Interface Tailoring by Exploiting Temporality of Attributes for Small Screens

M. Kumara Swamy, P. Krishna Reddy,
R. Uday Kiran, and M. Venugopal Reddy

Center for Data Engineering
International Institute of Information
Technology-Hyderabad (IIIT-H), Hyderabad,
Andhra Pradesh, India - 500 032
pkreddy@iiit.ac.in

Abstract. In the pervasive computing era, mobile phones and personal digital assistants are widely used for data collection. The traditional user interfaces which are employed for data collection in personal computer environments are to be modified appropriately for the mobile environment. Because, it is difficult to display full interface on a single mobile screen due to the limitation of the mobile phone screen size. Interface tailoring methods are investigated in the literature for designing user interface for mobile screens. In the literature, temporality-based approach is proposed for designing efficient user interface for personal computer environment. In this paper, we extend the notion of attribute temporality to interface tailoring methods and propose an improved user interface design approach for small screens. The analysis on the real-world datasets shows that the proposed approach can be used for better user interface design for small screens.

Keywords: user interfaces, context-based approach, interface tailoring, mobile interface forms.

1 Introduction

Mobile phones and personal digital assistants are used for data collection in modern information systems. It is true that mobile phones and personal digital assistants provide significant advantages due to pervasive nature of communication and computing abilities. However, due to small screen size and other factors, the design of UI forms for mobile devices is a challenging task as the approaches used to design UIs for personal computer (PC)-based systems can not be extended for mobile devices in a straightforward manner [5]. So, appropriate methods are to be investigated to device efficient UIs for mobile environment. Normally, the large UI form in PC environment is divided into multiple small UI forms by considering small screen size in the mobile environment. As a result, user has to go through multiple screens which increases navigational burden. In addition, it also increases both cost of data-entry and maintenance. So, developing efficient UI methods for mobile environment is an ongoing research problem.

In the literature, interface tailoring approaches are investigated to develop UI forms for mobile devices [1,4]. In these approaches, the PC-based UI is divided into multiple screens (or high-level components) in such a way that the UI can accommodate in a mobile screen. The research issue is how to divide a large UI form into small UI forms (or screens) to reduce the navigational burden and improve the efficiency to the extent possible.

For PC-based systems, context-based approach (CA) [11] is followed for designing the UI forms. Recently, context-based approach with attribute temporality (CAAT) [7] has been proposed to reduce the navigational burden in UI forms for PC-based systems by exploiting the notion of "attribute temporality". The attribute temporality value indicates the active period of the attribute during which the attribute receives values. The attributes of a given context are clustered into several active-contexts based on the temporality of attribute and UIs are designed for each active-context. It was found out that the CAAT approach improves the UI performance by reducing the navigational burden significantly.

It was observed that there is a scope to design efficient UI by reducing the size of UI for small screens by extending the notion of attribute temporality to mobile environments. In this paper, we made an effort to propose an improved UI approach for small screens by extending the notion of attribute temporality to interface tailoring approaches. We have carried out analysis by considering real world dataset. The analysis results show that the proposed approach has a potential to improve the performance for mobile environments.

The rest of the paper is organized as follows. In the next section, we discuss the related work. In section 3, we present the overview of the existing approaches. In section 4, we explain the proposed approach. The analysis results and discussion are provided in Section 5. The last section consists of summary and conclusions.

2 Related Work

In this section, we discuss the related work in the area of UI design tools and approaches in both mobile and PC-based environments.

2.1 UI Design Tools for Mobile-Based Systems

Multi-User Publishing Environment (MUPE) [2] is an open source platform developed to enable multi-users to publish and communicate with each other. MUPE allows a user to generate a mobile UI form interactively and send this form by expecting the data from his/her contact list. This application will also generate an integrated view of the data received by the mobile UI form.

A template-based approach to generate UI for mobile phone is presented in [3]. In this work, a UI design templates generator is proposed for UI designers to easily and quickly create the UI templates for mobile phone. The developed UI templates can be fine tuned with a visual UI authoring tool to generate the UI prototype for the target mobile phone system.

2.2 UI Design Approaches for Mobile-Based Systems

The problem of division of large interface into numerous small screens is identified in [1,4]. User interface tailoring [4] is a solution to such problem, which tailors information based on the mobile devices screen. In this approach, the information is tailored to display only the important information on the interface. This approach does not guarantee the integrity of information [1]. Another approach is dividing large PC interface into smaller ones such that it can match the size of the mobile phone screen [1]. In this method, the large interface is divided into small screens. The issue is to investigate improved approaches to reduce the navigational burden as far as possible.

