Coordinating Adaptations in Open Service Architectures

Markus Bylund

Computing Science Department
Uppsala University
Box 311
SE-751 05 Uppsala
Sweden

This work has been carried out at
Swedish Institute of Computer Science
Box 1263
SE-164 29 Kista
Sweden

Abstract

An Open Service Architecture (OSA) is a software structure that makes an open set of information services available to an open set of users. The World Wide Web constitutes the most outstanding example of an OSA as of today. An important feature of an OSA is personalization, i.e. adapting the user interface, functionality, and information of services to its users. However, designers of such a feature are facing many problems, perhaps the biggest one being coordination. If services fail to coordinate how they adapt to users, chances are that the whole point of performing the adaptation, i.e. helping the user, is lost.

In this thesis, I lay out a framework for describing and reasoning about adaptive systems in Open Service Architectures, with a special emphasis on coordination. This framework is mainly meant for analysis and design, but some of the ideas presented are also suitable as metaphors for implementations. An implementation of an adaptive system that was designed using this framework, adaptive help in the KIMSAC system, is also described.

Supervisor: Annika Waern
Examiner: Lars Thalmann

Passed:
Acknowledgements

I would like to thank my supervisor Annika Waern at SICS, and examiner Lars Thalmann at the Computing Science Department at Uppsala University.

Large parts of the work with this thesis were conducted within the KIMSAC project, and I would like to thank all the partners of the project; especially Olle Olsson and Fredrik Espinoza at SICS, who provided an excellent environment in which I could develop and try out my ideas.

I would also like to thank Fredrik Espinoza and Kia Höök for help with commenting and proof reading the thesis.
# Table of Contents

Acknowledgements ........................................................................................................ iii

Table of Contents ........................................................................................................... v

1 Introduction .................................................................................................................. 1
   1.1 Why should systems be adaptive? ................................................................. 1
   1.2 Problem Domain ............................................................................................. 2
   1.3 Contribution .................................................................................................... 2
   1.4 Structure ......................................................................................................... 3

2 Background and Literature ......................................................................................... 5
   2.1 Open Service Architectures ............................................................................. 5
      Examples of OSAs ............................................................................................ 5
   2.2 Adaptive User Interfaces ............................................................................... 6
   2.3 Adaptive Help ................................................................................................ 7
      The Microsoft® Office Assistant ....................................................................... 8

3 A Framework for Performing Adaptations in OSAs .................................................. 9
   3.1 Actors .............................................................................................................. 9
      Users ................................................................................................................ 10
      Service Providers ............................................................................................. 10
      Adaptation Coordinator .................................................................................. 11
   3.2 Information Models ....................................................................................... 12
      Domain Models ............................................................................................... 12
      User Model ..................................................................................................... 12
      Adaptation Model .......................................................................................... 13
   3.3 Adaptation Phases ........................................................................................ 14
      Classifying Adaptations .................................................................................. 14
      Interaction Schemes ......................................................................................... 16
   3.4 Adaptation Lifecycle ....................................................................................... 18
      During Design .................................................................................................. 18
      At Startup ......................................................................................................... 19
      During Runtime ............................................................................................... 19
   3.5 Summary .......................................................................................................... 19

4 Adaptive Help in KIMSAc ......................................................................................... 21
   4.1 Overview ......................................................................................................... 21
   4.2 User Interface ................................................................................................ 21
   4.3 Help in KIMSAc ............................................................................................. 23
      Instruction Text and Video ............................................................................. 24
      Glossary of Terms ......................................................................................... 25
      Where to Next ................................................................................................. 25
      How It Works ................................................................................................. 25
   4.4 Components for Realizing Help .................................................................... 25
      Adaptation Agent ............................................................................................ 25
      Service Agents ............................................................................................... 26
      Guide Agent ................................................................................................. 26
   4.5 Information Models ......................................................................................... 27
Interaction Model ................................................................. 27
Usage Model ................................................................. 27
Help Model ........................................................................ 28
Adaptation Model ............................................................ 28
4.6 Estimating the User’s Need for Help .................................. 29
4.7 Interaction Schemes ....................................................... 30
4.8 Limitations ................................................................. 31
5 Conclusions ...................................................................... 33
5.1 Related Work ............................................................... 33
5.2 Future Work ................................................................. 33
References ........................................................................... 35
1 Introduction

The computer systems of today are no longer stand-alone programs developed for a closed group of users/professionals in a well-known organization and environment. Instead, computer systems are turning into computer services, available to a large, distributed and heterogeneous user group. In the same way, the singular user is no longer faced with a singular computer system, but with a vast and perpetually changing space of computer services. Services move between platforms: they are no longer the singular desk-top computer, but run on information kiosks, portable and wearable devices, mobile phones and the home TV. This requires an Open Service Architecture (OSA): a framework in which an open set of users can access and interact with an open set of services.

A critical issue for OSAs is their integration. An OSA forms a particular kind of complex system, where components are developed independently of each other. Unless there is some kind of integration of components, the individual user will be faced with a multitude of interaction metaphors and interfaces, that all must be understood and learned. The situation is worsened rather than eased by personalization, since every service as well as platform may provide different ranges of adaptations and different means for accessing them.

In this thesis, I address the issue of how to adapt different aspects of services, such as information, behavior, and functionality, in an OSA. The aim is to analyze the process of performing adaptations, an analysis that can be used as a basis for designing mechanisms that handle adaptations in OSAs.

1.1 Why should systems be adaptive?

The group of end users of a system is prone to be heterogeneous. Users’ demands on the system are likely to differ in communicative and functional terms. For example, a novice user might prefer to use a pull down menu when interacting with a word processor, while a more experienced user might prefer the shortcut keys of the same system. One user might use the word processor mainly for writing business correspondence, while another uses it for writing a novel. Issues like these constrain the design of a system, sometimes in contradictory ways (Benyon, 1993).

In addition to changes of users, the environment may change over time, possibly affecting the system’s ability to solve its task. Hardware devices might be added or removed, databases may change in accessibility, communication channels are prone to differ in bandwidth and availability, etc. With the introduction of mobile computer systems (laptop computers, palmtops, and wearable computers), everything in the system’s environment is likely to change.

Ever since interactive systems were first used, the prime way to cope with contradictory design constraints and changing environments has been to make systems adaptable. Users of a system have been able to tailor systems to their individual needs and to the system’s environment. However, this has been done manually by end users, administrators, etc. This kind of system, often called flexible, suffers from that the task of manually adapting a system may require deep knowledge about the system and its environment.

In contrast to a flexible system, an adaptive system includes some mechanism that automatically performs customization or tailoring of the system. An adaptive system
relieves users from the task of manually adapting the system, as well as the requirement to have deep knowledge about the system.

1.2 Problem Domain

When moving from singular computer systems to Open Service Architectures, the task of adapting system functionality, dialogs, and interfaces to the user becomes very complex. There are several reasons for this.

- **Short life cycle** – individualizing software takes time and requires effort, either from the user or from the system. The user might not want to spend the time required to manually individualize a service if it only will be used once and for a short period. In addition, services might not get the amount of interaction with the user that is required to automatically select appropriate adaptations.

- **Many entities to adapt** – with a multitude of services to chose from, and frequent collaboration between services, the total number of entities that the user encounters will be large. The user cannot be expected to manually individualize new services if such are encountered frequently.

