Modelling the Process of Process Execution: A Process Model-Driven Approach to Customising User Interfaces for Business Process Support Systems

Udo Kannengiesser(✉), Richard Heininger, Tobias Gründer, and Stefan Schedl

Metasonic GmbH, Münchner Straße 29 – Hettenshausen, 85276 Pfaffenhofen, Germany
{udo.kannengiesser,richard.heininger,tobias.gruender,
stefan.schedl}@metasonic.de

Abstract. This paper presents a process-driven approach for developing the user interfaces (UIs) of business process execution frontends. It allows customising the UIs to the needs of individual users and processes. The approach is based on viewing UI behaviour as a process that can be modelled and executed in the same way as the core process: as a sequence of steps, each of which is associated with a business object that describes the UI content in terms of the information displayed to the user. As both the UI process and the core process are run on the same business process engine, the two processes can interact smoothly using existing backend functionalities. The approach is demonstrated using a manufacturing scenario where shopfloor workers are provided with simple UIs on mobile devices to support the execution of a production process.

Keywords: Model-driven design · Customised user interfaces · Business process support systems · Subject-oriented Business Process Management (S-BPM)

1 Introduction

The design of user interfaces (UIs) is widely regarded as a critical factor for the acceptance of IT systems by users as well as for the acquisition of these systems by potential buyers. Although many system vendors today employ user experience (UX) designers to make UI design more effective, the fundamental problem remains that the space of possible UI designs can be too large to satisfy all users with a single solution. On the other hand, customising UIs is often tedious and costly, making many vendors opt for standardising their user interfaces, and limiting any customisations to broad user categories such as "simple users" and "advanced users".

This approach is also followed by most vendors of business process support systems. They usually have a single UI that has the same look and feel for most of its users. However, as customers become more demanding and the scope of these systems becomes broader to include more domains and applications [18], the ability to customise UIs to wider ranges of user preferences and skills becomes an important competitive advantage. One example includes workflow management applications in the industry 4.0 domain that seamlessly integrate business processes with shopfloor processes.

R. Schmidt et al. (Eds.): BPMDS/EMMSAD 2016, LNBIP 248, pp. 34–48, 2016.
DOI: 10.1007/978-3-319-39429-9_3

Here, workers with sometimes limited IT skills need to be provided with a very simple UI in order to adopt and master a given process support system.

This paper addresses the problem of customising the UIs of workflow systems by focussing on the "process of process execution" – i.e. the sequence of tasks to be performed, conjointly by a user and a user interface, necessary to execute the actual business process (called the "core process"). Specifically, the process of process execution (here called the "UI process") is modelled and executed in the same way as the core process. Each of the tasks in the UI process is associated with data (called business objects) composing the content and appearance of the UI. The UI process is run on the same execution engine as the core process, readily allowing for the dynamic interconnection between the two processes at runtime. The existing method of Subject-oriented Business Process Management (S-BPM) [6] provides a uniform modelling formalism for both the UI process and the core process. The approach can be seen as a process model-driven method for customising UIs to the needs of different users, devices and core processes to be executed.

The paper is structured as follows: Sect. 2 introduces the relevant foundations of S-BPM modelling. Section 3 describes how UIs can be modelled as the processes of (core) process execution. It shows how the S-BPM notation and a tool suite for S-BPM modelling and execution, the Metasonic Suite (www.metasonic.de/en), can be used for defining the UI workflow and UI content, and establishing the connection between UI and core processes. Section 4 demonstrates the use of our approach in a shopfloor scenario developed within an ongoing EU FP7 research project (www.so-pc-pro.eu). Section 5 concludes the paper with a summary of the approach, including a brief discussion of how it is the result of applying a "good theory" in practice, in reference to Lewin's quote that "nothing is more practical than a good theory" [10].

2 The S-BPM Approach to Business Process Modelling

Subject-oriented Business Process Management (S-BPM) is a method and notational approach to modelling and executing processes in a decentralised way. In S-BPM, processes are understood as interactions between process-centric roles (called "subjects"), where every subject encapsulates its own behaviour specification [6]. Subjects coordinate their individual behaviours by exchanging messages. The S-BPM approach is based on extensions of the Calculus of Communicating Systems by Milner [12] and Communicating Sequential Processes by Hoare [7]. Abstract State Machines (ASM) [1] are used as the underlying formalism to allow instant transformation of S-BPM models into executable software. S-BPM mostly targets those applications where a stakeholder-oriented, agile approach to business process management is preferred over more traditional methods based on global control flow. An increasing number of field studies demonstrate the benefits of S-BPM [5].

Based on the strong emphasis on the role concept and on the communication between roles, S-BPM shares some similarities with Role-Activity Diagrams (RAD) [13], UML communication diagrams, and the DEMO methodology [2]. However, there are also a number of significant differences with respect to these approaches. For instance, S-BPM has rigorously defined execution semantics, allows asynchronous communication, and supports end-user involvement in process modelling based on the simplicity of the S-BPM modelling constructs.

S-BPM models include two types of diagrams: a Subject Interaction Diagram (SID) specifying a set of subjects and the messages exchanged between them, and a Subject Behaviour Diagram (SBD) for every subject specifying the details of its behaviour. SBDs describe subject behaviour using state machines, where every state represents an action. There are three types of states in S-BPM: "receive states" for receiving messages from other subjects, "send states" for sending messages to other subjects, and "function states" for performing actions (typically operating on business objects) without involving other subjects. Examples of a SID and a SBD are shown in Figs. 1 and 2, respectively. They represent parts of a production process implemented in a Slovakian manufacturing company (in this paper referred to as "Company A") within the EU FP7 project SO-PC-Pro. Here, the SID in Fig. 1 includes subjects that coordinate (via messages directed to one another) to prepare the actual manufacturing subprocess. The SBD in Fig. 2 represents the internal behaviour specification of the subject "Work Task Preparation". The colours of the different states in the SBD indicate their types: green for receive states, yellow for function states, and red for send states. State transitions are represented as arrows, with labels indicating the outcome of the preceding state. For more details about the S-BPM notation readers may refer to Fleischmann et al. [6].

