FoodManager: a Cooking, Eating and Appliance Controlling Support System for the Elderly

Rosa Iglesias, Ibai Ibarguren, Nuria Gómez de Segura
Ikerlan-IK4
J.M. Arizmendiarieta, 2
20500 Mondragón, Gipuzkoa, Spain
{riglesias, ibarguren, ngomezsegura, }@ikerlan.es

Julen Ugalde
FAGOR ELECTRODOMESTICOS
Bº San Andrés, 18
20500 Mondragón, Gipuzkoa, Spain
jugalde@fagorelectrodomestico.s.es

Leticia Coello, Miren Iturburu
Fundación Instituto Gerontológico Matia-INGEMA
Camino de los Pinos, 27 bajo
20018 Donostia, Spain
{leticia.coello, miren.iturburu}@ingema.es

ABSTRACT
These days, many support systems are being developed to improve independence and quality of life of elderly and impaired people at home. Most of them have been hitherto focused on providing home healthcare-related services, and little attention has been paid to cooking and eating activities. On the one hand, the role of supporting eating activities is likely to become increasingly important. Since old age is often associated with memory impairments, it would be useful to provide them with menus including a variety of healthier meals. Furthermore, it is of paramount importance that these suggested meals are built up based on disease pathologies, health condition and user preferences. On the other hand, comprehensive cooking guidelines and food and shopping list handling are also needed. With regard to cooking directions, it seems desirable that users do not need to manage complex household appliances like the oven. This can be achievable thanks to the technologies developed in the area of Home Automation.

In this paper, a cooking support system, namely FoodManager, is discussed. This was designed and developed to deal with all the requirements described above. Its interface design and navigation was specially designed for the elderly. Testing performed with ten elderly people (ages from 58 to 81) provided some understanding about its usability and simplicity to use.

Categories and Subject Descriptors
H.4.m [Information Systems Applications]: Miscellaneous;
H.5.2 [Information Interfaces and Presentation]: User Interfaces – Interaction Styles;

General Terms
Performance, Design, Experimentation, Human Factors

Keywords
Ambient Assisted Living, Elderly People, Eating, Cooking, Smart Kitchen

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright © 2010 ACM ISBN 978-1-4503-0071-1/10/06.. $10.00

1. INTRODUCTION
It is a fact that population aging pattern is growing in the world [1] and that it is of paramount importance to maintain good health and medical care for everybody. Due to the decline of visual, hearing, cognitive and mobility abilities, elderly people must often face up to any obstacle that presents itself when carrying out everyday activities. Help care technology-based support systems can help them in surpassing these barriers.

Since healthcare becomes more important as we age, the landscape of support systems has been traditionally driven by healthcare support systems. These range from home monitoring, home diagnostics, smart medicine cabinets, tele-medical assistants to medication reminders [2-5]. These result in less time in: assisting them in an emergency situation, diagnosing illnesses or assessing medications and less visits of elderly people to medical facilities. On the other hand, good health is tightly related to good nutrition and a balanced diet. Currently, this issue is being put in the limelight. A relationship between diet and disease has been recognized by scientific societies.

These days many household appliances are supposed to be better, however their use is increasingly getting complex. Clear examples are washing machines and ovens. In the beginning, their use was as simple as ‘throw the clothes in or put food in and press a button’. Now they are provided with several buttons, displays and other interface controls to add features that people may want to have. This complexity does not often help elderly or impaired people, who usually do not know how to use them.

This paper describes a system, called FoodManager: (1) for planning weekly menus meeting nutritional needs, health conditions and personal preferences of an individual (with easy cooking guidelines), (2) for managing the oven parameters wirelessly (hands-free operation) and (3) for food and shopping list handling. This paper also reviews the benefits brought by FoodManager according to ten elderly people.

2. RELATED WORK
These days, many attempts are being pursued to achieve a Smart Kitchen that assists users in cooking, that assesses balanced diets, that helps users find recipes, or that provides automated control of power measure. These new services can be classified into: food preparation or cooking, food recommendation and controlling
cooking appliances. So far, many systems have been developed to address one of these categories and most of them have been designed for non-impaired people. Another issue to be considered is that currently most of them are not affordable to everybody, which would be desirable. Most of them are based on numerous displays, webcams, speakers, sensors, and so on.

For example, many of the proposed cooking support systems give instructions for a chosen recipe by means of multimedia contents or by using augmented reality [6-8]. Others offer food recommendation based on social recommendations [9, 10]. In these systems many challenges still exist, mainly from equipment issues to adaptation or suitability for the elderly.

In [6], an ambient kitchen was designed to support people with cognitive impairments in preparing meals and drinks. By means of a set of pressure sensors, RFID (Radio Frequency IDentification), accelerometers attached to kitchen objects and webcams, user actions and intentions were captured, and accordingly situated cooking-related prompts were provided to the user through displays. This system was mainly focused on inferring context and prompting adequate kitchen tasks and no evaluation was carried out.

Other authors presented a kitchen augmented with projected information onto objects (i.e. fridge, countertops, cabinets) and surfaces to support users [7]. In this kitchen, virtual recipe guidelines are displayed and hand gesture recognition is provided to interact with, food temperature in a burner is calculated by an infrared thermometer, food inside the fridge is displayed in the fridge door (a webcam image) and leds inside objects (drawers) are used to show a user where a utensil or container is. However this system falls short of expectations for the elderly, since it seems to be more focused on fast and efficient kitchen tasks.

The work in [8] focuses on optimizing cooking procedures even when cooking several recipes in parallel. The authors proposed a cooking navigation system that provided text, video, audio and image-based information. The displayed information is based on an algorithm that considers action units (i.e. fry eggs, cut onions), the meal duration, and the serving time, among others. Again, this research is far from the goal of this paper.

In terms of food recommendation, a method to recommend easy recipes considering different parameters (i.e. seasonings at home, preferences, user schedule, data and time) is described [9]. The assessment of the recipe easiness is based on an algorithm that bears in mind the number of ingredients, user’s familiarity for the seasonings and the cooking methods involved in the recipe. Another similar work, called SuChef, is described in [10]. This prototype deals with meal recommendation suggested by the system or by friends and family members, and with supporting users while cooking.

On the other hand, with regard to the control of cooking appliances little is found in the literature. To the best of our knowledge, just a research work has been reported about it. In [11], a system is proposed to automatically control a microwave. The microwave is equipped with an RFID reader, and providing that foodstuffs contain RFID codes, these codes are read. This information determines the microwave power measure for the food to be cooked in the microwave. This would be very interesting for impaired and non-impaired people.

Unlike these research works, this research effort will focus on providing cooking and eating care for the elderly. In that way, FoodManager will provide them with weekly healthy menus, food and shopping list handling and automatic control of the oven parameters through a home automation line.

3. HOME INFRASTRUCTURE

The home infrastructure consists of a key component called the Home Gateway, the Fagor Home Automation Network and a display to show FoodManager’s interface to the user (Figure 1). Whereas FoodManager’s core (application logic) is hosted on the Home Gateway to provide users with eating and cooking-related services, the home automation network allows the control and communication with household appliances, sensors and actuators. The controller of this network, namely Maior-domo, centralizes all commands and status of appliances and domotic devices connected through BDF (Fagor power line Domotic Bus), and it has a 485 serial interface [12].

