A Classification of Self-Explanatory User Interfaces

Maximilian Kern, Marco Blumendorf, Sahin Albayrak
DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
forename.surname@dai-labor.de

ABSTRACT
In this paper a definition of Self-Explanatory User Interfaces (SEUI) is proposed. Furthermore, existing approaches on SEUIs are classified by identification of their significant features. Derived from these features, challenges and open issues are elaborated. Then, advantages of a model-based approach for the development of SEUIs are given. Finally, a conclusion is given with an outlook on an ultimate SEUI from the author's perspective.

Keywords
assistance, guidance, self-explanatory UI, adaptive UI, meta UI, MDUI

INTRODUCTION
The term Supportive User Interface (SUI) has been introduced recently and still needs to be defined clearly. In our work, we understand the term supportive as the goal to support the user while interacting with a user interface. Support thus aims at the ability of a user interface to provide optimal interaction capabilities and the necessary configuration options therefore as well as help for the user to understand the rationales of a user interface, provide a context-sensitive help if the user is lost in navigation or requests help explicitly. In this paper we focus on Self-Explanatory User Interfaces as a subtype of SUIs that especially emphasizes the help as explanatory features of SUIs.

The earliest approaches for built-in support on an interactive system emerged around 1966 with the HELP system developed under the Genie project [12]. The HELP system provides answers to questions about commands and entities available on a UNIX based terminal window. While such approaches were restricted to low computing performance at this time, the ongoing technological improvements enables recent assistants being capable of understanding, interpreting and speaking human language, capturing and considering context information and learn from users by observing their interaction. In the following, we propose a definition of SEUIs. Furthermore, we clarify the term SEUI and classify existing approaches by analyzing their features. Afterwards, challenges of the development of SEUIs are discussed. Then, we discuss how SEUIs can benefit from model-based development. Finally, a conclusion on SEUIs is given with an outlook to an ultimate SEUI.

A DEFINITION OF SEUI
Self-explanatory user interfaces in general are characterized and thus, can be defined by the ability to reason on the application state and generate additional explanations or useful hints of higher value which support users in fulfilling their desired task faster. Therefore, SEUIs introspectively read out information hidden from the actual user interface and evaluate them. By these hints, the user gains deeper insight of the rationales in terms of purpose and structure of the application [6]. Advanced SEUIs are generic by means of that they adapt at runtime to the current context-of-use and they are not bounded to a specific domain. In this manner, their characteristics conform to those of meta UIs and thus, can be comprehended as a kind of meta UI. By taking the idea of an SEUI being able of accessing and reasoning on artifacts of other applications or domains, SEUIs can be thought of to be an ever-present agent or companion who intermediates between the user and the applications. Depending on its mightiness, it is not only giving hints generated out of the underlying application but is also able to interact on behalf of the user. The agent could make use of natural language processing (NLP) and understanding (NLU) and the user can establish a dialog with him. Users could then accomplish their task by cooperatively talking to the agent. For instance, in [13] an information-seeking chat bot is presented. This chat bot supports a tourist resided in Potsdam to find sight-seeing places and gain background information related to those places such as architects, historical persons, entrance fees and public transports. It integrates an ontology with topic maps applied as the discourse of the dialogue with the user. Furthermore, this approach utilizes templates for generating natural utterances which wrap the requested information.

FEATURES OF SEUIs
Existing approaches on self-explanatory user interfaces differentiate mainly in the way, how they appear to the user (Appearance) and how they are activated (Trigger). Furthermore, they can be distinguished by the type of knowledge base they are using and their scope or

mightiness. Figure 1 gives an overview of the identified features. These features are discussed in greater detail in the following sections.

