements in the next generation of speech applications.[42] We have found the compact agents to be very useful when alternative interaction methods are included in existing applications. Jaspis agents together with the built-in adaptation mechanism of Jaspis make it possible to include additional and alternative functionality for different needs without modifications to the existing components. An example of this is the customized DTMF interfaces for telephone systems and spoken guidance in indoor environments that takes the needs (e.g., visual versus nonvisual landmarks) of different users into account.

In order to better support sophisticated speech-based systems, the Jaspis architecture has been extended in several ways. The main new feature of the Jaspis-2 architecture[30] is the support for concurrent interaction in a coordinated and synchronized way. In the Jaspis-2 architecture, indirect messaging (using triggers and transactions) between system components (i.e., agents) is used to support concurrent interaction, while the coordinated adaptation mechanism of the original architecture (using managers and evaluators) is preserved.

At the same time, the architecture is more blackboard-oriented, and the distribution of components is extended to support sharing of components among modules. All the changes have been made in a way that makes it possible to use the adaptive features of the original architecture. This is still preliminary work, and in the future we will continue the development of the Jaspis-2 architecture in close relation to pervasive-computing systems in order to find out how new application areas can be better supported by the system architecture.

The solutions offered by the Jaspis architecture are targeted mainly for speech-based and auditory applications. Similar challenges can be found in other application domains as well. In particular, non-speech-based approaches to support accessibility are among the issues that will be part of the future development of the Jaspis architecture and its applications.

