An Interactive Process Meta Model for Runtime User Interface Generation and Adaptation

Thomas Schlegel
University of Stuttgart
Universitätsstr. 38, Stuttgart, Germany
Thomas.Schlegel@vis.uni-stuttgart.de
+49 711 7816 209

ABSTRACT

Complex and distributed interactive systems today – in this case mainly in production and supply chain management – rely strongly on defined processes but as well on flexible interaction and dynamic adaptation. We describe an interactive process model that allows recognizing and deriving interactions with users on runtime. The model can be dynamically adapted to fit new requirements or offer additional interactions in specific contexts.

Author Keywords
Interactive Processes, Process Model, Production Process Control, Runtime User Interface Generation

ACM Classification Keywords
H.5.2 User interface management systems (UIMS), H.5.3 Theory and models, H.5.m Miscellaneous, C.3 process control systems, H.4.1 workflow management

INTRODUCTION

Industry has experienced a strong development in the direction of product customization (flexibility in production and products) and globalization (flexibility in organizational structures and local differences), leading to instable processes and user interfaces with frequent adaptations and necessary changes. Industrial systems, in production and supply chain, are developing today from monolithic systems towards a complex set of interconnected systems like Service Oriented Architectures (SOA, [2]). On the one hand, these complex industrial systems like networked production systems or supply networks [13] rely on defined processes to ensure proper and controlled execution of business processes in production companies. On the other hand, build-to-order, Mass Customization [1] and rapid reconfiguration (e.g. [3]) of the factory require flexibility and adaptability of processes that go beyond what is possible today.

Therefore, we experience today **interaction with a "system of systems"** [6], integrating different users and different systems in a dynamic **Multi-Stakeholder Multi-System (MS2) System**, which adds a new level of complexity and necessary flexibility to today's networked industrial systems and their interaction layer. Interactions with such a complex system occur at all levels of the processes. Hence, processes changing dynamically on runtime lead to the requirement of adapting or generating their adjacent user interface even during their execution.

To achieve this flexibility in process modeling and execution, we shortly motivate and present the concepts of interaction in this field, explain the necessary interactive process model integrating static and dynamic aspects in one model and show a first user interface prototype for executing, controlling, adapting and testing interactive processes and interacting with the user based on this model.

INTERACTIVITY IN COMPLEX SYSTEMS

We studied e.g. the production of dishwashers at BSH, where variability is expected to be low compared e.g. to special machines. Even in this company, 1000 variants exist and changes to the process have to be applied with high frequency depending on product changes, location and many other factors, like distributing formerly integrated interaction tasks of the workers.

Users and even software developers working on such an MS2 System normally only perceive and access some local parts of the system – often without a detailed mental model of the whole system, while the system itself may additionally be changing and evolving without notification. Full oversight and control over the processes for one stakeholder or organization is often not possible, if the system is spanning across organizational and technological borders. E.g. an operator in a production cell executes some steps of a complex production process for producing a car part that is assembled by other workers in a different factory of a different company, still participating in the same overall process. This makes decentralization and flexible interaction a key issue for these systems.

The need for integrating interaction with processes has been recognized by the Workflow community, e.g. leading to additional standards like BPEL4People [4] and WS-Human Task [5], which emphasize the importance of the user in process execution. But even these specifications do not fully cover dynamically and decentrally changing interactions in interactive processes. To overcome this issue, interactions cannot be embedded or occur in processes as **pre-programmed or interaction-based user interfaces** with predefined dialogs and functions anymore.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

Flexibility is offered by **model-based user interfaces** (e.g. [11] and especially for workflows [7]), which do not rely on statically predefined functions and dialogs anymore. In an MS2-System with dynamically changing processes, even **dynamic model-based user interfaces** are needed, which additionally allow changes to the underlying workflows and interactions by users on runtime and offer automatable creation of interactions and dialogs based on changing contexts. Disadvantages of such a flexible system include that usability tests of model-based and runtime-generated systems are not fully possible and that the system and models become complex and partially unpredictable and make software development more expensive and complex. Although, for decentralized and runtime-changing infrastructures there is currently no second option available.

A SEMANTIC PROCESS MODEL

Model- and context-based generation means that the underlying process model has to provide the semantics and abilities for generating user interfaces and also mechanisms for safely adapting the process model to new needs arising while the system is already in production state. To achieve a decentralized execution of interactive processes, the underlying process model has to provide the semantics and the dynamics of the processes and the adjacent interactions.