2.3 UI Design Tools in PC-Based Systems

Brad Myers [15] surveyed the state of the art of UI software tools. The existing UI design tools are classified into categories such as language-based tools, interactive graphical specification tools and model-based generation tools. All these models are based on the dynamic behavior of a UI as well as the way as to how the designer can specify the layout. There exist tools which allow to design UI interactively, or which automatically generate the UI from a high-level model or specification. These tools also enable the UI developer to customize their design-output.

User Interface Management Systems (UIMSs) consist of UI construction tools which are used mainly to construct the dialogue component of the interface. The UIMS significantly reduces the time required to develop an interface and also the chances of errors to a minimum because the designer is dealing with a compact and higher level of specification. A high-level UIMS which automatically generates the lexical and syntactic design of GUIs is presented in [19] based on the given description and user preferences. Similar approaches have been used in the Menu Interaction Kontrol Environment (MIKE) [16] and User Interface Design Environment (UIDE) [10]. MIKE automatically generates textual interfaces from a description of the semantic commands supported by the application. The designer then adds the graphical interaction facilities to the interfaces generated by MIKE. UIDE provides a high-level conceptual design tool in which the designer describes the UI as a knowledge base. UIDE can algorithmically transform the knowledge base into a number of functionally equivalent interfaces, each of which is slightly different from the original interface. The transformed interface definition can then be input to a UIMS which implements the UI.

Egbert Schlungaum [17,18] summarizes the area of model-based UI software tools in order to prepare a basis for automatic UI generation. Efforts are made to create a common declarative model to specify the necessary information for automatic UI generation by combining various model-based UI approaches. A system called TADEUS was developed that takes requirement analysis as the input and generates UIs.

2.4 UI Design Approaches in PC-Based Systems

A survey was conducted on context-aware system in [12]. This survey presented common architecture principles of context-aware systems and derived a layered conceptual design framework to explain the different elements common to most context-aware architectures. These design principles are used to introduce various existing context-aware systems focusing on context-aware middleware and frameworks, which ease the development of context-aware applications.

A survey was conducted in [11] to understand context and context-aware applications. The concepts such as context and context-aware computing are defined. These definitions assist to understand the boundaries of context-aware computing, and facilitate application designers for selecting the context to use, structuring the context in applications, and in deciding what context-aware features to implement.

For improved UI design, the notion of attribute temporality was exploited in [7]. Given a context and temporal values, this approach divides context into several active-contexts. UIs are designed for all active-contexts.

About the proposed approach: In the mobile environment, interface tailoring approaches are investigated to tailor the PC interface to fit in small mobile screens. However, it was observed that the notion of attribute temporality can improve the performance by further reducing the navigational burden. In this paper, we have made an effort to improve the performance of interface tailoring methods to design UI forms for small screens by extending the notion of attribute temporality.

3 Overview of Context-Based and Interface Tailoring Approaches

In information systems, UI is a place where the user interacts with the database systems to populate the data. In this section, we briefly discuss the context-based approach with attribute temporality and interface tailoring approaches.

3.1 Context-Based Approach with Attribute Temporality (CAAT)

Normally, context-based approach (CA) [11] is followed for designing UIs in PC-based systems. In context-based approach, the UI forms are designed for each context. The context refers to a particular task or activity of an information system.

In information systems, it can be observed that some attributes receive data for certain time period and they do not receive values during the remaining time period. This notion is called attribute temporality. By extending the notion of attribute temporality on CA, it is possible to design better UIs. In this approach, the notion of active-context is defined which is set of attributes that receives values for a fixed time period. In the context-based approach with attribute temporality (CAAT) [7], the attributes of a given context are divided into several small active-contexts and UIs are designed for each active-context.

The CAAT approach divides the context into several active-contexts. Based on the timing of the data-entry, appropriate active-context is displayed to the user. In CA, all the attributes are shown to the user for the data-entry. Whereas, in CAAT, only the attributes of corresponding active-contexts are shown. Since the number of attributes in active-context is small as compared to the number of attributes in entire context, the performance is improved.

