Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 98 (2016) 340 – 347

Procedia Computer Science

The 6th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2016)

A bottom up approach for synchronous user interaction design and workflow modelling

Rosaline Barendregt[a,*], Yngve Lamo[b], Fazle Rabbi[b]

[a]*University of Bergen, Norway, {rosaline.barendregt@uib.no}*
[b]*Bergen University College, Norway {yla@hib.no, fra@hib.no}*

Abstract

Traditionally, both interaction design and workflow modelling have been developed in a top down approach. The user interaction design process consist of several phases of user requirement engineering involving interviews, observations, etc. Likewise, workflow modelling traditionally involves a systematic process of requirement collection. In this paper we introduce a new approach where user interaction design is used in a systematic way to develop workflow models in an iterative bottom up approach. The methodology is illustrated by a practical case study of software development, supporting a blood transfusion workflow in a hospital.

Peer-review under responsibility of the Program Chairs

Keywords: Workflow simulation; Metamodelling; User Interfaces; Interaction Design; Health Care processes;

1. Introduction

In situations where lives are at stake or disasters are right around the corner, there is little to no room for mistakes. For technology to be of aid in such stressful situations, it should be easy and intuitive to use, help to prevent mistakes, and be effective. When technology does not meet these requirements, there is a risk that it obstructs the work. Technology and applications to be used in stressful situations, require careful design and extra security and reliability measures. It is of vital importance to achieve precise and correct application logic and combine this with an efficient design. In our approach we represent the application logic by formal workflow models hence the synergy between the workflow model and the interaction design is critical. The workflow model lines out the exact possibilities of the user for each step of the process, including 'offline' actions, while the interaction design guides the user towards making choices at the right time in this process and limits the possibilities for human mistakes.

* Corresponding author.
E-mail address: rosaline.barendregt@uib.no

Peer-review under responsibility of the Program Chairs
doi:10.1016/j.procs.2016.09.051

Workflow modelling (WM) and interaction design (ID) have a strong connection, still, these two activities are often separated [1]. Workflow modellers work towards a detailed and structured overview of the different activities and tasks in a business process, but do often not think about how this model will be presented to the end user. Interaction designers also need an overview of the user activities, with the goal to translate it into a usable design, however they are usually not focusing on the architecture and the structure of the software. Moreover they apply different terminology, technology, and visualisation methods than the workflow modellers. This complicates the communication between the two disciplines.

Taking a holistic approach where the workflow modelling incorporates aspects that are of importance for the interaction design, benefits the whole application design process, but most important, it will benefit the end user. For complex applications that require a high level of accuracy and cannot tolerate any mistakes, the combination of precise (workflow) modelling and efficient interaction design is vital. Blood transfusion is a safety critical operation, that is considered so important that it is regulated by law. The case study that is presented in this paper is about the development of an application used by nurses in the emergency department of Haukeland University Hospital in Bergen, Norway. The system is designed to save time in critical situations while at the same time improve the security of blood transfusions. By lining out the precise options that nurses can have at each step in the process of handling blood, interaction design and workflow modelling are used to visually guide the nurses through the process, giving them only the options they need at each step in the process.

In this paper we start by providing background information on workflow modelling and introduce the bottom up approach for WM in section 2. Section 3 contains background information about interaction design and in section 4 we present a case study on the development of an application for secure blood transfusion to illustrate how interaction design and workflow modelling mutually could benefit on a synchronised development. Section 5 shows how our approach is related to other workflow modelling and interaction design approaches. We conclude the paper and envision possible further works in section 6.

