UI Prototyping for Multiple Devices Through Specifying Interaction Design

Jürgen Falb[1], Roman Popp[1], Thomas Röck[2], Helmut Jelinek[2], Edin Arnautovic[1], and Hermann Kaindl[1]

[1] Vienna University of Technology, ICT
A-1040 Vienna, Austria
{falb, popp, arnautovic, kaindl}@ict.tuwien.ac.at
[2] Siemens Austria, PSE
A-1210 Vienna, Austria
{thomas.roeck, helmut.jelinek}@siemens.com

Abstract. While user interface (UI) prototyping is generally considered useful, it may often be too expensive and time-consuming. This problem becomes even more severe through the ubiquitous use of a variety of devices such as PCs, mobile phones and PDAs, since each of these devices has its own specifics that require a special user interface.

Instead of developing UI prototypes directly, we propose specifying one interaction design from which UIs can be automatically generated for multiple devices. Our implemented approach uses communicative acts, which derive from speech act theory and carry desired intentions in interactions. Models of communicative acts, UI domain objects and interaction sequences comprise interaction design specifications in our approach and are based on a metamodel that we have defined. We support the development of such models through an IDE, which is coupled with the UI generator. This allows a new form of UI prototyping, where the effects of each model change can be seen immediately in the automatically generated UIs for every device at once.

1 Introduction

User interface (UI) prototypes are generally considered useful, but the effort and time required for building one are often prohibitive. In order to address this issue, "low fidelity" prototyping offers a reasonable compromise.

We propose a novel approach to UI prototyping with "perfect fidelity", where the user interface is automatically generated. The result would be identical to the real UI due to this generation process, assuming that the same interaction design according to our approach is taken as input.

In addition, our approach allows UI prototyping for multiple diverse devices at once (PCs, mobile phones and PDAs), from a single interaction design specification. We are not aware of any previous approach like that.

Of course, this approach to UI prototyping requires certain prerequisites. We build upon our previous work on automatic UI generation for multiple devices.

C. Baranauskas et al. (Eds.): INTERACT 2007, LNCS 4662, Part I, pp. 136–149, 2007.

The underlying approach based on the use of *communicative acts*, and the rendering techniques used for generating user interfaces for multiple devices are published elsewhere. In [1] we sketched the underlying approach, and in [2] we presented the rendering techniques used for generating user interfaces for multiple devices. In essence, they allow automatic generation of UIs for information systems using given widgets as building blocks.

In the current paper, we focus on the support for the designer to specify an interaction design according to our approach and, in particular, the support for UI prototyping. In order to make this paper self-contained, we also describe our metamodel. It defines how models should look that represent interaction design specifications.

We have fully implemented this approach in a tool, which is already in industrial use for real-world applications supporting several diverse devices. As a running example, let us use a small excerpt from a unique and new hypermedia guide that has been put into operation by the Kunsthistorisches Museum (KHM) in Vienna. It had been created with our approach and tools.

The remainder of this paper is organized in the following manner. First, we sketch what communicative acts are and where they come from. Then we present and explain our metamodel of interaction design specifications. After that, we show how the modeling task is supported by an IDE, with a focus on UI prototyping. We also show examples of automatically generated UIs for multiple devices and summarize anecdotic evidence for the usefulness of our approach. Finally, we discuss our approach and compare it with related work.

2 Communicative Acts

By investigating human language communication, philosophers observed that language is not only used to describe something or to give some statement but also to do something with intention — to act. Most significant examples are so-called performatives, expressions such as "I nominate John to be President.", "I sentence you to ten years imprisonment." or "I promise to pay you back.". In these expressions, the action that the sentence describes (nominating, sentencing, promising) is performed by the sentence itself; the speech is the act it effects.