- **More of the same** – the information that adaptations is based upon is expensive to gather. This is true whether it is done automatically by the system or if the user manually enters it. However, this information is not always unique to a specific service. A user’s preferred language is an example of a preference that is likely to be stable across all services of the user. If adaptations have to be made individually for each service, a lot of work will be doubled.

- **Protection of information** – the user might not want to share all information that is needed to perform an adaptation with individual services. Another challenge is to reduce that need for sharing of information while still performing adaptations.

In addition, the information that controls an adaptation is generally distributed among several actors within the OSA (including the user). Therefore, the distribution of the process of performing adaptations must be controlled.

In this thesis, I will argue for a solution to these problems by introducing an adaptation coordination agent. Such a component could be used to reuse information about users from services that the user previously has subscribed to. It could also constitute a uniform interface for controlling adaptations made by multiple services, in order to reduce the user’s efforts. Lastly, it could hide user information from services, information that is needed when performing adaptations.

1.3 Contribution

The main contribution of this thesis is the presentation of a conceptual framework for synchronization and control of adaptive functionality in OSAs. The framework has three main components.

- Identification of the main roles that agents can take on when collaborating about performing adaptations.

- An analysis of the fundamental classes of information that agents need to exchange in order to coordinate adaptations.

- An analysis of the adaptation process itself.
The usage of the framework is exemplified with a section that describes the KIMSAC system.

1.4 Structure

This thesis consists of three sections. First, in section 2 Background and Literature, the background of this thesis is discussed and two major fields of research are introduced, Open Service Architectures and Adaptive User interfaces. A special case of Adaptive User Interfaces, Adaptive Help, is also discussed since this is the topic of the KIMSAC system example in the thesis.

Second, in section 3, A Framework for Performing Adaptations in OSAs, a framework for synchronization and control of adaptive functionality in OSAs is presented.

Third, in section 4, Adaptive Help in KIMSAC, an example of how the framework of section 3 can be used when designing adaptive OSAs is given.
2 Background and Literature

In this thesis, findings in two major research fields are combined: *Open Service Architectures* and *Adaptive User Interfaces*. In the following, I introduce these fields by referencing state of the art research and giving examples. *Adaptive Help*, a special case of *Adaptive User Interfaces*, is also treated as this topic closely relates to the example in section 4.

2.1 Open Service Architectures

An Open Service Architecture handles a collection of services with the following minimum requirements (Waern, 1998).

- The architecture must support an open set of users.
- The architecture must support an open set of services; i.e. it must be possible to develop and add services without knowing anything about other available services.
- There must exist a means for users to access the services.

In the above definition, an open set of users and services refers to the possibility to add and remove users and services of an OSA. This without having to modify either individual services (other than the one that is directly affected) or the architecture as a whole. Such a requirement does not necessarily mean that restrictions on the set of users or services cannot be enforced; it would for example be perfectly reasonable to restrict the services in an OSA to deal with the selling of cars only.

Examples of OSAs

Only a few systems as of today fit the above outlined definition of an OSA. Following are a few examples.

- **World Wide Web (WWW)** – Users with access to the Internet and a WWW browser can access services simply by specifying an HTTP address. Services can be virtually anything ranging from simple HTML documents to highly advanced search engines, stock market brokers, airline reservation systems etc. Services can be added to the architecture without any affect on other services at all. However, it is unfortunately not as simple to remove services, a problem (stale links) that is causing the WWW community great pain.

- **MarketSpace** – Eriksson, Finne, & Janson (1996) outlines the first steps towards an open agent-based market infrastructure. This system would allow any user to enter the market, and anyone should be able to offer any kind of service (e.g. brokering).

- **Cooperative Information Agents (CIA)** – Verharen, Dignum, & Bos (1997) describe an architecture and implementation that can be classified as an OSA. By extending the paradigm of Cooperative Information Systems (CIS) to make each CIS autonomous, capable of handling tasks and contracts, they get what they call a CIA. A group of CIAs, where one or several CIAs offer a service, constitute an OSA in which the services are able to collaborate with an open set of users, as well as other services (CIAs).
2.2 Adaptive User Interfaces

As an attempt to overcome the many usability problems of static and flexible user interfaces, especially complex ones, adaptive user interfaces have been explored for some years. The idea is to help users with adapting the user interface of a system to suit their needs, either by doing it automatically or by offering assistance in the task. Adaptations may also affect application functionality.

Adaptations of user interfaces can be made at several different system levels. Changes of physical aspects of a system constitute one such level. The restructuring of interface components such as menus, screens, windows etc., is an example of a physical adaptation. Another possibility is to adapt the very task that the system is to perform.

Browne, Totterdell, & Normann (1990) present a categorization of adaptations.

- **Adaptive** – “Hard wired” mappings from the set of states to the set of possible adaptations.
- **Self-Regulating** – The adaptive system includes functionality for evaluating an adaptation after it has been executed, which is used for improving following adaptations.
- **Self-Mediating** – The adaptive system evaluates the adaptation in advance on a model of the interaction, i.e. adaptations are simulated before they are executed.
- **Self-Modifying** – In addition, the adaptive system includes functionality for changing the model that is used for evaluating adaptations.

The above categorization is listed in increasing complexity, which is directly related to the effort that is required for designing and implementing such systems. It is worth noting that hardly any Self-Mediating or Self-Modifying systems exist. The help functionality in the KIMSA C system (see section 4) can be described as an Adaptive system according to the above definition. An extension that would make the adaptive system Self-Regulating is described in section 4.8.

Kühme, Dietrich, Malinowski, & Schneider-Hufschmidt (1992) further define terms and aspects of adaptive user interfaces, and survey related work and projects. In their paper, they also give a scheme for describing adaptation configurations in two dimensions: the phases of the adaptation process, and the actors that participate. For describing the adaptation process, Kühme et al. proposed the following phases.

- **Initiative** – One of the actors suggests a need for an adaptation.
- **Proposal** – Adaptation alternatives are proposed.
- **Decision** – One of the alternatives is chosen.
- **Execution** – The adaptation is executed.

In their discussion, Kühme et al. mention the user and the system as the most interesting actors in the adaptation process. They also mentioned the system designer, the system administrator, and local experts as potential actors, but disregarded them with the motivation that the user would not know the difference between them and the system. This division is not sufficient for the purpose of this thesis. First, speaking of the system is misleading, since OSAs are built on an open set of services. The division of actors must reveal the fact that the system is no longer a well-defined and singular entity, but a heterogeneous and open set of services.
Second, if the user is to be able to maintain any control of the adaptations, a central access point in the adaptation process is needed. No additional actions have to be taken in order to achieve this in a singular system, but in an OSA, we will need a third actor for this task.

Benyon (1993) describes adaptive systems using a model-based approach, in which he identifies three main information models: the user model, the domain model, and the interaction model. In section 3.2, I present a modified version of this division. Similar to the categorization by Kühme et al., Benyon assumes two actors: the user and the system. For the same reasons as given above, this categorization is not enough when dealing with OSAs.

Considerations that have to be taken when designing the very adaptations of an adaptive system are not discussed in this thesis. Usability aspects of adaptive systems are not treated either. For a discussion on these matters, see Höök (to appear) and Jennings, Benyon, & Murray (1991).