2. Workflow Modelling

Workflow modelling is a popular technique used to model the behaviour of a system. It provides an abstract visualisation of the different processes of a system and represents the order of process execution. Formal representation of a workflow model has many advantages which include e.g. better understanding of a system, modularisation, simulation, and model based analysis. There are several workflow modelling languages that offer visual modelling artefacts for behaviour modelling such as YAWL [20], BPMN [12], ADEPT2 [8], and Nova workflow [10]. However, these workflow modelling languages have predefined syntax and semantics which cannot be customized. WebDPF is a metamodelling tool that supports flexible diagrammatic development of domain specific modelling languges (DSML) [15]. The tool is built on the principles of Diagrammatic Logic [5], graph transformations [6], and Diagram Predicate Framework (DPF) [16,9]. WebDPF offers a multilevel metamodelling environment which can be used to develop a workflow modelling language such as DERF [17].

2.1. Background on DERF

Rutle et al. proposed a metamodelling approach for behavioural modelling and developed a workflow modelling language named DERF using coupled model transformation rules [17]. In general, a workflow or process modelling language consists of a collection of routing or branching operators to model the control flow of processes. In DERF, routing operators are modelled by routing predicates; DERF provides a predefined set of routing predicates including [*sequence*], [*xor_split*], [*xor_merge*], [*and_split*], and [*and_merge*]. The [*sequence*] predicate indicates a sequential execution order of tasks and the [*xor_split*] predicate indicates that exactly one of the two possible flows must be followed, for more details see [17]. The state of a workflow is determined by the states of task instances (i.e., processes). Each task instance is annotated with a predicate

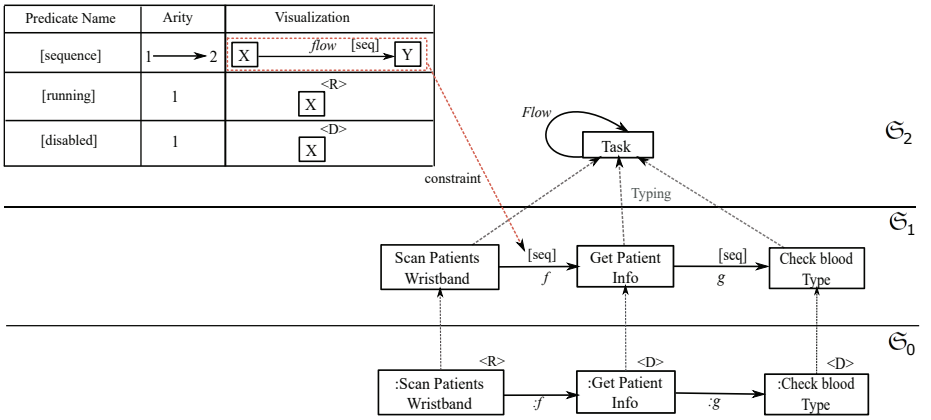


Fig. 1. Workflow metamodel specification $\mathfrak{S}_2$, model specification $\mathfrak{S}_1$, and an instance of $\mathfrak{S}_1$

from [*running*] and [*disabled*] (in short <*R*>, <*D*>, resp.) to indicate its state[1]. A task instance annotated with the predicate [*running*] indicates that the task instance is in the executing state and may transit to the disabled state (e.g., become annotated with [*disabled*]) when it has finished its execution.

Table 1. A routing predicate and its transformation rules

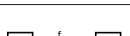
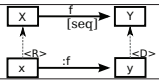
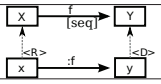
p	Visualization	Coupled transformation rules	
		LHS	RHS
[sequence]			