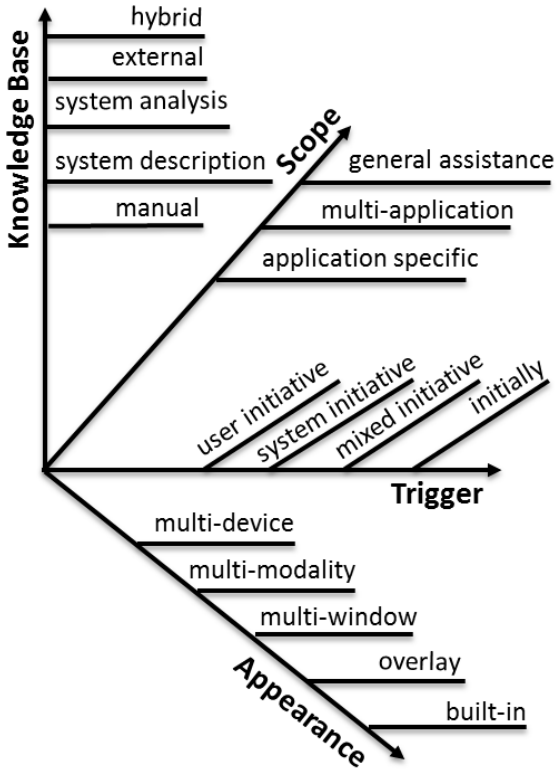

Figure 1. Overview of identified features of SEUIs

Appearance

The appearance of SEUIs is manifold. However, we can distinguish 5 basic ways of interaction:

1. Multi-device: shows the assistant on another device.
2. Multi-modality: utilizes one modality for the UI (e.g. graphics) and another one (e.g. voice) for assistance.
3. Multi-window: combines UI and assistance on one device and modality e.g. by using multiple windows, different voices or split screens.
4. Overlay: puts the assistance over the application which makes it easier to directly refer to specific elements.
5. Integration: integrates the assistance as part of the application so that the user perceives it as part of the application.

An example for multi-window, more in detail a split screen mode was applied in the DiamondHelp system introduced in [11], where the user still remains able to manipulate the underlying user interface directly. The user can choose between a 'guided' interaction in form of a chat with the system or 'unguided' interaction by interacting with classical user interface elements such as buttons, labels, etc.. Overlay mode is emphasizing the character of meta UI by overlaying the guided user interface in order to reach the user. This mode was applied to the MASP Guide [8] and is illustrated in Figure 2.

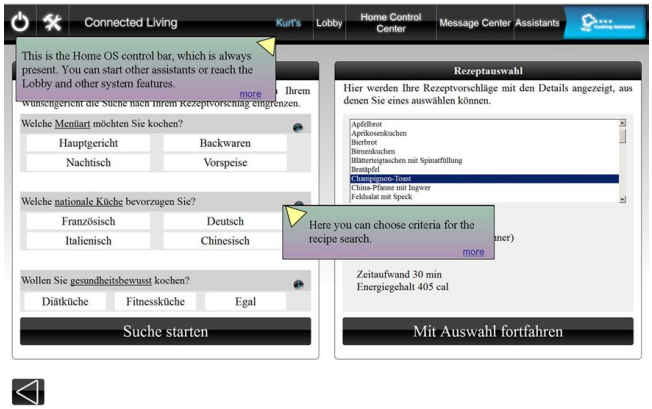

Figure 2. MASP Guide in overlay mode

Triggers

SEUIs either propose hints to the user pro-actively (system initiative) or the user is explicitly asking for help (user initiative). A third mode is called mixed-initiative which is a combination of both. A proactive SEUI, where the system takes over initiative, needs to recognize when support is actually required by the user. In order to be able to recognize the need for guidance, one option is to observe interaction history of a specific user and reason on the collected information. In [1] for instance, task models are used to connect sequences of observed user interactions to abstract tasks. Based on this information, possible interactions of users are predicted and could be proposed as a solution to the user. In [7] an approach for initial help is presented, which helps users using an application for the first time, i.e. it initially gives hints on startup. An important and reasonable issue for system initiative is to keep support decent in the sense of that the user is not flooded with hints and suggestions on what he is able to do next. In detail, system-initiated, self-explanatory user interfaces subtly appear in the moment, the user is lost in navigation or explicitly requests help. For the case the user explicitly requests help by asking for instance "Why does the menu bar appear all the sudden on the right hand side?", the SEUI may find the reason by analyzing the adaptation history and finds that the user is right-handed and switched using a touchscreen and they should not cover the user interface with her right arm. The crystal framework proposed in [9] enables the user to ask a wide variety of why-questions, the answer is generated by introspection of the current state of the application.