![Figure 1. Home infrastructure](image)

The Home Gateway developments are based on OSGi [13], a service oriented and component-based framework that promotes interoperability of applications and services. The Home Gateway allows accessing to the Internet, for example, the communication with an external server for downloading new recipes. On the other hand, it also allows communication with the commercial Fagor Automation Network.

4. FOODMANAGER

This section describes FoodManager’s general architecture and the services that it provides: (1) supporting weekly menus with simple cooking instructions, (2) managing the oven parameters according to a selected recipe and (3) food and shopping list handling.

4.1 FoodManager Architecture

The architecture consists of three separated layers: data, logic and interface. The logic layer execution is performed in the Home Gateway; whereas the interface layer can be running in one or more computers in order to display information to users. The application layer is the layer in which the application functionality
is defined and services will be accessible. The interface or presentation layer displays the user interface and after user interaction, the corresponding service invocation published by the logic layer will be carried out.

4.2 Supporting weekly menus with simple cooking directions

This section describes the functionality of supporting weekly menus by FoodManager and how the recipe database was created.

4.2.1 Description

FoodManager supports weekly menus based on user disease pathologies, health conditions and preferences. Weekly menus are displayed for seven days from the current day. There are two menus per day (lunch and dinner) (Figure 2). Each menu contains three recommendations: an appetizer or starter, a main course and a dessert. Starters include salads, cheese and vegetables, among others; whereas main courses are simple recipes mainly to be cooked in the oven. Desserts include yogurts, fruits or dessert baking recipes.

Users can also change the planned recipes by browsing and choosing from a list of recipes. These are categorized into five types: appetizers, vegetable, meat, fish and desserts.

4.2.2 Recipe database

A recipe database was created with 40 recipes. In the creation of the database several requirements were identified. First of all, all recipes/ menus were ‘labelled’ with attributes that defined whether or not that recipe was for diabetics or people with hypertension, overweight, gluten allergy, and/or high cholesterol rates, among others. This allows the system to search for suitable recipes according to a user’s disease, health condition or preferences.

With regard to preferences, users can select their preferences or food dislikes by choosing or discarding ingredients from a list. To achieve the latter, a search by ingredient can be carried out. Moreover, a variety range of recipes was pursued to cover all preferences. Figure 3 shows the cooking directions of a roast chicken’s recipe.

On the other hand, and since it was addressed to the elderly, simple food cooking directions were selected. Therefore, only recipes that involved a unique cooking method (in this case in the oven, baking or roasting) were selected. The ingredients of all recipes can automatically be updated according to the number of persons.

4.3 Managing the oven parameters

Home automation involves a group of technologies that allow the communication, control and manage of diverse functionalities in a building or at home. The widespread and well-known home automation services have traditionally focused on switching on/off different appliances or programming them remotely. Home automation can also bring new solutions to enhance our comfort and quality of life. For managing the oven parameters, a home automation controller, called Maior-domo, is used [12].

Once users select a recipe and they are ready to cook it, they have to press the “download to oven” button and this action triggers the delivery of the recipe oven parameters.

First, the information is sent wirelessly to the Home Gateway. The Home Gateway sends the oven frames to Maior-domo (controller) that sends adapted frames to the oven through power-line communications. Once all the information is received by the oven, the oven display shows the recipe name and parameters and the oven is started up. Parameters sent to the oven are: recipe name, oven function, temperature, time and special parameters such as preheat function.

4.4 Food and shopping list handling

In addition to providing weekly menus, FoodManager handles food in the fridge, and according to the planned weekly menu it computes the shopping list of food items for the user. Therefore, FoodManager takes into account the list of ingredients required for the weekly menu and the available foodstuffs in the fridge. This task was not tested by the users, since it was beyond the testing goal.

5. USER EVALUATION

The main objective of the evaluation was to assess the usefulness and the ease of interaction of FoodManager. Firstly, the laboratory where the usability test was carried out is described, and secondly the description of each test is introduced.
5.1 Environment Description
The participants tested FoodManager in the kitchen of an embedded systems laboratory (Figure 4). This laboratory is similar to a common kitchen provided with all mainstream household appliances. The FoodManager graphical interface was run in a computer provided with a 12” touch screen (Figure 5). It was considered that a finger-operated touch screen was more suitable for the elderly.

5.2 User Testing Description

5.2.1 Participants
The participants who tested the system were selected from a retirement home and none of them had participated in a testing like this before (Figure 5). They were 10 (5 males and 5 females) participants with ages ranging from 58 to 81 years old. Six of them were married (3 females and 3 males), two woman widows, a separated man and another man was single. Two participants only worked, and the remaining ones were retired. Five of them had secondary education and the rest had primary education. Before starting the experiment, each of them was asked several questions to know how familiar they were with the use of PCs, mobile phones and other related technologies. Six people said they had never used a computer, whereas two participants used it everyday, a person once a week and the other one several times over the week.

5.2.2 Task description
FoodManager was introduced to the participants with a brief description. First of all, the menu planner was described about how it planned what to eat for lunch and dinner according to health-related and user preferences issues. The health-related issues involve: diseases, health conditions and allergies. Next, the recipe viewing was shown and how they could view the recipes by food groups (i.e. vegetables, meat, fish, and desserts). Afterwards, it was mentioned how according to the food items in the fridge and the week menu planning it made the shopping list.

The following actions were carried out by the participants. They were in the kitchen and used the touch screen to interact with FoodManager. The task to be performed was to view today’s menu and to cook the main course for two people. So users had to first select ‘week planning’ and then ‘today’s main course’. This course was ‘roast chicken’. Then, they changed the number of servings. Once they read over, they selected the ‘Download to oven’ button and the oven was automatically switched on with the temperature and time required to cook the selected recipe.

5.2.3 Evaluation purposes
Usability (performance and the ease of interaction) and effectiveness of FoodManager were evaluated. To achieve it, design issues, including representative icons, colors, size and font size were also taken into account. Furthermore, each option was asked to be or not useful and whether or not the recipe information was easy to follow and to understand. On the other hand, they were asked to show their preferences about the fact of using the IDBlue pen instead of the touch screen. The IDBlue pen is a Bluetooth stylus tip for interacting with RFID tags [14].

To evaluate FoodManager, a questionnaire was filled up for every user. These questionnaires had five-point answers of the type: very easy, easy, indifferent, difficult and very difficult, and for other questions, useless, not too useful, indifferent, useful and very useful.

6. RESULTS AND FINDINGS
All the participants completed successfully each task, which suggests the intuitiveness and the ease of use of the system. This section shows the results and findings when using FoodManager.

6.1 Food Manager Evaluation
With regard to the use of mobile phones, all of them but one (with 66 years old) had been using it for 2 to 12 years. When it comes down to healthy problems, three of them had no problem, two people suffered from mobility problems, two from visual impairments and three had diabetes, hypertension and/or fatigue. Just four participants had home care services for cleaning chores.
some icons, in particular the accept (✓) and reject (X) icons. Once each icon was explained they associated with the corresponding action with no problem.

In terms of the recipe information given, the majority regarded it to be adequate. Following, they were asked about the fact of downloading the recipe to the oven. Half of the participants conceived it as very useful, 4 of them as useful and one of them did not answer.

6.2 User Interaction
The touch screen interaction was considered to be easy (8 participants) or very easy (2 participants) (Figure 5).

6.3 Other suggestions
Related to FoodManager, users gave several ideas about what they liked the most or how to improve the system. They liked the fact that the application planned a 7-day menu, so they did not need to think about what to cook every day. Particularly, it was by far valorated that the manager adapted the menu to the health situation and pathologies.

