2014 IEEE International Conference on Systems, Man, and Cybernetics
October 5-8, 2014, San Diego, CA, USA

Duality of Task- and Discourse-based Interaction Design for GUI Generation

Roman Popp, Hermann Kaindl, Simon Badalians, David Raneburger
Vienna University of Technology
Gusshausstrasse 27-29, 1040 Vienna, Austria
{popp, kaindl, badalians, raneburger}@ict.tuwien.ac.at

Fabio Paternò
CNR-ISTI, HIIS Laboratory
Via Giuseppe Moruzzi 1, 56124 Pisa, Italy
fabio.paterno@isti.cnr.it

***Abstract*—In general, Interaction Design is considered important for achieving usable user interfaces, but in terms of specification languages for this purpose, there is not much agreement. In contrast, for more specific Interaction Design in the context of automated (G)UI generation, Task Models are the most widely used approach. The more recent Discourse-based approach in this context is much less widely understood. It focuses on the specification of (classes of) dialogues in contrast to tasks for modeling activities that can be performed by the user or the application (system).**

We identified a *duality* of Interaction Design according to these approaches to high-level modeling for GUI generation, which we explain and elaborate in this paper. In this course, we contrast the different models, which specify largely corresponding abstractions of Interaction Design, but from different views and with a different design philosophy. As a consequence, mutual understanding of these different modeling approaches should be improved. Based on this duality, suggestions for mutual improvements can be made, so that the tool-box of interaction designers wishing to have GUIs generated automatically from their models should become more general and flexible.

I. Introduction

An Interaction Design [1] or a Requirements Specification may include a *usage scenario*, which may be specified in UML through an Activity Diagram (see, e.g., Figure 1) or a Sequence Diagram (see, e.g., Figure 2). These different types of diagrams specify the very same scenarios but from different views, with a focus on tasks or interaction, respectively.

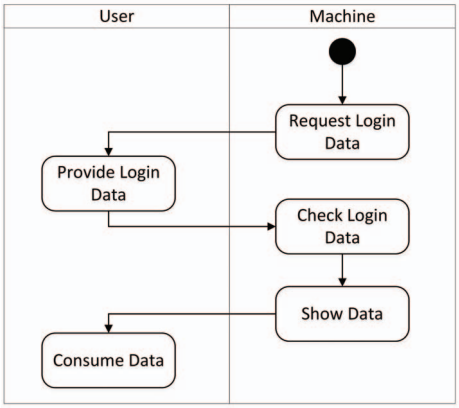


Fig. 1: Scenario Represented as a UML Activity Diagram

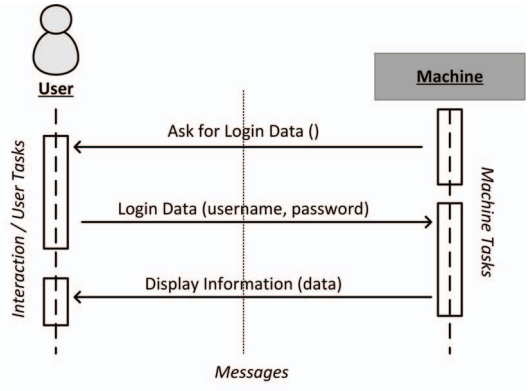


Fig. 2: Scenario Represented as a UML Sequence Diagram

We identified a similar *duality*[1] of task-based [2], [3] and discourse-based [4], [5], [6] Interaction Design for automated GUI generation. These approaches actually build on scenarios as represented through these UML diagrams, but their models are richer.[2] In this paper, we explain this duality of Interaction Design and contrast these approaches. Their current realizations have different strengths in specific respects.

From an engineering perspective, it is important to understand the conceptual relation between different modeling approaches, both when using UML and in Interaction Design. When they are better understood, it should become easier to rationally choose one under given circumstances. In addition, it is important to know about the respective strengths and weaknesses of the concrete realizations, so that they can inform each other and improvements in both realizations can be made for the tool-box of interaction designers.

