Univ Access Inf Soc (2003) 2: 342–358 / Digital Object Identifier (DOI) 10.1007/s10209-003-0050-1

Using consensus methods to construct adaptive interfaces in multimodal web-based systems

Ngoc Thanh Nguyen, Janusz Sobecki

Department of Information Systems, Wroclaw University of Technology, Wyb. St. Wyspianskiego 27, 50-370 Wroclaw, Poland; E-mail: thanh@pwr.wroc.pl, sobecki@pwr.wroc.pl

Published online: 24 June 2003 –

Abstract. This paper presents a concept of adaptive development of user interfaces in multimodal web-based systems. Today, it is crucial for general access web-based systems that the user interface is properly designed and adjusted to user needs and capabilities. It is believed that adaptive interfaces could offer a possible solution to this problem. Here, we introduce the notion of the user profile for classification, the interface profile for describing the system interface, and the compound usability measure for evaluation of the interface. Consensus-based methods are applied for constructing the interface profiles appropriate to classes of users.

Keywords: Web-based systems – Multimodal interaction – Consensus-based interface adaptation

1 Introduction

Let us consider the following scenario (depicted in Fig. 1): a manager of a computer company is visiting a neighboring country for a commercial conference. To be prepared appropriately for attending this conference, the manager is using the conference portal. It is easy to guess that the conference portal is being used not only to consult the general agenda of the conference and the detailed program but also to register for the chosen sessions, as well as for cocktails and parties. Needless to say, during such conferences face-to-face meetings are arranged and they can also be appointed through the portal. Quite often the manager can be asked to meet a person who has never been met before. It may also occur that the company represented by such a person is unfamiliar too. What the manager should do is to find it in the Web and visit its pages, as some official information about the potential partner will certainly be useful.

As the manager can be offered a few meetings, the aforementioned procedures will be repeated several times. It is quite probable that the manager will also wish to meet somebody at the conference. In this case he or she will have to consult the list of the conference participants and then ask for an appointment. Thus, visiting potential partner homepages, as well as revising his or her own pages, will be quite important.

Having arranged all the basic conference details, the next thing to do is to decide on accommodation and make a reservation. Searching for an appropriate hotel or choosing one from the conference offer also involves using the Net and visiting hotel web pages. As soon as the standard, location, facilities, and price have been checked, the manager can start on-line booking procedures.

Finally, the manager, and any accompanying person, may want to spend their spare time visiting restaurants, clubs, and beaches, and going sightseeing on guided tours. To collect all the necessary information, at least several information web-based systems will have to be visited. The very last thing to do is to take the necessary steps to arrange the journey to the conference place, by consulting timetables and then booking plane or train tickets. This obviously, can also be done on-line.

Alternatively, a car may be chosen as a means of transport. While driving a car, the manager can take advantage of one of the web-based e-maps to find the optimal route to reach the destination. Possibly, he or she will take a laptop and a handheld equipped with GSM or GPRS modem. The manager will surely remember to download some documents and web pages on these devices as they can turn out to be very useful on the road.

During the journey by car, train or plane, access to web-based information systems, such as the already mentioned e-maps, train information systems, like the very well known train Travel Service *Die Bahn DB*, and other systems, may be necessary. In addition, driving a car

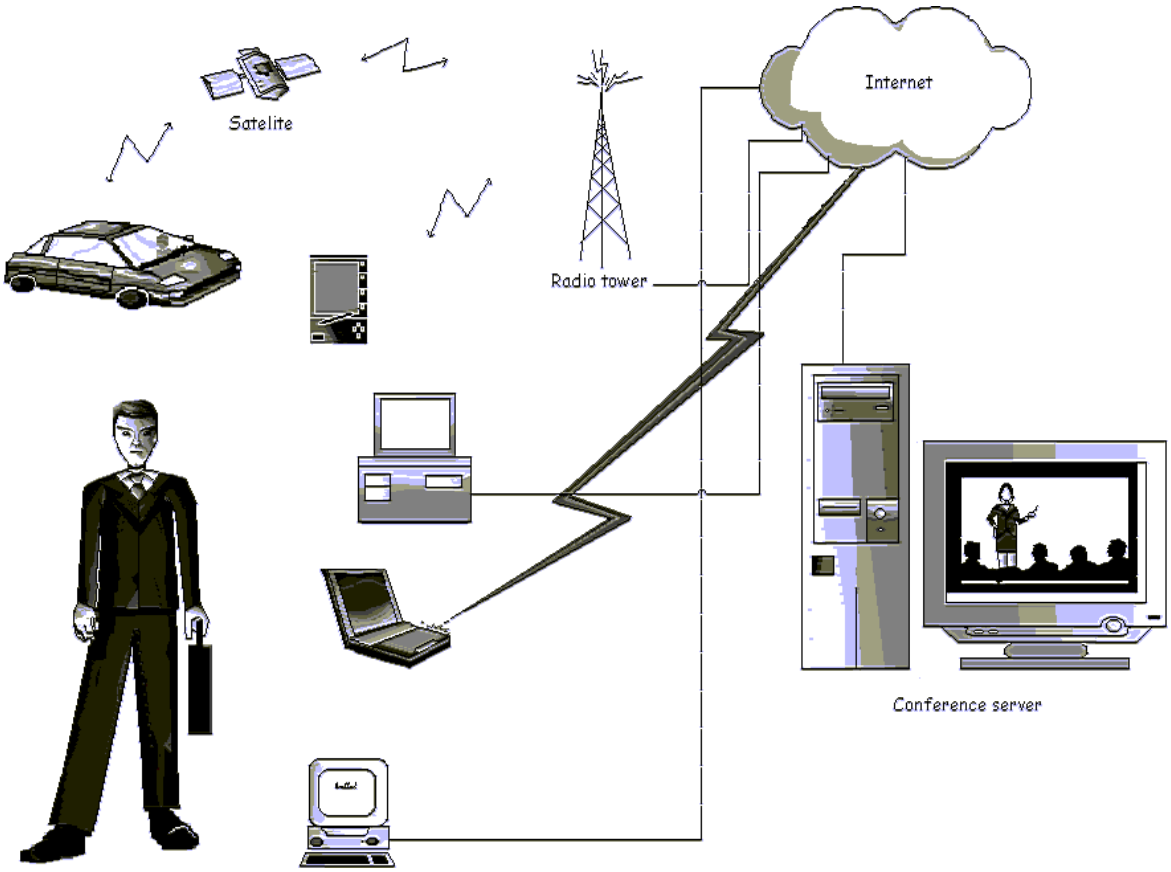

Fig. 1. Multimodal interaction of the manager with the conference information portal

equipped with GPS-based systems integrated with an e-map and up-to-date road information can be very comfortable and enjoyable. Nowadays these systems can be easily installed in the car. Even some compact cars can have these devices as an option and many higher class cars are equipped with them as a standard.

All the mentioned pre-conference preparations are only an introduction to the high-tech world. There are many other encounters to come at the conference. Firstly, at the conference registration desk the manager can use information kiosks to register and obtain further information. Secondly, the handheld and laptop can be used to improve working standards during and after the sessions. During the sessions, the handheld off-line or on-line, provided that it has mobile Internet access, enables the manager to view any information of interest, such as, for example, the detailed up-to-date conference program, the special guests' homepages, and the list of chairmen, lecturers, tutors, and other presenters. Besides, it enables the manager to look up all the incoming appointments, take notes, and read the conference papers. If the circumstances allow it, the laptop can be used instead.

As we can see, the use of all these information systems is multimodal, with different and possibly simultaneous input and output modalities. They are also accessed by means of different platforms: PC's, laptops, handhelds, navigation systems mounted in cars, and information kiosks. On these platforms many different interaction styles [23] can be used, such as direct graphical manipulation, menus, function keys, and natural language. In the interaction, different communication media can be used: typed text and handwriting, graphics, sound, speech, animation, and video. Also the input and output devices can be very different: from standard CRT and LCD to a touch screen used in information kiosks and handhelds, a keyboard, a mouse, a touch-pad, a joystick and a mini-joystick, a camera, a microphone, loudspeakers and earphones, and other devices with special keys and roller-keys.

These different interaction styles are accomplished by the aforementioned communication media and devices in all the platforms that the manager attending the conference is able to use on this specific occasion. The selection of the appropriate mode of interaction depends not only on the device used, but also on the user's general preferences and the circumstances. For example, while driving a car equipped with a navigation system, the driver can use speech input and the system response may be provided both in graphical and voice forms. However, while not driving, making use of more standard forms of human computer interaction, e.g. function keys or the touch screen as an input and the graphical output on the LCD screen, would be more natural and convenient.

As appears in the example, the same systems, i.e. the conference information portal, the navigation system, and the city information system can be accessed from many different platforms and in many different situations by means of multimodal interaction. The mode of the operation can be selected by the user or retrieved automatically from the platform's recent settings stored in the system files, i.e. cookies. So these systems have to be designed to serve multimodal interaction. What is worth noticing is that these systems, with the possible exception of the navigation system, were applied only on the occasion of the conference participation, had never been used before, and are not likely be re-used in the future by the same user. It is also obvious that the manager is not the only user of these systems and there are many other users. The user population is disparate, so it is almost impossible to design a single, equally appropriate user interface for each user and each system [34]. Today's trends in information system design, strongly supported by the European Commission 6th Framework Program [9], require further development of Universal Access systems [37].