Colors were fine, but they suggested that they should be more vivid (4 users) and with more contrast between foreground and background colors. On the other hand, they thought that the font size was enough, although three users thought bigger size would be better. The viewing inclination of the screen was also highlighted by 5 participants. The touch screen was on a table and its inclination could have improved the vision clarity. Only a person stated that the application could be used by a person who was not able to cook.

A user pinpointed that he would like the manager to be integrated in the oven, to avoid having many devices surrounding him. Another person suggested the buttons + and − to add servings should be removed. The button with the number of servings should be enough to increment them. Three users mixed up with the buttons ✓ y X, they preferred an OK button.

When it comes down to interaction, all the users showed their preferences to the touch screen due to handling and fast use rather than to the IDBlue pen.

Just a person thinks that, although the application is easy to use (the oldest person), she would not use it. The reason is that it conceived it as an autonomy loss.

7. CONCLUSIONS AND FUTURE WORK
This paper presents an assistive system to support the elderly in eating and cooking activities. This system called FoodManager provides menu planning meeting nutritional needs, health conditions and personal preferences of an individual, managing the oven parameters wirelessly (hands-free operation) and food and shopping list handling.

The results of the usability test with 10 elderly people were promising, especially for managing oven parameters. Most of the participants regarded the system as useful and they liked it quite a lot. Managing the oven parameters was by far positively seen as the most useful part of the system. In that sense, we envision Home Automation-related and other pervasive computing technologies to be useful to elderly individuals for automatic control of household appliances.

Regarding next steps, it is currently being studied whether by means of RFID tags on clothing and a home automation controller is possible to set up washing-machine controls (i.e. temperature, laundry program) according to clothing colors and types. Moreover, the washing machine could be stopped if users mixed clothing types or colors.

8. ACKNOWLEDGMENTS
This work has been funded by the AMIE project (ITEA2- TSI-020400-2008-50) within the AVANZA I+D program from the Spanish Ministry of Industry, Tourism and Commerce (Ministerio de Industria Turismo y Comercio).

9. REFERENCES


ABSTRACT

Pervasive technologies are essential at any field of healthcare issues. Especially the growing number elderly necessitates the development of measures to support care-dependent people at their private homes. One important research field are so called Smart-Rooms, local agglomerations of smart appliances, whose composition is prone to frequent, unforeseeable, and substantial changes. These compositions are able to react on the user's needs by interaction of individual appliances. Normally, these environments are equipped with smart sensors to infer the activities of the user. In case the environments know the current activity they can try to infer the next possible actions the user will take, thus assist the user pro-active. On the other side, these environments should be able to manage information or data the user is interested in. Therefore it is necessary to offer solutions for data access and information exchange in ubiquitous environments with respect to the users needs and skills. Therefore, we present BlueS (Bluetooth-Services), our service based solution for data access and information exchange in spontaneous linked smart environments, e.g. Bluetooth networks. In this paper we want to focus on smart caching approaches for data reuse. A full description of BlueS can be found in [10].

Categories and Subject Descriptors

C.2 [Computer-Communication Networks]: Distributed Systems;
H.2 [Database Management]: Languages;
H.4 [Information Systems Application]: Miscellaneous

1 Here we are talking about users of smart rooms with respect to nursing-service scenarios. So this could be the care-dependent person itself, relatives or caregivers.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright © 2010 ACM ISBN 978-1-4503-0071-1/10/06... $10.00

General Terms

Pervasive Computing

Keywords

Semantic Caching, Distributed Data Management, Mobile Databases

1. INTRODUCTION

At the university of Rostock we are researching on assistive environments in different points of views.

At MuSAMA (Multimodal Smart Appliance Ensembles for Mobile Application, [9]) supported by the DFG (German Research Foundation) we believe that ubiquitous machine intelligence envisioned for our future everyday environments will be provided by dynamic ensembles, agglomerations of smart appliances, whose composition is prone to frequent, unforeseeable, and substantial changes. To evaluate our models and algorithms we are equipped with a small lab, representing smart-room scenarios. Currently all devices like lights, sensors or data sources in general are connected via wire. Now we research on mobile P2P architectures for dynamic behavior of individual appliances. One part of it is data management and query processing in spontaneous linked environments like MANETs.

Following Franklin [4] we believe that data management is essential to reach the goal of pervasive assistance in smart scenarios:

"In order for computing to fade into the background while supporting more and more activities, the data required to support those activities must be reliably and efficiently stored, queried, and delivered. Traditional approaches to data management such as caching, concurrency control, query processing, etc. need to be adapted to the requirements and restrictions of ubiquitous computing environments. These include resource limitations, varying and intermittent connectivity, mobile users, and dynamic collaborations."

To evaluate our results healthcare scenarios are one important part. One project dealing with this fact is the MARIKA project (Mobile Assistance for Route Information and Electronic Health Record) we are currently working on.

This project focuses on the conception and development...
is easier to handle. As there is no need for join operations the cache management is simpler resulting in long time periods for data analysis. That’s why we had a look on mechanism which are able to identify correct regions as outlined in section 4.

As you can imagine there are some other cases as multiple region containment and multiple query containment which have to be considered. So the question is which regions have to be chosen to answer \( q \) and to keep \( q' \) quite simple. Wrong regions can cause very complex postprocessing steps resulting in long time periods for data analysis. That’s why we had a look on mechanism which are able to identify correct regions as outlined in section 4.

3. BLUES: DATA- AND INFORMATION SERVICES IN SMART ENVIRONMENTS

In general, information- and data-management systems are tailored for specific types of queries on restricted data.
sets according to a particular semantic. To illustrate this, consider a scenario of information service applications in pervasive health care scenarios as described above. Among other features all available devices should be able to inform about their stored data in real time. So standardized interfaces needed to support such information application.

Services are able to fulfill this task as observed in [2]:

"The proliferation of these services along with the ever increasing use of diverse computational devices such as mobile phones or PDA’s entails the potential to provide users with ubiquitous access to all kinds of relevant and up to date information."

Typically service interfaces are described by \( \{ f, X, Y \} : f \) the name or identifier of the service, \( X \) the input parameters and \( Y \) the output parameters. In [8] additional parameters like preconditions and postconditions are given. Using these information, heterogeneous software and hardware resources may be discovered and integrated transparently. BlueS is a framework for data- and information-service discovery and data exchange in spontaneously linked environments.

### 3.1 Information System Architecture

Currently there are four main layers:

1. **communication layer**
2. **selection layer**
3. **registration layer**
4. **data layer**

As observed in [10] layers 2 and 3 describe the basic application data model (3) and operations (2) which 1 and 4 are based upon so they are described first.

**registration layer:** Since data and information can be accessed via services, i.e. bluetooth services, local- and global services have to register at the registration layer. They are mapped to available semantic data representation like taxonomies via categorization, clustering algorithms or semi automatic approaches so they can be retrieved by browsing, zooming and pruning using the approach of dynamic taxonomies as represented by [11].

**selection layer:** The selection layer contains the dynamic part of the application data model, including basic operations for the data and graph part of the domain specific model. To support difference data mapping and ranking strategies we provide plugin-based interfaces.