### 2.3 Adaptive Help

As system availability and complexity grows, providing the user with adequate help becomes increasingly important. However, users' need for help vary greatly, both over time for individual users as they learn the system, and over the set of users as background and cognitive skills influence users' need for help. Surely, a help system that for every given point in time provides the user with exactly the right help, nothing more and nothing less, must be advantageous to a system that always provides the user with the same help. This is especially true as help systems themselves grow in complexity, often with a need for a help system for the help system consequently (search tools, topic selection, index, etc.).

The help system outlined above is yet to be seen; the techniques and theories needed for such a system simply do not exist, they are most likely utopian. Nevertheless, a help system can gain from less perfect adaptivity as well. By restricting the help that is shown to the user to topics that relate to the user’s current interaction, the user’s need to navigate the help space can be reduced significantly. This is much simpler to accomplish than to foresee the user’s true lack of knowledge and to map that information to the correct help item.

The situation of being overwhelmed with help that one does not need is almost as bad as having a hard time finding adequate help. An example of such a problem might occur with proactive help, where help is pushed to the user rather than provided when the user asks for it. For that purpose, proactive help systems often adapt the amount of help they provide to the user. In the simplest form, this can be as easy as allowing the user to disable help for tasks that the user is familiar with. However, if two or more services have individual help systems but offer the same task to the user, a user that is fluent in performing the task in one service might still be considered novice when for the first time performing the same task in another service (due to the second system's lack of experience with the user). By identifying the task as identical (or similar), the help system of the second service can adapt to the correct help level automatically.
The Microsoft® Office Assistant

An example of a commercial adaptive help system is the Microsoft® Office Assistant. The following description of the Office Assistant features can be found on Microsoft's website:\(^1\)

Breakthrough technology that unifies online user assistance, lets users ask questions in their own words, and interactively provides helpful suggestions.

The latter part of the description refers to an anthropomorphic agent that proactively suggests tips on using application specific features, keyboard shortcuts, and effective mouse usage. The user's most recent interactions with the system are used as the base for the selection of hints to display. The assistant also lets the user search the help database using free text queries.

The Office Assistant works as an add-in with all programs in Microsoft's Office 97 suite (word processor, spreadsheet, database manager, etc.), i.e. it operates in an environment that is quite similar to an OSA. Properties of the help system, such as for example which types of hints to display, can be set from any program that makes use of the assistant. These changes also affect the assistant's behavior when used from other programs.

However, the Office Assistant is not truly integrated.

- While all applications that make use of the Office Assistant share the user preference database, they have private versions of the model that constitute the base for tips selection. This in spite of the fact that several suggestions apply to more than one application.
- The suggestion selection mechanism and the free text search engine are completely separate components. A possible improvement would be to include knowledge from the suggestion selection mechanism when resolving free text queries, and using previously made free text queries when analyzing the user's need for tips.

The introduction of an adaptation coordinator in this thesis provides a possible solution to these problems. In the first case, this is done by storing the model for suggestion selection, shared between services, with the coordinator. In the second case, a mapping from the model for suggestion selections to the free text engine can also be assigned to the adaptation coordinator.

---

\(^1\) www.microsoft.com
3 A Framework for Performing Adaptations in OSAs²

This section will analyze how OSAs and adaptive technologies can be combined, and what is needed for doing so. The discussion is broken down into four parts. First, I suggest a set of actors that participate in adapting information in OSAs. Then information models that actors need to agree upon before adaptations can be realized are discussed. An analysis of the task of performing an adaptation, with a division of the adaptation into adaptation phases, follows. Last, the adaptation lifecycle is discussed.

Following is an example of a very simple adaptive user interface in an OSA; namely, the adaptation of colored links in WWW browsers. The example highlights many interesting aspects of adapting information in OSAs, and throughout the analysis, it will be referenced repeatedly.

Example
Most WWW browsers automatically adapt the color of links to whether the user has followed them or not. The color might for example be blue if the link has never been followed (or if a certain period has lapsed since it was most recently followed), and red after the user has explored it. Several ways to modify this adaptation exist, one being that of letting the user select the colors representing followed and not followed links respectively. A second is to let Service Providers define colors of their own, overriding the user’s settings. A third way to modify the adaptation could be to let the user decide whether services should be allowed to override link color settings or not.

Throughout the following discussion, I will refer to models of various entities (users, adaptations, service domains, etc.). I use the concept of models in order to present a simplified view of some entity. This view should highlight interesting aspects of the entity, but suppress unnecessary details.

This use of models frequently occurs in literature when discussing domain analysis and the design of computer systems. The models need not necessarily map to data object types or components of an implementation.

3.1 Actors

The first part of analyzing the process of adapting information in OSAs is to identify which actors participate, and what their contribution to the process is.

Categorizations of actors in adaptive systems similar to this occur in literature, either explicitly or implicitly (Benyon, 1993; Kühme et al., 1992, see also section 2.2). However, as discussed in section 2.2, they fail to address the issue that is central in OSAs: the support of an open set of both services and users. For that reason, I suggest three main categories of actors in OSAs: Users, Service Providers, and Adaptation Coordinators.

Note that although the categories of actors are most easily viewed as strict (i.e. the categories are not overlapping), the definition could be extended to allow otherwise. For example, nothing would prevent the Adaptation Coordinator to also be a Service Provider; i.e. providing means for processing adaptations could very well be a service in itself. In addition, if the OSA is agent based with agents representing

² A summary of this section is published in Bylund and Waern (1998).
Service Providers, a Service Provider could equally well make use of another to fulfill its goal; i.e. a Service Provider could also be a User. This extension would merely make the categorization of actors a description of roles that need to be played in the process of adapting information in an OSA. However, in the following discussion I always assume users human, just as I assume Service Providers separate from Adaptation Coordinators.

Users

In this analysis, I am viewing the User as just another agent that participates in the adaptation process. For this task, the user could need assistance with representation towards the other groups of agents in the OSA. Within the EdInfo project (Waern, Tierney, Rudström, & Laaksolahti, 1998), such a user representative was implemented in the form of a Personal Service Assistant (PSA). The PSA is used to help the user to select services, for which purpose the PSA maintains information about which services the user subscribes to.

To enable service interaction, it is useful if the system-user interaction can be integrated. A plain window system suffices to present the interfaces to different services, but it is desirable that services also are enabled to integrate their presentation into a single presentation. Espinoza (1998) describes how this can be solved with the introduction of an Interaction System (IS). The main tasks of the IS are to render different services' presentations, and to interpret user actions. The IS may also include functionality for sharing of information, user interface components, and models of its environment, between services. The user interface is now very much the user's private interface to the OSA, as no single service can claim a better right to it than any other. The user is also free to constrain the interaction with services using one entry point only, immediately affecting the presentations of all services at once.

Example

The prime example of an interaction system is a WWW browser, through which services of all kinds share a common interface. The browser knows how to render presentations specified in HTML directly, but can also be equipped with a set of plug-ins (shared between services) for rendering of presentations in other specifications (Shockwave°, VRML, Adobe°, etc.).