Early and seminal work on speech acts was done by Searle [3]. In this essay Searle claims that "the speaking a language is performing speech acts, act such as making statements, giving commands, asking questions, making promises and so on". Such speech acts are basic units of language communication in this work, and not tokens, words or sentences as previously believed. Each speech act can be represented in the form $F(P)$ where F is the *illocutionary force* (intention) and P its *propositional content*. For example, the speech acts

Sam smokes habitually.
Does Sam smoke habitually?
Sam, smoke habitually!

have the same proposition P (Sam smoking habitually) but different illocutionary forces F: making an assertion, asking a question and giving an order. Speech acts can be further characterized through:

- **Degree of strength**, which describes the intensity and importance of the speech act (e.g., a command and a polite request have more or less the same illocutionary force but different strength);
- **Propositional content conditions**, which describe conditions on the content (e.g., the content has to be about the future);
- **Preparatory condition** (e.g., the receiver must be able to perform the action).

Such speech acts are basic units of language communication. Since speech act theory provides a formal and clean view of communication, computer scientists have found speech acts very useful for describing communication also apart from speech or natural language, e.g., in the area of agent communication languages [4]. To emphasize their general applicability, the notion communicative act is used in this context. The communicative acts simply abstract from the corresponding speech acts in not relying on speech or natural language.

3 Metamodel of Interaction Design Specifications

The metamodel defines what the models of an interaction design specification should look like in our approach. It captures three main concepts used for modeling as well as their relations:

- the intention captured by a communicative act,
- the propositional content modeled by use of an ontology language, and
- the set of interaction sequences modeled with a finite state machine.

Figure 1 illustrates the metamodel for interaction design specification in the form of a UML class diagram containing metaclasses and their relations. The central concept here is the *communicative act*. As stated above, it carries an intention of the interaction: question, assertion, etc. For the purpose of this paper, we show only a selection of the classification of communicative acts.

According to the form of speech acts, the interaction intention as specified by the type of the communicative act, and the propositional content are separated. In the metamodel, the content is represented by UI domain classes and UI domain properties. We prefixed the content entities with "UI" to indicate that the classes and properties model the domain according to the structure presented to the user which should correspond to the user's point of view and need not to be identical to the internal system representation of the domain.

Since we are using the ontology language OWL (Web Ontology Language) [5] for domain modeling, properties are also "first class citizens" and therefore shown separately in the metamodel. Treating classes and properties equally allows us to refer to both independently from a communicative act. This has also practical advantages such as allowing a question about some property per se. For example,

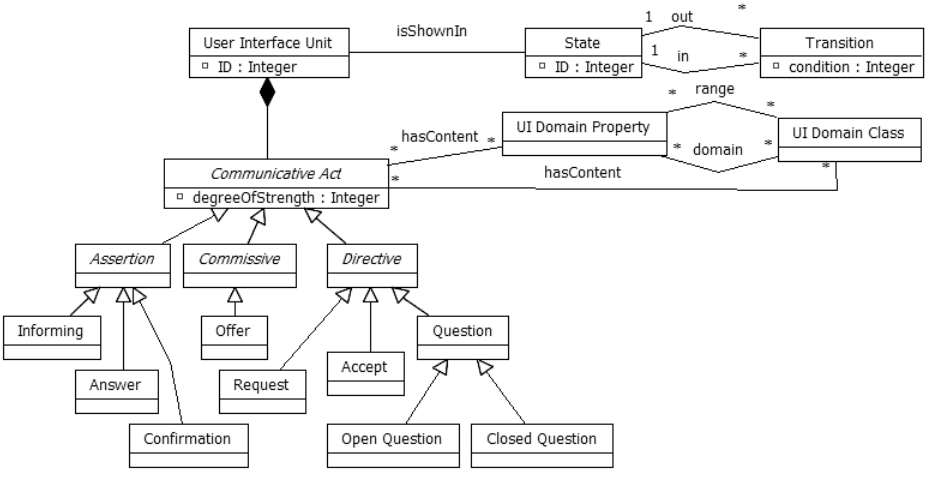

Fig. 1. The metamodel of interaction design specifications in UML

for the UI domain class property *maritalState* related to a UI domain class *Person*, a question about the percentage of married persons can be asked. This is clearly different from a question about a person's marital status. Furthermore, the content of a communicative act can be the class or property itself instead of an instance of a class or property.[1] This results in a discourse about the class or property itself, e.g., talking about what makes up a person in the particular domain/context. Regarding the transformation into user interfaces, a typical application of such a meta-discourse is the generation of help documents.