Table I shows a rule, which is used to change the annotation of a task instance x from <*R*> to <*D*> and the annotation of the next task instance y from <*D*> to <*R*>, respectively. Figure. 1 shows an example of a DPF model (specification) $\mathfrak{S}_1$ with its metamodel (specification) $\mathfrak{S}_2$. The graphs represent the structure of the models; constraints are added into the structure by predicates. The metamodel (specification) $\mathfrak{S}_2$ provides the typing for the model (specification) $\mathfrak{S}_1$. Nodes in model (specification) $\mathfrak{S}_1$ are of type 'Task'; edges in $\mathfrak{S}_1$ are of type 'Flow'. The [*sequence*] predicate is used for constraining the specification $\mathfrak{S}_1$. Each predicate has a name (p), a shape graph ($\alpha(p)$), a visualisation, and a semantic interpretation. The semantic of a predicate is provided by a set of instances. The (atomic) constraining constructs which are available for the users of the modelling language are provided in the signature Σ. A signature consists of a collection of diagrammatic predicates. The figure shows an instance $\mathfrak{S}_0$ of $\mathfrak{S}_1$ where the tasks are annotated with [*running*] and [*disabled*] predicates to indicate their states. The state space of a workflow model is produced by taking an initial workflow model instance and applying the model transformation rules associated with the routing predicates[17].

2.2. Bottom up approach for workflow development

The bottom up approach for workflow development suggests a customizable framework for workflow modelling. Using this approach, it is not necessary to have a workflow language with a set of predefined branching operators before a workflow model is being developed. The modeller starts building a workflow model using its metamodel. As soon as new language requirements are identified, the modeller updates the metamodel to adapt to the requirements for language changes. The modeller then continues building the workflow model using the updated version of the workflow language. However, there are some refactoring

[1] Here we use two states for task instances rather than four states ([*disabled*], [*enabled*], [*running*], [*finished*]) as proposed in DERF[17].

related issues of this approach such as: if a concept A is removed from a metamodel where A has number of instances. In this section we discuss these issues and present solutions.

WebDPF provides a multilevel metamodelling environment where a modeller interactively creates a metamodel stack. The tool allows the modeller to switch from one abstraction level to another while modelling. The editor shows two abstraction levels at a time consisting of a model and its metamodel. First we identify all the changes one may wish to perform to a metamodel and then we classify the change requirements into different categories to analyse their effect. Below is a list of changes one may wish to perform in a metamodel:

- Change the name of a concept or a relation between concepts (i.e. a node or an edge)
- Change which concepts a relation is affecting (i.e. the source / target of an edge)
- Add/delete a concept or a relation between concepts (i.e. a node or an edge)
- Add/delete a new requirement to/from the metamodel (i.e. an atomic constraint)

Let us consider a specific workflow model where changes in it's metamodel occur. Formally we represent the workflow model as (I, ι), and its corresponding metamodel as $\mathfrak{S}_1$. Renaming a node or an edge at $\mathfrak{S}_1$ does not have any side-effects (i.e., updated instances do not become inconsistent) and performing this change is allowed in WebDPF. If a node or an edge is renamed in $\mathfrak{S}_1$, the change is automatically reflected to the instance (I, ι).

WebDPF allows to perform the change of a source and/or target of an edge to some degree; the change is permitted if an edge $A \xrightarrow{e} B$ of metamodel $\mathfrak{S}_1$ has no instances in (I, ι). In case there are some instances of e in (I, ι), the instances can automatically be migrated if there is no ambiguity. For example, $a \xrightarrow{e1} b$ is an instance of e where a and b are instances of A and B respectively. Suppose, we introduce a change to e by changing the target from B to C. The new target node C has only one instance, c. Therefore, to migrate from $a \xrightarrow{e1} b$ to $a \xrightarrow{e1} c$ is obvious. However, this change may have some side-effects (i.e., updated instances may become inconsistent). Again, if there were more than one instance of C, the migration would not be straightforward. One solution would be to show different possible migrated instances to the modeller and let him select; another way to resolve this issue would be to create multiple instances of e for each instance of C. However, the migrated instance may become inconsistent which will require repairing. Coupled model transformation rules may be used to repair inconsistent instances as presented in[14].

Deleting a node from a metamodel is allowed in WebDPF if the removal of the node does not produce any dangling edges and the node does not have any instances. In case there are instances related to a node (incoming edges, outgoing edges, instances of incoming edges, instances of outgoing edges, etc.), the removal of the node requires the removal of all these connected modelling elements. Even though it is a straightforward operation, it may have some side-effects. Deleting an edge has similar issues as deleting a node from a metamodel.