The Process Flow Model Aspect (horizontal aspects)

Common process models like Event-Driven Process Chains (EPC) with BPEL [8] and UML Activity Diagrams [10] focus on the flow aspect of the process. Process steps with their sequence and dependencies are described in a graphical or other language-based model. The **Process Flow Model** contains these horizontal, dynamic aspects of processes necessary for correct execution. It also provides the dependencies between different elements to be able to create interactions, e.g. dialogs which consider causality of inputs. Flow relations determine where data, events and activation have to flow (process definition) or currently flow (process instance).

The Structural Model Aspect (vertical aspects)

While the flow aspect shows the sequencing, the Structural Model defines the vertical, static aspects of a process. Data type, activity types, event types and many other classifications also form an integral part of an executable and self-dependent process model.

Object-oriented inheritance mechanisms are used to create specializations of existing types of actions and interactions in order to provide rich semantics for every component in the model, like process steps, data or dialog elements. Aggregation allows for creating complex processes, data types and semantic groups of interactions, determining their meaning and integration. The inheritance mechanism allows creating specializations of complex, aggregated processes (activities) and their process steps. Figure 1 shows how the concept works for such a process: Activity A consists of Activities x, y and z. Activity B is created as a specialized process (child) of A by inheritance. While x' and z' are created by (sub-)inheritance, i.e. inherited without changes from A (gray, dashed line), y' is specialized from y as a full component (black, continuous line). Through instantiation, process instances are defined by and created from processes (types) and then executed decentrally according to the process specification, e.g. process instance c, which is an object-oriented instance of Activity C, with its components being instances of the Activities x'', y'' and z''.

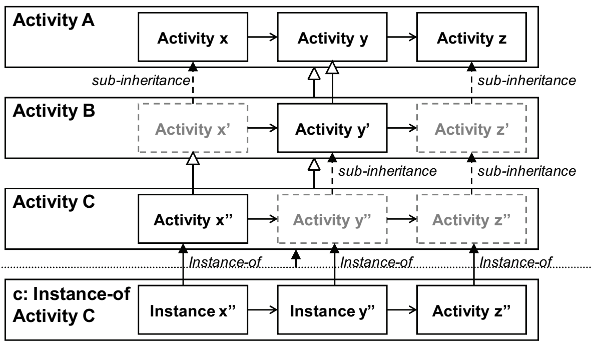

Figure 1. Specialization of aggregated activities by inheritance.

To execute interactive steps, dialog elements can be instantiated from type-compatible interaction element types. For such **explicit interactions** [12], the classification can be used to determine the most specialized interaction element available for the type of interaction requested. For **implicit interactions** [12], it can be used to determine the correct interaction elements to be used to gather the information needed from the users for the current process execution.

Integrated Semantic Process Model

One significant contribution to flexible interactive process systems is that we provide an integrated process and interaction concept, which offers the advantages of object-orientation like runtime behavior, type-safe extension and generation also for interactive processes.

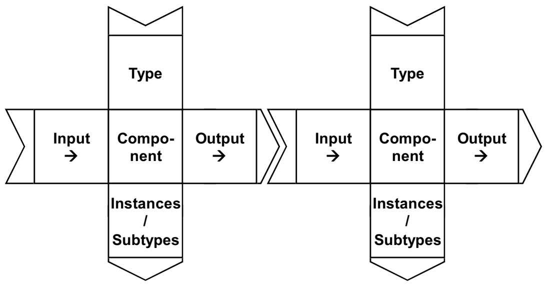

Figure 2. Integrated model containing the structural (vertical, semantic) and dynamic (horizontal, flow) aspects of processes.

To execute interactive processes in a decentralized environment, the integration of both aspects into one model framework is vital. This is done using different component and relation types for static aspects like inheritance / classification and aggregation than for the dynamic aspects like flow/activation, input and output. Figure 2 shows how two components are connected horizontally via their Input and Output interfaces. Inheritance and aggregation offer a vertical connection.

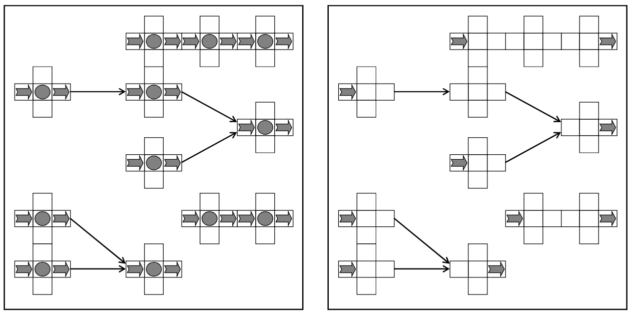

Figure 3. Interface of the component.

For each complex/aggregated component the data needed or produced can be determined semantically by identifying the "interface" of the component. Figure 3 shows how the input and output of every component (e.g. process step) can be connected either internally (arrows between components) or be aggregated as interface for the whole component (right), showing input necessary and output delivered to the next component in the process.