The remainder of this paper is organized in the following manner. First, we discuss high-level Interaction Design for GUI generation and explain a duality of such Interaction Design according to two different approaches. Then we contrast these approaches in the context of automated GUI generation. Finally, we discuss the whole issue more generally and conclude.

[1] In Operations Research, for instance, for a given problem formulation often a *dual* problem formulation exists in the sense, that solutions of either problem formulation are the same. Another, more well-known duality is the one of light in physics.

[2] Note, that task models can be transferred into (extended) activity diagrams [7].

978-1-4799-3840-7/14/$31.00 ©2014 IEEE

II. High-level Interaction Design for GUI Generation

As it currently stands, there are two different views for specifying Interaction Design at a high level of abstraction in such a way that a GUI can be automatically generated from it. They use task-based and discourse-based models, respectively. We explain the different modeling views of these approaches for highlighting their duality, and we use a simple login example for illustrating it.

A. Different Modeling Views

Figure 1 shows an excerpt of the login example as a scenario represented through a UML Activity Diagram and Figure 2 shows the same excerpt as a UML Sequence Diagram.

Both specify a single thread of possible interaction between a (human) user and a (machine) application / system in the course of logging in. First the application asks for the *username* and *password* of the user. After these data have been provided, the application validates them, and in case of successful validation, the user is logged in and the application displays some piece of information. Such a scenario alone (without extra annotations) is insufficient for generating a user interface automatically, as shown in [8]. Still, both modeling views that we contrast here build somehow on scenarios.

The first view models *tasks* to be performed by the two actors and relationships between these tasks. According to [9], it is systematically possible to determine such tasks from a scenario specification. First, there is a task that asks for login data. It is complementary to a subsequent task that replies with such data. It is followed by a task for checking the given login data. In case of a successful login, this is followed by tasks for getting and showing the information to be displayed. The earliest and most widely used task-based approach for modeling and automatic generation of user interfaces is the ConcurTaskTrees (CTT) notation [2]. It also defines several Temporal Operators for associating such tasks with each other.

Actually, there may even be other tasks only performed by users themselves (called User Tasks in CTT). In our simple example, this may be looking up login information, e.g., on a sheet of paper. In a more intricate example, this may even involve complex cognitive processes. However, for the interaction between user and application these tasks are not relevant, and especially not for generating user interfaces from such interaction models.

The second view focuses on the communicative interaction between the two actors. It is directly visible in the UML Sequence Diagram. Still, it adds information to the so-called message passing metaphor that is essential for automatic generation of user interfaces. In our example, the message of asking becomes an `OpenQuestion` raised by the application, and its complementary reply an `Answer` with these data. After they have been validated, an `Informing` provides the information for display. The major approach along these lines defines Discourse-based Communication Models, from which GUIs can be generated automatically [4], [5], [6]. These models also define several relations for associating such communication units with each other.

As exemplified by these (implemented) approaches, both views allow modeling Interaction Design for automatic generation of user interfaces. Both add in certain and largely different ways on the underlying scenario concept to enable that. Overall, there is a duality of Interaction Design according to these views and their respective approaches.

B. Task-based Model

The task-based approach has been considered in various contributions such as the following selection. UsiXML [10] supports UI development at design-time based on task models that offer a subset of the possibilities provided by CTT and its environment. UsiComp [11] uses UsiXML for the specification of the involved models and aims to support the dynamic composition of models at run-time. HAMSTER [12] provides different representations of task models in order to obtain more compact specifications but it assumes that subtasks can be composed only through the same Temporal Operator.

Still, let us use a concrete example of a CTT model partly based on the "educational scenario v3", which is packed together with the MARIAE tool.[3] CTT models are primarily represented in a specific graphical notation, which is supported by an editor (the CTT Environment [13]) for creating and modifying such models.