Adding an atomic constraint to a metamodel may introduce some side-effects to its instances. An instance may become inconsistent and will require repairment. However, deleting an atomic constraint from a metamodel is straightforward and does not have any side-effects.

These side effects can be automatically managed by using coupled transformation rules. Mantz et al. presented a co-evolution approach for updating a metamodel and migrating its instances[11]. It may be a time consuming and error prone task to migrate a large number of instances after the evolution of a corresponding modelling language. The co-evolution approach can be useful to adapt a large number of instances with respect to an updated metamodel. The approach uses coupled transformation rules for migrating instances and guarantees that the migrated instances confirm to the updated modelling language.

3. Interaction Design

The design for correct *use of systems* is just as important as the correctness of the system itself[21]. While software modellers are concerned about domain requirements, interaction designers focus on requirements

based on user behaviour. The concept of user experience is very important to interaction design; all efforts are directed towards optimising the interaction between the system and the user, by designing user interfaces that meet the user's needs and make the system easy to use[13]. To get an overview of what these user needs are, interaction designers usually start by identifying the current workflow of users by observation and conducting interviews. They identify e.g. the different tasks, goals, and take into account the environment the system will be used in. The workflow often gets modelled semiformally as e.g. flow charts, site maps, or architectural diagrams, which have less details than a formal workflow model and usually lack simulation and analysis capability. Based on the workflow model, the interaction designer comes up with an initial prototype of the user interface, which will in its turn be discussed and tested with users, in order to be improved and tested again etc. The quality and correctness of the workflow model, decides in large degree the quality of the user interface.

3.1. Problematic work environments

The design of a system should always take into account the environment in which it will be used; different circumstances require different design choices. In stressful work environments with little room for mistakes, such as hospitals, the design of the user interface should take into account the pressure that the users are dealing with by making it easy for the users to find the desired actions. The user interface should be designed with extra attention to prevent the user from making mistakes[21]. In order to do this, it is very important to get a good understanding of which actions should be available at specific times in specific medical procedures. Observing and interviewing health care specialists are a very important part in achieving this.

3.2. Health care specialists as users

Specialists in the healthcare domain often experience a heavy workload, especially in emergency departments[7]. Unfortunately this means that they are usually too busy for long in-depth interviews, which is an important technique for interaction designers during the development of new systems. Observations are also problematic; it is e.g. usually prohibited to observe in operation rooms, due to safety and ethical issues. In the emergency room, so much is happening at the same time that an observer would just stand in the way. While using cameras could be an option, health care specialists might still feel the extra pressure of the knowledge of being observed, which could potentially put lives at risk. We can conclude that the usual process of interaction design is not optimal for systems to be used in this kind of stressful situations. There are too many factors and too many potential risks that could put lives in danger.

3.3. Synchronous development

By synchronously developing the user interface and the workflow model, we ensure that design changes are incorporated in the software architecture. Moreover the domain constraints that are represented in the workflow model are implemented in the visual design and can rapidly be tested with users.

4. Case study: Blodappen

Blodappen is a *blood transfusion application* that aims to improve the security around blood transfusions at Haukeland University Hospital. It was developed in collaboration with the hospital, their ICT organisation, interaction design researchers from the University of Bergen, and software modelling researchers from Bergen University College. The initial request from the hospital was an application, to be used on a 7inch tablet with professional scanner, that would meet the following requirements:

- The application must read the identity of patients from the patients wristbands and control that the patient identity is matching the prescribed blood product.

- The application must access the electronic health record (EHR) to determine patients blood type.
- The application can be used to print tags for blood samples to be sent to the laboratory.
- Nurses should be able to order blood from the blood bank based on patient identity or tags.
- Laboratory technicians should be able to send screening results from laboratory to the blood bank.
- The application can be used to record any complications related to blood transfusion.