USER INTERFACE CREATION FROM THE MODEL

Once the elements required for execution of the process (or sub-process) have been identified, for each element it can be validated if the necessary data is available. If not, or if an interaction is explicitly foreseen to gather the data from the user [13], the contained User Information Model (UIM) is used to determine the type of the element. With this information, each peer in the system can check the User Interface Element Model (UIEM) to determine if an interpretation of the element is available for the defined context of use. The Dashboard Model is then used to integrate the elements correctly into the overall dialog set.

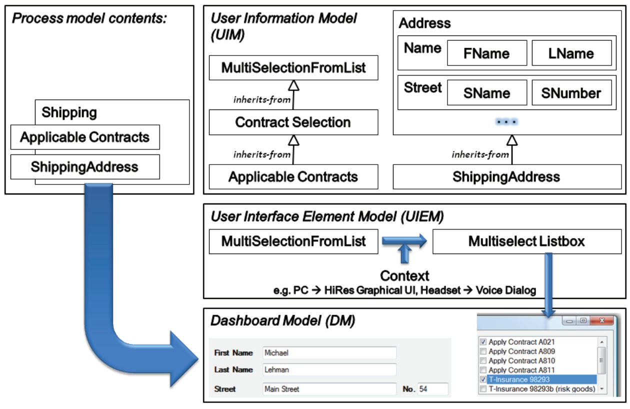

Figure 4. Example for the model-based execution of an interactive shipping process using UIM, UIEM and DM for model interpretation.

In the example in figure 4, we describe a partial view of a typical shipping process in logistics. It requires the shipping address and "Applicable Contracts" (insurance etc.) for the adjacent order and transport. From the UIM, the type can be derived. While for "Applicable Contracts" there is no interaction specification available in the UIEM, it is a specialization of MultiSelectionFromList for which the UIEM provides an implementation.

For ShippingAddress, the UIM provides all elements of it by resolving it from Address. For all partial elements like LName an implementation is derived from the UIEM. The semantic group ShippingAddress can be displayed together with the MultiselectListbox implementation of "Applicable Contracts". MultiselectListbox has been determined to be a valid implementation of MultiSelectionFromList, which in turn is the (still) most specialized generalization of "Applicable Contracts" in the UIM.

To create necessary dialogs for gathering missing information, the type of a missing element (complex or simple) is mapped via reference directly to a corresponding element in the UIM. Each possible realization of an element is listed in the UIEM and references its source element in UIM. For a specific context (e.g. WinFormsApp), the correct renderer in UIM can be identified.

PROTOTYPE FOR MODELING AND EXECUTION

When interactive processes are executed and adapted in a decentralized system environment, it is necessary to provide a modeling and control interface, which is context-oriented. The challenges of this kind of process models are the size and the connections of the model. Each process component has sub-components like input and output and a type hierarchy above and maybe also below as well as flow connections to other components at the same level. A graphical notation like BPMN[9], EPC[8] or UML [10] needs a lot of space, has low content editing abilities and already uses two dimensions without providing the static (vertical) aspects of the model. Therefore we are using a dialog based, navigational form of working with the process model, including execution, editing and generation.

Figure 5 shows a prototype of a dialog-based interface for process execution and editing using an object-oriented process and interaction model. It offers runtime inspection, editing and execution of the interactive process. In the center, information about the currently focused component is displayed and can be edited where possible. Each component has a unique key (identity) and carries a payload and comments with it. The left part contains all incoming data and trigger relations with their types, components and indication if data is missing. The right part contains all results and triggers generated by the current component.

Where possible, elements of the current component can be edited. Also, additional components can be connected using all available relation types. New components can be created directly as input, output or part of the current component in focus.

Navigation is possible **upwards** to the type layer and along the aggregation and inheritance relations, **downwards** to the sub-elements via different relation types like