**communication layer:** Based on the fact that data of (semi-) domain specific environments can be modeled with graph representations like taxonomies and data-and information services can be mapped on them we have to concentrate on query languages for graphs. With respect to the projects NuSAMA and MARICA the conceptual design of means for users, who have no notion of any kind of formal query languages, have to be taken into consideration. That is why we are using the conceptual model of dynamic taxonomies as a visual paradigm for accessing complex information bases and an effective, complete and easy to use query model. It serves as an interface between the internal operations and the user respectively, who has no previous knowledge of any formal query language.

**data layer:** In this paper we want to focus on the data layer. Indexing and caching approaches are necessary to support efficient query evaluation on mobile devices in pervasive scenarios. As it is not guaranteed that all necessary data sources are available all the time there is a need of data reuse in pervasive scenarios. Therefore we research on caching strategies like semantic caches as outlined in section 2: by representing the relational data and semistructured information with semantic descriptions like semantic regions, semantic caches are able to decide which tuples or objects in general are already available to answer incoming and local queries. So we are able to generate remainder queries describing non-available objects. They are exchanged within the dynamic ensemble. As there are some regions available for query answering the data layer has to decide which region should be chosen. At section 4 we present mechanism which are used to overcome this problem.

### 4. INITIAL WORK

As outlined in section 2 the use of semantic regions is well known in the research areas of relational and unstructured data management. We started to look at information retrieval systems. Here queries \( q \) (boolean queries) are terms \( t_1, \ldots, t_n \) combined with operators like \( \land, \lor, \neg \) and \( \sim \). Disjunctive queries \( q = t_1 \land t_2 \land \ldots \land t_{i-1} \lor t_i \land t_{i+1} \land \ldots \land t_n \) can be operated in two queries \( q_1 = t_1 \land t_2 \land \ldots \land t_{i-1} \) and \( q_2 = t_i \land t_{i+1} \land \ldots \land t_n \) which are answered separately. The results can be combined with \( P(q_1) \cup P(q_2) \). As we only have to answer conjunctive queries now it is guaranteed that each region is also a conjunctive query and obviously each remainder query \( q' \) as observed in [1]. Our aim is to keep the remainder query quite simple to ensure fast and efficient answering. That’s why we are using the similarity function \( \text{DIFF} \): Let \( q \) be a conjunctive query and \( r_i \) a semantic region (or a query describing the region). \( \text{DIFF}(q, r_i) \) counts the terms of \( r_i \) which are not available in \( q \). By using \( \text{DIFF}(q, r_i) \) there are some advantages:

Let \( T(q) \) be the set of terms available in \( q \)

- If \( \text{DIFF}(q, r_i) \leq 1 \): the level of complexity of \( q^* \) keeps small.
- If there are two regions \( r_1 \) and \( r_2 \) with \( \text{DIFF}(q, r_1) = 1 \) and \( \text{DIFF}(q, r_2) \geq 2 \): region \( r_1 \) is more helpful for answering \( q \) as the additional terms of \( r_2 \) restrict the set of available objects.
- If \( \text{DIFF}(q, r_i) = 0 \): all terms of \( r_i \) are in \( q \) \( (T(q) = T(r_i) \lor T(q) \supset T(r_i)) \). Here the probe \( q^p \) is identical with \( q \) and the remainder \( q^* \) is empty.
- If \( \text{DIFF}(q, r_i) = 1 \): \( q^p = q \land n \) and \( q^* = q \land \neg n \) with \( n \) the term of \( r_i \) which is not available in \( q \).

As we are now able to choose the correct regions at the client side we now have to look at the remainder queries which are broadcasted or directly send to other devices or base stations. Here there are two cost factors we are considering. Let \( A \) be the set of objects resulting in answering \( q \):

1. Costs \( C_A \) for sending answers based on the count of objects \( |A| \) contained in the answer \( A \)
2. Costs for calculating \( A \)

Let \( S(t) \in [0,1] \) be the selectivity of any term \( t \). This means if \( S(t) = 0 \) the answer contains all objects available. On the
other side $S[t] = 1$ results in $|A| = 0$. As we observed we are now able to calculate $C_A$ with respect to any remainder query $q'$:

$$C_A = C_{q'_i} = C_{q'_{r_i/n}} \cdot S(n)$$

Obviously the query describing region $r_i$ is more specific then the region $r_i/\{n\}$ because $r_i$ contains another term $n$. So it is clear that this region $r_i$ contains less objects ($|A_{r_i}| \leq |A_{r_i/n}|$). The only difference of $|A|$ and $|A_{r_i/n}|$ can be given by $S(n)$.

Example: Let $q$ be a query we want to answer given by $q = t_A \land t_B$. The SC contains two regions $r_1 = t_B \land t_C$ and $r_2 = t_B \land t_D$. As $DIFF(q, r_1) = DIFF(q, r_2) = 1$ the regions can be choosen for query $q$. If we know i.e. that $|r_1| = 10$ and $|r_2| = 100$ we also know that $S(t_C) > S(t_D)$ and that is why $S(\neg t_D) < S(\neg t_C)$. So the answer $A$ for the remainder query for region $r_2$ ($q'_r = t_A \land t_B \land \neg t_D$) would have less objects. Finally the costs for sending $A$ would also be less.

If we are not able to find any region $r$ with $DIFF(q, r) \leq 1$ $DIFF$ can also be used as a heuristic to identify devices the remainder should be send to:

Let $INT(q, r) = |r| - DIFF(q, r)$ be the number of terms of region $r$ which are available at $q$. Using this value we are able to calculate the average similarities of all regions $R_d$ belonging to a device $d$ in relation to a query $q$:

$$AVG_d(q) = \frac{\sum_{r_i} INT(q, r_i)}{|R_d|}; r_i \in R_d$$

Using this formula an ordered list of devices we want to communicate with is given by $max(AVG_d(q))$. To evaluate this results our prototype BlueS contains a device emulator which can be used for measurements of bytes transferred by-segments in relation to the number of available semantic regions available at any device. First tests indicated that there is a critical number $k$ of semantic regions which should be choosen. Using more than $k$ regions the communication gain is marginal but the management overhead increases.

5. CONCLUSION

In this paper we have outlined the problem of data- and information management in pervasive healthcare scenarios. Therefore we introduced the projects MuSAMA and MARIKA. As we observed here caregivers, care-dependent persons or relatives can be supported by information access and exchange during the whole period of caregiving. We outlined that there is a problem of data source availability which we tried to overcome by caching approaches like semantic caching. We are currently working on caching strategies for structured data sets. Here the calculation of remainder queries is far more complex so future work includes the conception, development and execution of evaluation strategies for structured data-sets.

6. ACKNOWLEDGEMENTS

The work of the author is supported by a grant of the German National Research Foundation (DFG), GRK 1424.

7. REFERENCES


Communicating Through Preferences
Nicolas Spyratos
Université Paris Sud 11 and CNRS LRI - Bat. 90, 91405 Orsay Cedex, France Tel.: +33169156629 spyratos@lri.fr

Dimitris Kotzinos
TEI of Serres and FORTH-ICS P.O. Box 1385, 71110 Heraklio, Greece Tel.: +302810391635 kotzino@ics.forth.gr

ABSTRACT
This paper is concerned with the creation and management of events in a community of persons with common interests. We introduce concepts and tools for performing the following activities by members of the community: (a) proposing events in which the members of the community can participate (b) expressing preferences over a set of proposed events (c) choosing the events that best fit the community with respect to the expressed preferences and (d) announcing to the community the chosen events that are to take place. We focus on the design of communication and collaborative decision making, and we outline the design of an online system that supports these activities. Our proposal can be seen as a generalization of existing tools like Doodle, an online system for setting up meeting times.