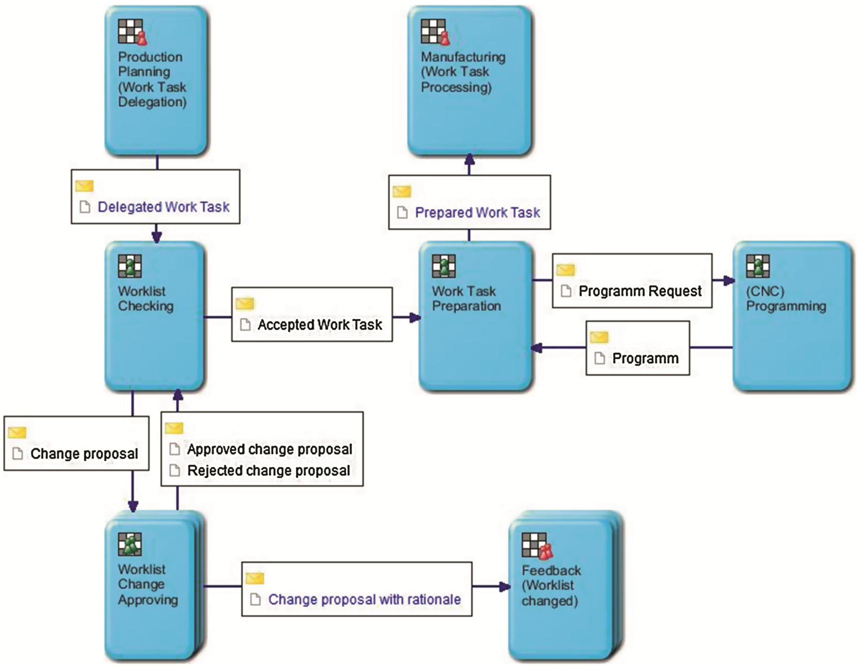


Fig. 1. Subject Interaction Diagram (SID) of a manufacturing preparation process on the shopfloor

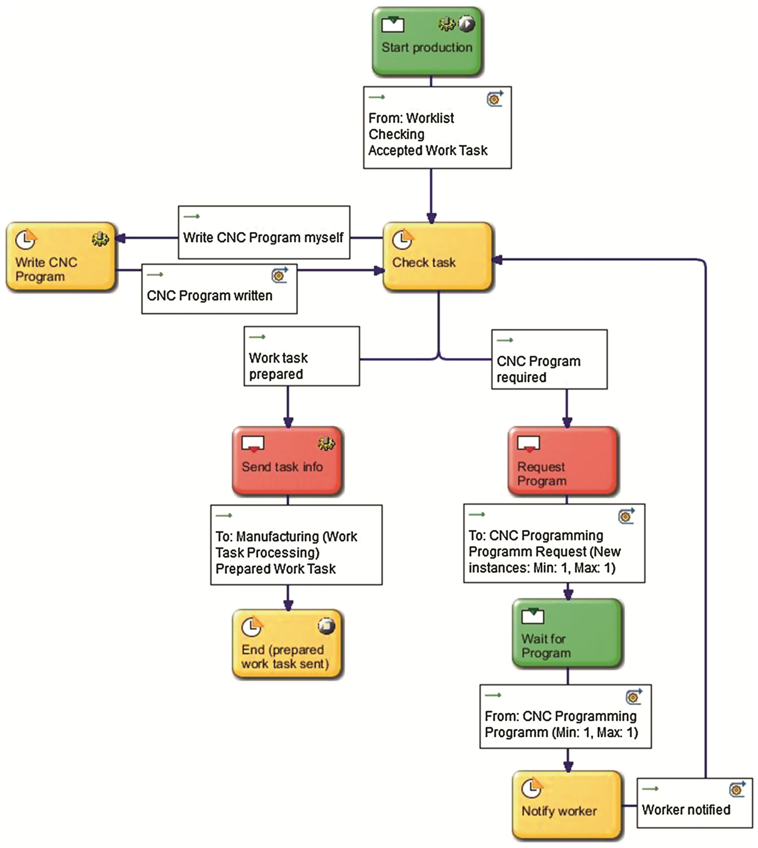

Start production
From: Worklist Checking Accepted Work Task
Write CNC Program myself
Write CNC Program
CNC Program written
Check task
Work task prepared
CNC Program required
Send task info
Request Program
To: Manufacturing (Work Task Processing) Prepared Work Task
To: CNC Programming Programm Request (New instances: Min: 1, Max: 1)
End (prepared work task sent)
Wait for Program
From: CNC Programming Programm (Min: 1, Max: 1)
Notify worker
Worker notified

Fig. 2. Subject Behaviour Diagram (SBD) of the subject "Work Task Preparation" (Color figure online)

Subjects may be executed by human or computational agents [4]. When executed by a human worker, parts of the subject may also be automated by associating pieces of code (called "refinements") to individual states in the behaviour. These states are marked in Fig. 2 using a cogwheel icon in their top right corner. Refinements are always triggered from within the process in which they are defined, irrespective of whether that process is controlled by a traditional user interface or by another process.

3 Modelling Process Execution as a Process

The process of process execution is often a hybrid set of manual and automated tasks, the former commonly being guided by a UI. Using S-BPM, this process can be represented as a subject where some parts of its behaviour are executed by a human user and other parts are executed by a computational agent. We call this subject a "UI subject", and the behaviour of that subject accordingly "UI behaviour". The UI behaviour includes two aspects:

1. *UI workflow*: consisting of a sequence of generic steps independent of the specifics of the underlying core process. For example, a UI workflow may include a particular ordering of steps such as starting a process, displaying a list of user tasks, and editing a function state. The UI workflow can be modelled using an SBD for the UI subject.
2. *UI content*: consisting of the graphical elements (e.g. text fields, buttons, etc.) and appearance of the UI in terms of the layout, shapes and colours. UI content can be modelled as a business object within the SBD of the UI subject.

In this Section we describe how the two aspects of UI behaviour can be modelled and finally connected to the core process, using the commercial S-BPM modelling and execution tool Metasonic Suite.