Table 1. Sample data consisting of attribute names and the corresponding durations

Name of the attribute	Duration of the attribute (days)
a	$1-5$
b	$2-14$
c	$5-12$
d	$12-15$
e	$1-15$
f	$9-14$
g	$1-15$
h	$10-15$
i	$2-8$
j	$8-10$

The method to divide the given context into active-contexts is as follows. The algorithm is given in [7]. The Jaccard coefficient [20] similarity criteria is employed to find the similarity between the two active-contexts. Initially, the attributes of a given context are clustered into day-wise active-contexts (the attributes valid on a day) at 'Level-0'. The merging process starts at 'Level-0' with the first two contiguous active-contexts. When the similarity value of these two contiguous active-contexts is greater than or equals to the given similarity threshold, they are merged into a single active-context. The newly generated active-context is again merged with the next contiguous active-context if the similarity value exceeds the threshold value. Otherwise, the same process is repeated from the next consecutive active-context. All these newly generated active-contexts form the active-contexts at 'Level-1'. The process is repeated for the active-context in 'Level-1' to form higher level active-context. In this way, finally, all the active-contexts are merged into a single largest active-context at 'Level-n'. The appropriate active-contexts are selected for designing UI.

We explain the process of finding the active-contexts through an example.

Example 1: Consider the sample data in Table 1. In this table, column one contains the name of the attributes and the other column contains active duration of the corresponding attributes in days. In this column, the starting day and ending day, during which the attribute receives the values, are shown. The

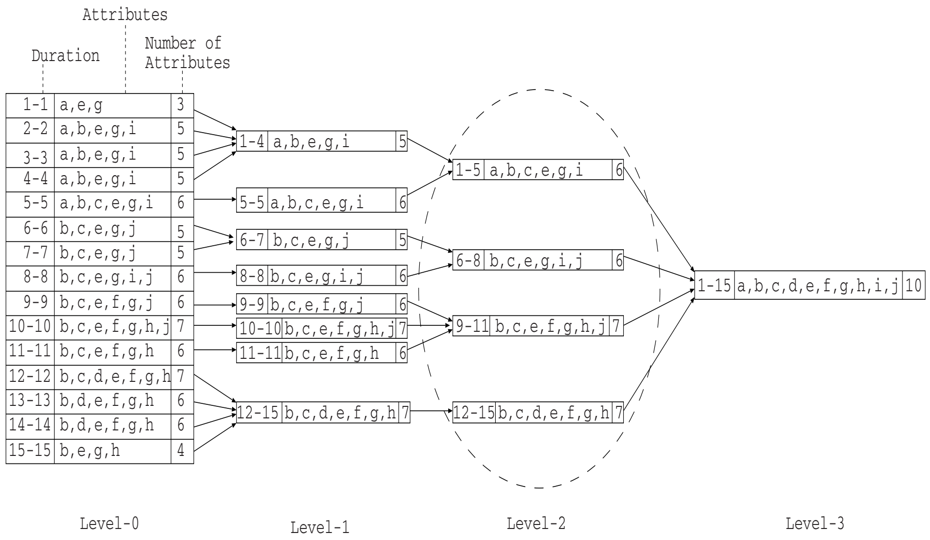

Fig. 1. Clustering processes to find active-contexts in CAAT approach

period which the attribute receives the values is called duration. The maximum duration for this sample data is 15 days. The attributes are clustered into different active-contexts. The hierarchy of derived active-contexts is shown in Figure 1. In this figure, each node (active-context) is consisting of three values. The first value shows the duration, the second value shows the attributes and the final value indicates the number of attributes for the corresponding duration. The 'Level-0' shows the initial active-contexts, 'Level-1', 'Level-2' and 'Level-3' are the active-contexts at different levels. The final hierarchy is generated based on the above procedure. Initially, day-wise active-contexts are generated, the attributes active in a day are grouped together based on the durations. As a result, 15 active-contexts are generated. These day-wise active-contexts are merged to higher level active-contexts based on the similarity between the active-contexts. Let us use Jaccard coefficient as the similarity metric to merge the two active contexts. Let the similarity threshold value be 0.9. The merging process is repeated to merge all the active-contexts into a single large active-context. In Figure 1, 'Level-0' contains day-wise active-contexts and 'Level-1' to 'Level-3' contain the corresponding active-contexts. At 'Level-3', all the attributes are merged into a single large active-context. The final active-contexts are extracted from 'Level-2', where the temporality of each active-context is distinct and average number of attributes in each active-context is minimum. The dotted circle in the figure shows the layer of active-context. As a result, four active-contexts are extracted for the durations 1-5, 6-8, 9-11 and 12-15. For these active-contexts the UI forms are designed.