The UI domain classes and properties also relate to the actual application logic. Thus, they have to be connected technically to the application interface (e.g., Enterprise Java Beans or CORBA). This is done by implementation technology-specific templates describing how UI domain classes and properties get associated with application classes and data. This mapping is not necessary for UI prototyping, it is only required if the prototype should operate on real data or should evolve into a product.

One or more communicative acts are contained in one *User Interface Unit* (UIU). The UIU is a basic unit presented to the user providing choices of different interaction sequences based on the contained communicative acts. However, it is still an abstract entity — it can be mapped to one or several concrete user interface screens according to concrete device profiles.

The set of interaction sequences is modeled with a finite state machine where each state can have multiple ingoing and outgoing transitions representing segments of the interaction sequences. Each state also implicitly defines and fulfills an associated communicative act's preparatory condition. After each state transition, the UIU connected to this state is presented to the user. More precisely,

[1] Instances of properties are usually equivalent to statements in ontology languages.

this happens while entering this state. In fact, the machine performs actions as specified in the communicative acts: e.g., it presents a question about some domain object — it asks. When the user interacts with the machine through the user interface, e.g. answering a question, a corresponding communicative act will be generated leading to the execution of its action and a subsequent state transition. The transition conditions between the states are defined in a simple expression language specified by us.

An interaction design specification according to this metamodel provides the essence of a user interface to be generated. For its concrete rendering, additional device profiles and style guides are used to take device characteristics into account. Combined with heuristics, the rendering engine generates UIs for PCs, PDAs and mobile phones. We currently support only information system UIs and no direct manipulation interfaces, since we have neither appropriate UI building blocks for the latter nor rules that would map communicative acts to them.

4 Tool Support for Specifying and Prototyping

Specifying such models directly in OWL or XML syntax is technically on a low level, takes a lot of effort, and does not support rapid prototyping. Therefore, we have also built an IDE for tool support of this modeling task through a graphical interface. This IDE compiles the OWL and XML representations of the models and displays at the same time the generated user interface in a separate window. If designers change parts of the models, the effects of the changes are shown immediately in the generated interface. These results and changes can be seen immediately for all supported diverse devices. In our running example, we present the output generated for PCs and PDAs.

The IDE provides support for modeling the three distinct metamodel parts and their relationships:

- the UI domain model,
- the state machine, and
- the communicative acts.

By using the tool, designers can create the UI domain model and communicative acts from scratch or from predefined templates and change their properties and references to the content. Then they can group the communicative acts into user interface units and assign them to states of the finite state machine. For actually generating a UI, some additional information is required, like style guides and rendering settings for each supported device. They can and should be provided by UI experts even before modeling the interaction design and may be assumed to be given. So, they are not integrated in the IDE and would have to be edited externally. Future experience will show whether their integration into the IDE may be necessary. In the following, we leave these aspects aside and describe the different modeling steps in detail.

The IDE enables designers to construct UI domain models directly or to import ontologies designed with an external ontology editor and stored in OWL

format. The designed classes, properties and instances are candidates for the propositional content of communicative acts and for state transition conditions in the subsequent design steps. UI domain models specified within the IDE are stored separately within an OWL file to allow easy reuse for other purposes.

State machines can be either built using graphical UML tools, which generate an XMI representation, or directly within this IDE. The advantage of using our IDE is an immediate check of the transition conditions during design in the following sense. The tool deduces possible occurring communicative acts and their associated content from the model and warns designers if they use information not available in the specified state. This feature relies on a designer having already associated a user interface unit with the state and assigned one or more communicative acts to the UIU.

Upon specifying the user interface units, designers can specify the communicative acts and assign them to a user interface unit. After executing this step, designers can have a first look at the resulting user interface and refine it afterwards. This allows incremental development of the UI prototype.

Figure 2 shows a screenshot of the IDE, that presents a dialogue for editing communicative acts. The left part of the screenshot displays all states, their outgoing transitions, and their associated user interface units (UIU), represented by the user interface description child node of each state. Each UIU contains a list of its aggregated communicative acts that should be presented together. In the example shown in this screenshot, the communicative act named *SearchQuestion* is selected and its properties can be edited in the right part of the IDE.