3.1 Modelling UI Workflow

UI workflows can be modelled in various ways, depending on the needs of the specific users and devices, and the kind of core process they are to be connected with. The SBD in Fig. 3 shows one possible outcome of modelling such a UI workflow. The only types of states used in the SBD are function states, as the process of process execution is modelled using a single subject without any communication with other subjects.[1]

The state "Initialize" is the start state of the SBD, including a refinement to load the initial user interface. In case of a technical failure occurring in this state, a transition is followed to the end state "End (init failed)". In contrast, when the initialization is successfully completed, the states "Select process" and "Select task" need to be performed by the user. Depending on the nature of the selected task as either a function state, a send state or a receive state, the UI behaviour proceeds along separate paths ("Edit function state", "Edit send state" and "Edit receive state"), after which the UI automatically executes the state "Compute next step" to loop back to one of the three paths. During task execution, the user may also switch back to the task overview and select a different task (i.e. follow the transition back to "Select task"), and, while doing that, may also select a different process ("Start process"). Upon termination of the core process, the UI behaviour reaches its desired "End" state.

3.2 Modelling UI Content

The UI content is composed of two groups of data queried from the associated core process instance: the business objects handled in that process, and some of the meta-data needed for process instance management (e.g. subject instance ID and currently active states). Both groups of data are loaded at runtime from the core process and are represented in a business object handled by the UI process. The definition of this business

[1] This modelling decision is based on the fast response times required for the UI behaviour, which would not be reached with the current implementation of the Metasonic Suite if messaging was included.

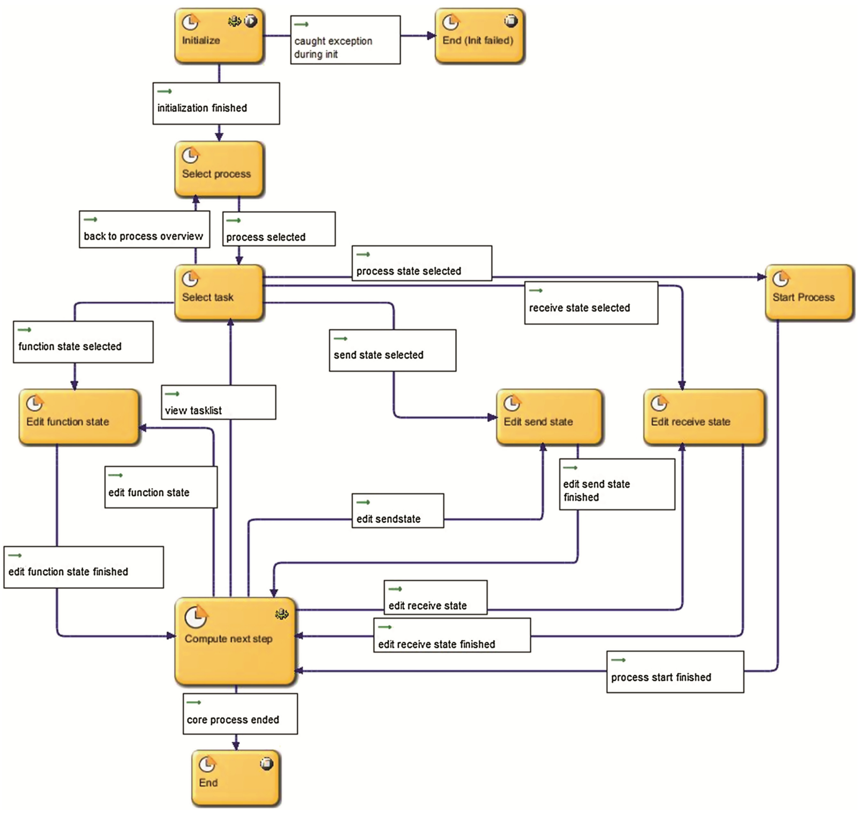
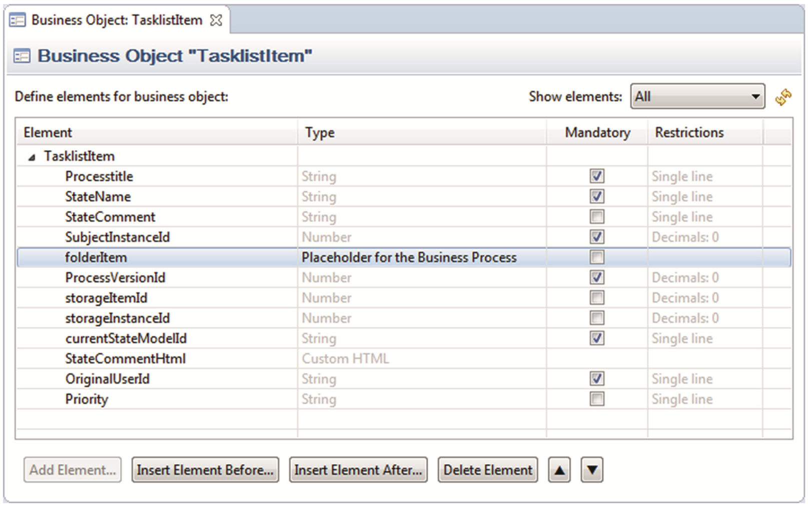


Fig. 3. Example of a SBD defining the UI workflow

object can use the existing data types provided by Metasonic's modelling editor (e.g. String, Number, Enumeration etc.), but also requires a new data type representing a placeholder for the business object of the core process. An example is provided in Fig. 4, showing the definition of a UI business object that contains data elements using standard data types and a placeholder for the core business object.

Out-of-the-box functionalities of the Metasonic Suite also allow defining custom views and layouts of the UI business object. Views [6] specify restrictions on the data elements, including whether an element is visible, hidden, or inactive for a particular state in a SBD. Views in the Metasonic Suite can be associated with client rules to define further attributes such as the colour to be used for displaying a data element. For every view a particular layout can be specified. In addition, the bootstrap framework (http://getbootstrap.com) is used for making the layout and shape of data elements responsive to different screen sizes, supporting conventional computer screens and mobile devices. All bootstrap functionalities such as CSS themes can be used to further customise the UI.