Figure 3a shows the CTT model of this login example, which specifies that the user enters username and password through the `Enter Login Data` Interaction Task, which is decomposed into the `Enter Login` and the `Enter Password` Interaction Tasks. The asterisks after these two task names signify that these tasks can be *iterative*. The `Trigger Login` Interaction Task disables the `Enter Login Data` task (and its two subtasks) and passes the entered information to the `Check Login` Application Task. This behavior is modeled through the corresponding *disabling* and *enabling with information passing* Temporal Operators, respectively. The result of the `Check Login` Application Task is passed to the `Login Deny` or the `Get Data` Application Tasks, which are mutually exclusive. This is modeled through the *choice* Temporal Operator visible in the diagram and an additional pre-condition, which is not included in the diagram. In case of a successful login, some data is shown (as modeled through the `Show Data` Application Task) and logging out is possible (as modeled through the `Log Out` Interaction Task).

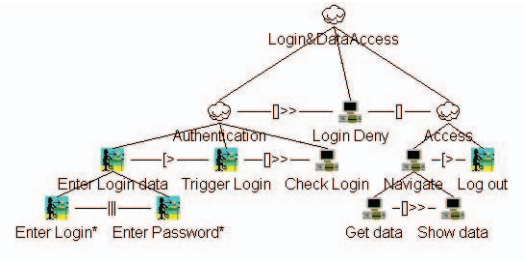
Tool support for the automated transformation of CTT models to multi-modal UIs is provided by the MARIA Environment (MARIAE) [3]. The first screen of the automatically generated GUI for the CTT login example is shown in Figure 4a.

C. Discourse-based Model

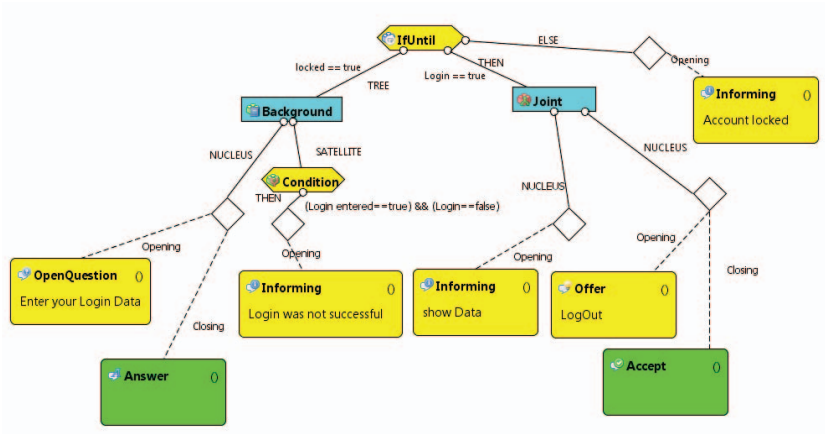
Let us also show and explain the discourse-based approach using this login example, which we modeled in such a way that it corresponds as closely as possible to the CTT model above.

Figure 3b shows our Discourse Model for this login example. The basic building blocks of Discourse Models are *Communicative Acts*, depicted through rounded rectangles. Each

[3] http://giove.isti.cnr.it/tools/Mariae/

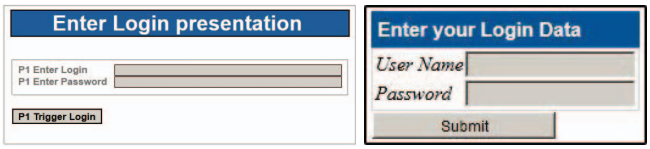


(a) CTT Model of Login Example



(b) Discourse Model of Login Example

Fig. 3: Interaction Design Models



(a) Login Screen Generated with MARIAE

(b) Login Screen Generated with UCP

Fig. 4: Automatically Generated Login Screens

Communicative Act is assigned to one of the communicating parties, yellow (light) for the application and green (dark) for the user. So-called *Adjacency Pairs* are used to model typical turn-takings (e.g., Question–Answer) in a conversation and are depicted as diamonds. Asking for the login data is modeled here as an `OpenQuestion–Answer` Adjacency Pair. The structure of the login data is specified in the so-called Domain-of-Discourse Model (not shown in this paper) and referenced through the propositional content of the `OpenQuestion` labeled `Enter your Login Data`, given here as a short-cut (for the actual query language see [6]).