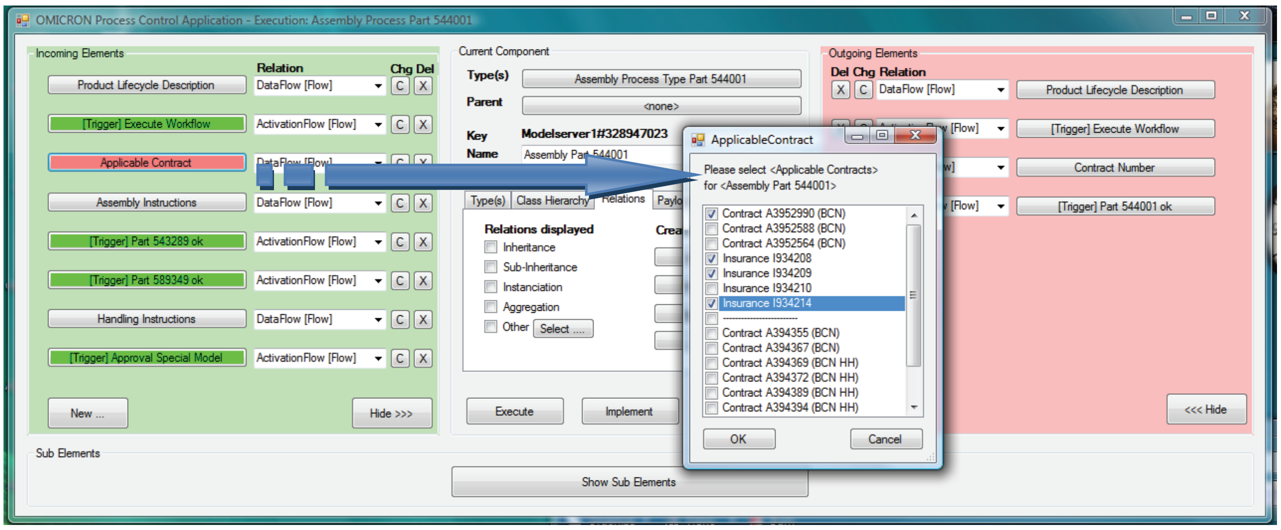

Figure 5. Integrated structural (vertical, semantic) and dynamic (horizontal, flow) model.

instantiation, inheritance, aggregation and others, depending on the required aspect, **leftwards** to the predecessors of the same and next sub-level along flow relations of data and activation and **rightwards** to the successors of the same and next sub-level along flow relations of data and activation.

Additional interfaces e.g. serve for browsing the hierarchy more easily in a tree-like view. The multi-relational hierarchy of the components is that of a Directed Acyclic Graph (DAG) for each relation type used, which requires aspect-oriented (mainly relation type-oriented) navigation.

The prototype has been implemented in C# using WinForms as generator target. An implementation for WPF is currently created in addition. The screenshot also shows how missing information in the process executed is added interactively: “Applicable Contract” input is not filled by the predecessors. Clicking on the button for this data required to execute “Assembly Part 544001”, a dialog opens that offers the user to interactively enter the contracts to be used with this order / process instance.

CONCLUSION AND OUTLOOK

This article has presented a concept for an interactive process model execution and modeling prototype for distributed process execution and derivation of interaction elements from the semantic structures in the model.

Further research will be carried out on model extension and specification to allow for runtime user interface generation also for complex types and extend the modeling capabilities in addition to a better integration with existing standards and concepts in an M2S System. This also includes user interface generator components and extended process modeling capabilities.

REFERENCES

1. Anderson, D.M. *Build-to-Order & Mass Customization,* Cambria CIM Press (2003).
2. Erl, T. *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall (2005).
3. Harrison, R., Colombo, A. W. Collaborative automation from rigid coupling toward dynamic reconfigurable production systems, *Proc. 16th IFAC World Congress 2005* (2006).
4. IBM et al. *WS-BPEL Extension for People (BPEL4People)*, Version 1.0, (June 2007).
5. IBM et al. *Web Services Human Task (WS-HumanTask)*, Version 1.0, (June 2007).
6. Jamshidi, M. System of systems engineering – New challenges for the 21st century, *IEEE Aerospace and Electronic Systems Magazine*, 23, 5 (2008), 4-19.
7. Guerrero García, J., Lemaigre, C., González Calleros, J.M., Vanderdonckt, J. Model-Driven Approach to Design User Interfaces for Workflow Information Systems, Journal of Universal Computer Science, vol. 14, no. 19 (2008), 3160-3173
8. Kopp, O., Unger, T., Leymann, F. Nautilus Event-driven Process Chains Syntax, Semantics, and their mapping to BPEL, *Proc. GI EPK* (2006).
9. OMG *Business Process Modeling Notation Specification*, OMG Final Adopted Specification, dtc/06-02-01, (2006).
10. OMG *Unified Modeling Language (UML) Superstructure* Version 2.1.1. formal/07-02-05, (2007).
11. Pinheiro da Silva, P., Griffiths, T., Paton, N. Generating User Interface Code in a Model Based User Interface Development Environment, *Proc. Advanced Visual Interfaces 2000* (2000), 155-160.
12. Schlegel, T. Object-oriented interactive processes in decentralized production systems. *Proc. HCI International 2009* (2009).
13. Schlegel, T., Kirn, S. Interacting with the Supply Swarm Towards an Interactive and Visual Management of Supply Webs. *Proc. IEEE INDIN* (2009).