A first version of Espinoza's IS was implemented as a component of the KIMSAc system (see section 4). In this system, agents are free to render presentations using graphical user interface building blocks called Content Handlers, for which the IS includes functionality for automatic layout. Content Handlers may be provided by the IS for any agent to use, or custom made and supplied by individual agents. They are highly parameterizable components that are individually addressable from agents. Content Handlers can also be responsible for handling user input, in which case they include functionality for this.

Service Providers

A Service Provider is responsible for the content and functionality of a service. Furthermore, Service Providers should control the interaction required when a user is accessing a service. Some of this work can be delegated to an IS, but it must always be initiated by a Service Provider.
Since Service Providers control the interaction with the user, they are bound to be the source of most of the knowledge about the information and behavior that can be adapted to the user. This knowledge can be either implicit in the implementation of Service Providers, or explicit, expressed in some knowledge representation format. In the continuation of this thesis, I will refer to this information as a subset of the service domain of a service.

**Adaptation Coordinator**

The task of the Adaptation Coordinator is to coordinate actions and knowledge between the User on the one hand, and Service Providers on the other. It maintains information that otherwise would be duplicated in several services. General information about the user is an example of such information that an Adaptation Coordinator can maintain. In the discussion that follows, I will find other kinds of information that fall into this category.

The Adaptation Coordinator may also perform adaptations if they are common between a set of services, or for other reasons are delegated to the Adaptation Coordinator by a service.

There are many reasons for why parts of the adaptation process should be delegated.

- The delegation relieves Service Providers from having to learn, represent, and reason about its users, a potentially very costly process.
- User Models can be based on interaction between the user and more than one Service Provider, which might lead to richer models.
- Since information about the user is shared between several Service Providers, the user need not enter the same information repeatedly.
- Information that is crucial for performing adaptations, but users feel is sensitive, may be hidden from individual Service Providers.

In Kobsa & Pohl (1995), a user modeling shell system called BGP-MS is described. This system is capable of modeling much of the information that is needed when adapting interactions between a computer system and its user. Models can be organized in so called partition hierarchies, which allows for reasoning about multiple service domains and models of the user at the same time. It also includes maintenance mechanisms for keeping the models consistent, and inference engines that may be used to trigger adaptations. The BGP-MS system is an example of a platform that could work as a base for an implementation of an Adaptation Coordinator.

**Example**

In the example mentioned earlier in this section, it is easy to identify the different actors. The User is of course the user of the WWW browser, using the browser as an interaction system that renders presentations of services, and interprets user input for forwarding to Service Providers. The functionality of the Adaptation Coordinator is integrated with the browser, and Service Providers are WWW recourses (e.g. web pages) on the Internet.
3.2 Information Models

To allow agents to take on these roles in collaboration, they must share common views on some classes of information. This common understanding may be built in, or agreed upon before executing adaptations.

Following is a division of the information models that together describe an adaptation: Domain Models, User Model, and Adaptation Model. While the division is influenced by Benyon (1993, see also section 2.2), I have switched focus from design aspects of adaptation to the matters of how the adaptation process can be distributed.

Domain Models

In order to perform an adaptation, a great deal needs to be known about the domain in which changes are to be made. Domain models describe adaptable aspects of services, including those for describing physical design, logical functions, and tasks of services.

When adapting services, it is important to recognize that in order to perform an adaptation at a certain level, the same level must be modeled in a domain model. For example, if a menu is to be adapted, a model describing the menu hierarchy must exist. If the set of tasks that may be performed is to be adapted, a task model is needed.

Example

In the example from the introduction of section 3, the domain model includes a common understanding of the object link. Also described are actions that can be performed on it (follow), as well as attributes of it (current_state, time_since_last_follow, not_followed_color, and followed_color).

Models of adaptable aspects in service domains are service specific and to the greater part owned by Service Providers, they are not likely to be duplicated in an OSA. However, it is often useful to store at least parts of domain models with the Adaptation Coordinator. This is true if multiple services share a service domain, in part or completely, and wish to delegate the task of performing adaptations to the Adaptation Coordinator. This would let the user control an adaptation that applies to multiple services from one entry point only.

Another reason for storing parts of domain models with the Adaptation Coordinator is that the user modeling capabilities of the Adaptation Coordinator may require domain-specific knowledge to allow for making inferences.

User Model

The User Model describes attributes that the system can adapt to; in particular, aspects of the user. Models that describe aspects of the user include profiles, cognitive models, and student models (Browne et al., 1990 pp. 86-91).

User Models can be implemented with general components such as the BGP-MS system mentioned in section 3.1. However, sometimes User Models are not easily expressed in a general form. For example, Malinowski (1993) describes a system that models the user's use of an interface as the base for adaptation of shading and coloring of user interface components. This adaptive system shades components that have not been used for a long time, and colors fields with values not commonly used red. Malinowski's User Model constitutes a very shallow description of the user, a model that is very specific to a particular service.
Example

Continuing the example from section 3, the model for describing what to adapt to includes all attributes that are needed to decide the color of the link (i.e. the adaptation). These attributes include one set of values each for representing: default values, the user’s values, and Service Providers’ values. Also defined is an attribute that states whether the user accepts Service Provider overrides or not. Note that this model need not be completely distributed between the actors. Only the Adaptation Coordinator needs to access all parts of it.

One of the requirements of an OSA (see section 2.1) is that an open set of users must be supported. This raises the issue of how the user is modeled: the adaptive system may model each user individually, users may be categorized to belong to a group that is modeled, or all users of the OSA may be modeled together (Calistri-Yeh, 1991).

General parts of the user model naturally reside with the Adaptation Coordinator. However, even in the case when the User Model is specific to a particular service domain, there may be reasons to store it with the Adaptation Coordinator. It is always desirable to let users have control over their own models. A user should for example be able to specify what information about itself that a service should be allowed to share with other services. In some cases, the user might even want to hide the User Model from the very service that the adaptation stems from. If the Adaptation Coordinator is implemented as a component that the user can trust, the user can store information with it instead of with individual services. Given that the Service Providers agree to delegate the necessary parts of their domain models, the Adaptation Coordinator can perform the adaptation in place for the service.

Based on this discussion, it seems like a good idea to place large parts of the User Model with the Adaptation Coordinator. This makes for a situation where adaptations that are directed towards the user can be based on a uniform model, no matter which Service Provider is responsible.

Adaptation Model

The Adaptation Model describes mappings from attributes of the User Model, to attributes of the set of Domain Models. These mappings constitute descriptions of the very adaptations that the adaptive system performs.

The Adaptation Model also includes mechanisms for detecting when a certain adaptation should be triggered (i.e. a mapping from the User Model to the Adaptation Model itself). Adaptation Models of self-regulating, self-mediating and self-modifying adaptive systems (Browne et al., 1990) would also include functionality for evaluating the effect of an adaptation, together with functionality for adapting the adaptations themselves (i.e. mappings both from and to the Adaptation Model).
Example

The Adaptation Model includes an adaptation function as follows:

```java
if(isDefined(sp_not_followed_color) &
   isDefined(sp_followed_color) &
   allow_override==true)
   if(current_state==followed)
       setLinkColor(sp_followed_color)
   else
       setLinkColor(sp_not_followed_color)
else if(isDefined(user_not_followed_color) &
   isDefined(user_followed_color))
   if(current_state==followed)
       setLinkColor(user_followed_color)
   else
       setLinkColor(user_not_followed_color)
else
   if(current_state==followed)
       setLinkColor(default_followed_color)
   else
       setLinkColor(default_not_followed_color)
```

The information model must also describe the attribute `current_state`, which holds information about whether the link has been followed or not.