To get more detailed requirements as basis for the software development, several sessions were organised where all collaboration partners were present, see figure 2. An innovator was functioning as a mediator, supporting the communication between the different experts. During these sessions we experienced communication problems between software developers and clinicians, as well as between the different departments within the hospital.

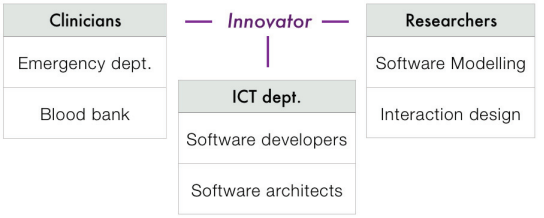


Fig. 2. Participants in the Blodappen project

We also experienced that presenting a workflow model with different types of routing operators and/or constraints to domain experts (clinicians) is not an effective way to get their feedback. The clinicians seemed to have trouble relating the conceptual model to their practical work. On the other hand, discussing user interface designs resulted in useful feedback, but there was still confusion about at which step in the work process the specific interface would be used; a lot of time was spend explaining in which context the specific user interface would be used.

To be able to simulate the workflow model and present it to the users we developed a simulation tool that synchronised the workflow execution with the user interface designs. Figure 3 shows a workflow model instance in the state of getting the patient info from the EHR. On the right side of the figure the actual user interface for this task is displayed.

The simulation tool allowed the clinicians to interact with the user interface yet simultaneously observe the states of the workflow in the background, this helped them to communicate requirements and constraints more precisely. This set-up helped to create a common ground for all participants during the sessions where the blodappen was discussed and saved a lot of time and confusion. The domain experts were very positive about the sessions and the simulation tool: it was a new and pleasant experience for them to participate so actively in an iterative software development process.

The simulation tool also created a close collaboration between the interaction designers and workflow modellers: the connection between the user interface and workflow model allowed interaction designers to base the user interface on a formal workflow model and the workflow modellers could use the user interface design as a tool to increase the quality of the requirements and constraints. The simulation tool has a flexible infrastructure, which makes it easy to carry out changes in the workflow model in case e.g. the hospital will change procedures, it also clarifies at once when the user interface needs to be updated.

5. Related Work

Borchers[1] argues for the need for a common language for software engineers, human computer interaction designers and domain experts to communicate and exchange ideas in development projects. His central idea is that members of a cross disciplinary team should express their expertise in the form of a common pattern language. This makes their knowledge and assumptions more explicit, and easier for the other disciplines to understand and refer to. The proposed patterns are representing ontological relations

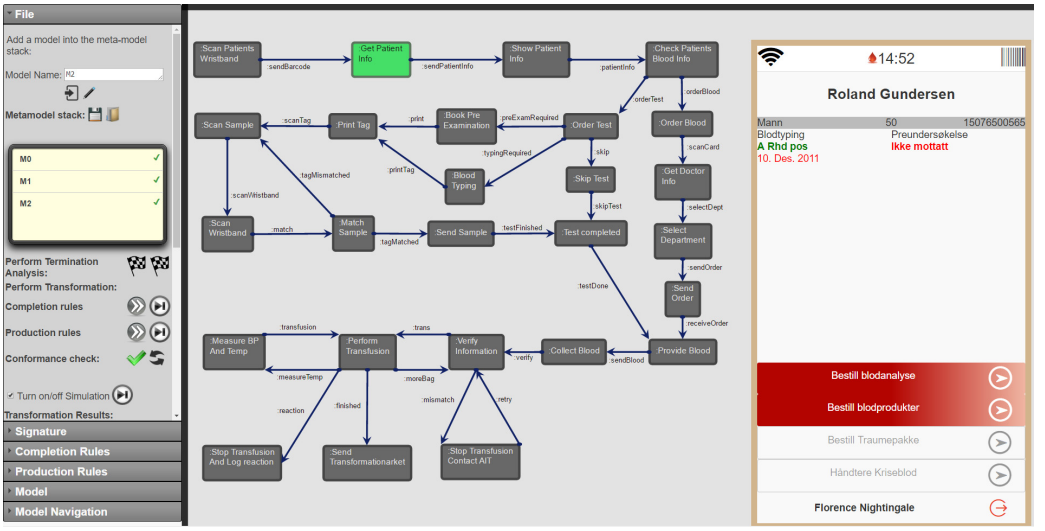


Fig. 3. A screenshot of our tool for synchronous simulation of workflow and user interfaces