Discourse Relations like `Background`, `Joint`, `IfUntil` or `Condition` can be used to link such Adjacency Pairs for modeling classes of dialogues and thus more complex flows of interaction. The `IfUntil` relation specifies that the `Tree` branch is executed until the condition of the `Then` branch (`Login==true`) is fulfilled. The `Background` and `Condition` relations are used to inform the user after an unsuccessful login attempt in this example. In case of a successful login, data is shown (modeled through the `show Data` Informing Communicative Act) and the user is enabled to logout (modeled through the `LogOut` Offer–Accept Adjacency Pair) concurrently (modeled through the `Joint` relation).

Strictly speaking, however, the models in Figures 3a and 3b are different with respect to handling the usual possibility of restricting login attempts (e.g., to three) and to lock the account thereafter. The GUI generated through MARIAE from the CTT model allows an unlimited number of tries, but actually this is an iteration that includes the whole `Login&DataAccess` (although this is not explicitly represented in this model). The Discourse Model, however, represents this case explicitly through the condition `locked==true` and the ELSE branch, while there is no iteration modeled here for the overall discourse.

Tool support for the creation / modification of such models and automated transformation to GUIs is provided by the Unified Communication Platform [6], more precisely its part UCP:UI.[4] The first screen of the automatically generated GUI for the Discourse-based Communication Model login example is shown in Figure 4b.

III. Contrasting these Interaction Design Approaches in the Context of GUI Generation

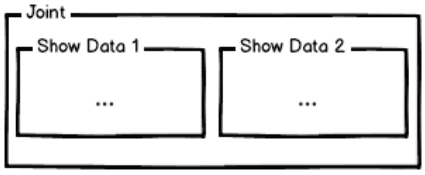
While these approaches are dual to each other, they have certain differences, which we highlight and discuss with the purpose of contrasting them.

A. High-Level Models

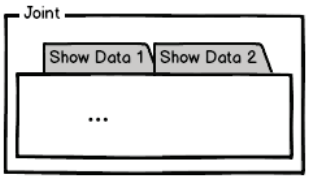
The source models of both these approaches are on the highest level of abstraction of the Cameleon Reference Framework [14], i.e., the Tasks & Concepts Level. More specifically, a major goal is to obtain GUIs for different devices (especially with large difference in screen size) through automatic generation from the same source model. This tailoring for a specific device in the course of generation includes the use of different widgets and different layout, and even an adaptation of the presentation units.

Both approaches (CTT/MARIAE and UCP:UI) support tailoring at the level of device-specific widgets and layouts. There is a difference, however, with respect to adaptation of the presentation units. Figure 5 shows two mock-ups presenting the same content in two different ways. These offer alternatives for device tailoring of the GUI. The first one, Figure 5a, shows all the information in one (typically large) screen. The second one, Figure 5b, splits the presentation into two (smaller)

[4]More information can be found at http://ucp.ict.tuwien.ac.at.



(a) Common Presentation Unit



(b) Split Presentation Units

Fig. 5: GUI Mock-ups of Alternative Presentations

screens. Of course, this kind of tailoring is not restricted to displaying information only, it can be done with any kind of interaction specified.

The current version of CTT/MARIAE allows specifying such content to be displayed in two presentation units, with the *interleaving* Temporal Operator, but there is no tool support available for automatic selection of the best fitting rendering alternative in the course of automatic generation of a GUI. However, if a desktop-oriented presentation is provided, it is still possible according to [15] to automatically adapt it into a mobile version through the *semantic redesign method*, which provides adaptation to the size of the device at hand.

In contrast, UCP (UCP:UI) supports automatic tailoring for a given device through optimization in the course of automatic GUI generation, exploring the space of such alternatives and determining an optimum for given optimization criteria [16]. One of the relations for modeling that allow for the alternatives illustrated in Figure 5 is the `Joint` relation (for more details, see [6]).