Business Object: TasklistItem

Business Object "TasklistItem"

Define elements for business object: Show elements: All

Element	Type	Mandatory	Restrictions
TasklistItem			
Processtitle	String	✓	Single line
StateName	String	✓	Single line
StateComment	String	☐	Single line
SubjectInstanceId	Number	✓	Decimals: 0
folderItem	Placeholder for the Business Process	☐	
ProcessVersionId	Number	✓	Decimals: 0
storageItemId	Number	☐	Decimals: 0
storageInstanceId	Number	☐	Decimals: 0
currentStateModelId	String	✓	Single line
StateCommentHtml	Custom HTML		
OriginalUserId	String	✓	Single line
Priority	String	☐	Single line

Fig. 4. Example for the definition of a business object in the UI process, containing a placeholder for the business object of the core process

3.3 Connecting the UI Process to the Core Process

UI processes can be modelled either generically for any core process, or for a specific core process. For example, the UI behaviour shown in Fig. 3 is very generic and may be used for all core processes. Other UI behaviours may be defined to tailor the UI to a specific core process and turn some of the "fixed" UI components such as generic menu items and navigation buttons into dynamically generated components that depend on where you currently are in the core process. For example, the "Next" button that is normally used in many workflow UIs to proceed from one user task to the next, may be turned into a set of buttons labelled according to the specific user options defined in the core process.

The concept of generic and specialised UI processes as well as their interplay is shown in Fig. 5. Instances of generic UI processes (designed for all core process models) may be used for providing the UI for instances of any core process. Instances of specific UI processes can be used for providing the UI only for instances of that core process they have been designed for. All process instances are run on the same runtime environment, the Metasonic execution engine.

Running core and UI processes on the same platform allows utilising a number of built-in mechanisms to establish the communication needed between the two processes. As shown in Fig. 6, the Metasonic frontend executing the UI process uses Java Remote Method Invocation (RMI) via API calls and connectors to access core process instance data from the Metasonic backend. That instance data is stored in a DBMS that is queried using Java Database Connectivity (JDBC). The frontend can be accessed by web browsers via HTTP.

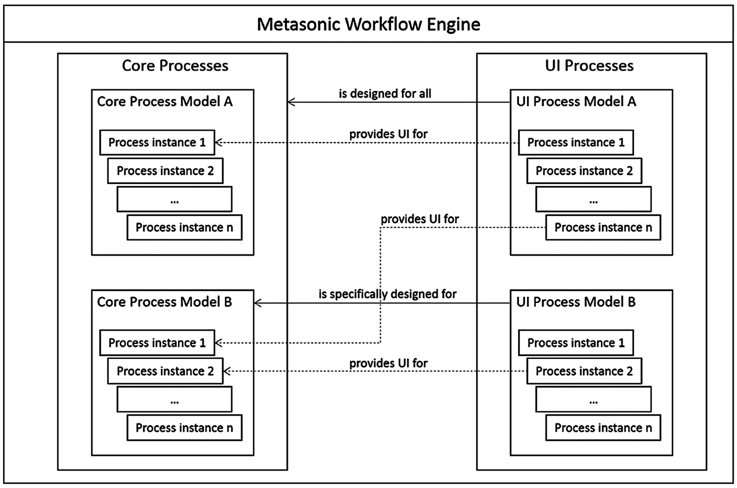


Fig. 5. Conceptual model of the interconnections between core process and UI process

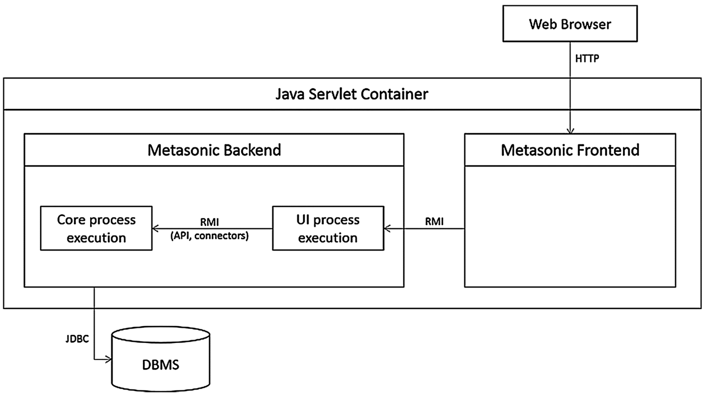
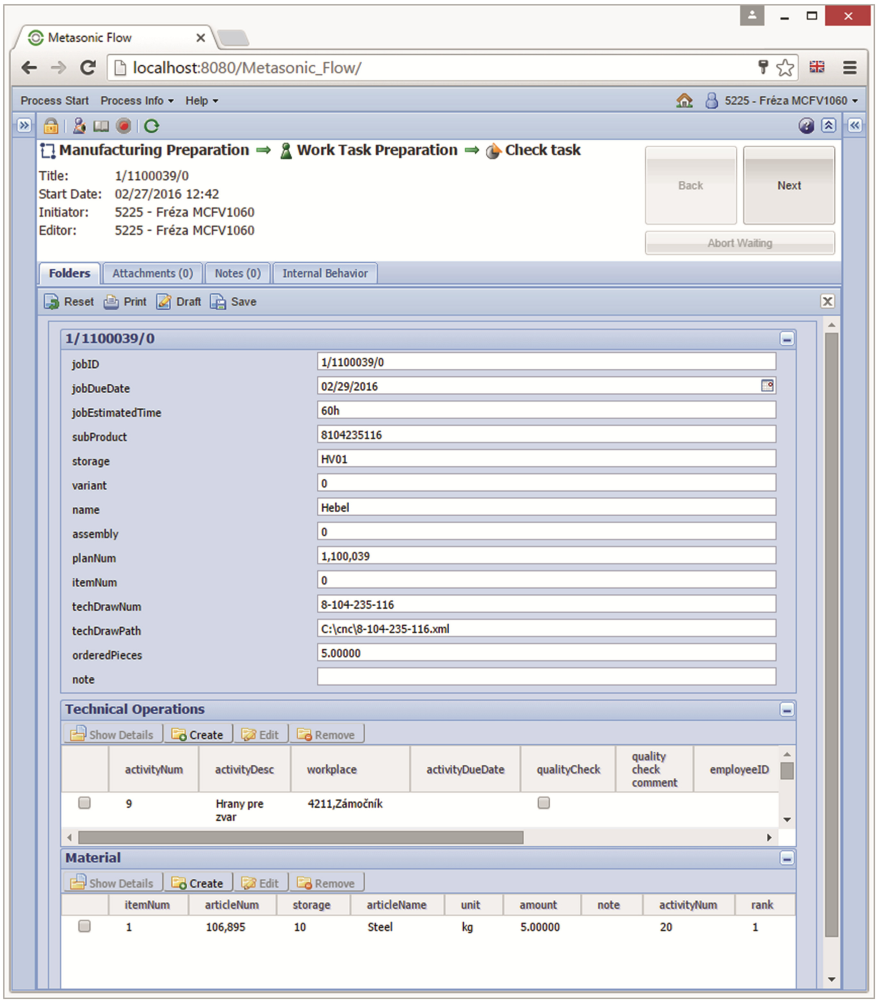