Web-based system interfaces should be designed and implemented in accordance with proper norms [13] and guidelines [11]. This does not mean that their implementation is deterministic and leads to a single solution. It is just the opposite – these guidelines can lead to the development of very different interfaces. Differences concern not only their graphic elements, such as background, icons, and buttons, but also textual elements, sound effects, and soundtracks. Quite often these differences can be even deeper, i.e. they can lead to the implementation of different interaction styles as well [23].

The problems of interactive system design and development are relevant for many scientific and technological fields: Human Computer Interaction (HCI) and particular Interaction Styles and User Models, Artificial Intelligence (AI), Natural Language Processing, or Information Retrieval (IR), and in particular User Profiles. There are already more multidisciplinary approaches to this problem, such us Interface Agents as emerging from more general Agent Paradigm [39] and Adaptive Interfaces.

In this paper, automatic interface adaptation using consensus methods is presented. The paper is structured as follows: In the second section two current approaches to the development of hypermedia systems interfaces are presented, i.e., system personalization and interface agents. In the third section, the suggested architecture of the adaptive interface using consensus methods is defined. The section is divided into four parts. In the first part, the notion of user profiles is introduced; in the second, the method of user classification is presented; in the third, the notion of the interface profile is introduced and the measures of interface usability are discussed; and, finally, in the fourth, the user interface adaptive construction method is presented. In the fourth section, the consensus determination of user profiles is explained in detail. First, consensus methods are introduced, then the adopted consensus model is presented, and the corresponding algorithms for consensus determination for different types of attributed values are presented and referenced. Finally, consensus application to the problem of the interface profile construction is defined. In the fifth section, an example of the proposed architecture is illustrated. Conclusive remarks, along with the future work, are addressed in the last section.

2 Current approaches to web-based systems user interface adaptation

Today's information systems are getting more and more complex and their users are differentiating, so classical HCI approaches [23, 27] are not necessarily the most efficient. Also the user's information environment is becoming more and more sophisticated, so it is quite difficult for every individual to have full control over it. Application of interface personalization enables users to customize their interfaces to meet their information needs and interaction preferences. Software agents, on the other hand, enable the delegation of some of the user's tasks, especially those that are repeatedly invoked and require a lot of effort.

2.1 Interface personalization

Interface personalization can be seen as part of a more general notion of web personalization, which is defined '*as any action that makes the Web experience of a user personalized to the user's taste*' (p. 142 of [21]). A more precise definition can be found on p. 113 of [16], in which personalized hypermedia application is defined '*as a hypermedia system which adapts the content, structure, and/or presentation of the networked objects to the individual user's characteristics, usage behavior and/or usage environment*'. Personalization is a common feature of many web-based systems, Internet portals and many other information and entertainment services.

Usually, users can choose several of many interesting topics or system elements. They can also set the way in which the information is presented to them or how they can interact with the system. First of all, the user of a multimodal web-based system should choose the mode of operation. For example, PC, handheld or kiosk mode can be chosen, and then the interface layout, i.e., the specification of the location of elements on the screen, can be created. Furthermore, the user can choose one of the several offered interface templates, with, for example, different graphic (graphics, color, buttons, or background) or sound (sounds associated with different events or background music) patterns. As a matter of fact, these templates define the user interface itself. By choosing, for instance, templates with very dynamically reacting buttons, flashing colors (such as orange or shocking pink) and loud modern music, an interface likely to be suitable for younger users is created. On the contrary, when

we choose templates with rather standard buttons, toned colors and calmer music, the resulting interfaces are presumably more suitable for more conservative users.

Mainly because of commercial purposes, i.e., placement of advertisements, web-based information system providers often collect user data. The most popular part of user data is demographic data. According to [16], one can distinguish the following elements of demographic data: record data (name, address, e-mail, etc.), geographic data (zip-code, city, state, country), user characteristics (sex, education, occupation), and other customer qualifying data. These data are usually entered by the users themselves, or acquired from some external databases. The user data may also contain information about the users' knowledge, their skills and capabilities, their interests and preferences, and also their plans and goals.

During the user's interactions with web-based systems, the usage data can be observed and recorded to build, along with the user data, the model of the user. The usage data may concern selective operations that express users' interests, unfamiliarity or preferences, temporal viewing behavior, as well as ratings concerning the relevance of these elements [16].

Creating an appropriate model of the user is the key feature in personalized multimedia. This is because it is believed that if we are in the possession of full information concerning the web presence of users, we can utilize this information in delivering them appropriate services. It is obvious that for different users, different presentations, content, and structures are required. In the past, information systems were developed mainly for specialized applications, e.g. accounting and booking systems. Therefore, only differences such as novel versus very experienced users were distinguished [23, 27]. In later developments, the user's role in the system was also distinguished, like in the user cube [36], in which three types of user in the tourist system were distinguished: a software vendor, a tourist agent, and a tourist consumer.

There are very different methods for building a user model. For some systems, information about the zip code of living places is sufficient for drawing quite detailed assumptions on people's social status, interests, and various purchasing behaviors. This method has its origin in the work of Jonathan Robbin, a marketing specialist from the U.S., who noticed over thirty years ago that the address zip code might serve as a very good indicator of many assumptions on people's characteristics [28]. Even if this method is now criticized, web-based system users are still often asked to enter their zip code, along with answers to more or less personal questions. Users are usually very reluctant to provide any information about themselves because they treat it as being too private or simply do not want to lose time doing this. A good solution to this problem is based on automatic user information gathering and classification.

Usage data are usually collected automatically by the client and/or system sides by means of very different technologies ranging from CGI, PHP, ASP, Flash and Cookies to DoubleClick [38]. Cookies are entries stored in the special file on the user's local computer each time the user enters a web page. These entries can contain a lot of different data and settings, such as URL's of visited pages, links followed by the users, data entered by them into the forms on the pages, and interface settings made by the users themselves. These settings can be remembered and restored each time the user enters the system. This is especially useful in multimodal interfaces because they can restore the most recent settings made on a particular platform. These data, however, can also be used to identify the user's preferences and interests, which, if appropriately identified, can serve, for example, for advertisement placement or information delivery (e.g. weather information based on the user's address).

2.2 Interface agents

System personalization is a facility that is provided for the user by the web-based system. However, there are also solutions that are determined more by the users and are designed to serve primarily their purposes. These solutions are called interface agents. Interface agents can also be seen as an alternative to system personalization, because they do not require filling in online forms and questionnaires [32]. Maes defines an interface agent as an agent that acts as a kind of intelligent assistant to a user with respect to some computer application [39]. Interface agents are also defined as Personal Service Assistants that act as mediators between the human and the cyberspace and are able to personalize the interface by monitoring and sensing users' capabilities [1]. The interface agents developed at MIT [10] act primarily by observing their users, and by applying some machine learning mechanisms. For each new situation, the agent computes the distances between the current state and each past state stored in the memory. While, together with these past states, corresponding actions taken by the user in the past are stored, a recalled action which bears the largest resemblance to the current situation, or in other words, which has the smallest distance from it, is selected. The distance function can be constructed in many different ways, for example, as a sum of weighted features describing the particular state.

An interface agent should meet several requirements. Features such as adaptivity, autonomy, and collaboration are essential for all agents. Some researchers also add another requirement, namely robustness, defined as an ability to perform in restricted conditions [5]. To properly predict the user's intents, interface agents must work on accurate user model, which are difficult to achieve. There are many different reasons for this, but the most important are differences between users and their behavioral changes over time. Having defined some metrics and functions, such as precision, reactive, and autonomy metrics, as well as usability measures, it is possible to predict

the user's intentions and to model the user's behavior. Some authors [10] identify the uncertainty of these metrics and sometimes suggest consulting the user before making a decision.

In some cases, adaptive interfaces are composed not from a single agent, but rather from sets of agents that usually are simple processes that run 'in the background' and all together compose the interface implementation. Generally, interface agents try to help the users by assigning them a particular user's model, comparing their activities with the population of other users and offering appropriate assistance. Some of them, as mentioned above, also take into account the user's prior interaction with the system. Both methods are focused on a specific application and its environment [15]. For some researchers, interface agents are even perceived as an opposite solution to HCI achievements, and especially direct manipulation for improvement of user's tasks performed with computer assistance.

Interface agents are becoming very useful not only because of the ever increasing complexity of user interfaces [32], but also because of the increasing number of naïve users [30]. We can find many applications of interface agents. For example, Letizia, an autonomous interface agent for Web browsing [17], records URL's of visited pages and constructs the user profile out of them. Then, using a simple keyword-frequency measure, adopted from the field of Information Retrieval, the agent searches the neighborhood of pages currently visited for potentially relevant pages. Another type of interface agent is Apt Decision, which learns the user's real estates rental preferences to suggest appropriate apartments [32]. The Apt Decision agent uses the initial profile provided by the user as well as descriptions of apartments extracted from offers the user has analyzed so far.

Another type of interface agent is collaborative filtering agents, also called recommended systems, which are built on the assumption that a proper way to find the relevant content for a particular user is to find other people that resemble the current user in some way, and then recommend titles that those similar people like [4]. These ideas are of course not completely new. They were already developed in the field of the Information Retrieval in the 1960's. Collaborative filtering implementations use many methodologies, which have been divided into two classes [4], namely memory-based algorithms and model-based methods. Methods belonging to the first class are correlation, vector similarity, default voting, inverse user frequency, and case amplification, while the second class includes cluster methods and the Bayesian Network Model. Bayesian belief networks can represent cause-and-effect relationships between nodes, which are weighted to express how they affect each other. One of the most interesting aspects of belief networks is their non-determinism, which enables the most reasonable solution to be inferred from the given network of relationships, probabilities, and trade-offs inherent in the situation. The experiments presented in [4] have shown that Bayesian networks with decision trees and correlation methods are the best performing algorithms. A good example of a commercial application of an interface agent using Bayesian belief networks is the Microsoft Office 97 Office Assistant [12]. These implementations have been further developed, namely into the MS Agent, which is part of the Microsoft .NET technology.