B. Operators and Relations

Both approaches have defined operators/relations between their basic building blocks described above. In CTT, these relations are *Temporal Operators*, in UCP *Discourse Relations*. Temporal Operators are partly based on the LOTOS operators [17]. Discourse Relations distinguish between relations based on Rhetorical Structure Theory (RST) [18] (e.g., `Elaboration`) and Procedural Constructs (e.g., `IfUntil`). The currently available Temporal Operators partly overlap with the Discourse Relations (e.g., both include a sequential relation each), but both approaches also provide operators/relations that cannot be mapped directly to the other approach.

One example is the `Choice` Temporal Operator, which allows the user to select one branch and disables the other branches as soon as the user selection has been made. There is no directly corresponding relation in UCP. The most similar relation in UCP is `Alternative`, where the other branches are only disabled as soon as the first branch is terminated. In contrast to `Choice`, this does not necessarily have to be the initially selected branch.

In RST, the `Elaboration` relation is used to provide additional information about a situation or some element. Its *satellite* provides the additional information for its *nucleus*. RST actually provides a whole list of specific relation types of this kind, e.g., set/member or whole/part. For the `Elaboration` relation in UCP, it is defined that the satellite part of a discourse is only available under the condition of a selected element in the nucleus part. In both cases (with and without available satellite), the nucleus part is available. The Procedural Semantics of the `Elaboration` relation in UCP is shown in Figure 6 as a UML statemachine. It allows for executing either the nucleus only, or nucleus and satellite in parallel, depending on the internal state of the corresponding communication party (e.g., a presentation of products of a specified category can only be provided, if this category is specified through the internal state of the corresponding communication party). This dependency is modeled with the choice element of UML in the statemachine. Finishing one of these branches terminates the execution of the whole `Elaboration` relation.

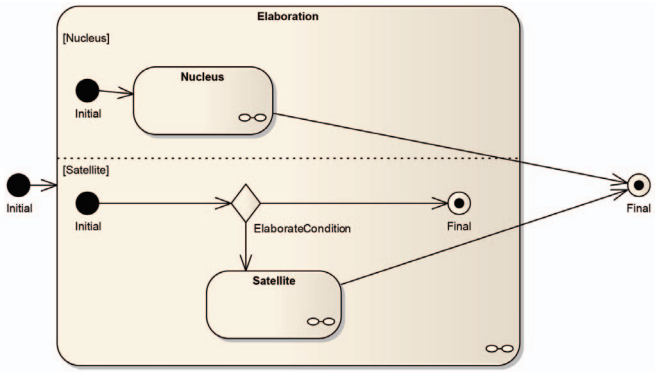
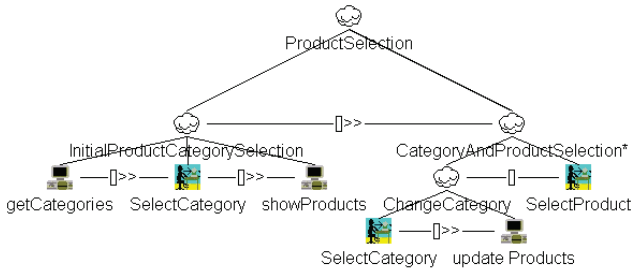


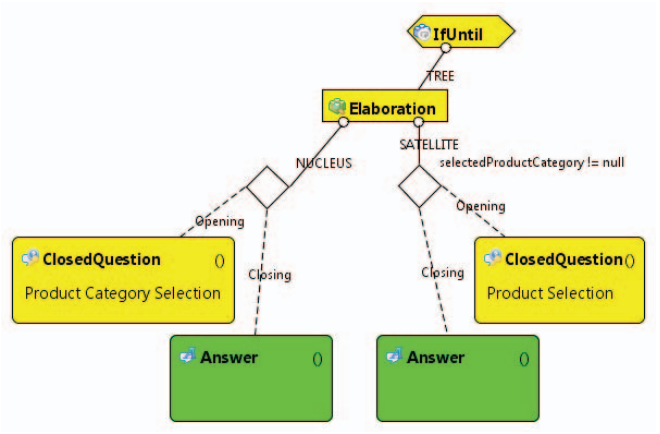
Fig. 6: Procedural Semantics of the UCP Elaboration Relation [19]

Figure 7b shows an application of this construct for modeling the dependency between a selected Product Category and the possibility of selecting a Product from it. The corresponding GUI part is sketched in Figure 8a for the case that no Product Category has been selected yet, and in Figure 8b for the case that one has been selected. The `IfUntil` on top of the Elaboration construct specifies the repeatability of these two selections.