The Adaptation Model can exist in both service domain specific and general forms. Service Providers generally hold knowledge about adaptations that are specific to the service, e.g. “if the user is unidentified, disable secure transmission”. The Adaptation Coordinator may complement such information with general knowledge about adaptations, e.g. “activate help if the user makes three consecutive errors when interacting with a new service”. In most cases, we would expect adaptation models to be hard-wired into special purpose agents, such as an Adaptation Coordinator, that realize adaptive help.

3.3 Adaptation Phases

When moving from singular adaptive systems to adaptive OSAs, the number of ways to actually perform adaptations grow immensely. In this section, I make a comparison between adaptive OSAs and traditional adaptive systems, suggesting a way to classify adaptive OSAs. I then introduce the concept of Interaction Schemes, which in detail describe the process of performing adaptations in a distributed fashion.

Classifying Adaptations

To classify adaptations, I extend the analysis by Kühme et al. (1992) to include an Adaptation Coordinator, and replace `system` with a set of Service Providers. By doing these adjustments, I get a scheme for describing adaptation configurations in OSAs (see Figure 1).
This model of adaptation configurations partially determines how the models described in section 3.2 are distributed between actors. For example, the example configuration that is pictured in Figure 1 indicates that the Service Providers must possess large parts of the Domain Model. Otherwise, it would be difficult to actually execute the adaptation. The Adaptation Model that describes what is triggering the adaptation and the direct mapping from the User Model to the Domain Model must be held by the Adaptation Coordinator. For this purpose, the Adaptation Coordinator as well as the User need to share an ontology of terms with the Service Providers for describing the adaptation.

While this model holds for describing individual adaptation configurations, most adaptive systems include more than one adaptation configuration. In order to classify the adaptive properties of an OSA with enough precision, we need a classification scheme that defines the complete space of possible (and reasonable) configurations. This is achieved by adding a dimension, the Degree of Distribution of the OSA, to the multi-dimensional classification scheme proposed in Kühme et al.

The six configurations (see Figure 2) in the x-y plane cover the configurations that Kühme et al. proposed as the most interesting set of configurations. A configuration is arranged in two dimensions: System Intelligence for Context Analysis and Plan Recognition, and System Intelligence for Proposal Generation and Evaluation. When the Degree of Distribution dimension is added, we get the description of the complete space of available adaptation configurations in OSAs.

Along the Degree of Distribution axis, the distribution of the adaptation process between the Adaptation Coordinator and Service Providers gradually increases. This happens as the phases of adaptation shift from being preformed by the Adaptation Coordinator (at orig.), to being done, completely distributed, by the community of Service Providers (to the left in Figure 2). Therefore, the Degree of Distribution is a measurement of how the adaptation process is distributed between the Service Providers and the Adaptation Coordinator.

Note that some of the configurations, such as the one were the user executes all actions, are not affected by the Degree of Distribution. This also highlights the fact that the user is not directly effected by the Degree of Distribution.
In section 0, an example of an application of the multi-dimensional classification scheme is given.

**Interaction Schemes**

The actors that participate in the adaptation process must agree upon which interactions are necessary in order to complete the adaptation. I will refer to such an agreement as an *Interaction Scheme*. This construct constitutes a detailed description of the interaction that takes place between the User, Service Providers, and the Adaptation Coordinator, while performing an adaptation.

An Interaction Scheme should describe the interactions required in order to perform an adaptation in terms of:

- the actors that participate in the adaptation process,
- the message sending involved, together with information about the sender and recipients of each message, as well as the order and timing between messages, and
- pre and post conditions of the interaction as a whole, but also of individual messages.
An example of an Interaction Scheme, taken from the KIMSAI project (Charlton et al., 1997b), is illustrated in Figure 3. In this example, the User interacts with Service Providers via an Interaction System (see section 3.1 - User), and the task is to adapt the selection of terms in a glossary to the content of the workspace of the Interaction System. The adaptation is triggered when the User requests extended help, an action that will expose an option for the user to further request a context sensitive glossary. Multiple Service Providers can sign up to contribute with terms for the glossary, and the compilation of the selection of terms is coordinated by the Adaptation Coordinator.

While the Interaction Scheme illustrated in Figure 3 could certainly be used in the design phase of an OAS, it would hardly be suitable as notation for agreeing upon

Figure 3. An example of an Interaction Scheme taken from the KIMSAI system (see section 4). Horizontal arrows represent messages being sent between actors. Arrows with a single head represent a single message, and arrows with two heads represent multiple messages. Most pre and post conditions have been left out.
interactions dynamically. Labrou (1997) introduces the concept of conversations as a means for controlling threads of interactions between agents speaking KQML\(^2\) (Finin, Labrou, & Mayfield, 1997). Examples of conversations implemented in Prolog as definite clause grammars, complete with constraining clauses, are presented. However, the intended use of conversations does not quite map to the use of Interaction Schemes as described above. For example, conversation policies are designed for handling two types of actors at the same time, while Interaction Schemes need to handle three. Another issue is that conversation policies are an important part of the definition of the semantics of KQML, i.e. they constitute very static constructs that are not easily modified. While the Interaction Schemes described above need not necessarily be negotiated between the actors at runtime, it is desirable to define them in a way that makes them at least modifiable. This, however, would require a notation for describing Interaction Schemes that does not exist (see section 5.2).

### 3.4 Adaptation Lifecycle

In an ideal OSA, the information models discussed in section 3.2 would be defined at runtime as services and users join the OSA. This would also be true for the Interaction Schemes discussed in section 3.3, as different actors in the OSA would be able to negotiate new schemes for interaction as new situations emerged. This would lead to a truly open system, in which adaptations of information, behavior, and functionality would be added and removed as the OSA changed. However, a system that is fully open, both to the set of users and the set of services as well as to underlying concepts such as Interaction Schemes, is very costly to implement\(^4\). In reality, designers of adaptive systems must compromise between openness and the cost of implementation.

One choice that has to be made during design is when ontologies of information models and Interaction Schemes should be defined. Following, I identify three possible points in time when this might be done: during the design phase, at system startup, and at runtime.

#### During Design

The simplest solution is to define all models, including the Interaction Schemes, at design time. If this approach is used, a new service can introduce itself to an Adaptation Coordinator simply by stating the domain models it is able to handle. This introduction is in itself a very simple negotiation, similar to the KQML approach, where agents publish themselves to a facilitator by stating their competencies (Finin et al., 1997). We could envision that the Adaptation Coordinator is allowed to reply 'no' to this information, implying that Adaptation Coordinator is unaware of the domain model that the service has mentioned.

An obvious drawback with this option is that all ontologies need to be set when designing the system. In practice, this is a very disturbing limitation; the open nature of OSAs makes it likely that new concepts and service domains will be introduced. New aspects of the user to adapt to might also show up, as well as services with new

\(^2\) Knowledge Query and Manipulation Language

\(^4\) Another issue is of course that such a degree of openness might not be desired at all, no matter how easy to implement.
requirements of the Interaction Schemes needed to support their type of adaptation respectively.