3 Adaptive interface architecture for multimodal web-based systems

Adaptivity is one of the obligatory features of interface agents [5]. However, so far most applications concentrate, despite the name, more on the task to be done by the user with the help of the agent than on improving the process of the interaction itself. On the other hand, adaptive hypermedia [16] are more interesting in this respect, as they concentrate on the adaptation of information content, presentation, and structure. This adaptation is necessary because of universal usability challenges, namely the variety of interface platform technology, the diversity of users, and gaps in the user's knowledge [31]. We must also remember that users, their preferences and information needs, as well as their knowledge, are changing all the time. Therefore, the model of the adaptive system should take such factors into account.

Many web-based systems can be personalized by the users themselves. But, because of gaps in users' knowledge about the system, the process of personalization is quite often time consuming, boring, and not very effective. That is why we postulate the use of an adaptive system that will be able to offer for each new user of a particular web-based system, an adapted interface constructed accordingly to settings that have already been made by users somehow similar to the current one. Therefore, we propose to build an architecture in which the user's platforms form a multi-agent system. Agents collect information about the environment, i.e., users, in the form of the user profiles. These profiles may consist of both the user as well as usage data [16]. Further details about this matter will be provided in Sect. 3.1. Users whose profiles are very much alike form a class of users. The classification methods will be presented in Sect. 3.2.

Users of web-based systems are often extremely disparate. They have different interests, information needs, and interaction preferences. As a consequence, for different classes of users different user interfaces are most suitable [33]. As explained above, users of such systems are able to personalize them to suit their preferences by selection of different settings of information content and structure, as well as of presentation. These settings are called the interface profile. Because of the great differences among the population of users, while interacting with the same interface, they may achieve completely different performance and satisfaction measure values [14].

These measures that provide the basis for overall interface usability are described in detail in Sect. 3.3.

Each interface agent, besides the user profile, also stores the interface profile of each web-based system with the corresponding compound usability measure values. This information forms the agent's knowledge, which can be used to compose more efficient interface than the default one, for each novel user belonging to the same class of users. The agent's knowledge, because of the nature of human computer interaction, is often inconsistent, i.e., for different users, even belonging to the same class, different interface settings can have equal usability measure values. In this situation the necessity of reconciling the agent's knowledge states arises. By applying user interface usability measures, it is possible to find consensus between the agent's knowledge states, and so determining more efficient interfaces for new users in the adaptive way. As the domain of consensus methods may not be familiar to specialists in the field of interactive systems, a short introduction is provided in Sect. 4.

In Sect. 3.4, the adaptive interface construction method is described thoroughly.

3.1 User models and profiles

The user model can be defined as a user's description from the design point of view that delivers information on the user's behavior and task performance [27], or as a collection of the user's habits and projections of how the system may work [35], which are necessary to adapt a computer system to the modeled user's needs [36]. Building an appropriate user model requires knowledge from several scientific disciplines, such as psychology, pedagogy, sociology, anthropology, and ergonomics. Models are usually based on a specific theory. This can be an explanatory theory that explains only observed human behavior, or an empirical law that allows simple quantitative predictions to be made, or, finally a dynamic model of more complex user actions. The combination of all these approaches is called the Human Virtual Machine [23].

To create an appropriate user model we can take into account achievements of psychology, i.e., models of human information processing such as information processing subsystems, cycle times, and task performance. Quite often information systems address specific social environments. Sociological and anthropological theories of human behavior should therefore be applied, because sociological factors can have a greater influence on human behavior and task performance than cognitive ones.

Tasks that are performed with the help of computerized systems are becoming more and more complex. So, to complete them, specific sequences of different actions should be made. Organization theories should be implemented to address this aspect. We can introduce problem-solving models, such as the Theory of Human Problem Solving by Newell and Simon, or the more specific Norman's task performance [23].

We must remember that the solutions offered by HCI approaches concentrate upon statistically centered interface adaptation to the so-called average user. Nowadays, however, systems are used by users who are very different from each other, and this kind of approximation becomes inefficient. Luckily, the emergence of AI (Artificial Intelligence) user modeling techniques and methodologies has enabled a shift from traditional, rather implicit, user model representation used in software engineering to more explicit ones [36]. Traditional user models can be considered as static because once they have been developed and tailored into the software system they cannot be updated automatically during system operation. Today's interface agents, as well as adaptive hypermedia applications, are provided with a current state of knowledge or beliefs representing the user model in a form that enable some changes in the system operation, and especially the user interface, to be made.

Collecting the user data, along with the usage data, is a key feature of personalized hypermedia [16], described also in Sect. 2.1. In our architecture, we propose to call these data the user profile. As the agent resides on each platform used by the user, some basic information about the system platform itself is also placed in the profile as part of the usage data. These data can contain the name of the platform (e.g., PC, Mac, or iPAQ), as well as of the operating system (e.g., MS Windows, Mac OS, or Pocket PC), and, additionally, more detailed information on screen resolution, sound, and manipulation devices. In our architecture we assume that in many cases the user data entered by the user is completely unavailable because of the user's general reluctance in doing so. However, if they are available, they can be used properly. In our architecture, the user profile is used only for finding similar users and not for deriving from these data any stereotypes used to model the system itself. System personalization is made by the users themselves, and we only find appropriate consensus of these settings and deliver it to the new users who belong to the same class.

3.2 User classification

User classification is a very well-known problem that has been dealt with by many information retrieval specialists since early 1960's [29]. Today, in the era of the e-economy and of the development of Information Society, this research is being continued [8, 18]. In e-business it is very important to gather as much information as possible about users, i.e. potential clients. The users' data files can be very large, so to extract some useful information from them the methods of data mining are widely used. Clustering is a central process in data mining [8]. As the results of research in this field can be extremely valuable for e-business, it is obvious that much of them will never be published.

To classify users, we can use different types of user data or usage data. Usage data for web-based systems

may simply contain user's browsing history, i.e., the list of URL's of visited documents. Usage data, however, could also contain other more qualified actions, such as filling-in different forms that are part of the pages, joining mailing lists, purchasing any goods that are offered on the Web, etc. One of the easiest forms to describe users can be the URL's of visited pages or keywords from these pages. These data are then transferred into attribute-vectors with real values. These could be simple binary vectors, denoting the presence or absence of a specific feature in the user profile or vectors with keyword frequencies in visited pages [29]. These vectors can also contain other data types from the user profile. They could be in numerical form, such as age, in binary form, e.g., sex where 0 denotes male and 1 female (or opposite), but also in typical text form, e.g., education, address, etc. These types of data can be transformed into binary values by using modified attributes that denote attribute value pairs. It is also possible to represent attribute values by means of fuzzy values.

The process of user classification results in grouping the heterogeneous set of users $U = \{u_1, ..., u_n\}$ into disjoint subsets of the set U by some measure of similarity or distance. Finding an appropriate measure, as well as a user's representation, are key problems of user classification. The implementation of classification/clustering methods is also essential. We can distinguish three major types of clustering algorithms [8]: hierarchical, Euclidean or similar metric space, and similarity matrix.

The most popular are those algorithms that belong to the family of Euclidean or similar metric space clustering. Their common property is that of delivering information about each user from the set U in the form of an attribute-vector, in which all the values are real numbers and for which it is possible to distinguish a metric function to measure the distance between them, i.e., the Euclidean distance. Very popular among these measures is the cosine distance function [29]:

$$d(u_j, u_k) = \frac{\sum_{i=1}^{m} u_{j,i} * u_{k,i}}{\sqrt{\sum_{i=1}^{m} (u_{j,i})^2 \sum_{i=1}^{m} (u_{k,i})^2}}, \tag{1}$$

where u_j and u_k are vectors of real values of m attributes describing users j and k, respectively. However, it is also possible to use some similarity functions instead of distance functions. Many different similarity functions can be found, e.g., in [29]. The similarity function used in the Dattola classification algorithm [6, 29] is shown below:

$$s(u_j, u_k) = \sum_{i=1}^{m} \min(u_{k,i}, u_{j,i}). \tag{2}$$

The clustering optimization criteria can be described as finding a partition of the set U into p disjoint subsets C_i $i = 1, \ldots, p$ of users such that the distance among all the members of each class – $d(C_i)$ is minimal:

$$d(C_i) = \sum_{j=1}^{r} \sum_{k=1}^{r} d(u_j, u_k), \tag{3}$$

where $r = \mathrm{Card}(C_i)$;
or the distance between all p classes as stated below is maximal:

$$d(U^p) = \sum_{i=1}^{p} \sum_{j=1}^{i-1} d(C_i, C_j), \tag{4}$$

where

$$d(C_i, C_j) = \sum_{l=1}^{r} \sum_{k=1}^{q} d(u_l, u_k) \tag{5}$$

and $r = \mathrm{Card}(C_i)$, and $q = \mathrm{Card}(C_j)$.