In CTT, no dedicated Temporal Operator for such a relation is currently defined, but the corresponding behavior can be modeled through a combination of available operators. The corresponding CTT Model is shown in Figure 7a. The left sub-tree, with the Abstract Task `InitialProductCategorySelection` as root node, models the initial product category selection. The Application Task `getCategories` is needed to get the available product categories from a Web Service. The Application Task `showProducts` presents the products of the selected product category, after the user has selected one in the `SelectCategory` Interaction Task. The right sub-tree, with the Abstract Task `CategoryAndProductSelection` as root node, models the interaction after the initial product category selection. It concurrently offers the two tasks,

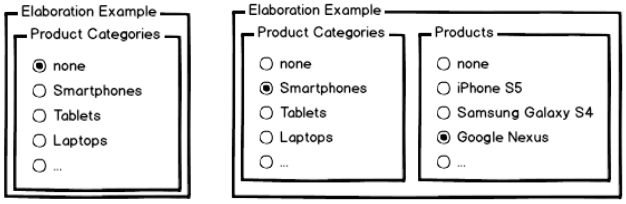


(a) CTT Model for Product Selection



(b) Discourse-based Communication Model for Product Selection

Fig. 7: Interaction Models for ProductSelection



(a) Selection of Product Category

(b) Selection of Product Category and of Product

Fig. 8: GUI Mock-up for Product Selection

`ChangeCategory` and `SelectProduct`, allowing either the selection of a new product category, or of a product of the previously selected product category. These two tasks are related via a `choice` Temporal Operator, which implies that only one of them can be finished. The Abstract Task `ChangeCategory` is decomposed into an Interaction Task for selecting a category, and an Application Task for updating the available product list. These two tasks are executed sequentially. After the product list has been updated, the iterative Abstract Task `CategoryAndProductSelection` is repeated and the `SelectProduct` task again allows the user to select a product from the previously retrieved product list (through `updateProducts`).

C. Coupling between GUI and Application Logic

These contrasted approaches employ different ways of coupling a generated GUI with a given application logic. In MARIAE it is possible to bind Web Service operations (that implement each a part of the application logic) with Application Tasks. The binding is then automatically mapped in the generated user interface, i.e., the binding between the Web Services and the interactive application is encoded in the generated Java Server Pages (JSP).

In contrast, UCP provides a dedicated run-time environment, which distinguishes between a service provider (the application) and a service consumer (the user interface), according to the Service-oriented Architecture notions. The Discourse-based Communication Model defines the interface between these two services [19]. This interface decouples the application and the user interface and makes it easy to exchange either part. Therefore, the application does not depend on a specific user interface. This has the advantage that the same application can be used from different device-tailored user interfaces. The UCP run-time architecture is based on the well-known Model-View-Controller (MVC) architecture [20], with its decoupling of different concerns and the resulting software properties. The application logic may still be implemented through Web Services, but this integration is more intricate.

IV. Discussion and Future Work

CTT/MARIAE and UCP are contrasted above in the context of GUI generation only, but both approaches also provide further features. For example, CTT/MARIAE supports also multi-modal UI development [21], in particular for combining graphical and vocal UIs. In principal, Discourse-based Communication Models do not depend on a specific modality and can also be used for the semi-automatic generation of multi-modal UIs [22]. However, this is not (yet) integrated in the current implementation of UCP.

Another interesting field of research are nomadic UIs [23]. While this is a topic in CTT, in UCP this is currently out of scope.