Fig. 6. Software architecture of the interconnected processes

4 Example: Customising UIs for a Shopfloor Process

This Section illustrates our approach based on a case scenario used in the SO-PC-Pro project. Parts of the core process in this scenario – a manufacturing preparation process at Company A – were already introduced in Sect. 2. We will focus on the subject "Work Task Preparation" whose behaviour (shown in Fig. 2) is to be executed by shopfloor workers, guided by a UI running on mobile devices. The standard UI provided in the Metasonic Suite for executing the state "Check task" in this subject is shown in Fig. 7. As it is not modelled as a UI process, it is always the same no matter who executes the process or what core process is executed.



Manufacturing Preparation ⇒ Work Task Preparation ⇒ Check task

Title:	1/1100039/0
Start Date:	02/27/2016 12:42
Initiator:	5225 - Fréza MCFV1060
Editor:	5225 - Fréza MCFV1060

1/1100039/0

jobID	1/1100039/0
jobDueDate	02/29/2016
jobEstimatedTime	60h
subProduct	8104235116
storage	HV01
variant	0
name	Hebel
assembly	0
planNum	1,100,039
itemNum	0
techDrawNum	8-104-235-116
techDrawPath	C:\cnc\8-104-235-116.xml
orderedPieces	5.00000
note	

Technical Operations

	activityNum	activityDesc	workplace	activityDueDate	qualityCheck	quality check comment	employeeID
☐	9	Hrany pre zvar	4211,Zámočník		☐		

Material

	itemNum	articleNum	storage	articleName	unit	amount	note	activityNum	rank
☐	1	106,895	10	Steel	kg	5.00000		20	1

Fig. 7. Standard UI of metasonic flow, showing the data required to execute the function state "Check task"

In the specific case at Company A this UI was deemed too complex to be used by shopfloor workers, as many of them had only limited IT skills. Therefore, a UI process was modelled with the aim to simplify the UI for the workers. The model of this process is almost identical with the one in Fig. 3; it is a generic UI process that may be used for any core process. The appearance of the resulting UIs for the consecutive states "Select process", "Select task" and "Edit function state" in the UI process (cf. Fig. 3) is shown in Figs. 8, 9 and 10, respectively. The UI produced for the state "Edit function state" uses as a header the label of the state "Check task" imported from the core process.

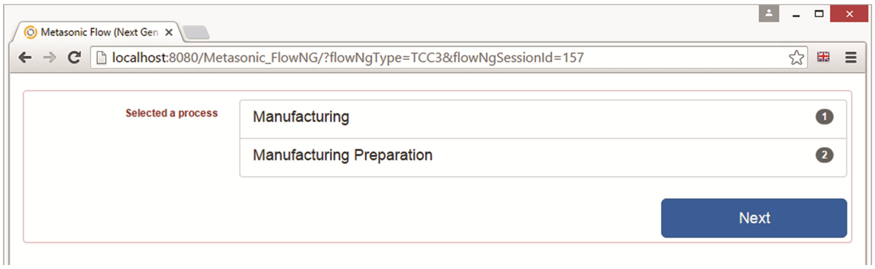

Metasonic Flow (Next Gen
localhost:8080/Metasonic_FlowNG/?flowNgType=TCC3&flowNgSessionId=157
Selected a process
Manufacturing 1
Manufacturing Preparation 2
Next

Fig. 8. User interface for the “Select process” state in the generic UI process

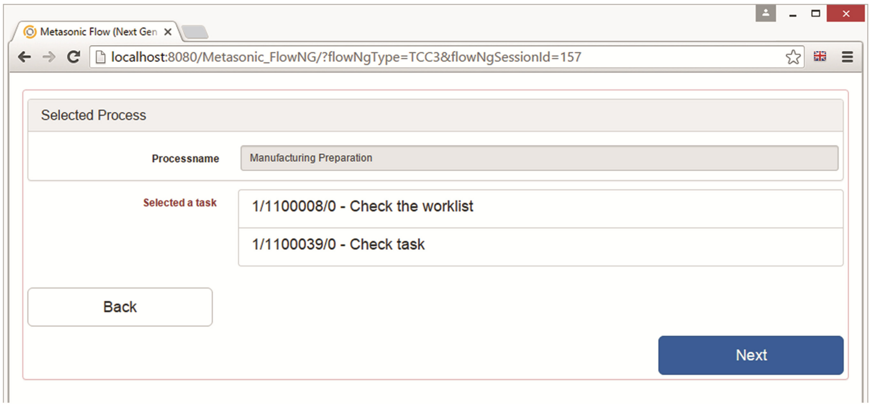

Metasonic Flow (Next Gen
localhost:8080/Metasonic_FlowNG/?flowNgType=TCC3&flowNgSessionId=157
Selected Process
Processname
Manufacturing Preparation
Selected a task
1/1100008/0 - Check the worklist
1/1100039/0 - Check task
Back
Next