Knowledge base

Another aspect is the source from where to retrieve information for giving the user desirably useful hints. One option is that the designer or developer of the SEUI is manually identifying possible critical states of the user interface at design time. Practically, due to the nature of adaptive user interfaces, this is difficult since the designer might not be able to foresee each state of the application during runtime (Even if she could, she should prevent critical situations at design time by revising the design of

the application.). Thus, it is preferable that the supportive user interface is giving generic support during runtime. At this time, the SEUI can retrieve information either from the system description or from an external resource, e.g. the Internet. The former option would require that the system description offers more information than is held on the surface of the user interface and this information is available during runtime. By this way, hints are generated out of hidden artifacts of the system description. The latter option represents a bigger challenge since the information on the Internet needs to be matched to a machine-processible structure, i.e. a structure which is comprehensible by the SEUI. For this purpose, the use of some kind of ontology matching or well-formed source is inevitable. In [4] a hybrid approach "The Companions" is introduced, which is able to incorporate knowledge retrieved from local resources, but also from a social network or news site into a local rdf-based knowledge base (KB). In order to give the user the impression of talking to a human, the face of the avatar is displayed. The system was designed to enrich photo albums with semantic information about recognized people and places such as their relations or detailed information.

Scope

The previously mentioned possibility of retrieving information from an external resource yields to another aspect of SEUIs - the scope of an SEUI. Generated hints might be more useful to the user when the SEUI has knowledge which goes beyond the intended domain of the application, i.e. it has also knowledge of other applications and their domains. For instance, for an interactive application for preparing recipes, the SEUI gives reasons if a step is not feasible due an electric device is missing, which is controlled by another application for device management. Mightiness of an SEUI is addressing the potential of controlling the application itself or other applications. For instance, if a user asks for a missing device, the SEUI can implicate that the user wants to use the device and activate the device in the device management application. General assistance applies for fully generic SEUI approaches. Such approaches require no certain structure from the guided application.

CHALLENGES

Based on the identified features, we can identify various challenges for the development of SEUIs. The major general issue of giving support to the user is the understanding of the user and their needs. Getting this right is crucial so the user actually feels supported rather than annoyed. The users are playing the key-role in HCI, so they should not be displeased by the amount of hints and the moment hints are communicated by the system.

This directly leads to the appearance of the SEUI. It should please the user without disturbance and therefore needs to be well designed and provide the necessary integration into the application depending on the needs. Learning from many bad examples of help systems, it seems advisable to provide some kind of adaptation and personalization capability, which allows the continuous adaptation, based on the users behavior, and also requires the continuous monitoring of adaptation results and the performance of the help system in terms of user satisfaction.

Looking at the triggers to start the assistance, system-, user- and mixed initiative also pose different challenges. A system-initiative SEUI needs to be aware of situations, where users are not certain of how to proceed, and then find a reason (and a solution) in order to solve the users' problem. For instance, Microsoft's Paper Clip discourages users due to the lack of information about the context-of-use, i.e. it is not aware of the context. For user-initiative SEUIs the major issue lays in the ambiguity of a user's utterance, the system has to rely on the terms of the current domain, current task and the discourse of the user interface, i.e. it needs to be aware of the system state.

The issue of ambiguity then also refers to the knowledge base (KB) of an SEUI. As discussed earlier, the usage of an ontology or presumption of certain structures of the KB is inevitable. Then, the challenge is accounted to the quality of the ontology matching algorithm and the way of extracting and processing information. Furthermore, this quality depends also on the fineness of the world knowledge and common knowledge for SEUIs with knowledge which goes beyond the intended domain of the application.

Relating to the scope of SEUIs, there might not be one best way for supportive UIs. It depends on the needs of the user, the usage situation and the application if SEUIs are integrated parts or separate applications. Being external applications, this however also poses requirements on the application in terms of traceability of the current state and access to design information and semantic meaning of elements. An application might need to conform to a specific structure in order to integrate self-explainability. This has direct impact on the effort for application developers/designers, which should be ideally minimal. Thus, the challenge is to develop an open or standardized programming/controlling interface for applications in order to ease integration of SEUIs and access application knowledge.

A MODEL-BASED APPROACH TO SEUI DEVELOPMENT

From our point of view, model-based development comes along with major advantages in order to cope with previously mentioned challenges. Models provide explicit information about the application state and the contextual space instead of weaving information in unstructured program code. For the sake of separation of concern, information is held in several models each covering a certain aspect (e.g. context model, interaction model, abstract UI model, concrete UI model, final UI model, etc.). An SEUI can access this information easily and needed information can be retrieved from these models. For inferring on semantics, the SEUI benefits from the self-explanatory nature of models. The MASP has built-in

features for monitoring the application state and interactions [2], which lower the development effort for recognizing trigger situations of an SEUI. Another model-based approach on Automated Usability Evaluation (AUE) described in [10] is simulating a user model at run-time in order to identify lacks in usability. This approach could also be applied in order to identify problematic states of an application during runtime and provide hints to the user (for system-initiative SEUIs). Models have been proposed and utilized as basis for adaptive systems [2][3][5]. Regarding the appearance, an SEUI integrated into such systems needs to be as adaptive as the surrounding environment.