In the spirit of Interaction Design in general [1], an iterative process is defined for developing applications with UCP, which even makes use of the automatically generated GUIs for prototyping [24].

CTT/MARIAE can be used for generating service front-ends using JSPs, which define the composition of the access to the Web Services [25]. An interesting feature of the UCP run-time in the context of service composition is its additional support of machine-machine interaction. In particular, UCP supports the composition of interactive applications based on existing, smaller applications. For example, a Travel Booking application can be composed of a Flight Booking application

and a Hotel Booking application [19]. This facilitates the creation of a new application based on existing ones.

Contrasting the realizations of these approaches in CTT/MARIAE and UCP revealed certain differences that are not inherent to their task-based vs. discourse-based approaches. These realizations may inform each other for further improvements of the tool-box of interaction designers.

An open question for future research is, why designers in practice would choose one or the other conceptual modeling approach. Of course, the available tool support for a concrete method will also be important for practical application.

V. Conclusion

We identified a duality of Interaction Design according to two conceptual approaches in the context of automated (G)UI generation, task-based and discourse-based Interaction Design. While we investigated and showed this duality with two concrete approaches, it is fundamental. When such a design is specified in one approach, a dual formulation in the other approach exists in the sense, that the specified interaction is (essentially) the same. They just differ in terms of the modeling concepts, in one case the focus is on the tasks to accomplish, in the other one on the communicative interaction between user and system. We conjecture, that this identified duality exists even apart from automated (G)UI generation, for Interaction Design in general.

References

[1] J. Preece, Y. Rogers, and H. Sharp, *Interaction design: beyond human-computer interaction*, 3rd ed. John Wiley & Sons, 2011.

[2] F. Paternò, C. Mancini, and S. Meniconi, "ConcurTaskTrees: A diagrammatic notation for specifying task models," in *Proceedings of the IFIP TC13 Sixth International Conference on Human-Computer Interaction*, 1997, pp. 362–369.

[3] F. Paternò, C. Santoro, and L. D. Spano, "MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments," *ACM Trans. Comput.-Hum. Interact.*, vol. 16, pp. 19:1–19:30, November 2009. [Online]. Available: http://doi.acm.org/10.1145/1614390.1614394

[4] J. Falb, H. Kaindl, H. Horacek, C. Bogdan, R. Popp, and E. Arnautovic, "A discourse model for interaction design based on theories of human communication," in *Extended Abstracts on Human Factors in Computing Systems (CHI '06)*. ACM Press: New York, NY, 2006, pp. 754–759.

[5] C. Bogdan, J. Falb, H. Kaindl, S. Kavaldjian, R. Popp, H. Horacek, E. Arnautovic, and A. Szep, "Generating an abstract user interface from a discourse model inspired by human communication," in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS-41)*. IEEE Computer Society Press, January 2008.

[6] R. Popp, D. Raneburger, and H. Kaindl, "Tool support for automated multi-device GUI generation from discourse-based communication models," in *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '13)*. New York, NY, USA: ACM, 2013.

[7] J. Brüning, A. Dittmar, P. Forbrig, and D. Reichart, "Getting SW engineers on board: Task modelling with activity diagrams," in *Engineering Interactive Systems – EIS 2007 Joint Working Conferences, EHCI 2007, DSV-IS 2007, HCSE 2007. Selected Papers*, ser. Lecture Notes in Computer Science, vol. 4940. Springer, 2008, pp. 175–192.

[8] H. Kaindl, R. Popp, and D. Raneburger, "Automated generation of user interfaces: Based on use case or interaction design specifications?" in *Proceedings of the 7th International Conference on Software Paradigm Trends (ICSOFT'12)*, S. Hammoudi, M. van Sinderen, and J. Cordeiro, Eds. SciTePress, July 2012, pp. 303–308.

[9] H. Kaindl and R. Jezek, "From usage scenarios to user interface elements in a few steps," in *Proceedings of CADUI'02*, C. Kolski and J. Vanderdonckt, Eds. Kluwer, 2002, pp. 91–102.