It is obvious that one can build analogous criteria for the similarity functions. As has been proved, the computational complexity of the problem of partitioning the set U with n users into two classes is exponential with respect to n. So, practically, other sub-optimal but more efficient algorithms should be used. They are usually based on the selection of some initial partition, as for example in the Dattola method [6], presented below:

Start

In the beginning we divide the set $U = \{u_1, \ldots, u_n\}$ of n users into k initial classes: $C_{1,1}$, $C_{1,2}$, $\ldots$, $C_{1,k}$. For all the classes we calculate the centroid. The centroid of the class C_j is denoted as $O_j = \{o_{j,1}, o_{j,2}, \ldots, o_{j,m}\}$, where

$$o_{j,s} = \begin{cases} 0 \quad \text{if} \quad f_{j,s} = 0, \\ b - r_{j,s} \quad \text{otherwise}, \end{cases} \tag{6}$$

where b is a constant assumed priori.

$$f_{j,s} = \sum_{u_i \in C_j} u_{i,s}, \tag{7}$$

$$r_{j,s} = (1 + \max\{f_{j,t} : t = 1, 2, \ldots, m\} - f_{j,s}). \tag{8}$$

Then in $t-1$ iteration we have the partition $C_{t-1,1}$, $C_{t-1,2}$, $\ldots$, $C_{t-1,k}$, with adequate centroids $O_{t-1,1}$, $O_{t-1,2}$, $\ldots$, $O_{t-1,k}$. Let T be the threshold used to construct new classes $C_{t,j}$, which is determined in the following way:

$$\begin{aligned} C_{t,j} = \{a_i : s(a_i, O_{t-1,j}) \geq T \text{ and} \\ s(a_i, O_{t-1,j}) = \max\{s(a_i, O_{t-1,l}) \\ for\ l = 1, 2, \ldots, k\}\}. \end{aligned} \tag{9}$$

All the user profiles that were not included into any class should be inserted into the set L_t of isolated objects. A new centroid is determined as described above if the following condition is fulfilled:

$$\sum_{a_i \in C_{t,j}} s(a_i, O_{t,j}) > \sum_{a_i \in C_{t,j}} s(a_i, O_{t-1,j}), \tag{10}$$

Otherwise the centroid preserves its value ($O_{t,j} = O_{t-1,j}$).

The iteration ends when for a particular t and all $j = 1, 2, \ldots, k$, $O_{t,j} = O_{t-1,j}$ occurs. Then the objects from the set of isolated objects L_t are treated as a separate class or joined to those classes to which they are most similar.

End.

The type of partition that is presented in the Dattola algorithm could be especially useful in classifying web-based systems users. When a new user comes to use the system, it is necessary to calculate only the similarity values of the user and all the centroids. When we find the class with the greatest similarity function value, we add the user profile to this class and calculate the new centroid for it as described in (6) and then we check the condition described in (10). If it is true, we start the Dattola classification algorithm with the given partition, centroids and new greater set of users. Otherwise we leave the partition as it is.

3.3 Compound usability measure

A key concept in HCI is usability, which focuses on making interactive systems easy to learn and use [27]. Although it is very important, this feature is very difficult to measure. Usability is also a general quality concept used in the international standard ISO 9241 (Part 11) [14], in which it is defined in the following way: '*Usability of a product is the extent to which the product can be used by specific users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specific context of use*'. Effectiveness expresses: '*the accuracy and completeness with which users achieve specified goals*', while efficiency considers: '*the resources expended in relation to the accuracy and completeness with which users achieve goals*' and finally satisfaction reveals the user's '*comfort and acceptability of use*'. We can see from its definition that usability is a compound concept and, as a consequence, its measure should reflect this.

Interface evaluation is not straightforward and could not be measured by a simple function. Quite often different empirical methods are used, but analytical methods or combination of different methods are also applied. Usability factors usually represent different users' relations with the interface, i.e., how the user interacts with it, the related mental efforts, or the interface's ergonomic attributes. In most information systems, however, we must consider also other parties than the users themselves, such as system providers, ISP's, content providers, and advertising agencies that are also involved in the whole life-cycle of the interactive system. So it is believed that the efficiency of the system could be considered from very different points of view. In the ISO definition of system usability these objectives could be included in the effectiveness part, in which the accuracy of completing the stated goals is taken into account. We should remember, however, that the objectives of these different parties could be contradictory. For example, web portal users usually do not want to see any advertisements, but the system providers live from selling the advertisement placing.

Each web-based system could be built and maintained for different reasons, mainly profit. A web portal can earn money on advertisements, an Internet shop can sell different goods, or an Internet auction server can enable the exchange of different goods between their users, and so on. Each of these applications has different goals, and hence different factors express the achievement of these goals. Additionally, the goals of their users are different, e.g., finding relevant information, finding desired goods to buy, or selling their own items.

Usually, for web portals, the more time users spend on browsing the portal pages the better because, in the meantime, they have at least visual contact with advertisements. So for such systems, the usability measures should consider metrics such as the number of pages visited, the number of advertisements sent to the user, the number of clicks on links presented on the advertisements, the frequency of visits on the site, etc. Of course, some of these metrics could be normalized by dividing them by the total number of items.

Besides the system effectiveness, which reveals the system's achievement of different goals and objectives, the efficiency is connected with resources spent, as well as the completeness and accuracy of goal achievement. For e-shops, the most important measure is the amount of money spent by the user. There are, however, other possible factors that should be considered in the usability measures, such as the number of visits to the shop, the number of visited pages, the number of goods that were put into the basket, the speed of shopping, the utilization of lists and retrieval mechanisms of goods, the purchase of goods on special offer, etc.

For e-auction sites, like eBay, usually there is no charge for browsing, bidding on or buying items, but clients have to pay fees to list and sell items. Nevertheless, some start-ups in the field of e-auction business or auctions that cooperate with web portals cannot charge any fees. Usability measures of any on-line trading center should consider the number of sold and bought items, the number of bids made by the user, the number of auctions the user took part in, the frequency of bids made by the user, the number of items on sale, etc.

The last element of system usability, i.e., user satisfaction, is the most subjective factor. The most popular way for measuring its values is by asking the users, for example by questionnaires. There are also some more objective forms of measuring users satisfaction, for example by observing users behavior while they interact with the system [14].

In our model of adaptive multimodal web-based systems, a single usability measure value is needed, but because of its complexity we need some compound measure. An interesting example of a compound measure can be

found in the area of the interface agents [5]. The utility function is defined as follows:

$$U_{requirements} : \omega^n \times R^n \times H \to \Re\,, \tag{11}$$

where for each previous action or event denoted as $h \in H$, $\omega \in [0,1]$ is a weighting factor for each of n requirement metrics R, and $\Re$ is a set of real numbers. This measure could also be applied as a compound measure of the system usability, where R metrics express values of effectiveness, efficiency, and user satisfaction in using the system.

3.4 Adaptive user interface construction by means of knowledge states reconciliation

The adaptive user interface construction functions like many Internet-based systems in the client-server mode. The client side, as the interface agents, is responsible for the interaction with the user and communicates with the server side not only for exchanging content information, but also for exchanging data necessary for the interface construction. The general schema for interface construction is quite simple. At every start of the interaction with an interactive system, the agent checks whether there already exists an interface profile for that system, which is efficient enough for the user making use of this particular platform, which means the compound usability measure value is above the specified threshold. If this condition is satisfied, the interface is generated and presented to the user. If it happens that a user is using the particular system on the particular platform for the first time, or if prompted, the user agrees to modify the interface settings, the user profile containing the platform settings is sent to the server where the user is classified. Then the server, applying consensus-based methods creates a new interface profile for the given platform settings and sends it back to the client (agent). Moreover, every user interaction with the system is evaluated, by means of the usability measures [14], and the evaluation results, together with any updates of the user profile made by the users themselves, are sent to the server for further processing. If there are too few interface profiles constructed for a particular user, new ones are generated randomly and presented to the new user. The schema is also shown in Fig. 2, and the method of consensus-based interface profile determination is presented in the following section.

The interface profile, as mentioned before, consists of different parameters that are used in the particular interface construction. For every system, as well as for every platform, this process should be designed sepa-