Example
The colored links adaptation is an example where ontologies and the interaction scheme are fixed during design. This makes the implementation very straightforward. Service Providers do not even have to participate in the adaptation process at all. If they do want to affect the process though, all they have to do is to add a few statements to the HTML code that specifies how the service is rendered. However, if a Service Provider wishes to extend the adaptation, for example by introducing a third color for links that the user followed more than a week ago, it would not be possible. There is no means for the Service Provider to express this to the Adaptation Coordinator or to the Users.

At Startup
A more flexible solution is obtained if agents are implemented in a general manner, so that they can be specialized by loading appropriate parts of the domain, user, and adaptation models. The models are then stored in a separate database that is accessed by all agents upon startup of the system.

This solution allows the designers to make changes to the models that the system supports, without having to redesign and re-implement an arbitrary large subset of the agents. There are two main drawbacks of the method, though. First, obviously the option is only open to systems that can be shut down and restarted each time a novel type of service is added. Second, all agents must share a common language for the descriptions of these models, so that they are able to load appropriate parts of the domain models at startup. As discussed in section 2.1, a useful requirement of an OSA is that developers can develop services independent of each other, adhering to as small a set of standards as possible. It may be difficult to incorporate this method for specialization, and still adhere to this principle.

The Kimsac system (Charlton et al., 1997b) was made open for maintenance using this technique.

During Runtime
The most flexible solution is found if agents are allowed to negotiate service contracts at runtime. In this case, agents must share a communication language in which they can negotiate parts of the runtime service contract prior to executing it. This alternative, however, requires very rigid ontologies for describing information models and interaction schemes, including methods for signing contracts that in detail describe different actors’ obligations. A discussion on negotiation of service contracts can be found in Bylund & Waern (1998).

3.5 Summary
In this section, I have analyzed the process of adapting information, behavior, and functionality in Open Service Architectures. I began with identifying the actors that participate in the process of performing an adaptation (User, Service Providers, and Adaptation Coordinator), and describing their roles respectively. I have also divided the information that describes an adaptation into three different models: Domain Models, User Model, and Adaptation Model.
In section 3.3, an existing model for describing adaptation configurations was extended to take into account the full set of actors in an OSA. Using this model, I defined a scheme for categorizing adaptive OSAs in three dimensions (System Intelligence for Context Analysis and Plan Recognition, System Intelligence for Proposal Generation and Evaluation, and Degree of Distribution). Further more, I introduced the concept of Interaction Schemes, which describe the interaction between actors during the process of performing an adaptation.

Finally, I discussed the issue of when models that describe adaptations can be defined.
4 Adaptive Help in KIMSAC

In this section, I describe an implementation of a service architecture, KIMSAC (Kiosk-based Integrated Multimedia Service Access for Citizens), that makes use of adaptive help. The analysis in section 3 was performed during the early phases of designing the infrastructure that is used for adapting help in KIMSAC, and it constitutes the base for the current implementation.

This section is meant to exemplify how the categorizations, techniques, and models introduced in the previous section can be used when designing mechanisms for handling adaptations in OSAs.

4.1 Overview

KIMSAC was an EU ACTS supported program, which aimed at providing citizens with flexible means for accessing complex information using public information kiosks (Charlton et al., 1997a; Charlton et al., 1997b; Charlton et al., 1997c).

The perhaps most important goal for the project was to enhance usability of kiosk-based information systems to allow for otherwise complex services by applying new technologies. Intelligent agent technology is one example of such technology, which was being employed with the motivation to increase the efficiency of the dialog between the user and services. Hopes were that this could be realized by adapting the information to the user to be as relevant as possible.

KIMSAC was built upon the Agent Services Layer (ASL), a proprietary agent platform developed at Broadcom Eireann Research Ltd. (Kerr, O’Sullivan, Evans, Richardson, & Somers, 1998). Based on CORBA distribution technology, ASL provided the developers of KIMSAC with an environment that is transparent both in terms of distribution and implementation language.

The main user groups for the application itself were unemployed people and people in social welfare programs. It supported search for personalized information in the domains of unemployment benefits, job search, and training. Each one of these three domains was implemented as a separate service. During the second trial period, the system was (to a limited extent) available to citizens via public kiosks in places like government offices, libraries, shopping malls etc.

4.2 User Interface

The user interface of the KIMSAC system is quite restrained with respect to what the user can do. By constraining the interface, the interface designers could control the user’s task, which made it easy to provide relevant help. It also made it easier to foresee problems that the user might experience, since the means for improvisation is limited.

Another reason for tight user interface constraints is that no keyboard or mouse is available to the user, but only a touch-screen. This fact greatly reduces the practical amount of input that a user can produce. Different interface technologies that would relieve these limitations were explored. An example is a hierarchical structure of buttons (see Figure 5 - Profile Forms) as a replacement for form filling procedures. The KIMSAC system can also automatically supply forms with available user data, by identifying the user using a card reader, and fetching user data from remote databases.
The following user interface components are present at all times (Figure 4).

1. **Kim button** – The user can press this button at all times for access to a *Glossary of Terms*, *Where to Next* help and *How It Works* help. Instruction video clips are also shown here.

2. **Instruction text area** – In this area instruction text will appear. This area is also used for error and warning messages.

3. **Header area** – Here a title of the current screen is visible. A short version of the title is used in the *Navigation bar* when stacking previous screens.

4. **Workspace** – A generic area that is free for services to use, most of the interaction between services and the user takes place here.

5. **End button** – By pressing this button, the user exits the current session and a new is initiated.

6. **Navigation bar** – This component works as a stack for previously visited screens. As the user navigates the *Workspace*, titles of visited screens are pushed to the stack as buttons. At any time, the user can pop the stack to return to a previous screen by pressing one of the buttons on the bar.
7. Yes button – The semantics of this button changes over time as it is up to individual services to program what it should do. However, most often it is programmed to acknowledge some interaction in the Workspace.

The user is interacting with the system by navigating in a set of screens (Figure 5), where each screen is designed to solve a well defined task (e.g. select a service, specify your profile, or browse search results). The user navigates downwards in the screen structure by pressing buttons in the Workspace (e.g. when selecting service or identification procedure). The Navigation bar can be used to take the user back to any screen higher up in the hierarchy. In some screens (e.g. Profile Forms), the Yes button also brings the user back to previously visited screens, in which case the user both navigates and performs an acknowledgement of input at the same time.

The Help screen (Figure 4) can always be reached by pressing the Kim button.

The user can end a session at any time by pressing the End button, which resets the system and takes the user back to the Service Selection screen. By ending the session, all user information (profile data, identification information, usage history, etc.) that the system previously held is reset. If the user wishes to initiate a new session, identification procedures need to be performed again.

Most screens are shared by different services, although some details of their content may differ. For example, the task of specifying a user profile is similar for all services, but the values that are required for a job search might be different from the ones that are needed to sort out income support issues. The fact that services share screens is useful when adapting proactive help to the user's history with the system. A user that repeatedly has modified a profile for one service is not likely to need further assistance with completing the same task for another service, given that only the profile values differ. This of course only refers to the task of modifying a profile, not to the semantics of values in the profile.

The sharing of screens also makes it possible to streamline help content for the screens in question. This guarantees that the user never needs to be exposed to several different versions of a help item, when solving an identical task.