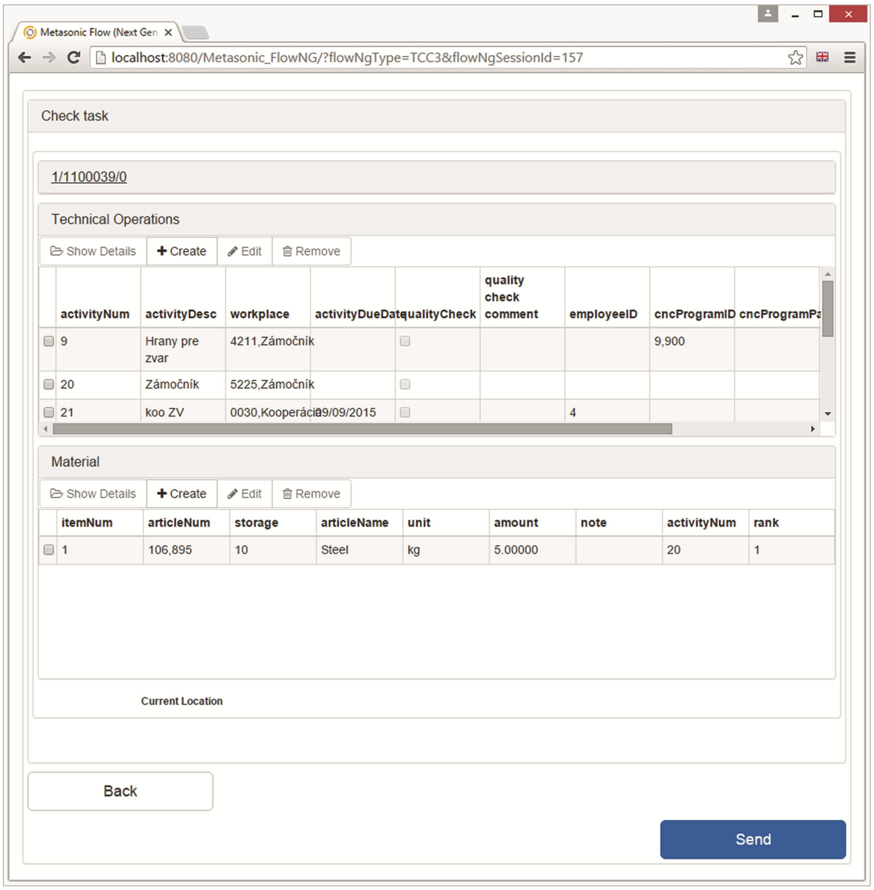
Fig. 9. User interface for the “Select task” state in the generic UI process

These UIs contain only those pieces of information and functionality that a worker would interact with, eliminating all those general menu items, buttons and tabs previously displayed in the standard UI that were regarded unnecessary for the workers. The examples also show that different UIs can be created for different core process steps.

The UI process was later specialised, as shown in Fig. 11, to increase comfort for workers when navigating in the core process from one state to another. For example, the resulting UI shown in Fig. 12 includes three new buttons – “Start production”, “Request CNC Code” and “Write CNC Code myself” – to proceed from “Check task” along the corresponding transitions in the core process (cf. Fig. 2).

A final design of the workers’ UIs was established after a few iterations in which the UIs were tested by workers in Company A using real process data. Desired UI adaptations were fairly easy to be implemented, simply by changing the UI process model without incurring major programming effort or changes to the core process.

A few technical limitations still exist related to the connection between core process components and the UI process. So far only (core) function states with one going transition can be controlled by the UI process, but not receive states, send states or functions states with multiple outgoing transitions. Development is already underway to address these limitations.



Metasonic Flow (Next Gen
localhost:8080/Metasonic_FlowNG/?flowNgType=TCC3&flowNgSessionId=157
Show Details
Create
Edit
Remove
Back
Send

Check task

1/1100039/0

Technical Operations

	activityNum	activityDesc	workplace	activityDueDate	qualityCheck	quality check comment	employeeID	cncProgramID	cncProgramPa
☐	9	Hrany pre zvar	4211,Zámočník		☐			9,900	
☐	20	Zámočník	5225,Zámočník		☐				
☐	21	koo ZV	0030,Kooperácia	09/09/2015	☐		4		

Material

	itemNum	articleNum	storage	articleName	unit	amount	note	activityNum	rank
☐	1	106,895	10	Steel	kg	5.00000		20	1

Current Location

Fig. 10. UI for the “Check task” state (from the core process) used in the generic UI process

5 Related Work

Process modelling in the context of UI design has been proposed for a number of purposes, including usability analysis, requirements specification, and model-driven development [11, 21].

An early approach to modelling user tasks for UIs is the one by Parnas [14] based on state transition diagrams. These diagrams are fundamentally very similar to our simplified SBDs that have only function states but no send or receive states. However, the sole purpose of the models is to represent UI requirements that are then interpreted by human UI designers. The execution of state transition diagrams for automatically generating and controlling the behaviour of UI software is not within the scope of his work.

Dubé et al. [3] proposed hierarchically-linked statecharts (HIS) consisting of UML class diagrams and state machines for specifying the structure and behaviour of UIs, respectively. In particular, the state machine formalism has been chosen based on its