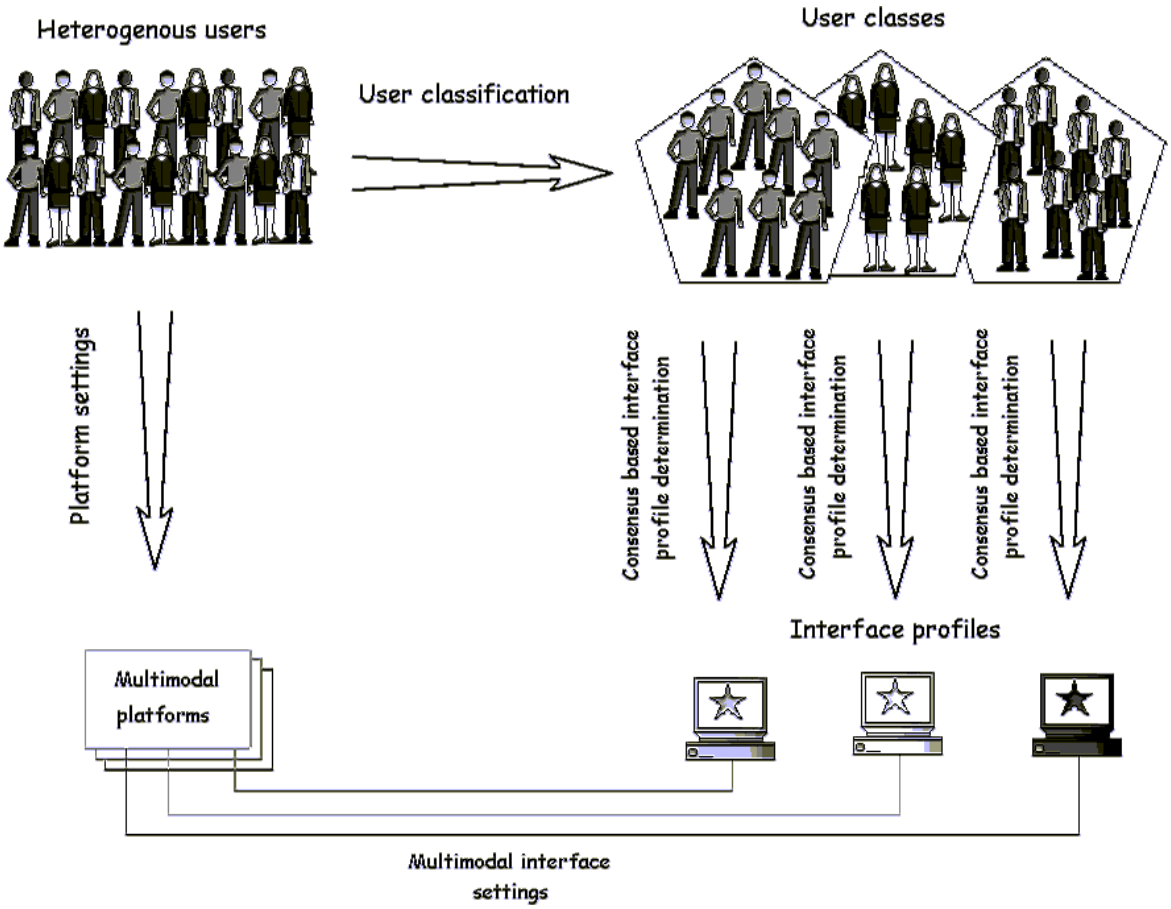

Fig. 2. General architecture of the consensus based adaptive interfaces in multimodal web-based systems

rately. Therefore, in this paper only some general ideas concerning this matter can be presented. The potential interface attributes that can be selected range from quite simple ones, such as fonts, styles, colors and sizes, background colors or graphics, buttons in the form of icons or text, to more complex ones, such as parameters of sets of fonts (their correlation), interface templates (for example, themes in Microsoft FrontPage), or styles of button animation. These can also be more general parameters such as used metaphors, or attributes describing information content and its hierarchy.

The method of consensus-based interface profile determination is based on the following two assumptions: first, it is possible to construct the distance function between all the values of each attribute, and second, the adaptively constructed interface is consistent.

4 Consensus determination for interface profiles

The first applications of consensus methods can be found in the social and sociological sciences. They have been used for standardization and working out agreements in solving conflicts [7]. A conflict is most often understood as a situation in which at least two bodies have different opinions on the same matter. Consensus methods are particularly useful for systems in which uncertainty of information is assumed, but decision making is nevertheless required. Consensus theory concerns two very distinct problems [2]. First, searching for a latent structure and reconciling disagreeing data. In this case, the data versions are considered as reflecting (not necessarily correctly) an unknown structure. The discards are due to measurement problems, missing information, or errors in evaluation criteria, or the heuristic nature of algorithms. In the second case, the data versions differ from each other because they are obtained in different ways (observations, experiments, etc.), and the proper version is unknown. Up to now, in the related literature, all solved consensus problems have been characterized by two approaches. The first concerns three properties: concrete structures of data versions [3, 7, 19, 20], i.e., transitive reference relations, equivalence relations, n-trees , semilattices, weak hierarchies, etc.; concrete distance (or similarity) functions defined on the basis of the structures; and concrete consensus choice functions. The most often used consensus function is called the *median*, which minimizes the sum of distances between the consensus and given data versions [20]. The other approach refers to using axioms to determine the positions of elements in a ranking, or to determine a class of consensus functions [26].

It is known that Bayesian networks are a good tool for classification. A classification task using Bayesian networks is most often based on finding the classes themselves from a given set of unclassified objects. Once such a set of classes has been found they can be the basis for classifying new objects. However, to use this tool as a classification task one should have a procedure that enables the conditional probabilities representing the degrees of membership of objects in given classes to be determined. If the consensus choice task formulated above was treated as a classification task, then of course Bayesian networks would be applied, but under the condition that the number of profiles forming the basis of the consensus choice is large enough. The reason is that only in this case can the probabilities be determined as credible. However, if the number of the profiles is not large enough, then Bayesian networks are not useful.

We can see that in all the approaches mentioned above the structures of versions are omitted (abstract consensus) or defined concretely (concrete consensus). If the structures are defined, they are uniform in the sense that the data versions are determined on the basis of the same universe of elementary objects. For example, if partitions (i.e., equivalence relations) are the structure of data versions, then all of them should be partitions of the same set whose elements are indivisible. However, in the related literature there is a lack of attribute paradigm. It means that the elements of the universe that is the basis of the data versions considered for consensus choice are not objects with varied features described by attributes, but have only one feature. In this work we assume that the data versions representing the conflicting content are built by means of some universe of tuples representing potential objects or events, etc. The tuples are represented by a set of different attributes and their values, each of which is a set of elementary values. In other words, we use multi-valued attributes to represent data versions. One should underline that the attribute paradigm is very convenient for describing real world objects, and is very often used. It even seems to be irreplaceable in database models, from hierarchical to object-oriented ones, or in knowledge representation.

The situation of inconsistency of interface agent profiles considered in this work can be treated as a conflict because, for the same user class, the agents propose different instances for the interface profiles. Thus, we can employ the consensus methods for solving the problem. We will use multi-valued attributes for representing this kind of conflict, and propose consensus choice algorithms for determining consensus.

4.1 The model of consensus

4.1.1 Basic notions

We assume that a real world domain is described by means of a finite set $\boldsymbol{A}$ of attributes and a set V of attribute *elementary values*, where $V = \bigcup_{a \in \boldsymbol{A}} V_a$ (V_a is the domain of attribute a). Let $\prod(V_a)$ denote the set of subsets of set V_a and $\prod(V_B) = \bigcup_{b \in B} \prod(V_b)$ for any $B \subseteq \boldsymbol{A}$. We accept the following assumption. For each attribute its value is a set of elementary values from V_a, thus it is an element of set $\prod(V_a)$. By an elementary value we mean

a value which is not divisible in the system. Thus it is a relative notion. One can assume, for example, that time units, numbers, partitions, etc., are elementary.

We define the following notions: let $B \subseteq \boldsymbol{A}$, and let a tuple of type B be a function $r : B \to \prod(V_B)$, where $(\forall b \in B)(r(b) \subseteq V_b)$. Instead of $r(b)$ we will write r_b, and a tuple of type B will be written as r_B. The set of all tuples of type B is denoted by $TYPE(B)$. A tuple is elementary if all attribute values are empty sets or 1-element sets. The set of elementary tuples of type B is denoted by $E\text{-}TYPE(B)$. An empty tuple whose values are all empty sets is denoted by the symbol ϕ. A partly empty tuple with at least one empty value is denoted by the symbol θ. A non-empty set R of tuples of type B is called a relation of type B, thus $R \subseteq TYPE(B)$. A sum of 2 tuples r and r' of type B is a tuple r'' of type $B (r'' = r \cup r')$ such that $(\forall b \in B)(r''_b = r_b \cup r'_b)$. A product of 2 tuples r and r' of type B is also a tuple r'' of type $B (r'' = r \cap r')$ such that $(\forall b \in B)(r''_b = r_b \cap r'_b)$. Let $r, r' \in TYPE(B)$. We say that tuple r is included in tuple r' (that is $r \prec r'$), iff $(\forall b \in B)(r_b \subseteq r'_b)$.

4.1.2 Definition of conflict system

We assume that a real world domain is commonly considered by agents that are placed in sites of a distributed system. The subjects of agent interest consist of events occurring in the world. The task of the agents is based on determining the values of event attributes (an event is described by an elementary tuple of some type). The elements of the system defined below will describe this situation [25].

Definition 1. *A conflict system is a quadruple:*

$$Conflict\_Sys = (\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{P}, \boldsymbol{Z})$$

where:

- $\boldsymbol{A}$ *– is a finite set of attributes, including a special attribute Agent; each attribute* $a \in \boldsymbol{A}$ *has a domain* V_a *(a non-empty and finite set of elementary values) such that values of a form a subset of* V_a*.; values of the attribute Agent are 1-element sets, which identify the agents.*
- $\boldsymbol{X}$ *– is a finite set of conflict carriers,* $\boldsymbol{X} = \{\prod(V_a) : a \in \boldsymbol{A}\}$.
- $\boldsymbol{P}$ *– is a finite set of relations on carriers from* $\boldsymbol{X}$*. Each relation is of some type A (for* $A \in \boldsymbol{A}$ *and* $Agent \in A$*).*
- $\boldsymbol{Z}$ *– is a finite set of logic formulas for which the model is a relation system* $(\boldsymbol{X}, \boldsymbol{P})$.

The purpose of Definition 1 relies on representing two kinds of information, namely information about conflicts in the distributed system, which require solving, and information needed for consensus determination.

In the conflict system, an event is described by an elementary tuple of type $B \subseteq \boldsymbol{A} \backslash \{Agent\}$. The values of attributes represent the parameters of the event. For example, the tuple presented in Table 1 describes the event "*In region* r_1 *at* 8 a.m. *the wind speed is* 10 m/s".