The benefits of having services sharing screens can be illustrated by highlighting the lack of this feature on the WWW. Today, many services (e.g. web mail servers, product support, and web shops) on the WWW require their users to identify themselves with a username and password. However, there is yet to be seen a login screen that works in the same way as another. Identification screens differ from each other in terms of which characters are allowed in the username and password, key assignment (e.g. TAB for jumping from username to password, and ENTER as default for the OK button), usage of terms (e.g. UID, username, userid), etc. By agreeing on a shared design for login screens, Service Providers on the WWW could relieve their users with the burden of having to remember trivial details on how to identify themselves.

4.3 Help in KIMSAC

The KIMSAC user can seek help from four different sources, of which three are adaptive.
Figure 5. The KIMSAC screen structure. A session is initiated in the Service Selection screen. After that, all other screens are shared among the services (the Help screen exempted, which can be reached from all screens in the structure, see Figure 4).

Instruction Text and Video

The user will be given instructions describing the purpose of the screen and how to interact with it when navigating to a screen. Instructions are available both in text and as a video clip, both revealing approximately the same information. The purpose
of presenting both text and video is that while the video clip is only showed when
the user enters a screen, the text remains throughout the visit reminding the user of
what to do. Instructions are adapted to the current screen, the actual content of the
screen, and the user’s experience with the current interaction.

Glossary of Terms
Via the Kim button, the user can request a glossary of difficult terms. This glossary is
adapted to both the current screen and the actual content of the screen\(^1\), but not to the
user. By limiting the glossary to only showing terms that are relative to the current
screen, the glossary becomes lucid and easy to browse.

Where to Next
To get further instructions on how to interact with the current screen, and on how to
proceed after it, the user may request Where to Next help. This type of help is not
adapted to the user, it is only sensitive to the current screen. As this help is not
proactive, the user has to ask for it manually via the Kim button.

How It Works
If the user is completely novice about what to do and how to interact with the
KIMSAC system, an introductory video clip is available for manual activation. The
video clip describes the main interface components and how to start a session. The
video clip is accessible at all times via the Kim button, but also shown at the
beginning of every new session. This help source is not adaptive.

4.4 Components for Realizing Help
The help system in KIMSAC is realized with 4 different (types of) components:

- the Adaptation Agent (AA),
- the Guide Agent (GA),
- *sicsDAIS* (an implementation of an Interaction System as described in section
  3.1 – Users), and
- a community of Service Agents.

In Figure 6, the screenshot of the user interface represents the set of Content
Handlers that are responsible for rendering and interpreting user interaction. Each
Content Handler roughly corresponds to the user interface components pictured in
Figure 4. The workspace is composed of three Content Handlers, one for each button.

Adaptation Agent
The Adaptation Agent (AA) is responsible for maintaining the User Model needed
for Where to Next help and Instruction Text and Video. It is also the entity that
triggers all adaptations of help content in the KIMSAC system. Most important
however, the Adaptation Agent coordinates all activities that are needed in order to
provide the user with adapted help. Therefore, the Adaptation Agent can be seen as
an implementation of the Adaptation Coordinator discussed in section 3.1.

\(^1\) Some screens may have different content over time, for example search result screens
Figure 6. An architectural overview of the help system in KIMSAC.

Service Agents

Service Agents are responsible for the main interaction with the user. While doing so, they supply the Adaptation Agent with information on the current state of the interaction with the user. The Adaptation Agent is for example updated on which service domain the Service Agent currently works in, and which screen it currently employs.

Service Agents are also responsible for suggesting adaptations of glossary help content, as well as deciding which should be executed. For this to work, each service agent must maintain parts of both a shared domain model as well as a private (to the Service Provider) user model. The reason for the latter part being private is that this model contains information that is specific to each individual Service Provider (e.g. information regarding the actual content in a dialog with the user).

Each Service Agent in KIMSAC is in itself a realization of a Service Provider as defined in section 3.1.

Guide Agent

In KIMSAC, providing help is considered a general task and the responsibility for maintaining a help dialog with the user cannot be assigned to any Service Agent in particular. Instead, there is a special component, the Guide Agent (GA), whose sole task is to control the rendering of help user interface components. However, the component does not provide any help content at all. Instead, all user interface components that it controls are instructed to wait for the Adaptation Agent to provide help content. For this reason, the Guide Agent can be seen as just another service; a service that provides help to the user, but completely relies on the Adaptation Agent for help content.
4.5 Information Models

Four different models constitute the base for adapting help in the KIMSAC system. An Interaction Model describes the user interface navigation structure and is one of two models that describe what to adapt help to. The other model is a Usage Model, which for every state in the Interaction Model describes the user’s usage of the system. The Adaptation Model describes mappings between the Interaction Model and the Usage Model on the one hand, and the Interaction Model and the Help Model on the other. The Help Model in turn describes the kinds of help that are available in KIMSAC.

These models map to the models described in section 3.2 in the following way.

- **Interaction Model and Help Model** - These two models describe different aspects of the domain and what to adapt to, they therefore make instances of the Domain Model.

- **Usage Model** - This is the User Model, however in a specialized form.

**Interaction Model**

The Interaction Model consists of three parameters that together define the state of the user interface. The first parameter (the domain id) identifies the service that is currently controlling the interaction with the user. The second parameter defines the current screen (the screen id). In order to give different help depending on in which phase of completing a screen the user is, the third parameter identifies this phase (the phase id). Phases are for example used for the form filling screens, where different items need different help texts. The triple that the described parameters make is hereafter referred to as the state of the user interaction.

The service agent that is currently responsible for the interaction with the user maintains all three parameters described above. They are communicated to the Adaptation Agent via a database that resides within the Interaction System.

**Usage Model**

For every combination of the three parameters described in the previous section (i.e. for every user interface state), the Adaptation Coordinator is gathering statistics. The following parameters are saved:

- The number of times the user has visited the state during the current session.
- The number of visits to other states since the last visit.
- The time that has passed since the last visit of the state.
- The total amount of time spent in this state.
- A listing of the glossary terms that the user has requested in this state.

At present, the Usage Model can only hold information about the current session. This limits what can be inferred about the user’s need for help.\(^1\)

---

\(^1\) See section 4.8 for a more complete discussion on limitations.
Help Model

The Help Model is not represented explicitly in the KIMSAc system, but implicitly in the way the help components are handling help content. The Adaptation Coordinator is responsible for providing the content handlers that render help content. The format of this content varies with help type.

- *Instruction Text* – the filename of a text file.
- *Instruction Video* – the filename of an AVI video file.
- *Where to Next* – the filename of an HTML file.
- *Glossary* – a set of HTML strings. One string for the overall structure and one for each term in the glossary.
- *How It Works* – the filename of an AVI file.

Adaptation Model

The adaptation model describes mappings from the Interaction Model and the Usage Model to actual help content that content handlers are to render. Different types of help are adapted with different Adaptation Models, which in turn makes use of different parts of the Interaction Model and Usage Model. In the following discussion, it is assumed that unless otherwise stated, the Adaptation Agent performs the mappings.

*Instruction Text and Video* – the mapping is performed in two steps. First, selected parameters from the Usage Model are mapped to either one of three levels of help: *none, brief, or extensive* (see section 0). Second, the help level together with the current domain id, screen id and phase id are mapped to a filename. This second mapping is hard-wired in an index file containing help properties.