[10] J. M. Vanderdonckt, "Model-driven engineering of user interfaces: Promises, successes, and failures," in *Proceedings of 5th Annual Romanian Conf. on Human-Computer Interaction*. Matrix ROM, Sept. 2008, pp. 1–10.

[11] A. García Frey, E. Céret, S. Dupuy-Chessa, G. Calvary, and Y. Gabillon, "UsiComp: an extensible model-driven composer," in *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '12)*. New York, NY, USA: ACM, 2012, pp. 263–268. [Online]. Available: http://doi.acm.org/10.1145/2305484.2305528

[12] E. Barboni, J.-F. Ladry, D. Navarre, P. Palanque, and M. Winckler, "Beyond modelling: An integrated environment supporting co-execution of tasks and systems models," in *Proceedings of the 2Nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*. New York, NY, USA: ACM, 2010, pp. 165–174. [Online]. Available: http://doi.acm.org/10.1145/1822018.1822043

[13] G. Mori, F. Paternò, and C. Santoro, "CTTE: Support for developing and analyzing task models for interactive system design," *IEEE Transactions on Software Engineering*, vol. 28, pp. 797–813, 2002.

[14] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt, "A unifying reference framework for multi-target user interfaces," *Interacting with Computers*, vol. 15, no. 3, pp. 289–308, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0953543803000109

[15] F. Paternó and G. Zichittella, "Desktop-to-mobile web adaptation through customizable two-dimensional semantic redesign," in *Human-Centred Software Engineering*, ser. Lecture Notes in Computer Science, R. Bernhaupt, P. Forbrig, J. Gulliksen, and M. Lrusdttir, Eds. Springer Berlin / Heidelberg, 2010, vol. 6409, pp. 79–94. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16488-0_7

[16] D. Raneburger, R. Popp, S. Kavaldjian, H. Kaindl, and J. Falb, "Optimized GUI generation for small screens," in *Model-Driven Development of Advanced User Interfaces*, ser. Studies in Computational Intelligence, H. Hussmann, G. Meixner, and D. Zuehlke, Eds. Springer Berlin / Heidelberg, 2011, vol. 340, pp. 107–122.

[17] T. Bolognesi and E. Brinksma, "Introduction to the ISO specification language LOTOS," *Comput. Netw. ISDN Syst.*, vol. 14, no. 1, pp. 25–59, Mar. 1987. [Online]. Available: http://dx.doi.org/10.1016/0169-7552(87)90085-7

[18] W. C. Mann and S. Thompson, "Rhetorical Structure Theory: Toward a functional theory of text organization," *Text*, vol. 8, no. 3, pp. 243–281, 1988.

[19] R. Popp, "A unified solution for service-oriented architecture and user interface generation through discourse-based communication models," Vienna University of Technology, Vienna, Austria, Doctoral Dissertation, 2012.

[20] R. Popp, H. Kaindl, and D. Raneburger, "Connecting interaction models and application logic for model-driven generation of Web-based graphical user interfaces," in *Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC 2013)*, 2013.

[21] M. Manca and F. Paternò, "Supporting multimodality in service-oriented model-based development environments," in *Proceedings of HCSE*, 2010, pp. 135–148.

[22] D. Ertl and H. Kaindl, "Semi-automatic generation of multimodal user interfaces for dialogue-based interactive systems," in *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI'12)*. New York, NY, USA: ACM, 2012.

[23] F. Paternò, C. Santoro, and A. Scorcia, "User interface migration between mobile devices and digital TV," in *Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams (HCSE-TAMODIA 2008)*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 287–292.

[24] D. Raneburger, H. Kaindl, R. Popp, V. Šajatović, and A. Armbruster, "A process for facilitating interaction design through automated GUI generation," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC'14)*, 2014.

[25] F. Paternó, C. Santoro, and L. D. Spano, "Engineering the authoring of usable service front ends," *Journal of Systems and Software*, vol. 84, no. 10, pp. 1806 – 1822, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121211001257