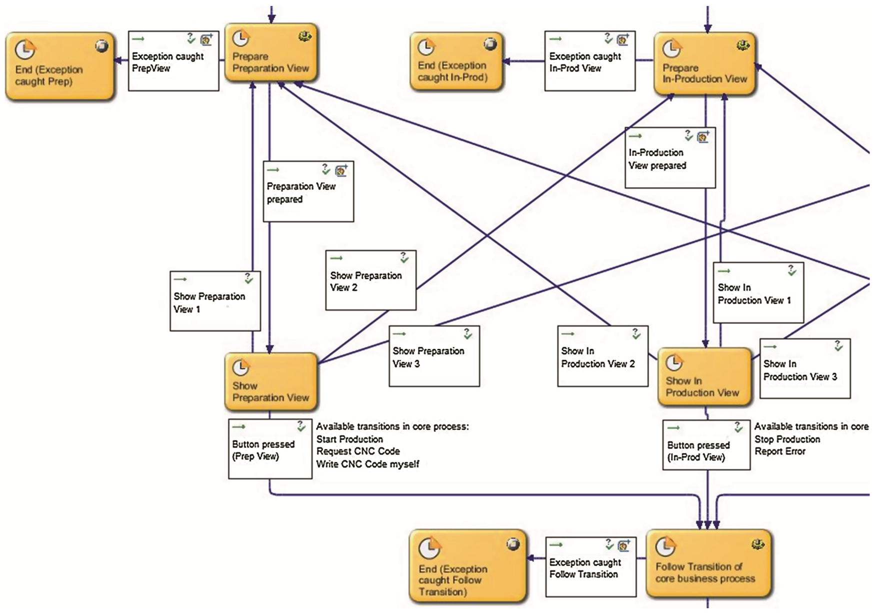
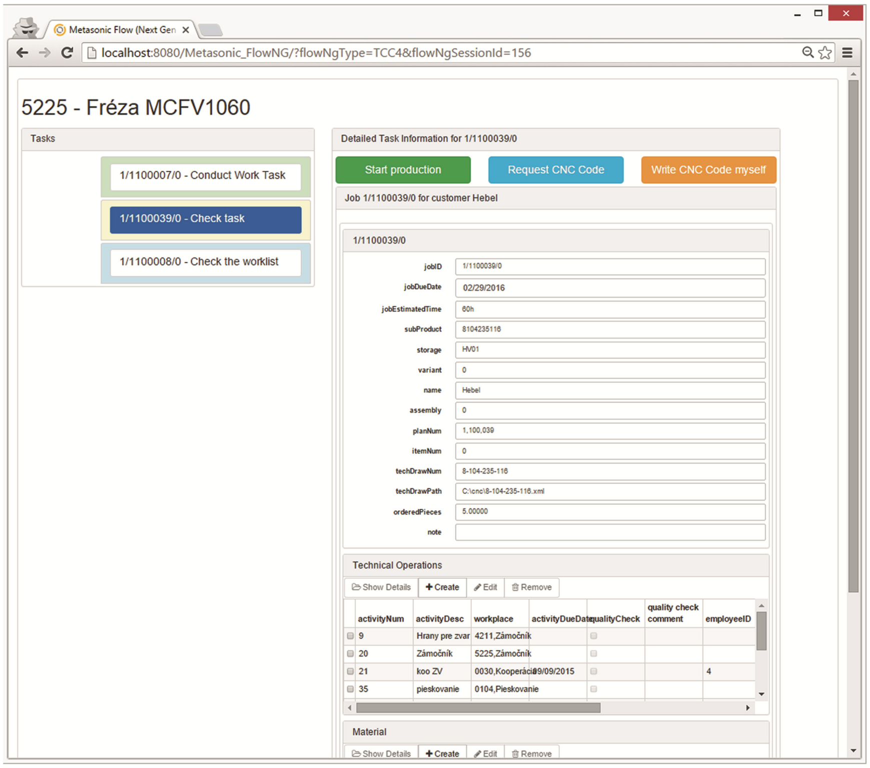

End (Exception caught Prep)
Exception caught PrepView
Prepare Preparation View
End (Exception caught In-Prod)
Exception caught In-Prod View
Prepare In-Production View
In-Production View prepared
Preparation View prepared
Show Preparation View 1
Show Preparation View 2
Show Preparation View 3
Show In Production View 1
Show In Production View 2
Show In Production View 3
Show Preparation View
Show In Production View
Button pressed (Prep View)
Available transitions in core process: Start Production Request CNC Code Write CNC Code myself
Button pressed (In-Prod View)
Available transitions in core Stop Production Report Error
End (Exception caught Follow Transition)
Exception caught Follow Transition
Follow Transition of core business process

Fig. 11. Excerpt from the specialised UI process modelled for Company A

suitability for connecting UI responses with user events such as mouse clicks and key presses. Every visual element in the UI is defined with its own state machine. There are a number of similar approaches to model-driven design of UIs, using behavioural diagrams that are directly or indirectly drawn from UML [15, 16].

Work by Trætteberg and Krogstie [20] aims to realise individualised UIs by means of a model-based design approach that uses the core process model as a starting point. A task model representing the user's tasks is first extracted from the core process model, and then transformed into a dialog model representing the interaction logic of the UI. The core process and the task model are both modelled using BPMN, whereas the dialog model is modelled using the Diamodl notation [19] that is partially based on UML state charts. This approach requires significant manual work for the individual transformations. Transforming core process models into task models involves splitting lanes into pools to make explicit the data flow between them and annotating the task model with pre- and post-conditions. Transforming task models into dialog models then requires additional manual translation effort due to the separate notations used.

Kolb et al. [8] have proposed mappings between task-oriented process models, represented in BPMN, and the logic and contents of UIs, with the aim to generate UI components in a model-driven way. Based on that work, Schobel et al. [17] have implemented a system for designing the UIs of electronic questionnaires using process modelling, and for executing the UI logic on a workflow engine. Other work [9] proposes state-flow representations of data objects as micro-level processes that can be used for generating UIs. These approaches do not include modelling the core process separately from the UI process: There is only one process model that seems to represent both



5225 - Fréza MCFV1060

Detailed Task Information for 1/1100039/0

Job 1/1100039/0 for customer Hebel

1/1100039/0

jobID	1/1100039/0
jobDueDate	02/29/2016
jobEstimatedTime	60h
subProduct	8104235116
storage	HV01
variant	0
name	Hebel
assembly	0
planNum	1,100,039
itemNum	0
techDrawNum	8-104-235-116
techDrawPath	C:\cnc\8-104-235-116.xml
orderedPieces	5.00000
note	

Technical Operations

	activityNum	activityDesc	workplace	activityDueDate	qualityCheck	quality check comment	employeeID
☐	9	Hrany pre zvar	4211,Zámočník		☐		
☐	20	Zámočník	5225,Zámočník		☐		
☐	21	koo ZV	0030,Kooperácia	09/09/2015	☐		4
☐	35	pieskovanie	0104,Pieskovanie		☐		

Material

Fig. 12. User interface for "Check task", generated by a specialised UI process that has been designed only for the manufacturing preparation (core) process in Company A

processes at once. While such a tight coupling has advantages regarding UI maintenance (i.e. if the core process is modified, the UI process is automatically updated accordingly), it prevents customising UIs independently of the core process (i.e. the ability to model multiple UI processes for the same core process).