*Where to Next* – the current domain id and screen id are mapped to a filename. This mapping is also hard-wired in an index file.

*Glossary* – This model is out of reach for the Adaptation Coordinator since the service agents are responsible for the proposal and decision phase when adapting the glossary. However, the Adaptation Coordinator is initiating the adaptation, and while doing so it forwards the current parameters from the Interaction Model as well as parameters from the Usage Model (a listing of the glossary terms that the user has requested in this state) to the service agents. It is then up to the individual service agents to perform a suitable adaptation of the set of terms that should be available to the user.

*How It Works* – no adaptation of this help type is performed.
Figure 7. Available adaptation configurations in KIMSAC expressed using the multi-dimensional categorization scheme proposed in section 3.3. Only the two palettes that are actually used are shown (the other 14 on the Degree of Distribution axis have been left out). The left palette describes the Glossary configuration and the right the Instruction Text and Video configuration. Note that the columns of configuration palette stand for: Service Provider, Adaptation Coordinator, and User respectively. The rows stands for the actions: initiative, proposal, decision, and execution.

4.6 Estimating the User’s Need for Help

This section describes the algorithm that is used to calculate the estimate of the user’s need for help in a given screen. The estimate is based on the weighted sum of two functions. The functions depend on the number of times the user has visited the screen during the past session, and the amount of time that has passed since the user’s last visit respectively.

\[ e = w_a \times f_a(n) + w_b \times f_b(t) \]
\[ f_a(n) = k_a \times n + m_a \]
\[ f_b(t) = k_b \times t + m_b \]

In the above equations, \( e \) represents the estimate, \( n \) the number of visits, and \( t \) the time since the last visit (in minutes). The following symbols are constants:

- \( k_a \) - counterSlope
- \( k_b \) - timeSlope
- \( m_a \) - counterOffset
- \( m_b \) - timeOffset
- \( w_a \) - counterWeight
- \( w_b \) - timeWeight
Figure 8. An Interaction Scheme that in detail describes the process of adapting Instruction Text and Video. The first part of the interaction represents user activity that constitutes the base for the usage model. The other part describes the adaptation as such.

The estimate is compared with two thresholds. If it is above a certain value $a$, the user's need for help falls into category 0. If it is below a value $b$, it falls into category 2. For a value of $e$ between $a$ and $b$, the user belongs to category 1. The different categories can be assigned more readable names by setting three system properties, they are currently set to none, brief, and extensive.

4.7 Interaction Schemes

Adaptations of help in Kimsac can be described with two adaptation configurations (see Figure 7 and section 3.3). Instruction Text and Video accounts for a self-adaptation configuration (Browne et al., 1990) that is completely controlled by the Adaptation Coordinator. Adaptation of the Glossary can also be described as self-adaptation, but the process is distributed between the Adaptation Coordinator and Service Providers.

However, as also mentioned in section 3.3, the categorization scheme does not cover the whole interaction that takes place when adapting help in Kimsac. Figure 3 and Figure 8 give a more complete picture of the process.

---

\[^{7}\text{These symbolic names are used in the index files for instruction text and video.}\]
4.8 Limitations

A few obvious help features are currently missing in the KIMSAC system. One is to give the user the option to request all currently available, or to interrupt the currently running video.

Another limitation is that if the user feels that the help level is incorrect; there exist no means for the user to correct it. Such a change of help level would provide useful feedback to the adaptive system, indicating if the adaptations work as intended. This could be used for improving the adaptation model. Such an improvement would change the adaptive system from being adaptive to self-regulating, according to the taxonomy of Browne et al (1990, see also section 2.2).

In the current implementation of KIMSAC, the lifetime of the Usage Model is limited to one session. This, of course, greatly limits the values of the adaptive system, since the system needs to relearn the user’s need for help for every new session, no matter how many times the individual user has used the system.
5 Conclusions

This thesis came to as a result of a request to implement adaptive help for the KIMSAC system; a task that required coordination of help content from an open set of services, as well as individualization of the output from that process. In order to manage this task, a framework that described the process was needed. In specific, the actors that participate in the process, the information that needs to be exchanged, and the order in which activities need to be performed had to be identified. This thesis is the formal description of that framework (section 3), and as such, it has been successful. The design of the help system was based on this framework (section 4), and it was easy to integrate the help systems with the rest of KIMSAC.

5.1 Related Work

In parallel with my work on this thesis, the Foundation for Intelligent Physical Agents (FIPA) has released the FIPA98 Specification (FIPA, 1998). This document introduces a User Personalization Service (UPS), which is an agent that is responsible for "...information about the user upon which personalized behavior can be based." The rational for the UPS is to large extent similar to the rational for the Adaptation Coordinator. Among other things, both constructs aim at:

- relieving agents (Service Providers) from the potentially complex task of acquiring, reasoning, representing and maintaining user models,
- utilizing the fact that user models can be based on interaction between the UPS (Adaptation Coordinator) and many agents (Service Providers) which may lead to a richer user model, and
- relieving the user from having to provide agents (Service Providers) with the same information over and over again.

In addition, the work on the FIPA-UPS includes an access control model that lets the user specify access control properties on information that is stored with the UPS. Ontologies that describe registration, user model handling, learning procedures, etc. are also defined. These ontologies cover much of what is needed for negotiating service contracts at runtime as discussed in section 3.4.

However, the FIPA-UPS differs from the aims of the Adaptation Coordinator in several ways.

- The possibility to negotiate Interaction Schemes with the UPS is limited, an option that the Adaptation Coordinator is open to. This is mainly due to the fixed interaction ontologies that FIPA has defined for the UPS.
- An instance of the FIPA-UPS is not specific to a user; instead, several users typically share the same UPS. This weakens the individual user’s (feel of) control of the adaptation process and the information that is stored with the UPS (Adaptation Coordinator).

5.2 Future Work

If adaptive functionality in OSAs is to work in a truly open manner, at least two issues need to be solved. First, the information models described in section 3.2 need to be further investigated; notations and ontologies for describing these models must be defined. A challenge in this work is to make these definitions open enough to
allow for future changes, but expressive enough to actually be useful. Second, runtime contract negotiation in general, and negotiation of Interaction Schemes in specific need to be analyzed.

Of these two problems, the first one is currently being investigated by for example the FIPA consortium, as well as other research organizations that attack the problem from a more general point of view\(^1\). The second problem has faced little attention so far. It is my intention to attack this problem from several angles.

- The problem needs to be described in the perspective of a concrete and realistic domain. The domain must be wide enough to allow for an OSA to be meaningful, but at the same time limited or the design and implementation work will be too big. I am currently investigating the domain of Pharmaceuticals, including the handling of information by professionals at hospitals and pharmacies, for this purpose.

- Contract negotiation within OSAs in general needs to be explored. Introductory research in this field has been described in Waern (1998), and Bylund & Waern (1998). However, practical experiences from this work is needed.

- A suitable notation for describing Interaction Schemes, including the definition of ontologies for sharing of concepts and terms relating to Interaction Schemes, needs to be described.

\(^1\) For example KQML, KIF (Knowledge Interchange Format), and Ontolingua within the ARPA-sponsored Knowledge Sharing Effort (http://www.cs.umbc.edu/kse/).
References