Categories and Subject Descriptors
H. [Information Systems] H.4 [INFORMATION SYSTEMS APPLICATIONS] H.4.3 [Communications Applications]

General Terms

Keywords
Preference, keywords, meeting, event, communication.

1. INTRODUCTION
This paper is concerned with the electronic introduction and management of events in a community of persons with common interests. It is acknowledged today that in several activities carried out collaboratively, people need a way to choose among a set of alternatives and to communicate their choices to others. Since more often than not this can't be the decision of a single person, we need to have in place mechanisms (if possible online) supporting the decision making process. Stating preferences ([3]) over a set of alternatives becomes increasingly a way of communicating among the members of a community, replacing traditional (and more labour intensive) methods like phone calls or e-mails. It becomes also a more robust way of communicating when one can state his/her own preferences and the system evaluates them ([4], [5], [6]) against existing choices, taking into account preferences from other users at the same time as the work we present in this paper.

A typical example is that of organizing meetings for a group of people working on a common project. The meeting can be seen as an event. The person organizing the meeting (the coordinator) proposes a list of possible dates; these are the possible alternatives for the event. Each potential participant indicates availability by placing a “yes” or a “no” next to each date in the list. Then the date with the largest number of “yes” is the one chosen by the coordinator as the date of the meeting; and in case of a tie (i.e. two or more dates with the same (largest) number of "yes") it is up to the coordinator to decide on one date.

In fact, there is already (at least) one tool for organizing meetings in precisely the way just described. This tool is called Doodle [1], it is accessible through the internet, and it is gaining popularity rapidly. However, in many ways, the tools like Doodle provide limited capabilities to end users. In this paper, we propose a generalization to the basic concepts and tools offered by existing online meeting setup tools like Doodle in several ways, and along several dimensions. Our motivation comes from a number of observations:

First, one can generalize the concept of “event”. Indeed, in our previous example, where the event is a meeting, each alternative is defined by just one parameter of the form dd/mm/yy indicating a date. However, an event might be more complex. For example, suppose that the teachers of a high school are planning a cultural excursion for their students in some European city, and they are considering several options for the city to visit (e.g. Rome, Madrid, etc.) but also for the transportation means (e.g. coach, train or airplane), without forgetting the overall cost per student. Here, the event is the excursion and has three features, or attributes: the city, the transportation means and the overall cost of the excursion. Therefore, each alternative needs three attributes to be described: a specific city (such as Rome), a specific transportation means (such as train) and an overall cost. In our earlier example of organizing a meeting, the event was the meeting and had just one attribute, the date.

Second, one can do with a “milder” form of coordination, or even without coordination at all in some cases. Indeed, in our example of excursion, one can imagine that each teacher proposes one or more alternatives; the students are asked to express their preferences on the set of all proposed alternatives; the alternative with the largest number of votes wins; and if there is a tie between two or more alternatives it is the principal of the school that makes the final decision. This is a milder form of coordination as opposed to the meeting organization example, where the person calling the meeting also proposes (or rather imposes) the possible dates and takes the decision in case of a tie.

Third, some of the participants in an event might be more important than others. In our example of organizing a meeting, it
might very well be that the presence of some of the participants is indispensable for the meeting to take place, while the presence of others might not be indispensable.

In this paper, we present an approach that takes into account all the generalizations mentioned above. To this end we introduce concepts and tools for performing the following activities by the members of a community: (a) proposing events in which the members of the community can participate (b) suggesting attributes that these events carry, (c) expressing preferences over the set of events, either directly or through their attributes, (d) making the choices that best fit the community with respect to the expressed preferences and (e) broadcasting to the community the chosen events that are to take place.

We focus on the design of communication and collaborative decision making, and we outline the design of an online web based system that supports these activities. The system combines robustness and speed in calculations (that can be rather complex for a large number of events and attributes) with an intuitive and easy to use User Interface (UI). It also tries to be as open as possible allowing for “open” or “restrictive” participation in the events and for coordinated or not events. Our proposal can be seen as a generalization of existing systems like Doodle.

2. PROBLEM STATEMENT

2.1 Problem Definitions

In order to state the problem formally we need some preliminary definitions and notation regarding the basic concepts.

As we explained in the introduction, we view an event as an abstract concept with one or more alternatives, where each alternative is described by a set of attributes. More formally, we have the following definition:

**Definition 1 - Event**

An event is defined by an expression of the form \( E(Alt\text{-id}, A_1, \ldots, A_n) \), where \( E \) stands for the name of the event, \( Alt\text{-id} \) stands for the alternative identifier and \( A_1, \ldots, A_n \) stand for the attributes of alternatives. An alternative of \( E \) is defined by giving an identifier \( i \) and a value for each of the attributes \( A_1, \ldots, A_n \).

For example, the excursion event of our running example is defined by the expression

\[
\text{Excursion}(\text{Alt-id}, \text{City}, \text{Transp.}, \text{Cost})
\]

Each teacher can propose an alternative by specifying a city, a transportation means and the cost associated with the alternative.

In fact, we can view the expression \( \text{Excursion}(\text{Alt-id}, \text{City}, \text{Transp.}, \text{Cost}) \) as defining the headings of a table, in the sense of the relational database model, and the definition of an alternative as the definition of a tuple. Following this intuition, the table of alternatives shown in Table 1 shows 9 alternatives of the event Excursion. Clearly, the attribute Alt-id is a key of the table, thus acting as a tuple identifier (or as a “surrogate” for the whole tuple).

Given a tuple \( t \) of an event \( E(\text{Alt-id}, A_1, \ldots, A_n) \), we use the notation \( t.A_i \) to denote the value of tuple \( t \) on attribute \( A_i \). For example, in Table 1, if \( t \) is the second line of the table then \( t.\text{City}=\text{London} \), \( t.\text{Transp.}=\text{Train} \) and \( t.\text{Cost}=\text{900} \).

A table like the one in Table 1 is a basic tool for the creation and management of the alternatives of an event. Indeed, an event has participants and each participant can propose one or more alternatives. The set of all proposed alternatives is conveniently represented in the form of a table such as the one of Figure1. We shall call such a table the event table.

Once all proposed alternatives are collected and represented in the event table, the next step is to filter the proposed alternatives based on certain criteria. For example, it might be decided that only the alternatives with cost less than 1500 should be considered, but all alternatives involving Athens or Rome must be considered - even if their cost exceeds 1500.

The following step is to have all participants “vote” that is expressing their preferences. This can be done in one of two ways, ranking the alternatives directly or ranking the alternatives indirectly by ranking their attributes:

1/ **Ranking the alternatives directly**

After inspecting the attribute values of each alternative, the participant expresses preferences directly on the alternatives (i.e. directly on the values of \( Alt\text{-id} \) in Table 1); this can be done either quantitatively (by assigning a number between 0 and 1 to each alternative); or qualitatively, by expressing preferences between alternatives (e.g. I like alternative number 2 more than alternative number 5).

2/ **Ranking the alternatives indirectly**

In this case, the participant expresses preferences over one or more attributes of the alternatives (either quantitatively or qualitatively), and eventually priorities between the attributes. Based on this input, the system then infers a ranking of the alternatives.