6 Conclusion

The UI design of business process support systems is a critical issue in the execution of human-centric processes. For a long time, standardised UIs have been favoured based on the high cost of UI customisation. However, as customer demands become more heterogeneous, partly driven by the increasing scope of business process management to cover new domains such as factory and supply chain processes, there is a clear need for more customised UI solutions. This paper has shown how process modelling can be leveraged to develop UIs that can be customised fairly easily using existing functionalities of a BPM suite and integrated frameworks such as bootstrap. The self-referential approach of using process modelling for specifying process execution enables using not only the same technical platform but also the same type of knowledge: the knowledge of process modelling. As a result, developers and integrators of process support systems can address many UI customisation needs without having to employ dedicated UI design specialists.

Our approach is an example for the practical application of a “good theory”, using Lewin’s [10] terms. Here the theory is the S-BPM methodology; it can be seen as “good” because it includes two concepts whose practical application has been demonstrated in this paper: the genericity and the formality of S-BPM process models. The concept of genericity results from the highly abstract modelling constructs in S-BPM: Processes are modelled independently of their embedding in particular organisations and IT infrastructures [4], using only five abstract symbols. This allows modelling any kind of process, including human-centric (business) processes, computational processes and manufacturing processes. In this paper we have shown how the process of process execution, which is often a mixture of human-centric and computational activities, can be modelled with S-BPM. The other concept, formality, is established by the well-defined execution semantics of S-BPM. It allows model-driven transformation of graphical process models into executable software. We have shown that this concept enables turning the modelled process of process execution directly, i.e. without manual intervention, into a running software – the UI of a process support system. We expect that this would be very difficult to be achieved using traditional BPM methodologies such as BPMN, due to their insufficient formal foundations.

Finally, the application of S-BPM for customising UI design can be seen as an example of the “eat your own dog food” principle: As a vendor of an S-BPM suite, we use our own methodology (S-BPM) and our own tool (Metasonic Suite) as a basis for generating customised UIs for our process execution frontend. The technical extensions needed for the realisation of this approach have now matured to product-level quality and will be available on the market with the next feature release (version 5.3) of the Metasonic Suite.

Acknowledgements. The research leading to these results has received funding from the EU Seventh Framework Programme FP7-2013-NMP-ICT-FOF(RTD) under grant agreement no 609190 (www.so-pc-pro.eu).

References

1. Börger, E., Stärk, R.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer, Berlin (2003)
2. Dietz, J.L.G.: DEMO: towards a discipline of organisation engineering. Eur. J. Oper. Res. **128**(2), 351–363 (2001)
3. Dubé, D., Beard, J., Vangheluwe, H.: Rapid development of scoped user interfaces. In: Jacko, J.A. (ed.) HCI International 2009, Part I. LNCS, vol. 5610, pp. 816–825. Springer, Berlin (2009)
4. Fleischmann, A., Kannengiesser, U., Schmidt, W., Stary, C.: Subject-oriented modeling and execution of multi-agent business processes. In: 2013 IEEE/WIC/ACM International Conferences on Web Intelligence (WI) and Intelligent Agent Technology (IAT), pp. 138–145, Atlanta, GA (2013)
5. Fleischmann, A., Schmidt, W., Stary, C.: S-BPM in the Wild: Practical Value Creation. Springer, Berlin (2015)

6. Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Börger, E.: Subject-Oriented Business Process Management. Springer, Berlin (2012)
7. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM **21**(8), 666–677 (1978)
8. Kolb, J., Hübner, P., Reichert, M.: Automatically generating and updating user interface components in process-aware information systems. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 444–454. Springer, Berlin (2012)
9. Künzle, V., Reichert, M.: A modeling paradigm for integrating processes and data at the micro level. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 201–215. Springer, Berlin (2011)
10. Lewin, K.: Field Theory in Social Science: Selected Theoretical Papers. Harper & Brothers, New York (1951)
11. Limbourg, Q., Vanderdonckt, J.: Comparing task models for user interface design. In: The Handbook of Task Analysis for Human-Computer Interaction, pp. 135–154. Lawrence Erlbaum Associates, London (2004)
12. Milner, R.: Communicating and Mobile Systems: The Pi-Calculus. Cambridge University Press, Cambridge (1999)
13. Ould, M.A.: Business Processes: Modelling and Analysis for Re-Engineering and Improvement. Wiley, Chichester (1995)
14. Parnas, D.L.: On the use of transition diagrams in the design of a user interface for an interactive computer system. In: ACM/CSC-ER, pp. 379–385. ACM Press, New York (1969)
15. Paternó, F.: Towards a UML for interactive systems. In: Nigay, L., Little, M. (eds.) EHCI 2001. LNCS, vol. 2254, pp. 7–18. Springer, Berlin (2001)
16. Pinheiro da Silva, P., Paton, N.W.: User interface modelling with UML. Information Modelling and Knowledge Bases XII, pp. 203–217. IOS Press, Amsterdam (2001)
17. Schobel, J., Schickler, M., Pryss, R., Reichert, M.: Process-driven data collection with smart mobile devices. In: Monfort, V., Krempels, K.-H. (eds.) WEBIST 2014. LNBIP, vol. 226, pp. 347–362. Springer, Switzerland (2014)
18. Sinur, J., Odell, J., Fingar, P.: Business Process Management: The Next Wave. Meghan-Kiffer Press, Tampa (2013)
19. Trætteberg, H.: Dialog modelling with interactors and UML statecharts - a hybrid approach. In: Jorge, J.A., Jardim Nunes, N., Falcao e Cunha, J. (eds.) DSV-IS 2003. LNCS, vol. 2844, pp. 346–361. Springer, Heidelberg (2003)
20. Trætteberg, H., Krogstie, J.: Enhancing the usability of BPM-solutions by combining process and user-interface modelling. In: Stirna, J., Persson, A. (eds.) PoEM 2008. LNBIP, vol. 15, pp. 86–97. Springer, Heidelberg (2008)
21. van Welie, M., van der Veer, G.C., Eliëns, A.: An ontology for task world models. In: Markopoulos, P., Johnson, P. (eds.) Design, Specification and Verification of Interactive Systems 1998. Eurographics, pp. 57–70. Springer, Vienna (1998)