Table 1. An example tuple

Region	*Time*	*Wind_Speed*
r_1	8 a.m.	10 m/s

Relations belonging to set $\boldsymbol{P}$ are classified in such a way that each of them includes relations representing similar events. For identifying relations belonging to a given group, the symbols "+" and "−" are used as the upper index. If P is the name of a group, then the relation P^+ is called a positive relation (i.e., it contains positive knowledge) and P^- a negative relation (i.e., it contains negative knowledge). If $r \in P^+$, then we have the following interpretation: in the opinion of agent r_{Agent} one or more events included in r_A should take place. If $r \in P^-$, then we say that in the opinion of agent r_{Agent} none of the events included in r_A should take place. The same agent cannot simultaneously state that the same event should take place and should not take place, as the same event cannot be classified by the same agent into positive and negative relations simultaneously.

4.1.3 Conflict profiles

We define a conflict situation containing information about a concrete conflict as follows:

Definition 2. *A conflict situation is a pair* $\langle \{P^+, P^-\}, A \to B \rangle$*, where* $A, B \subseteq \boldsymbol{A}$*,* $A \cap B \neq \phi$ *and* $r_A \neq \theta$ *and* $r'_A \neq \theta$ *for every tuples* $r \in P^+$ *and* $r' \in P^-$.

According to the above definition, a conflict situation consists of agents (conflict body) which appear in relations P^+ and P^- (conflict content) representing the positive and negative knowledge of agents referring to subjects represented by set A of attributes. These relations are the basis of consensus. Expression $A \to B$ means that the agents do not agree when referring to combinations of values of attributes from A with values of attributes from B, and the purpose of the consensus choice is that for a tuple type A at most one tuple of type B should be assigned.

For a given situation s, we determine the set of agents which take part in the conflict as follows:

$$Agent(s) = \{a \in V_{Agent} : (\exists r \in P^+)(r_{Agent} = \{a\}) \vee (\exists r \in P^-)(r_{Agent} = \{a\})\},$$

and the set of *subject elements* (or *subjects* for short) as follows:

$$Subject(s) = \{e \in E - TYPE(A) : (e \neq \theta) \wedge [(\exists r \in P^+)(e \prec r) \vee (\exists r \in P^-)(e \prec r)]\}.$$

The set $Subject(s)$ then includes subject elements that have been occupied by agents. For example, for the situation $\langle\{Wind^+, Wind^-\}, \{Region\} \rightarrow \{Time, Wind\_Speed\}\rangle$ the subjects are the regions for which the agents present their forecast for time and speed of the wind. Now for each subject $e \in Subject(s)$ let us determine sets with repetitions $Profile(e)^+$ and $Profile(e)^-$ which include the positive and negative knowledge of agents on subject e, as follows:

$$Profile(e)^+ = \{r_{B\cup\{Agent\}} : (r \in P^+) \wedge (e \prec r_A)\},$$
$$Profile(e)^- = \{r_{B\cup\{Agent\}} : (r \in P^-) \wedge (e \prec r_A)\}.$$

These sets are called positive and negative profiles of a given conflict subject e.

4.1.4 Consensus definition and determination

Below we present the definition of consensus [25].

Definition 3. *Consensus on subject $e \in Subject(s)$ of situation $s = \langle\{P^+, P^-\}, A \rightarrow B\rangle$ is a pair of two tuples $(C(s,e)^+, C(s,e)^-)$, where $C(s,e)^+, C(s,e)^- \in TYPE(A \cup B)$ and the following conditions are fulfilled:*

a) $C(s,e)^+_A = C(s,e)^-_A = e$,
b) $C(s,e)^+_B \cap C(s,e)^-_B = \phi$,
c) Tuples $C(s,e)^+$ and $C(s,e)^-$ fulfil logic formulas from set $\boldsymbol{Z}$,
d) One or more of the following postulates are satisfied:

P1. $C(s,e)^+_B \prec \bigcup_{r\in profile(e)^+} r_B$ *and* $C(s,e)^-_B \prec \bigcup_{r\in profile(e)^-} r_B$.

P2. $\bigcap_{r\in profile(e)^+} r_B \prec C(s,e)^+_B$. *and* $\bigcap_{r\in profile(e)^-} r_B \prec C(s,e)^-_B$.

P3. *For $s' = \langle\{P'^+, P'^-\}, A \rightarrow B\rangle$ and $s'' = \langle\{P''^+, P''^-\}, A \rightarrow B\rangle$*

if $C(s',e)^+_B \cap C(s'',e)^+_B \neq \theta$
then $C(s,e)^+_B = C(s',e)^+_B \cap C(s'',e)^+_B$
where $s = s = \langle\{P'^+ \cup P''^+, P'^- \cup P''^-\}, A \rightarrow B\rangle$.

P4. *For $s' = \langle\{P'^+, P'^-\}, A \rightarrow B\rangle$ and $s'' = \langle\{P''^+, P''^-\}, A \rightarrow B\rangle$*

if $C(s',e)^-_B \cap C(s'',e)^-_B \neq \theta$
then $C(s,e)^-_B = C(s',e)^-_B \cap C(s'',e)^-_B$
where $s = \langle\{P'^+ \cup P''^+, P'^- \cup P''^-\}, A \rightarrow B\rangle$.

P5. *For $x \in Profile(e)^+$ and $x_B \neq C(s,e)^+_B$ there exists a natural number n such that $x = C(s',e)^+_B$ where s' is a situation with the same subject $A \rightarrow B$, in which $Profile'(e)^+ = Profile(e)^+ \cup \{n * x\}$.*

P6. *For $x \in Profile(e)^-$, $x_B \neq C(s,e)^-_B$ there exists a natural number n such that $x = C(s',e)^-_B$ where s' is a situation with the same subject $A \rightarrow B$, in which $Profile'(e)^- = Profile(e)^- \cup \{n * x\}$.*

Some remarks should be made about the above postulates. The first postulate states that the consensus should be included in the sum of profile′ elements, that is, the so-called *knowledge closure*. The second postulate requires knowledge consistency, meaning that the common part of profile elements should be included in the consensus. Postulates P3 and P4 are the Condorcet consistency condition for choice. Postulates P5 and P6 state that if a tuple x is not a consensus of a profile, then it should be a consensus of a new profile including the old profile and a sufficient number of tuples x.

The following theorem should enable a consensus satisfying all postulates **P1**–**P6** to be determined [25].

Theorem 1. *If there is a defined distance function φ between tuples of $TYPE(B)$, then for a given subject e of situation $s = \langle\{P^+, P^-\}, A \rightarrow B\rangle$, tuples $C(s,e)^+$ and $C(s,e)^-$, which satisfy conditions (a)–(c) of Definition 3 and minimize the expressions $\sum_{r\in profile(e)^+} \varphi(r_B, C(s,e)^+_B)$ and $\sum_{r\in profile(e)^-} \varphi(r_B, C(s,e)^-_B)$, should create a consensus satisfying all postulates **P1**–**P6**.*

4.2 Consensus determination for user profiles

For our web-based information system we specify the parameters of the conflict system as follows:

- $\boldsymbol{A} = \{Agent, System, Class, A_1, A_2, \ldots, A_n\}$,

where:

Agent represents interface agents,
Class represents classes of users, and
$A = \{A_1, A_2, \ldots, A_n\}$ is a set of attributes describing the interface profiles for user service.

- $P = \{Profile^+, Profile^-\}$, where $Profile^+, Profile^- \subseteq \prod(V_{Agent}) \times \prod(V_{Interface}) \times \prod(V_{Class}) \times \prod(V_{A1}) \times \prod(V_{A2}) \times \ldots \times \prod(V_{An})$.

We interpret a tuple of relation $Profile^+$, for example, $\langle Agent : a_1, Class : c_1, A_1 : a_1, A_2 : a_2, \ldots, A_n : a_n\rangle$ as follows: in the opinion of agent a_1 the appropriate interface profile for serving users from class c_1 on the web-based information system should be the tuple $\langle A_1 : a_1, A_2 : a_2, \ldots, A_n : a_n\rangle$. A tuple $\langle Agent : a_2, Class : c_1, A_1 : a'_1, A_2 : a'_2, \ldots, A_n : a'_n\rangle$ belonging to relation $Profile^-$ means that according to agent a_2, for users from class c_1 the profile $\langle A_1 : a'_1, A_2 : a'_2, \ldots, A_n : a'_n\rangle$ has unacceptable usability.

- $\boldsymbol{Z}$: Logical formulas representing conditions that have to be satisfied by the tuples belonging to the relations from $\boldsymbol{P}$.

A conflict situation is then defined as follows:

$$s = (\{Profile^+, Profile^-\}, \{Class\} \rightarrow \{A_1, A_2, \ldots, A_n\}).$$

The set $\{Profile^+, Profile^-\}$ is then the basis of consensus, the set of attribute $\{Class\}$ represents the consensus subject, and set $\{A_1, A_2, \ldots, A_n\}$ describes the content of consensus. Methods for consensus determination are provided by the algorithms described in the next subsection.

4.3 Algorithms for consensus determination

All algorithms briefly presented below are reported in the Appendix.

4.3.1 Algorithm for number intervals

The idea of the first algorithm is that, for given intervals, the consensus interval must be a sub-interval of the minimal interval, which contains the given intervals. First, the minimal interval is calculated (Step 2). Next, each sub-interval of the minimal interval is checked and is determined to be the consensus if it minimizes the sum of the distances (Steps 3 and 4). The distance function between number intervals is defined as follows [24]:

$$\partial_1(i, i') = \begin{cases} |i^* - i'^*| + |i_* - i'_*| & for\ i \cap i' \neq \varnothing, \\ \max\{i^*, i'^*\} - \min\{i_*, i'_*\} + 1 & for\ i \cap i' = \varnothing. \end{cases}$$

4.3.2 Algorithm for rankings

The idea of this algorithm relies on the following steps:

- Transformation rankings (as binary relations on some set) to binary matrices $\boldsymbol{A}^{(k)}$ as the input of the algorithm.
- The calculation of the sum of weights for each pair of elements so as to order them in the consensus ranking (matrix $\boldsymbol{B}$).
- The transformation of matrix $\boldsymbol{B}$ into ranking $\boldsymbol{E}$.