We note that quantitative preferences are rather difficult to express by a casual participant, as the participant is confronted with an assignment of percentages to individual alternatives; whereas qualitative preferences are quite easy to express as the participant only has to indicate preference between two alternatives.

For the purposes of this paper, we consider only qualitative preferences ([7], [8], [9], [10]); therefore the term “preference” will mean “qualitative preference” from now on. We introduce now some definitions and notation regarding (qualitative) preferences.

**Definition 2 – Preference**

A preference over attribute \( A \) of the event table is defined to be just a pair of \( (v, v') \) of values of \( A \). Such a pair is denoted as \( v \rightarrow v' \) and interpreted as “\( v \) is preferred to \( v' \)”, or “\( v \) precedes \( v' \)”.

The set of all preferences expressed by a participant over an attribute \( A \) is called a preference relation over \( A \). A preference relation over \( A \) is denoted as \( P.A \). For example, in Table 1, the following could be a preference relation over the attribute City, consisting of three individual preferences:

\[
\text{P.City: Athens} \rightarrow \text{Vienna}, \text{Rome} \rightarrow \text{Berlin}, \text{Athens} \rightarrow \text{Berlin}
\]
We note that a preference relation over attribute A is considered to be just a binary relation over the values of A without any particular properties or constraints (such as transitivity, acyclicity and the like). The idea here is to make things easy for a participant, namely, the participant only has to indicate preference without having to pay attention to any constraint.

**Definition 3 – Precedence between tuples**

Given two tuples s, t of E, and a preference relation P.A, we say that s is preferred to t with respect to P.A, denoted s\(\rightarrow t\), if s.A \(\rightarrow t.A\) is in P.A.

For example, with respect to the preference relation P.City given in Definition 2, we have the following precedence between tuples in Table 1: 1 \(\rightarrow\) 3, 4 \(\rightarrow\) 6, 5 \(\rightarrow\) 6, 1 \(\rightarrow\) 6. Note that no other precedence between tuples is possible to infer from the given preference relation.

### 2.2 Problem Formalization

Having defined the basic concepts needed, we can now state formally the problem considered in this paper. It consists of five parts as follows:

**Proposing alternatives**

Each participant proposes one or more alternatives by giving the attribute values for each of them; the proposed alternatives enter the event table.

**Filtering alternatives**

The coordinator might filter the set of proposed alternatives based on certain criteria. For example, it might be decided that only the alternatives with cost less than 500 should be considered but all alternatives involving Athens or Rome must be considered - even if their cost exceeds 1500.

**Setting priorities**

In some contexts, the coordinator might want to assign a weight or priority to each participant (either with the participant knowing about it or not knowing). For example, in organizing a meeting, the presence of some of the participants might be considered as indispensable for the meeting to take place whereas the presence of other participants might not be necessary (e.g. if they participate only as observers).

**Voting (i.e. expressing preferences)**

Once the creation and filtering of alternatives is over, each participant expresses preferences over the retained alternatives, in one of two ways:

- directly, by first inspecting the attribute values of each alternative then expressing preferences directly on the alternatives (e.g. P.Alt-id: 3 \(\rightarrow\) 2, 5 \(\rightarrow\) 2, 5 \(\rightarrow\) 7);
- or indirectly, by expressing preferences over the values of one or more attributes, together (eventually) with priorities between the attributes (e.g. P.City: Rome \(\rightarrow\) Vienna, Athens \(\rightarrow\) Madrid, P.Transp: Train \(\rightarrow\) Coach, Train \(\rightarrow\) Airplane)

**Ranking the alternatives**

Whatever the way of expressing preferences, the ranking procedure described in the following section is applied to derive one ranking of the set of alternatives per participant (ranking of the alternatives means associating a positive integer rank with each alternative).

### Result (of voting)

Following the previous steps, and assuming we have m alternatives and n voters, we will have an mXn matrix with one row of n rankings for each alternative; it then remains to apply the ranking procedures presented in the following section in order to aggregate the n rankings and produce the final ranking of the alternatives.

The overall goal of the paper is to develop a system capable of supporting the coordinator in carrying out all the above tasks and ensuring good communication with the participants.

<table>
<thead>
<tr>
<th>Alt-id</th>
<th>City</th>
<th>Transp.</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Athens</td>
<td>Airplane</td>
<td>1900</td>
</tr>
<tr>
<td>2</td>
<td>London</td>
<td>Train</td>
<td>900</td>
</tr>
<tr>
<td>3</td>
<td>Vienna</td>
<td>Airplane</td>
<td>1600</td>
</tr>
<tr>
<td>4</td>
<td>Rome</td>
<td>Airplane</td>
<td>1450</td>
</tr>
<tr>
<td>5</td>
<td>Rome</td>
<td>Train</td>
<td>1050</td>
</tr>
<tr>
<td>6</td>
<td>Berlin</td>
<td>Airplane</td>
<td>1550</td>
</tr>
<tr>
<td>7</td>
<td>Madrid</td>
<td>Train</td>
<td>950</td>
</tr>
<tr>
<td>8</td>
<td>Amsterdam</td>
<td>Train</td>
<td>700</td>
</tr>
<tr>
<td>9</td>
<td>Amsterdam</td>
<td>Coach</td>
<td>550</td>
</tr>
</tbody>
</table>

### 3. THE RANKING PROCEDURES

We have seen that, during voting, each participant expresses a preference relation either on the column Alt-id of the event table (this is the direct way) or over one or more attributes of the alternatives (this is the indirect way). The basic ranking procedure takes as input a preference relation over an attribute (any attribute, including Alt-id) and produces as output a ranking of the alternatives.

We shall assume that the event table is implemented as a relational table (i.e. a table residing in a relational database) and that the set of alternatives to be considered might be a subset of those in the event table E. For example it might be decided that only alternatives containing Athens as a destination, or having a cost less than 1500 will be considered. This can be expressed as follows: (City=\text{Athens}) \lor (\text{Cost}\leq 1500). We call such an expression a query over E; and a query over E together with a preference relation over some attribute of E is what we call a preference query.

**Definition 4 – Preference Query**

A preference query over E is a pair <Q, P.A> such that:
1/ Q is a query over E, that is a Boolean combination of elementary conditions of the form A=v, A≠v or A≤v where A is an attribute and v is a value of A
2/ P.A is a preference relation over attribute A

The answer to a preference query, denoted ans(<Q, P.A>), is defined to be a sequence of blocks of tuples, E0, E1, .., Em, Em+1 such that:

1/ E0, E1, ..., Em+1 form a partition of ans(Q) (i.e. they are mutually disjoint and their union is ans(Q))
2/ Each Ei contains tuples that are incomparable with respect to P.A
3/ E0 contains the “best” tuples, that is for every tuple t in E0 there is no tuple s in E0 such that s precedes t with respect to P.A (and this is true for i=1, ..., m). Also note that the block Em+1 in the above definition contains all tuples in the answer of Q that cannot be compared to other tuples.
4/ for each i= 1, 2, .., m, that the answers produce the sequence of blocks E0, E 1, E 2, Em, Em+1 of the answer directly we shall show how we can rewrite the preference query <Q, P.A> into a sequence Q0, Q1, .., Qm, Qm+1 of ordinary queries such that the answer of Q0 is E0, the answer of Q1 is E1, and so on. In doing so, for notational convenience, we shall drop the attribute names whenever they are easy to understand from context. For example, we shall write:

Q= Train ∨ Airplane
instead of
Q= (Transp. = Train) ∨ (Transp. = Airplane)

Following this convention, let us illustrate Definition 4 and the rewriting of <Q, P.A> using a very simple example. Consider the preference query <Q, P.City> defined as follows:

Q= Train ∨ Airplane
P.City: Rome→Vienna, Rome→Madrid, Athens→Madrid

First, let us observe that the answer to Q, denoted ans(Q), contains all tuples that are of interest; it is precisely this set that we need to partition into a sequence of subsets E0, E1, .., Em, Em+1, that will constitute the answer to <Q, P.City> (see point 1/ in the above definition). To compute E0, we look at the preference relation P.City and we observe that (a) of the four cities appearing in P.City, Rome and Athens are the only two cities that are not preceded by any other city and (b) none of them precedes the other (see point 2/ of the definition). Therefore, all tuples of ans(Q) that contain either Rome or Athens are the “best” with respect to the preferences in P.City (see point 3/ of the definition).