The right part in the screenshot displays the type and the associated content of the currently selected communicative act. The most important element on the screenshot is the type selection of the communicative act. For asking the search question in our running example, the *Open Question* type is selected that will result in a certain kind of input widgets. Finally, designers can specify the propositional content the communicative act refers to, by graphically specifying instance selectors via predefined queries (like selecting all instances, the current one or instances referred to by another instance). These queries allow designers to specify paths along the properties of classes and to select instances. All queries within the specification are replaced with the actual instances at runtime. In Figure 2, a predefined lookup instance is selected as content for the communicative act, acting as placeholder for the not yet known instances of exhibits. In addition, designers can group multiple communicative acts — not shown in the screenshot — to declare stronger relationships between particular communicative acts. The grouping gives additional hints for rendering regarding the placement of the parts generated from communicative acts, and the splitting into multiple pages on more constraining devices like PDAs or mobile phones.

The generated user interfaces corresponding to the UIU selected in Figure 2 are shown in Figures 4 and 6, for the PC and the PDA, respectively. These are generated immediately after each modification and allow rapid prototyping for multiple diverse devices. In our example, the screenshots display a form for searching exhibits. The communicative act named *SearchQuestion* of type

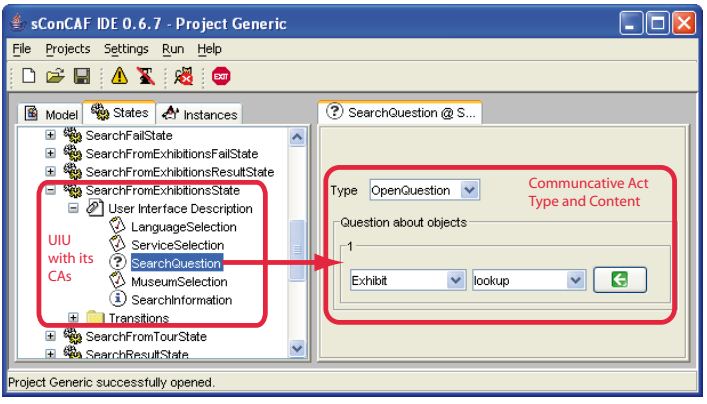

Fig. 2. IDE dialogue for editing communicative acts: specifying an Open Question

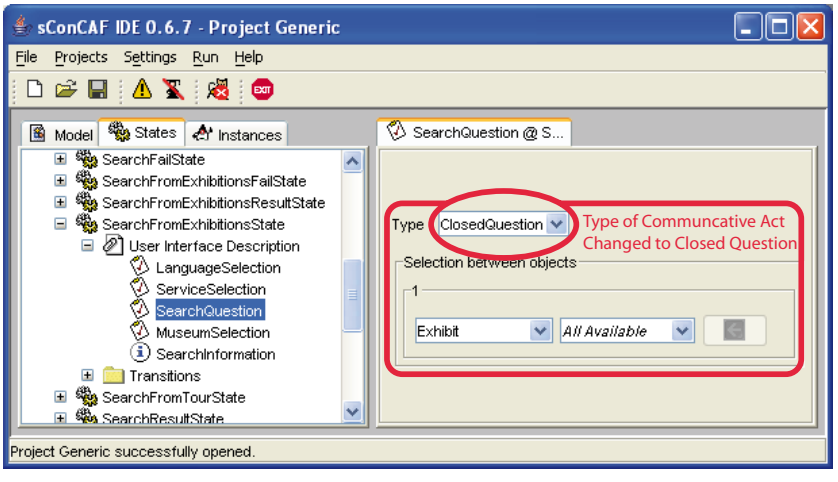

Fig. 3. IDE dialogue for editing communicative acts: specifying a Closed Question