3.2 Interface Tailoring Approach

Interface tailoring refers to the ability to customize and optimize an interface according to the context in which it is used [6]. Tailoring of the interface design includes the capability of adaptation of content delivery to various devices, which preserve consistency and usability of the service [4]. One of the interface tailoring approaches namely, a constrained-based UI method [1] is proposed to tailor the PC-based UI such that it matches the mobile screen using component-tree.

In this approach, the attributes are divided into several components by considering the constraints of mobile screen. Suppose, we take number of attributes to be displayed in the screen as a constraint. Based on this constraint, a component tree is built starting from high-level components to attribute levels. The attributes in the component-tree can be displayed in mobile UI using either depth-first principle or breadth-first principle.

In summary, using interface tailoring approaches, the attributes of given UI form for PC-based systems are divided into multiple high-level components such that each component or sub-component can fit into the mobile screen.

4 Proposed Interface Tailoring with Attribute Temporality (ITAT) Approach

The notion of "attribute temporality" can be exploited to interface tailoring approach to improve the performance. So, given the context, we apply attribute temporality and select appropriate active-contexts which match the screen size of mobile phone. If number of attributes are more in an active-context such that it can not fit into a mobile phone, interface tailoring approach is followed to divide the active-context into small screens.

We call the proposed approach as "Interface Tailoring with Attribute Temporality (ITAT)". It contains two phases. In the first phase, we identify the active-contexts from the given context and in the second phase, we apply interface tailoring approach for such active-contexts whose size is more than the user-specified mobile screen size.

– **First phase of ITAT:** To find the active-contexts, we modify the CAAT algorithm [7] by including the notion of active-context tree (AC_tree) which contains all the extracted active-contexts. The algorithm takes the context 'C' as the input and generates AC_Tree as the output. Initially, smaller day-wise ACs are generated. These day-wise ACs are merged into larger ACs at various levels and finally they form a single largest AC. All the extracted ACs, starting from day-wise to the single large AC, are inserted into AC_tree. Next, we find set of ACs which fit in a mobile screen by traversing the AC_tree. The tree traversal starts from the large AC. If the number of attributes in a node either equals or less than the number of attributes that can fit in the mobile screen, the node is selected as an AC for mobile screen and all the other child nodes are not processed further. If the number of attributes in a node exceeds mobile screen size, the lower level nodes are further traversed. This process continues until the entire *AC_tree* is covered.

– **Second phase of ITAT:** In the phase, we find the large *ACs* from a set of the selected ACs for applying the interface tailoring approach. The algorithm reads ACs from the selected ACs sequentially starting from the beginning. If the number of attributes are less than or equal to the mobile screen size then normal UI is generated. Otherwise interface tailoring approach is followed to generate UI. We take the mobile screen size as a constraint, that is number of attributes to be displayed on the mobile screen as a constraint. Based on this constraint, an attribute-tree is built starting from high-level components to attribute levels. The attributes in the attribute-tree can be displayed in mobile UI using breadth-first principle.

The proposed approach is explained with the following example.

Example 2: Continuing with the Example 1, let the mobile screen size, (MSS) be 6. It means a mobile screen can accommodate 6 attributes for taking input. We explain the method to extract active-contexts, later we explain the interface tailoring approach.

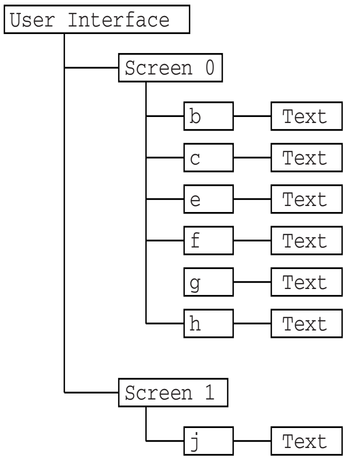

Fig. 2. A sample attribute-tree for active-context at duration "10-10"

An *AC_Tree* is built as shown in Figure 1. However, the active-context lies between 'Level-0' to 'Level-3' that is from day-wise active-context to final single active-context. The tree traversal starts from 'Level-3' to 'Level-0'. The node at 'Level-3' consists of 10 attributes. This node cannot fit in the mobile screen. The traversal goes to the first node in 'Level-2', start reading node by node at this level. The nodes having the durations 1-5 and 6-8 are selected as the *AC* by discarding the child nodes of the selected nodes, that is, child nodes at 'Level-0' and 'Level-1'. The next two nodes are having 7 attributes which cannot fit in the mobile size. Now, the control goes to the corresponding child nodes at the lower level that is at 'Level-1'. Here nodes having durations 9-9 and 11-11 are selected as ACs. At the end, the control goes to 'Level-1' to the final leaf nodes, find the rest of the nodes having durations 10-10, 12-12, 13-13, 14-14 and 15-15, and are selected as ACs as these are the final leaf nodes. After completion of this phase, we have the durations 1-5, 6-8, 9-9, 10-10, 11-11, 12-12, 13-13, 14-14 and 15-15 as the active-contexts.