More formally, this is expressed as follows:

E0= ans(Q)\[ans(Rome)∪ans(Athens)]

Now, if we call Q0 the query whose answer is E0, then Q0 can be expressed in terms of Q, Rome and Athens as follows:

Q0= Q∧¬(Rome∧Athens)

A similar reasoning shows that the ordinary query whose answer is E1 is defined as follows:

Q1= Q∧¬(Vienna∧Madrid)

There are no more cities to consider in the preference relation P.City, therefore m=1. Next we compute the last block Em+1 (see point 4/ of the definition), that is E2 in this case. We have:

E2= ans(Q)\(E0∪E1)

Note that E2 contains all tuples that are not possible to compare with other tuples, with respect to P.City.

Now, if we call Q2 the ordinary query whose answer is E2, then we can express it in terms of Q, Q0 and Q1 as follows:

Q2= Q ∧¬(Q0∧Q1)

As a result, the sequence of ordinary queries Q0, Q1, Q2 is such that the answers produce the sequence of blocks E0, E1, E2 answering the preference query of our example. Clearly, if the preference relation P.City is more complex, then we need an algorithm in order to produce the sequence of queries that answers the preference query. In the remaining of this section we present such an algorithm. Our algorithm takes as input an abstract binary graph, which is assumed to be acyclic; and it returns as output an ordered partition of the set of nodes.

More precisely, let G be an acyclic binary graph, and define the rank of a node t as follows:

if t is a root of G then rank(t)= 0
else rank(t)= the length of a maximal path among all paths from a root of G to t

Next, let us denote by Bi the set of nodes with rank i, and let m be the maximal path length among all paths starting from a root. Then it is rather easy to see that the sequence B0, B1, .., Bm has the following properties:

1/ B0, B1, .., Bm form a partition of the set of nodes of G
2/ for each i= 0, 1, 2, ..., m, there is no arc of G connecting two nodes of Bi
3/ for each i= 0, 1, 2, ..., m, and each node in Bi there is an antecedent s of t in Bi (i.e. there is a node s and arc s→t in G)

In our previous example, we have:

B0= {Rome, Athens}, B1= {Vienna, Madrid}

To find the sets B0, B1, .., Bm in a systematic way, we use the following algorithm (which is a variant of the well known topological sorting algorithm):

Algorithm Ordered-partition(G)

Input: An acyclic graph G
Output: A sequence B0, B1, .., Bm of sets of nodes

Aux: G; i:= 0 ;
The complexity of this algorithm is linear in $n+a$, where $n$ is the number of nodes and $a$ is the number of arcs of $G$. Now, in a preference query $<Q, P.A>$, the preference relation $P.A$ can be represented as a binary graph that we shall denote by $G(P.A)$. With this observation at hand, the following algorithm produces the sequence of queries $Q0, Q1, ..., Qm, Qm+1$ that answers the preference query $<Q, P.A>$:

**Algorithm Evaluate-Pref-query**

*Input*: A preference query $<Q, P.A>$ such that the graph $G(P.A)$ is acyclic.

*Output*: The sequence $Q0, Q1, ..., Qm$ answering the preference query

*Method*:

1/ Ordered-partition($G(P.A)$)

   {the output is a sequence $B_0, B_1, ..., B_m$ of sets of values of A }

2/ For each $i=0, 1, ..., m$ do

   begin
   
   $Q'i :=$ conjunction of all values of $A$ in $B_i$
   
   $Qi := Q \land Q'i$
   
   output $Qi$

   end

3/ $Qm+1 := Q \land \neg(Q0 \lor Q1 \lor ... \lor Qm)$;

output $Qm+1$

So far, we have considered that the preference relation $P.A$ is expressed over a single attribute. Clearly, if the preference relation is expressed over two or more attributes, then the way of defining the answer to the preference query remains the same. In other words, if instead of having $P.A$ we now have $P\{A, B\}$ then the only difference is that each node of the graph $G(P\{A, B\})$ is a conjunction of two values, a value of $A$ and a value of $B$ (instead of being just a single value). For example, consider a preference query $<Q, P\{City, Transp.\}>$, where

$P\{City, Transp.\}: \{\text{Athens}\lor\text{Airplane} \rightarrow \text{Rome}\lor\text{Train}, \text{London}\lor\text{Train} \rightarrow \text{Rome}\lor\text{Train}\}$

Then the answer to the preference query is the following sequence:

$Q0 = Q \land (\text{Athens}\lor\text{Airplane})\lor(\text{London}\lor\text{Train})$

$Q1 = Q \land (\text{Rome}\lor\text{Train})$

$Q2 = Q \land \neg (Q0 \lor Q1)$

However, things become more complex when the preference query has the form $<Q, \{P.A1, ..., P.Ak\}>$. This is the general case where the participants declare preference relations over different attributes, independently from each other, and we want to aggregate all expressed preferences in order to derive a ranking of the alternatives. Clearly, in this case we can proceed as if we had just one preference relation, provided that we can derive one preference relation $P\{A1, ..., Ak\}$ from the given preference relations $P.A1, ..., P.Ak$. However, in order to do this, we need to know whether the preference relations $P.A1, ..., P.Ak$ all carry the same weight or there are priorities among them; a priority is a linear ordering over $\{P.A1, ..., P.Ak\}$. For the purposes of this paper, we assume that it is the coordinator that decides whether there is a priority over the attributes, and if so then the coordinator declares this priority and broadcasts it to all participants. The following definitions state how the preference relation $P\{A1, ..., Ak\}$ is derived from the given preference relations $P.A1, ..., P.Ak$ in each case. In these definitions, “Pa” stands for “Pareto”, meaning that there is no priority over the attributes; and “Pr” stands for “Prioritized” meaning that there is a priority over the attributes.