Fig. 4. UI resulting from use of Open Question for search on a desktop PC

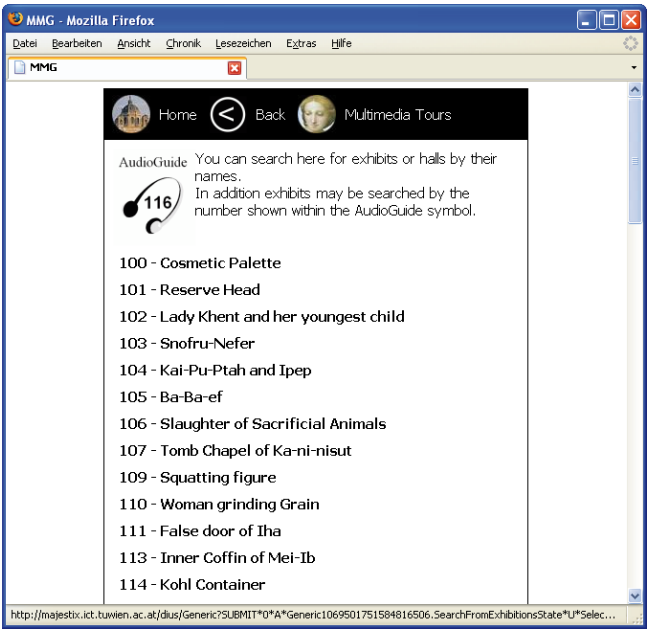

Fig. 5. UI resulting from use of Closed Question for search on a desktop PC

Fig. 6. UI resulting from use of Open Question for search on a PDA

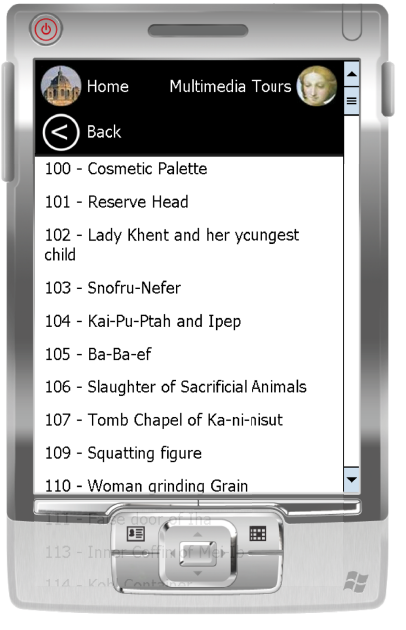

Fig. 7. UI resulting from use of Closed Question for search on a PDA

Open Question is rendered as a textbox with a button for submitting the search string. This part of the running example is rendered quite similarly on both devices to assure consistent behavior of the multiple UIs. One device-specific difference is the different order. On more constraining devices, the important interactive elements are placed before the additional information to avoid unnecessary scrolling.

In Figure 3, the communicative act type of *SearchQuestion* has changed from *Open Question* to *Closed Question* and the propositional content from a not yet known exhibit to the list of all available exhibits. After having made these two changes, the designer can immediately see the effect on the user interfaces as shown here in Figures 5 and 7. In this case, the textbox and its associated button for free text search are replaced by a list of links, each entry representing a selectable exhibit.

The design steps described above can be executed in an iterative and incremental way, allowing designers to select different alternatives and to incrementally extend the UI prototype. In our example, the alternatives are free text search and a selectable list of exhibits. After refining style guides and rendering settings, the resulting UI prototype can even be used as a product, as it happened in the case of the museum.

5 Anecdotic Evidence

We have interviewed four designers, who provided us with subjective evaluations of the approach and the tools implementing it. They do not have very strong formal education or experience in building user interfaces. After a short training period, all designers grasped the modeling concepts and were able to design a user interface for an application using our approach and tools. They state that the design process is intuitive and enables a fast and convenient design of a user interface for diverse devices. The designers also stated, that they need little technical knowledge to model the user interface.

For UI prototyping, the immediate feedback on look and feel of the concrete user interface through its automatic generation is appreciated. We observed a designer exploring the space of design alternatives in terms of various communicative acts and UI domain objects (as well as their attributes to be shown or not). Instead of envisioning how the resulting user interfaces might look on a PC and a PDA, our approach allows immediately seeing and even comparing them. Therefore, it was possible for the designer to select from the design alternatives those resulting in the user interfaces with the highest appeal to him.

An important question for the usefulness of our approach to UI prototyping is, of course, whether the space of user interfaces that can be automatically generated even contains usable ones to select from. After all, usability is said to be a major problem for generated UIs. Based on data gained from the real-world application in the museum, we tend to give an affirmative answer (at least for such kinds of user interfaces).