**Trademark, service mark, or registered trademark of Turner Entertainment Corporation.

CITED REFERENCES

1. E. Mynatt and G. Weber, "Nonvisual Presentation of Graphical User Interfaces," *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'94)*, Boston, MA (1994), pp. 166–172.
2. A. Edwards, "Multimodal Interaction and People with Disabilities," in *Multimodality in Language and Speech Systems*, B. Granström, D. House and I. Karlsson, Editors, Kluwer Academic Press, Dordrecht, The Netherlands (2002), pp. 73–92.
3. D. Garlan and M. Shaw, "An Introduction to Software Architecture," in *Advances in Software Engineering and Knowledge Engineering, Series on Software Engineering and Knowledge Engineering, Volume 2,* V. Ambriola and G. Tortora, Editors, World Scientific Publishing Company, Singapore (1993), pp. 1–39.
4. J. Allen, G. Ferguson, and A. Stent, "An Architecture for More Realistic Conversational Systems," *Proceedings of the International Conference on Intelligent User Interfaces 2001 (IUI-01)*, Santa Fe, New Mexico (2001), pp. 14–17.
5. M. McTear, "Spoken Dialogue Technology: Enabling the Conversational Interface," *ACM Computing Surveys* **34**, No. 1, 90–169 (March 2002).
6. S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-II: A Reference Architecture for Conversational System Development," *Proceedings of the Fifth International Conference on Spoken Language Processing (ICSLP98)*, Sydney, Australia (1998), pp. 931–934.
7. D. L. Martin, A. J. Cheyer, and D. B. Moran, "The Open Agent Architecture: A Framework for Building Distributed Software Systems," *Applied Artificial Intelligence: An International Journal* **13**, No. 1–2 (January–March 1999), pp. 91–128.
8. A. Rudnicky and W. Xu, "An Agenda-based Dialog Management Architecture for Spoken Language Systems," *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop* (1999), pp. 337–340.
9. D. Bohus and A. Rudnicky, "RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda," *Proceedings of the Eighth European Conference on Speech Communication and Technology (Eurospeech 2003)*, Geneva, Switzerland (2003), pp. 597–600.
10. I. O'Neill, P. Hanna, X. Liu, and M. McTear, "The Queen's Communicator: An Object-Oriented Dialogue Manager," *Proceedings of the Eighth European Conference on Speech Communication and Technology (Eurospeech 2003)*, Geneva, Switzerland (2003), pp. 593–596.
11. B. Pakucs, "Towards Dynamic Multi-Domain Dialogue Processing," *Proceedings of the Eighth European Conference on Speech Communication and Technology (Eurospeech 2003)*, Geneva, Switzerland (2003), pp. 741–744.
12. M. Turunen and J. Hakulinen, "Jaspis—A Framework for Multilingual Adaptive Speech Applications," *Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, Beijing, China (2000), pp. 719–722.
13. M. Turunen, *Jaspis—A Spoken Dialogue Architecture and Its Applications*, Ph.D. dissertation, University of Tampere, Finland, Department of Computer Sciences, Report A-2004-2, ISBN 951-44-5896-6, ISSN 1459-6903 (2004).
14. N. Blaylock, J. Allen, and G. Ferguson, "Synchronization in an Asynchronous Agent-Based Architecture for Dialogue Systems," *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialog*, Philadelphia, PA; Association for Computational Linguistics (2002), pp. 1–10.
15. O. Lemon, A. Bracy, A. Gruenstein, and S. Peters, "The WITAS Multi-Modal Dialogue System I," *Proceedings of the Seventh European Conference on Speech Communication and Technology (Eurospeech 2001)*, Aalborg, Denmark (2001), pp. 1559–1562.
16. I. O'Neill and M. McTear, "Object-Oriented Modeling of Spoken Language Dialogue Systems," *Natural Language Engineering* **6**, No. 3–4, Cambridge University Press (2000), pp. 341–362.
17. M. Turunen, E.-P. Salonen, M. Hartikainen, and J. Hakulinen, "Robust and Adaptive Architecture for Multilingual Spoken Dialogue Systems," *Proceedings of the Eighth International Conference on Spoken Language Processing (INTERSPEECH 2004-ICSLP)*, Jeju Island, Korea (2004), pp. 3081–3084.
18. E.-P. Salonen, M. Hartikainen, M. Turunen, J. Hakulinen, and A. Funk, "Flexible Dialogue Management Using Distributed and Dynamic Dialogue Control," *Proceedings of the Eighth International Conference on Spoken Language Processing (INTERSPEECH 2004-ICSLP)*, Jeju Island, Korea (2004), pp. 197–200.
19. M. Turunen and J. Hakulinen, "Agent-Based Error Handling in Spoken Dialogue Systems," *Proceedings of the Seventh European Conference on Speech Communication and Technology (Eurospeech 2001)*, Aalborg, Denmark (2001), pp. 2189–2192.
20. J. Hakulinen, M. Turunen, and E.-P. Salonen, "Agents for Integrated Tutoring in Spoken Dialogue Systems," *Proceedings of the Eighth European Conference on Speech Communication and Technology (Eurospeech 2003)*, Geneva, Switzerland (2003), pp. 757–760.
21. J. Chu-Carroll, "MIMIC: An Adaptive Mixed Initiative Spoken Dialogue System for Information Queries," *Proceedings of the 6th ACL Conference on Applied Natural Language Processing*, Seattle, WA (May 2000), pp. 97–104.
22. M. Walker, J. Fromer, G. Fabbrizio, C. Mestel, and D. Hindle, "What Can I Say?: Evaluating a Spoken Language Interface to Email," *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 98)*, Los Angeles, CA (1998), pp. 582–589.
23. B. Hockey, D. Rossen-Knill, B. Spejewski, M. Stone, and S. Isard, "Can You Predict Responses to Yes/No Questions? Yes, No, and Stuff," *Proceedings of the Fifth European Conference on Speech Communication and Technology (Eurospeech 1997)*, Rhodes, Greece (1997), pp. 2267–2270.
24. M. Turunen and J. Hakulinen, "Mailman—a Multilingual Speech-Only E-Mail Client Based on an Adaptive Speech Application Framework," *Proceedings of the Workshop on Multi-Lingual Speech Communication (MSC 2000)*, Kyoto, Japan (2000), pp. 7–12.
25. M. Turunen, E.-P. Salonen, M. Hartikainen, et al., "AthosMail—A Multilingual Adaptive Spoken Dialogue System for the E-mail Domain," *Proceedings of the Workshop on Robust and Adaptive Information Processing for Mobile Speech Interfaces*, Geneva, Switzerland (2004), pp. 77–86.
26. E.-P. Salinen, M. Hartikainen, M. Turunen, J. Hakulinen, J. Rissanen, K. Kanto, and K. Jokinen, "Adaptivity in a Speech-Based Multilingual E-mail Client," *Proceedings of NordiCHI 2004*, Tampere, Finland (2004), pp. 437–440.
27. B. Suhm, "Towards Best Practices for Speech User Interface Design," *Proceedings of the Eighth European Conference on Speech Communication and Technology*