4.3.3 Algorithm for sets of values

The idea of this algorithm relies on accounting for the number of occurrences of each element in the given sets. If the count is bigger or equal to half of the given sets, then the element should appear in the consensus.

5 An example

Let us return to the situation described in the introduction. We considered a manager who was planning to take part in a conference. As was described, in order to be prepared for this conference he had to use several multimodal web-based information systems. So let us consider one of them, namely the information system S of the city that hosts the conference. There are many different things that can be of special interest to city visitors and inhabitants, for example, public transportation, museums, hotels, cinemas, shopping centers, boutiques, etc. Different users can be interested in different information. For example, younger visitors could be interested in information on discos, whereas older users could be interested luxurious restaurants.

We assume that system S has already been used by a large number of users who have made some system settings in the process of personalization. These settings have been stored by their agents and evaluated to be appropriate (included in $Profile^+$) or inappropriate (included in $Profile^-$) for their users. The system settings that consist of the interface parameters and the information content can be described by the set of attributes $\boldsymbol{A}$ and their values. The overall interface template, the number of text columns or the window size, as well as information topics, described in Sects. 2 and 3.3, can serve as typical elements of the set $\boldsymbol{A}$. So the conflict system can be defined as follows:

- $\boldsymbol{A} = \{Agent, Class, Window\_size, Sound\_volume, Number\_of\_col., Template, Topics\}$
- $\boldsymbol{X} = \{\prod(V_{Agent}), \prod(V_{Class}), \prod(V_{Window\_size}), \prod(V_{Sound\_volume}), \prod(V_{Number\_of\_col.}), \prod(V_{Template}), \prod(V_{Topics})\}$, where $V_{Agent} = \{a_1, \ldots, a_8\}$; $V_{Class} = \{c_1, c_2\}$; $V_{Window\_size} = \{240 \times 320,\ 640 \times 480,\ 800 \times 600,\ 1024 \times 768\}$; $V_{Sound\_volume} = [0, 1]$; $V_{Number\_of\_col.} = \{1, 2, 3\}$; $V_{Template} = \{classical, normal, modern, vanguard\}$, $V_{Topics} = \{restaurants, boutiques, shoppingcenters, publictransport, discos, cafes, museums, hotels\}$.
- $\boldsymbol{P} = \{Profile^+, Profile^-\}$, where relations $Profile^+$ and $Profile^-$ are presented in Tables 2 and 3.
- $\boldsymbol{Z} = \{(Window\_size = 240 \times 320) \Rightarrow (Number\_of\_col. = 1); (Template = classical) \Rightarrow (Sound\_volume = 0)\}$.

For situation $s = \langle \{Profile^+, Profile^-\}, Class \rightarrow \{Window\_Size, Sound\_volume, Number\_of\_col., Template, Topics\} \rangle$ the set of subject is $Subject(s) = \{\langle Class : c_1 \rangle, \langle Class : c_2 \rangle\}$ and the profiles are presented in Tables 4–7.

The distance functions between tuples of the above relations are defined as follows. We assume that in sets V_{Window_size}, $V_{Number\_of\_col.}$, and $V_{Template}$ it is possible to define the following linear orders $\langle 240 \times 320, 640 \times 480, 800 \times 600, 1024 \times 768 \rangle$, $\langle 1, 2, 3 \rangle$, and $\langle classical, normal, modern, vanguard \rangle$, respectively. Thus, the distance between two values of one of the above sets may be determined as the difference between their positions in the order. For measuring distances between values of set V_{Sound_volume} one may use the Euclidean metric. Lastly, values of the attribute $Topics$ in the above relations are subsets of set $V_{Template}$. Thus the distance of these sets is defined as the cardinality of their symmetrical difference.

Table 2. Relation $Profile^{+}$

Agent	*Class*	*Window_size*	*Sound_volume*	*Number_of_col.*	*Template*	*Topics*
a_1	c_1	240×320	0	1	classical	{restaurants, museums, hotels}
a_2	c_2	800×600	0.3	2	normal	{boutiques, cafes, galleries, museums}
a_3	c_1	240×320	0.7	1	normal	{public transport, museums, cafes, boutiques}
a_4	c_2	1024×768	0.4	3	vanguard	{shopping centers, cafes, discos, museums}
a_5	c_2	1024×768	0.4	3	vanguard	{boutiques, shopping centers, galleries, museums}
a_6	c_1	240×320	0.8	3	normal	{pubs, museums, cafes}

Table 3. Relation $Profile^{-}$

Agent	*Class*	*Window_size*	*Sound_volume*	*Number_of_col.*	*Template*	*Topics*
a_2	c_1	640×480	0	1	classical	{hotels, discos, galleries, museums}
a_5	c_2	640×480	0.3	1	vanguard	{public transport, museums, cafes, shopping centers}
a_6	c_1	800×600	0.7	2	vanguard	{shopping centers, museums, hotels}
a_7	c_2	240×320	0	1	normal	{boutiques, cafes, crème, museums}
a_8	c_1	640×480	0.6	2	modern	{shopping centers, cafes, discos, museums}
a_3	c_2	240×320	0.2	1	modern	{boutiques, cafes, shopping centers, hotels}

Table 4. Relation $Profile(c_1)^{+}$

Agent	*Window_size*	*Sound_volume*	*Number_of_col.*	*Template*	*Topics*
a_1	240×320	0	1	classical	{restaurants, museums, hotels}
a_3	240×320	0.7	1	normal	{public transport, museums, cafes, boutiques}
a_6	240×320	0.8	3	normal	{public transport, museums, cafes}

Table 5. Relation $Profile(c_1)^{-}$

Agent	*Window_size*	*Sound_volume*	*Number_of_col.*	*Template*	*Topics*
a_2	640×480	0	1	classical	{hotels, discos, galleries, museums}
a_6	800×600	0.7	2	vanguard	{shopping centers, museums, hotels}
a_8	640×480	0.6	2	modern	{shopping centers, cafes, discos, museums}

Table 6. Relation $Profile(c_2)^{+}$

Agent	*Window_size*	*Sound_volume*	*Number_of_col.*	*Template*	*Topics*
a_2	800×600	0.3	2	normal	{boutiques, cafes, galleries, museums}
a_4	1024×768	0.4	3	vanguard	{shopping centers, cafes, discos, museums}
a_5	1024×768	0.4	3	vanguard	{boutiques, shopping centers, galleries, museums}

Table 7. Relation $Profile(c_2)^{-}$

Agent	*Window_size*	*Sound_volume*	*Number_of_col.*	*Template*	*Topics*
a_5	640×480	0.3	1	vanguard	{public transport, museums, cafes, shopping centers}
a_7	240×320	0	1	normal	{boutiques, cafes, galleries, museums}
a_3	240×320	0.2	1	modern	{boutiques, cafes, shopping centers, hotels}

The distance between two tuples belonging to the above relations is defined as the sum of distances between values of corresponding attributes.

In this way the consensus satisfying postulates **P1**–**P6** can be determined (Table 8).

In the consensus the values of attributes $Window\_size$, $Number\_of\_col.$, and $Template$ are determined by Algorithm 2, values of attribute $Sound\_volume$ by Algorithm 1, and values of attributes $Topics$ by Algorithm 3.

Table 8. The consensus

Consensus	*Window_size*	*Sound_volume*	*Number_of_col.*	*Template*	*Topics*
$C(s,a_1)^+$	240×320	0.7	1	normal	{public transport, museums, cafes}
$C(s,a_1)^-$	640×480	0.6	2	modern	{shopping centers, discos, hotels}
$C(s,a_2)^+$	1024×768	0.4	3	vanguard	{boutiques, galleries, shopping centers}
$C(s,a_2)^-$	240×320	0.2	1	modern	{museums, cafes}

We interpret the first tuple of the consensus as the best profile for the service for users from class a_1; the second tuple can be interpreted as the profile that should not be used for users from class a_1. For the third and fourth tuples the interpretation is similar. One can see that the conditions of Definition 3 and formulas from set $\boldsymbol{Z}$ are satisfied by the consensus.

6 Conclusions

Adaptive user interfaces are becoming more and more popular among different web-based information services. The ever growing complexity of such systems, as well as the growing number of users, along with the limited user knowledge in using each particular system, bring about requirements for elaborating design solutions that enable systems to adapt to user needs, preferences and the system platforms used. Most of the solutions commonly applied are commercial ones and quite often their methodologies are rather confidential. In this paper we have presented a complete general methodology for constructing adaptive user interfaces for multimodal web-based systems. More precise procedures, however, can be developed and applied only for a particular system application.

The potential application of the method presented here is quite promising. We hope that in the near future we will be able to evaluate this method by implementing it in a running application. The application shown in our example, i.e., the conference portal, is one of the candidates.

At present we are working on different adaptive web-based applications. One is a conference information system to be used for a national conference on multimedia and web-based applications, as well as other conferences organized by our University. For the interface construction we are using mainly Flash, but also other technologies, such as XML, WAP, and ASP. The potential users of that system are quite differentiated but cannot be easily defined. They have different sexes, ages, positions (from students to full professors), and educational backgrounds (e.g., computer and information scientists, electronics, and librarians). They are, however, difficult to classify manually. Many of the conference participants also took part in past conferences, so their papers can be used for extracting data. Based on the keywords from their past papers, as well as abstracts and full texts from current papers, we can compose their profiles. The final efficiency of the system could be evaluated by filling in questionnaires during or after the conference.