Let us sketch the essential results here.The evaluation through the museum users is restricted to the UI for the PDA only, since there is still a copyright issue about selling CDs with the PC version pending. For a certain period after the installation of the museum guide running on the PDA, a sample of subjective evaluations by 363 subjects was collected (all of them having used the same version). These users provided answers to a few standardized closed questions and free comments in a subjective questionnaire. We used a multi-point rating scale, more precisely a *Likert scale*. The scale for all closed questions was (*very good, good, medium, less good, not good*).

The answers to our key questions can be summarized as follows:

1. "How did you like the MMG?"[2]
   A clear majority of the subjects replied to this question with *very good* or *good*. Although a portion of visitors did not find the MMG so good, there is some empirical evidence that it has some appeal to museum visitors.
2. "Was the MMG easy to handle?"
   A majority of the subjects replied to this question with *very good* or *good*, and a clear majority with *very good*, *good* or *medium*. While the handling was rated less good than the appeal, there is still some empirical evidence that the handling is at least acceptable to many visitors.

In general, these aspects of usability of the UIs generated was assessed informally as good. It is clear, however, that expert UI designers and implementers can still provide better user interfaces, especially from an artistic perspective or in the details. This situation is reminiscent of code generated by compilers, which can often be "beaten" by experienced humans programming directly on the level of machine code.

6 Discussion

A major advantage for designers is that they can immediately see the resulting UIs, even for multiple devices and in "perfect fidelity", since these UIs are fully automatically generated from the same interaction design specification. There is no separation between UI prototyping and the development of a final UI. Both are served by specifying an interaction design. So, they become essentially the same activity.

Still, such a UI may not be "good enough". Our prototyping approach allows the interaction designer to perform a kind of optimization search for a better UI due to the possibility of quickly trying out alternatives. This is reminiscent of a hill-climbing approach that does not guarantee to find a global optimum, but the resulting UI can be expected to be better than the one from specifying a design without seeing the resulting UI. Whether it is really "good enough" can only be decided by tests with users, of course.

[2] MMG is the acronym used there for "multimedia guide", the not very accurate name of this application as used officially, which is rather a hypermedia guide.

Our UI prototyping approach is also usable by non-technicians to create user interfaces, since our interaction design model is based on communicative acts derived from a theory about human communication. Trying out alternatives also supports exploring different UIs by non-technicians, thus making this approach usable by a wide range of designers with different background. Finally, rendering the interaction design to diverse devices supports a variety of user groups.

7 Related Work

Rosenberg states in his review on 20 years of UI prototyping [6] that in order to achieve the best possible UI design it is still most important to enable the designer to iterate through design variations as fast as possible, particularly of the more complex interfaces nowadays. Our approach supports this fast iteration even for multiple devices by easy modification of models based on communicative acts.

So-called low-level prototyping techniques (e.g., [7]) provide great freedom but are used mostly for exploration. In contrast, our approach is more "high-level" and evolutionary. Our prototypes matches the final UIs in layout, colors, sizes, etc. One disadvantage of "high-level" prototyping mentioned by Sefelin in [8] is that designers show less willingness to realize their suggestions. On the other hand, with low-level prototyping the designer can overlook limitations and constraints of the real product.

Earlier approaches had their focus on the data view of the application and have thus prototyped mostly database UIs (e.g., [9]).

Elkoutbi et al. [10] present an approach for requirements engineering that generates a user interface prototype from scenarios. Scenarios are enriched with user interface (UI) information and are automatically transformed into UML statecharts. These statecharts are then used for UI prototype generation. We believe that our approach is more intuitive, since it enables the designer to specify the interaction semantics "naturally" as question, answer, informing, offer, etc. The design process also seems to be more efficient since the designer can see the implications of the interaction specification change in the change of the UIs of multiple devices immediately.

Puerta et al. recently presented a model-based tool to guide early user interface design [11]. Their tool guides the initial specification of user interfaces, so called *wireframes* — a set of elements to be present in, e.g., a Web page or a screen of a desktop application. The elements in wireframes are related to domain objects and user tasks. For the representation of wireframe elements, XIML [3] is used. Our approach differs in taking a more abstract view and concentrating on the intentions of interaction, whereas they try to hide the models. In addition, our UIs are generated immediately to support rapid prototyping, whereas their tool is used to support manual design of UIs.