(Eurospeech 2003), Geneva, Switzerland (2003), pp. 2217–2220.

28. R. Rosenfeld, X. Zhu, S. Shriver, A. Toth, K. Lenzo, and A. W. Black, "Towards a Universal Speech Interface," *Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, Beijing, China (2000), http://www-2.cs.cmu.edu/~stef/papers/ICSLP00-usi.pdf.
29. N. Dahlbäck and A. Jönsson, "Knowledge Sources In Spoken Dialogue Systems," *Proceedings of the Seventh European Conference on Speech Communication and Technology (Eurospeech 1999)*, Budapest, Hungary (1999), pp. 1523–1526.
30. M. Turunen and J. Hakulinen, "Jaspis^2 - An Architecture For Supporting Distributed Spoken Dialogues," *Proceedings of the Eighth European Conference on Speech Communication and Technology (Eurospeech 2003)*, Geneva, Switzerland (2003), pp. 1913–1916.
31. N. Yankelovich and C. McLain, "Office Monitor," *Proceedings of the Conference on Human Factors in Computing Systems (CHI '96)*, Vancouver, British Columbia, Canada; ACM Press, New York (1996), pp. 173–174.
32. B. Pakucs and H. Melin, "PER: A Speech Based Automated Entrance Receptionist," *Proceedings of the 13th Nordic Conference of Computational Linguistics (NoDaLiDa '01)*, Uppsala, Sweden (2001).
33. S. Werner, B. Krieg-Brückner, H. A. Mallot, K. Schweizer, and C. Freksa, "Spatial Cognition: The Role of Landmark, Route, and Survey Knowledge in Human and Robot Navigation," in *Informatik '97*, M. Jarke, K. Pasedach, and K. Pohl, Editors, Springer-Verlag, Vienna, Austria (1997), pp. 41–50.
34. K. Hook, "An Approach to a Route Guidance Interface," Licentiate Thesis, Stockholm University, Department of Computer and Systems Sciences, ISSN 1101 8526 (1991).
35. M. Weiser and J. S. Brown, "The Coming Age of Calm Technology," in *Beyond Calculation: The Next Fifty Years*, Copernicus, New York, NY (1997), pp. 75–85.
36. K. Mäkelä, J. Hakulinen, and M. Turunen, "The Use Of Walking Sounds In Supporting Awareness," *Proceedings of the International Conference on Auditory Display (ICAD 2003)*, Boston, MA (2003), pp. 144–147.
37. P. Heiskari, "The Finnish AthosNews: A News-Reading Application for Visually Impaired Users," *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland (2004), p. 43.
38. S. Klemmer, A. Sinha, J. Chen, A. Landay, N. Aboobaker, and A. Wang, "SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces," *CHI Letters, The 13th Annual ACM Symposium on User Interface Software and Technology (UIST 2000)* **2**, No. 2 (2000), pp. 1–10.
39. K. Mäkelä, E.-P. Salonen, M. Turunen, J. Hakulinen, and R. Raisamo, "Conducting a Wizard of Oz Experiment on a Ubiquitous Computing System Doorman," *Proceedings of the International Workshop on Information Presentation and Natural Multimodal Dialogue* (2001), pp. 115–119.
40. M. Hartikainen, E.-P. Salonen, and M. Turunen, "Subjective Evaluation of Spoken Dialogue Systems Using SERVQUAL Method," *Proceedings of the Eighth International Conference on Spoken Language Processing (INTERSPEECH 2004-ICSLP)*, Jeju Island, Korea (2004).
41. J. Hakulinen, M. Turunen, E.-P. Salonen, and K.-J. Räihä, "Tutor Design for Speech-Based Interfaces," *Proceedings of Designing Interactive Systems (DIS2004)*, Cambridge, MA (2004), pp. 155–164.
42. M. McTear, "New Directions in Spoken Dialogue Technology for Pervasive Interfaces," *Proceedings of the Workshop on Robust and Adaptive Information Processing for Mobile Speech Interfaces* (2004), pp. 57–64.