CONCLUSION AND OUTLOOK

Self-explanatory user interfaces raise supportiveness of user interfaces significantly. We have proposed a definition for SEUIs, which is "the ability (of a user interface) to reason on the application state and generate additional explanations or useful hints of higher value which support users in fulfilling their desired task faster.". It was stated that SEUIs mainly differentiate in their activation mechanism (user-/system-/mixed-initiative, initially), their appearance (multi-device, multi-modality, multi-window, overlay, built-in), their knowledge base (manual, system description, system analysis, external, hybrid) and their scope (application specific, multi-application, general assistance). We are conscious that our classification is not completive but consider it as a first step towards a better understanding of SEUI as a special kind of SUI. The challenges and open issues on SEUI lay in the design and the understanding of users and their needs. Furthermore, it was elaborated, how development of SEUIs can benefit from a model-based approach.

As a conclusion, the ultimate SEUI from our perspective is a companion, which is ubiquitously accessible and provides useful hints at any time. It would only take initiative if a user needs help and would incorporate knowledge beyond the current application's domain. For retrieving external information, it would apply approved algorithm for matching terms against ontologies. In order not to allocate space on the screen, the user could communicate entirely via voice, but it remains optional for overlay mode. Moreover, the SEUI would act in the same way as an expert knowing your personal needs and observing any of your interactions.

REFERENCES

1. Bezold, M. Describing user interactions in adaptive interactive systems. In UMAP (2009), 150–161.
2. Blumendorf, M., Lehmann, G., and Albayrak, S. Bridging models and systems at runtime to build adaptive user interfaces. In EICS '10: Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, ACM (2010).
3. Clerckx, T., Luyten, K., and Coninx, K. Dynamo-aid: A design process and a runtime architecture for dynamic model-based user interface development. In EHCI/DS-VIS (2004), 77–95.
4. Dingli, A., Wilks, Y., Catizone, R., and Cheng, W. The companions: Hybrid-world approach. In International Joint Conference on Artificial Intelligence (IJCAI)(Pasadena, CA, 2009).
5. Duarte, C. Design and Evaluation of Adaptative Multimodal Systems. PhD thesis, Department of Informatics, University of Lisbon, March 2008. DI/FCUL TR-08-9.
6. García Frey, A., Calvary, G., and Dupuy-Chesa, S. Xplain: an editor for building self-explanatory user interfaces by model-driven engineering. In Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, EICS '10, ACM (New York, NY, USA, 2010), 41–46.
7. Kang, H., Plaisant, C., and Shneiderman, B. New approaches to help users get started with visual interfaces: multi-layered interfaces and integrated initial guidance. In dg.o '03: Proceedings of the 2003 annual national conference on Digital government research, Digital Government Society of North America (2003), 1–6.
8. Kern, M., Trollmann, F., Blumendorf, M., and Albayrak, S. Adaptive user interface assistance in smart environments. In Proceedings of the Workshop on Meaning and Matching (AISB2010), De Montfort University Leicester, SSAISB (2010).
9. Myers, B. A., Weitzman, D. A., Ko, A. J., and Chau, D. H. Answering why and why not questions in user interfaces. In CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, ACM (New York, NY, USA, 2006), 397–406.
10. Quade, M., Blumendorf, M., and Albayrak, S. Towards model-based runtime evaluation and adaptation of user interfaces. In Proceedings of International Workshop on User Modeling and Adaptation for Daily Routines (UMADR2010): Providing Assistance to People with Special and Specific Needs (2010).
11. Rich, C., and Sidner, C. DiamondHelp: A generic collaborative task guidance system. AI Magazine 28, 2 (2007).
12. Roberts, R. Help: a question answering system. In AFIPS '70 (Fall): Proceedings of the November 17-19, 1970, fall joint computer conference, ACM (New York, NY, USA, 1970), 547–554.
13. Stede, M., and Schlangen, D. Information-seeking chat: Dialogue management by topic structure. In Proceedings of the 8th Workshop on Semantics and Pragmatics of Dialogue, CATALOG 04, Barcelona, 2004 (2004).*Autonomous Systems (ICAS 2008)*