We apply the second phase on the selected active-contexts. Among the selected active-contexts, the active-contexts at the durations 10-10 and 12-12 cannot fit in the mobile screen as they have 7 attributes in each of the ACs. For these two ACs interface tailoring approach is followed and for the other ACs normal approach is followed to generate UI. The attribute-tree is built with the attributes in active-contexts for duration 10-10 and 12-12 nodes. The sample attribute is shown in the Figure 2 for active-context at duration 10-10. The attributes in this figure are divided into two 'Screens', 'Screen 0' consisting of 6 attributes and the 'Screen 1' has one attribute.

5 Performance Analysis and Discussion

5.1 Performance Analysis

We have conducted performance analysis by considering the UI form data for eSagu [9,13] system. In eSagu system, the farmer communicates about the crop problem by capturing photographs and filling in the corresponding crop observation (UI) form and sends it to agricultural experts through eSagu portal. Agricultural experts deliver the expert advice based on the crop photographs and data received through the crop observation form. Each crop observation form consists of about 120 attributes on an average. All these attributes are spread over to the entire crop season that starts from sowing stage to harvesting stage. Individual UIs are created by considering each crop as a context.

We have taken contexts of five crops and applied IT approach and calculated the number of mobile screens, by considering mobile screen size as 5 attributes. We applied the proposed approach and calculated the number of mobile screens required. It was found out that the proposed approach reduced the number of mobile screens required as compared to IT approach. The results of the analysis for the five crops are shown in Tables 2. In this table, the first column indicates serial number, the second column shows the crop name (context), and the third column shows the number of mobile screens in IT and the final column shows the number of mobile screens in ITAT approach. It can be observed that, there is a reduction of 7, 6, 7, 13, and 12 screens are reduced for cotton, chilli, rice, maize and sunflower data-entry forms respectively.

We now discuss how reduction in number of mobile screens are obtained in case of UI form for cotton crop. The cotton crop has 106 attributes with 180 days cycle. The temporality values of the attributes are collected with the help

Table 2. Performance of CA, IT and ITAT CA approaches in eSagu^{TM} system

S.No.	Crop Name	Total Attributes	No.of Screens in IT	No.of Screens in ITAT
1	Cotton	106	22	15
2	Chilli	108	22	16
3	Rice	111	23	17
4	Maize	105	21	14
5	Sun Flower	106	22	14

of domain experts. In IT approach, all 106 attributes are used to generate mobile UI. We assumed the screen size as 5, that means the mobile screen can accommodate 5 attributes on one screen. As a result, these attributes are divided into 22 screens. In ITAT approach, 106 attributes are divided into 70 active-contexts where each active-context contains about 75 attributes. So, for 75 attributes, 15 mobile screens are required. Based on the time of the data-entry, appropriate active-context is displayed to the user. As a result, at a given point of time only 75 attributes are required with 15 screens in ITAT. Whereas, in IT approach, 106 attributes are shown all the days with 22 screens. With ITAT approach, there is a saving of 7 screens for any given time period, that means there is a saving of 35 attributes for each active-context.

Similar kind of analysis has been carried out for UI forms of other four crops. The results are shown in the Table 2.

5.2 Discussion

The proposed approach provides opportunity to improve the performance of UI for mobile screens. However, the performance improvement depends upon several factors which are discussed as follows.

1. **Performance improvement is application dependent:** The proposed approach depends on the notion of temporality. But UI attributes of only certain class of applications exhibit attribute temporality. Also, the proposed approach performs better if more number of attributes in UI form exhibit temporality property.
   In our analysis on real world dataset, it was observed that there are about 30 attributes which could not exhibit temporality property and appeared in all active-contexts. So, the performance could be higher, if more number of attributes exhibit temporality property. Still, the results with the proposed approach are encouraging.
2. **Requires extra effort to extract temporality values:** The proposed approach requires extra effort in identification of attribute temporality in a proper manner. Temporality values can be extracted in two ways: one is during requirement analysis phase and the other is by analyzing the past data-entry patterns.