Accepted for publication February 16, 2005.
Published online August 5, 2005.

Markku Turunen
University of Tampere, Department of Computer Sciences, Tampere Unit for Computer Human Interaction, Speech-Based and Pervasive Interaction Group, Kanslerinrinne 1, FIN-33014 University of Tampere, Finland (mturunen@cs.uta.fi). Dr. Turunen received a Ph.D. degree in computer science from the University of Tampere in March 2004. He joined the Tampere Unit for Computer-Human Interaction in 1998, and has been coordinating one of its research groups, the Speech-Based and Pervasive Interaction group. His research interests are in speech-based, auditory, and pervasive applications.

Jaakko Hakulinen
University of Tampere, Department of Computer Sciences, Tampere Unit for Computer Human Interaction, Speech-Based and Pervasive Interaction Group, Kanslerinrinne 1, FIN-33014 University of Tampere, Finland (jaakko.hakulinen@cs.uta.fi). Mr. Hakulinen is a researcher in the Speech-Based and Pervasive Interaction Group. He received an M.Sc. degree in computer science from the University of Joensuu in 1998. He joined the TAUCHI (Tampere Unit for Computer Human Interaction) unit in 1998 and has been working since on the area of speech interfaces. Currently, he is working on software tutoring of speech interfaces.

Kari-Jouko Räihä
University of Tampere, Department of Computer Sciences, Tampere Unit for Computer Human Interaction, FIN-33014 University of Tampere, Finland (kjr@cs.uta.fi). Dr. Räihä is a professor of computer science at the University of Tampere. He received a Ph.D. degree in computer science from the University of Helsinki in 1982 and moved to the University of Tampere soon after. He has worked in human-computer interaction for 15 years and is the founder and head of the Tampere Unit for Computer-Human Interaction, a research unit of more than 50 researchers, faculty members, and graduate students. His research interests are in interaction design, particularly the use of nonstandard modalities, such as eye gaze and speech.

Esa-Pekka Salonen
University of Tampere, Department of Computer Sciences, Tampere Unit for Computer Human Interaction, Speech-Based and Pervasive Interaction Group, University of Tampere, Kanslerinrinne 1, FIN-33014 University of Tampere, Finland (esa-pekka.salonen@cs.uta.fi). Mr. Salonen is a researcher and a Ph.D. candidate at the University of Tampere. He received B.Sc. and M.Sc. degrees in computer science from the University of Tampere in June 2002 and in December 2002, respectively. Mr. Salonen has worked in the Speech-Based and Pervasive Interaction Group since 2001 on various projects and applications.

Anssi Kainulainen
University of Tampere, Department of Computer Sciences, Tampere Unit for Computer Human Interaction, Speech-Based and Pervasive Interaction Group, University of Tampere, Kanslerinrinne 1, FIN-33014 University of Tampere, Finland (anssi@cs.uta.fi). Mr. Kainulainen is a researcher in the Speech-Based and Pervasive Interaction Group. He is working on auditory awareness information in ubiquitous systems.

Perttu Prusi
University of Tampere, Department of Computer Sciences, Tampere Unit for Computer Human Interaction, Speech-Based and Pervasive Interaction Group, University of Tampere, Kanslerinrinne 1, FIN-33014 University of Tampere, Finland (prusi@cs.uta.fi). Mr. Prusi is a researcher in the Speech-Based and Pervasive Interaction Group. He received an M.Sc. degree in computer science from the University of Tampere in February 2005. He is working on speech-based and audio-based guidance systems. ■