In [12], they support "low-fidelity" prototyping like with common GUI builders but keep higher-level models consistent with the concrete user

[3] http://www.ximl.org/

interface. As a result, they can instantly derive other types of user interfaces (e.g., using HTML or Java GUI) from the high-level model. For the purpose of prototyping, we believe that our approach is more straight-forward since it has only two levels of abstraction and the designer specifies the common basis of the different generated user interfaces.

Furthermore, our approach to specifying interaction design and generating the UI for "high-fidelity" prototyping is related to common model-based UI approaches. Van Setten et al. [13] provide a comparison of classical interaction design techniques (Goals, Operators, Methods and Selection rules (GOMS), Task-Action Grammar (TAG), Command Language Grammar (CLG), and External-Internal Task Mapping (ETIT)) and propose a metamodel for the conceptual basis of interaction design techniques. According to their metamodel, our approach covers almost all concepts like task and object mapping between external (e.g., user view of objects) and internal entities (e.g., system view of objects) and action design (e.g., designing the different communicative acts and their reference to actions in the system). In contrast to their metamodel, we do not include system functionality design and task analysis. Since our approach allows the generation of user interfaces, however, it takes implicitly into account the feedback from the system to the user (e.g., errors) which is not covered by any of the approaches in van Setten's comparison.

Another model-based approach is the User Interface Markup Language (UIML) [14] is a XML-based language used for device-independent descriptions of user interfaces. Abstractly defined user interface structures in UIML are mapped to the individual widgets by the render engine. This work is very important but on a much lower level of detail than ours, which has its focus on specifications of intent for UI prototyping.

Seffah et al. state in [15] that there is a need for new integrative frameworks for modeling, designing and evaluating multi-device user interfaces. They further conclude that model-based and pattern-based approaches are most suitable for designing "multiple user interfaces" once that also obey the different device characteristics. Our approach integrates both design approaches by specifying an interaction design model through the IDE and allowing the designer to select appropriate styles for the different devices. Since we automatically generate UIs for prototyping, the designer can immediately evaluate the consistency and the usability of the selected styles.

An advanced approach to specifying multi-device user interfaces based on task models is presented in [16]. It is widget-oriented employing abstract descriptions of the UI elements as the basic abstraction. Several of the transformations between models have to be done manually. When used for UI prototyping it only provides an abstract structure of the UI as immediate feedback while editing the task model.

Gajos and Weld [17] treat interface generation for different devices as an optimization problem, taking into account interface elements, device properties and usage patterns. The interface elements represent abstract UI widgets and are specified in terms of data exchanged between the user and the application.

The abstraction level is relatively low, since the "raw" data to be shown on the screen are specified. This tool concentrates on the definition of single screens and does not provide any possibility to model communication paths like in our approach.

The work of Eisenstein et al. [18] is in line with ours concerning the importance of high-level UI models for multi-device user interfaces but relies more on designer assistance than our approach to automatically generating the UI prototype.

Generating UIs based on an abstract definition of the user interface and in combination with knowledge of the capabilities of the target display was listed as a system challenge for ubiquitous and pervasive computing recently [19]. In this context, "high-fidelity" prototyping can be a valuable contribution, since it allows incorporating the capabilities and limitations of the various devices without requiring the designer to obey them during prototyping. We think that our implemented approach is a major step in this direction.

8 Conclusion

In this paper, we present a new approach to UI prototyping with "perfect fidelity", where the user interface is automatically generated using our rendering approach described elsewhere. Especially UI prototyping for multiple diverse devices at once, from a single interaction design specification, is new according to our best knowledge.

We also present tool support for specifying such models in the form of an IDE. Through the coupling of this IDE with the generator tool, our new approach to UI prototyping for multiple devices becomes feasible.

Our approach is fully implemented and already in industrial use. A unique and new hypermedia guide has been put into operation by a major museum, which had been created with our approach and tools. Since important aspects of usability of this application have been judged favorably by a sample of random museum users, we think that the usability of our generated UIs is "good enough" for our approach to UI prototyping.