**Pareto preference relation** (i.e. there is no priority over the attributes):

For all tuples $s$ and $t$ in $E$, $s \rightarrow_{Pa} t$ if and only if $s.A1...Ak \neq t.A1...Ak$ and either $s.A1 = t.A1$ or $s.Ai \rightarrow t.Ai$, $i=1,...,k$

We treat the relation $Pa$ up to tuple equivalence, where equivalence is defined as follows: $s \equiv_{Pa} t$ if and only if $s.A1..Ak = t.A1..Ak$

**Prioritized preference relation** (i.e. a priority has been declared over the attributes):

Let the preference relations $P.A1, ..., P.Ak$ be prioritized as follows: $P.A1 \rightarrow P.A2 \rightarrow ... \rightarrow P.Ak$ where we use the arrow $\rightarrow$ to also show priority. Under this assumption, we have the following definition:

For all tuples $s$ and $t$ in $E$, $s \rightarrow_{Pr} t$ if and only if $s.A1...Ak \neq t.A1...Ak$ and either $s.A1 \rightarrow t.A1$ or $(s.A1 = t.A1$ and $s.A2...Ak \rightarrow Pr t.A2...Ak)$

We treat the relation $Pr$ up to tuple equivalence, where equivalence is defined as follows: $s \equiv_{Pr} t$ if and only if $s.A1..Ak = t.A1..Ak$

We note that the well known lexicographic ordering is a special case of prioritized relation $Pr$ as defined above. Indeed, the lexicographic ordering is a prioritized ordering with the additional assumption that the domain of each of the attributes $A1, ..., Ak$ is totally ordered (i.e. given any two values $v$ and $v'$ of attribute $Ai$, either $v \rightarrow v'$ or $v' \rightarrow v$)

**4.1 OUTLINE OF THE SYSTEM**

**4.1 System functionality**

Obviously, the discussion in the previous paragraphs should not reach the end users. People that need to arrange various kinds of events and express their preferences on possible actions related to
them should be able to access a (preferably) web based system that provide the necessary functionalities. These functionalities should include:

- The ability to create an event and assign to it specific attributes (columns, \( C_i \)) that would be subsequently used to decide on the “most-wanted” solution
- The ability of the creator to assign (closed participation) or not (open participation) participants / voters / users to the event
- Users should have already been registered to the system if they want to participate to closed participation events
- One user who can be designated as coordinator (i.e. someone who would have the ability to perform special functions like starting and ending the voting). Events can have one or none coordinator. One usual case would be that the event creator will also act as coordinator.
- The ability of the users to express their preferences either in a total order (“I prefer A over B over C, etc.”) or in a partial order (“I prefer A over B, C over B, etc.”). This can be seen as adding preferences as rows \( R_i \) in a table.
- The ability of the users to express preferences over more than one attribute as described earlier in the text
- The ability of the users to cast preferences for more than one event concurrently
- The ability for a user to decide when to start and when to stop voting for an event (either manually by explicitly stating so or automatically by setting beforehand start and/or end times). In any case the system should have the ability to automatically “calculate” the most preferred solution and automatically notify the participants for it (e.g. via e-mail, RSS, posting to a web site, etc.)

### 4.2 System Design

The main actors involved in the system are the participants that can be registered or not with the system, the events that can have usually more than one attribute, and the choices over each attribute that the participants can choose from. So the system is to be designed around these, taking into account their interactions and providing an intuitive and easy to use UI (User Interface).

The system is an online web based system where users can connect and vote (either by logging in or not). This means that the system should be a multiuser system and be able to handle multiple votes on the same event simultaneously and voting on multiple events simultaneously. Moreover for having a more clear development and maintenance process for such a system we propose to follow the MVC (Model-View-Controller) [2] design pattern, that allows the separation of concerns among the data model, the system actions and the user interface.

Thus we suggest that we model as Java Beans [11] the basic actors of the system that are the users and the events. The User Java Bean will carry both logistic information about the user (if he is a registered user) like username, password, etc and information on the votes that he has initiated, coordinated or participated in. The Event Java Bean will also carry logistics information about the event that it models like the title, the initiator, the coordinator, start and end dates, etc. It will also carry additional Java Beans that will model the attributes of the event and the different choices for every attribute. It will also allow for different combinations of the different choices as was described earlier. All objects involved in the system will carry persistence capabilities and will be responsible to persist the information they carry as they see fit during the system operation. A sketch of the internal system architecture is depicted in Figure 1.

![Figure 1. The internal system architecture](image)

From an information flow point of view the situation is rather simple. The user will fill up her choices in a form that will be subsequently submitted to the system. The choices can be either over one or more attributes, and can be either in total or partial order. The system will make every possible effort to hide from the user the choice complexities and will pre-calculate the possible choices that she has to choose from. Then the user will submit the form with her choices to the system and after the process is completed (either because the end time has passed or because the coordinator has decided so) the system will use this information to compute the outcome of the voting by running the algorithm(s) described in the previous section of this paper. On the other hand the user should be able to setup an event at any time. By specifying an event one should be able to add information about it, add its attributes and finally add the different choices for each attribute and whether they can be chosen one by one or in combination.

### 4.3 System UI

The system should provide a simple and intuitive interface to the end users. As described so far they only need to know their choices (and their possible allowable combinations) and have a simple way to cast their votes. This means that the users will be presented by a set of choices and choose those that suit them. Users will be able to view whatever the rest of the participants have voted (giving the system openness) and will also be allowed to post comments about the event.
The UI for the initiator / coordinator will be equally simple: apart from specifying the event, its attributes and the different choices for every attribute, the coordinator would be allowed to specify the participants (if this is an event with “restricted” participation). The coordinator would also have the capability to explicitly state the allowed (or none or all) combinations among the attributes; thus allowing the system to pre-compute them and present them ready to the user. Referring to the example presented earlier, the choice of going to Athens could be coupled only with the Plane and the Coach options and not the Train, so we need to be able to make that available to the event coordinator. It should be noted here that the system would allow to handle events as completely unmanaged where all the possible participants would be able to act as coordinators and alter (by adding or deleting) the attributes and the choices of the event. Moreover in case of a “restricted” participation event, the coordinator would be able to favor one or more participant’s vote over the rest, for example in cases where someone’s participation is deemed necessary. The last two capabilities will be added in a second phase of the development. Finally the coordinator will be able to evoke the vote / preference calculation algorithm and decide on the number of reported preferences (the system would be able to report an ordered set of the most preferred (combination of) choices) by stopping the voting process at the same time. Subsequently the system will be able to inform each participant of the event of the final outcome of the voting process.

5. CONCLUDING REMARKS

It is deemed important in our collaborative communications and interactions to be able to express our opinion and make decisions in a way that allows us to exploit the different choices presented to us. In this paper we have presented both algorithms and tools that give the ability to end users to express their preference over any event. Moreover we allow the users to specify the alternative choices and even combine them as the decisions they usually face are more complex. The system, which is currently in implementation phase, will allow users to use the algorithms presented here without requiring any prior knowledge and will compute the “best” possible alternative.

In the near future we plan to move the research presented in this paper to a real world implementation that would allow users to actually exploit the system. We can log users’ actions (not their actual preferences in order to avoid conflicts with personal data issues) and then draw conclusions on how the users actually use the system, what are the features they exploit most, and to check if our intuition is validated by the facts. Our plan includes offering the system free for the community to use so as to allow for its widespread adoption. In such a case the system should also scale seamlessly since we need to keep the same level of service even when the number of users grows.

We also plan to expand the capabilities of the system to allow for tie-breaking decisions, if such situations occur. And here we do not mean only the system part where, in the absence of a coordinator the system needs to decide which choice should be chosen in case of a tie; but also situations where cyclic preference relations (cases when two or more choices create a cycle while one is preferable over the other in a cyclic manner but none is preferred overall) occur due the nature of a specific event and the proposed algorithms need to be extended to support that.

To the best of our knowledge there is no other work presented so far that combines to such extent the algorithmic support and the system implementation and allows users to specify choices in such a flexible and non-restricting way.

6. REFERENCES