The system will be accessed from PC platforms, but in the future handhelds with GPRS-based Internet access or UMTS platforms can also be considered. This will be possible since Flash, the selected technology for interface implementation, can also be used on many different platforms such as PC's, information kiosks on the conference site, and handhelds.

Acknowledgements. The authors would like to thank the reviewers for their useful remarks. Special thanks go to Ms Marta Mrozek for her help in working out the graphical form of the figures.

References

1. Arafa Y, Mamdani A (2000) Virtual personal service assistants: towards real-time characters with artificial hearts. In: Proc intelligent user interfaces. New Orleans, LA USA, pp 9–12
2. Barthelemy JP (1988) Thresholded consensus for n-trees. J Classif 5:229–236
3. Bogart KP (1973) Preference structure I: distance between transitive preference relations. J Math Sociol 3:455–470
4. Breese J, Heckermen D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. Technical report, Microsoft Research, October
5. Brown SM, Santos E, Banks SB, Oxley ME (1998) Using explicit requirements and metrics for interface agent user model for interface agent user model correction. In: Proc 2nd Int Conf Autonomous Agents, Minneapolis, Minnesota, USA, pp 1–7
6. Dattola RT (1968) A fast algorithm for automatic classification. Report ISR-14 to the National Science Foundation, Section V, Cornell University, Department of Computer Science
7. Day WHE (1988) Consensus methods as tools for data analysis. In: Bock HH (ed) Classification and related Methods of Data Analysis. Proc IFCS'87, North-Holland, pp 317–324
8. Estivil-Castro V, Yang J (2001) Categorizing visitors dynamically by fast and robust clustering of access logs. In: LNAI 2198. Springer, Berlin Heidelberg New York, pp 498–507
9. Expression of Interest (2002) Thematic Priorities. FP6. http://www.cordis.lu/fp6/eoi-instuments/infosoc.htm
10. Fleming M, Cohen R (1999) User modelling in the design of interactive interface agents. In: Proc 7th Int Conf User Modeling, pp 67–76
11. Hackos JT, Redish JD (1998) User and Task Analysis for Interface Design. Appendix C. John Wiley and Sons, New York, pp 457–478
12. Hedberg SR (1998) Is AI going mainstream at last? A look inside Microsoft Research. IEEE Intell Syst 3/4:21–25
13. International Standard ISO 9241-10 (1996) Ergonomic requirements for office work with visual display terminals (VDTs) – Part 10: Dialogue principles
14. International Standard ISO 9241-11 (1997) Ergonomic requirements for office work with visual display terminals (VDT's) – Part 11: Guidance on Usability
15. Keeble RJ, Macredie RD, Williams DS (2000) User environments and individuals: experience with adaptive agents. Cognition Technol Work 2:16–26

16. Kobsa A, Koenemann J, Pohl W (2001) Personalized hypermedia presentation techniques for improving online customer relationships. Knowl Eng Rev 16(2):111–155
17. Lieberman H (1997) Autonomous interface agents. Proc CHI 97, ACM, pp 67–74
18. Maglio PP, Barrett R, Campbell CS, Selker T (2000) SUITOR: an attentive information system. In: Proc 2000 Int Conf Intelligent User Interfaces, pp 169–176
19. McMorris FR, Mulder HM, Powers RC (2000) The median function on median graphs and semilattices. Discrete Appl Math 101:221–230
20. McMorris FR, Powers RC (1995) The median procedure in a formal theory of consensus. SIAM J Discrete Math 14:507–516
21. Mobasher B, Cooley R, Srivastava J (2000) Automatic personalization based on Web usage mining. Commun ACM 8(43):142–151
22. Nasraoui O, Frigui H, Joshi A, Krishnapuram R (1999) Mining Web access logs using relational competitive fuzzy clustering. In: Proc 8th Int Fuzzy Systems Association World Congr, August 1999, pp 28–36
23. Newman WM, Lamming MG (1996) Interactive System Design. Addison-Wesley, Harlow
24. Nguyen NT (2001) Consensus-based timestamps in distributed temporal databases. Comput J 44(5):398–409
25. Nguyen NT (2000) Using consensus methods for solving conflicts of data in distributed systems. In: LNCS. vol 1963, pp 409–417
26. Nguyen NT (2001) Using distance functions to solving representation choice problem. Fundamenta Informaticae 48(4):295–314
27. Peerce J, Rogers Y, Benyon D, Holland S, Carem T (1996) Human-Computer Interaction. Addison-Wesley, Harlow
28. Quinn LM, Pawasarat J (2001) Confronting anti-urban marketing stereotypes: a Milwaukee economic development challenge. June 2001. http://www.uwm.edu/Dept/ETI/purchasing/markets.htm
29. Rijsbergen C v (1979) Information Retrieval. 2nd edn. Butterworths, London
30. Schneiderman B, Maes P (1997) Debate: direct manipulation vs. interface agents. Interactions 4(6):42–61
31. Shneiderman B (2000) Universal usability. Commun ACM 43(5):85–91
32. Shearin S, Lieberman H (2001) Intelligent profiling by example. In: Proc Conf Intelligent User Interfaces, ACM Press, 2001
33. Sobecki J, Nguyen NT (2001) Consensus-based adaptive user interface for universal access systems. In: Stephanidis C (ed) Proc 9th Int Conf Human-Computer Interaction and 1st Int Conf Universal Access in Human-Computer Interaction. LEA, London, vol 3: pp 112–116
34. Sobecki J (2001) One suits all – is it possible to build a single interface appropriate for all users? In: Grzech A, Wilimowska Z (eds) Proc 23rd Int Scientific School ISAT, PWr Press, Wroclaw, pp 125–131
35. Spolsky J (2001) User interface design for programmers. Apress LP
36. Stary C (2001) User diversity and design representation: towards increased effectiveness in Design for All. Int J UAIS 1:16–30
37. Stephanidis C, Savidis A (2001) Universal Access in the Information Society: methods, tools, and interaction technologies. Int J UAIS 1:40–55
38. Whalen D (2002) The Unofficial Cookie FAQ, Version 2.54. Contributed to Cookie Central by David Whalen. http://www.cookiecentral.com/faq/#2.7
39. Wooldridge M, Jennings NR (1995) Intelligent agents: theory and practice. Knowl Eng Rev 100(2):115–152

Appendix

Algorithm 1

Given: Set J (with repetitions) of n number intervals, $J = (i_j | i_j \in I, j = 1, 2, \ldots, n)$, and distance function ∂_1.

Result: Consensus $i = (i_*, i^*)$ satisfying postulates **P1**–**P6**.

Program

```
BEGIN
    If n = 1 then set i = i_1 and go to END else
    Create interval i' = (i'_*, i'^*) such that
        i'_* = min{i_{j*} | i_j = (i_{j*}, i*_j) ∈ J} and
        i'^* = max{i*_j | i_j = (i*_j, i*_j) ∈ J};
    Set S = Σ_{l∈J} ∂1(i', l)/* serving to determine such i
        that the sum Σ_{l∈J} ∂1(i, l) is minimal */
    For k_* = i'_* to i'^* do
      For k* = k_* to i'^* do
      Begin
          Create interval k = (k_*, k*);
          Calculate D1(k) = Σ_{l∈J} ∂1(k, l);
          If D1(k) < S then set i = k and S = D1(k);
      End;
END.
```

Algorithm 2.

Given:

- m binary matrices $\boldsymbol{A}^{(k)}$ (for $k = 1, \ldots, m$) of dimension $n \times n$ representing rankings of n values.
- Matrix $\boldsymbol{W}$ of dimension $1 \times m$ representing the weights of the interfaces, the values of which belong to the interval [0,1].

Result: Matrix $\boldsymbol{C}$ of dimension $1 \times n$ representing consensus satisfying postulates **P1**–**P6**.

Program

```
Var
  B: array[1..n, 1..n] of real;
  E: array[1..n] of {0,1};
BEGIN
      For i := 1 to n do for j := 1 to n do B[i, j] := 0;
      For l := 1 to m do
        For i := 1 to n do
        For j := 1 to n do
          B[i, j] := B[i, j] + A^(l)[i, j] * W[l];
        For i := 1 to n do
        For j := 1 to n do
          Begin
            If B[i, j] < (m/2) then B[i, j] := 0;
            If B[i, j] ≥ (m/2) then B[i, j] := 1;
          End;
        For i := 1 to n do E[i] := 0;
        For i := 1 to n do
          Begin
            k := 0;
            For j := 1 to n do k := k + B[i, j];
            k := n − k + 1;
            If E[k] = 0 then E[k] := i
            Else
            Begin
              l := k + 1;
```

```
            While E[l] > 0 do l := l + 1;
            E[l] := i;
          End;
        End;
      C := E
END.
```

Algorithm 3.
Given: Collection K of k subsets s_j of a finite set $X = \{x_1, x_2, \ldots, x_n\}$
Result: Consensus s^* satisfying postulates **P1**–**P6**.

Program

```
BEGIN
    s* := ∅;
    For i := 1 to n do
      Begin
        Determine Occ(i) as the number of
        occurrences of element x_i in the sets
        of collection K;
        If Occ(i) ≥ n/2 then s* := s* ∪ {x_i}
      End;
END.
```