References

1. Falb, J., Popp, R., Röck, T., Jelinek, H., Arnautovic, E., Kaindl, H.: Using communicative acts in interaction design specifications for automated synthesis of user interfaces. In: ASE'06. Proceedings of the 21th IEEE/ACM International Conference on Automated Software Engineering, Piscataway, NJ, USA, pp. 261–264. IEEE Computer Society Press, Los Alamitos (2006)
2. Falb, J., Popp, R., Röck, T., Jelinek, H., Arnautovic, E., Kaindl, H.: Fully-automatic generation of user interfaces for multiple devices from a high-level model based on communicative acts. In: HICSS-40. Proceedings of the 40th Annual Hawaii International Conference on System Sciences, Piscataway, NJ, USA, Jan 2007, IEEE Computer Society Press, Los Alamitos (2007)

3. Searle, J.R.: Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, Cambridge, England (1969)
4. Foundation for Intelligent Physical Agents: FIPA communicative act library specification. Technical report, Foundation for Intelligent Physical Agents (2002), http://www.fipa.org
5. W3C: OWL Web Ontology Language Reference (2004), http://www.w3.org/2004/OWL/
6. Rosenberg, D.: Revisiting tangible speculation: 20 years of UI prototyping. interactions 13(1), 31–32 (2006)
7. Snyder, C.: Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces (The Morgan Kaufmann Series in Interactive Technologies). Morgan Kaufmann, San Francisco (2003)
8. Sefelin, R., Tscheligi, M., Giller, V.: Paper prototyping — What is it good for?: A comparison of paper- and computer-based low-fidelity prototyping. In: CHI '03. Extended Abstracts on Human Factors in Computing Systems, pp. 778–779. ACM Press, New York, NY, USA (2003)
9. Janssen, C., Weisbecker, A., Ziegler, J.: Generating user interfaces from data models and dialogue net specifications. In: CHI '93. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 418–423. ACM Press, New York, NY, USA (1993)
10. Elkoutbi, M., Khriss, I., Keller, R.K.: Automated prototyping of user interfaces based on UML scenarios. Automated Software Engg. 13(1), 5–40 (2006)
11. Puerta, A., Micheletti, M., Mak, A.: The UI pilot: A model-based tool to guide early interface design. In: IUI'05. Proceedings of the 10th International Conference on Intelligent User Interfaces, pp. 215–222. ACM Press, New York, NY, USA (2005)
12. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: USIXML: A language supporting multi-path development of user interfaces. In: EHCI/DS-VIS, pp. 200–220 (2004)
13. van Setten, M., van der Veer, G.C., Brinkkemper, S.: Comparing interaction design techniques: A method for objective comparison to find the conceptual basis for interaction design. In: DIS'97. Proceedings of the Conference on Designing Interactive Systems, Amsterdam, The Netherlands, pp. 349–357. ACM Press, New York, NY, USA (1997)
14. Abrams, M., Phanouriou, C.: UIML: An XML language for building device-independent user interfaces. In: Proceedings of the XML 99 (1999)
15. Seffah, A., Forbrig, P., Javahery, H.: Multi-devices "Multiple" user interfaces: development models and research opportunities. J. Syst. Softw. 73(2), 287–300 (2004)
16. Mori, G., Paterno, F., Santoro, C.: Design and development of multidevice user interfaces through multiple logical descriptions. IEEE Transactions on Software Engineering 30(8), 507–520 (2004)
17. Gajos, K., Weld, D.S.: SUPPLE: Automatically generating user interfaces. In: IUI '04. Proceedings of the 9th International Conference on Intelligent User Interfaces, pp. 93–100. ACM Press, New York, NY, USA (2004)
18. Eisenstein, J., Vanderdonckt, J., Puerta, A.: Applying model-based techniques to the development of UIs for mobile computers. In: IUI'01. Proceedings of the 6th International Conference on Intelligent User Interfaces, pp. 69–76. ACM Press, New York, NY, USA (2001)
19. Want, R., Pering, T.: System challenges for ubiquitous & pervasive computing. In: ICSE'05.Proceedings of the 27th International Conference on Software Engineering, pp. 9–14. ACM Press, New York, NY, USA (2005)