3. **Performance depends on mobile screen size:** Currently, mobile devices are available with varying screen sizes. If the screen size increases, more active-contexts can be fit into one mobile screen. As a result, performance could be improved.

6 Summary and Conclusions

For mobile devices, UIs design is a challenging task due to small screen size and other constraints. Interface tailoring methods are employed to design UIs for small screens. In this paper, we have proposed an improved approach to design UI forms for small screens by extending the notion of attribute temporality to interface tailoring approaches. The analysis results on the real dataset show that the proposed approach has the potential for improving the UI performance in mobil environment.

As a part of future work, we are planning to investigate the index-based approaches using the notion of attribute temporality for designing improved UI for mobile environments.

Acknowledgments

This work has been carried out with the support from Nokia Global University Grant.

References

1. Niu, Y., Li, X., Meng, X., Sun, J., Dong, H.: A Constraint-based User Interface Design Method for Mobile Computing Devices. In: 1st International Symposium on Pervasive Computing and Applications, August 3-5 (2006)
2. Chade, S., Koivisto, A.: Mobile Form-Editor: A push based group communication tool. In: Mobility 2006, Bangkok, Thailand, October 25-27 (2006)
3. Tsai, M.-J., Chen, D.-J.: Generating User Interface for Mobile Phone Devices Using Template-Based Approach and Generic Software Framework. Journal of Information Science and Engineering 23, 1189–1211 (2007)
4. Menkhaus, G., Pree, W.: User Interface Tailoring for Multi-Platform Service Access. In: International Conference on Intelligent User Interfaces: IUI 2002, San Francisco, CA, January 13-16, pp. 208–209 (2002)
5. Luyten, K., Coninx, K.: An XML-based runtime user interface description language for mobile computing devices. Springer, Heidelberg (2001)
6. Szekely, P.: Retrospective and Challenges for Model-Based Interface Development. In: 3rd Int. Workshop on Computer-Aided Design of User Interfaces CADUI 1996, June 5-7. Presses Universitaires de Namur, Namur (1996)
7. Kumara Swamy, M., Krishna Reddy, P.: An Efficient Context-based User Interface by Exploiting Temporality of Attributes. In: 16th Asia-Pacific Software Engineering Conference (APSEC 2009), Penang, Malaysia, December 1-3, pp. 205–212 (2009)
8. Fletcher, L.A., Erickson, D.J., Toomey, T.L., Wagenaar, A.c.: Handheld Computers a Feasible Altgernative to Paper Forms for Field Data Collection. Evaluation Review 27(2) (April 2003)

9. eSagu: IT-based Personalized Agro-Advisory system (January 2010), `http://www.esagu.in`
10. James, F.D., Christina, G., Won, C.K., Srdjan, K.: A Knowledge-Based User Interface Management System. In: CHI 1988 Human Factors in Computer Systems, Washington, D.C, May 15-19, pp. 67–72 (1988)
11. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: Conference on Human Factors in Computing Systems (CHI 2000), The Hague, Netherlands (April 2000)
12. Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey on Context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4) (2007)
13. Krishna Reddy, P., Ramaraju, G.V., Reddy, G.S.: eSagu^{TM}: A Data Warehouse Enabled Personalized Agricultural Advisory System. In: Proceedings SIGMOD 2007, Beijing, China (2007)
14. Myers, B.A.: Challenges of HCI Design and Implementation. Interactions 1(1), 73–83 (1994)
15. Myers, B.A.: User Interface Software Tools. ACM Transactions on Computer-Human Interaction 2(1), 64–103 (1995)
16. Dan, O.R.: MIKE: The Menu Interaction Kontrol Environment. ACM Transactions on Graphics 4(12), 33–42 (1984)
17. Schlungbaum, E., Elwert: Automatic user interface generation from declarative models. In: CAD UZ 1996, pp. 3–18. Namur University Press, Namur (1996)
18. Schlungbaum, E.: Individual User Interfaces and Model-based User Interface Software Tools. In: IUI 1997, Orlando Florida, USA (1997)
19. Singh, G., Green, M.: A High-level User Interface Management System. In: Proceedings SIGCHI 1989, April 1989, pp. 133–138 (1989)
20. Guha, S., Rastogi, R., Shim, K.: ROCK: ARobust Clustering Algorithm for Categorical Attributes. In: 15th Int. Conf. on Data Engineering (1999)