of a domain in form of a directed acyclic graph. This is similar to how the typing hierarchy is represented in WebDPF, but the pattern approach is not considering constraints or process behavioural.

Christov et al. presented a set of medical examples including a blood transfusion process definition created in Little-JIL language[4]. They also presented a set of properties that a blood transfusion process must satisfy, however, they did not show any technique for requirement analysis or user interaction design.

There are many workflow modelling languages such as YAWL[20], ADEPT2[8], and Declare[19] that support visual construction of workflow models, simulation and runtime migration of instances. However, these languages are not based on multilevel metamodelling and have limitation on customization such as adding a new branching operator into the modelling language.

Cuadrado et al. presented an interactive approach for bottom up metamodelling by domain experts[18]. They addressed issues related to the construction and customization of metamodels. The authors proposed an iterative process to automatically induce and/or refactor metamodels from model fragments given by sketches or textual annotations. Several other techniques have been presented to construct models or MDE artefacts in[2,3]. Authors in[3] have presented an approach to build metamodels from model examples using a concrete syntax.

The approach we presented differs from these approaches as we incorporate user-interface design during the customization phase. In our approach, the modeller uses an integrated environment where the user-interface design and workflow models are visualised to gather early feedback from domain experts. This approach may be used for gathering requirements as well as developing an early prototype of a system by combining multilevel metamodelling with interaction design.

6. Conclusion and Further Work

During the blodappen case study we experienced that the bottom up approach to workflow modelling and user interface design is a promising approach that helps to establish a common language to communicate between different disciplines. One benefit with this approach is that the domain experts were allowed to communicate in their conceptual space without need to think about technological details and software development concepts. The approach is so promising that we already started two new projects with the Haukeland University Hospital: a project on secure preparation, control, and delivery of medicines and a project that should support patients and medical doctors in the treatment of multiple sclerosis.

The bloodapp was developed as a proof of concept and there was limited time for formal user tests of the approach. A systematic study of usability is planned during the execution of these new projects. To test the interaction between the clinicians, the patients, and the application, without disturbing factors in the hospital, blodappen will be tested in Bergen University College's simulation hospital with the help of nurses in training. There are also plans to develop an educational version of blodappen which can be used in nurse education on the subject of blood transfusions; this will help students get a better understanding of the blood transfusion workflow and become familiar with the application.

We documented the need for better tools for synchronous simulation of workflows and user interfaces. Even though the tool presented in this paper got much positive feedback, there were still some issues regarding tight coupling of the user interface and the workflow model. Currently the tool is being reimplemented to make it easier for designers to create and connect user interfaces to the workflow model, based on methods they are already familiar with and thus without needing advanced software development expertise.

References

1. J. O. Borchers. A pattern approach to interaction design. *Ai & Society*, 15(4):359–376, 2001.
2. H. Cho, J. G. Gray, and E. Syriani. Creating visual domain-specific modeling languages from end-user demonstration. In *Proceedings of the 4th International Workshop on Modeling in Software Engineering, MiSE 2012, Zurich, Switzerland, June 2-3, 2012*, pages 22–28. IEEE Computer Society, 2012.
3. H. Cho, Y. Sun, J. Gray, and J. White. Key challenges for modeling language creation by demonstration. *ICSE 2011 Workshop on Flexible Modeling Tools, Honolulu HI*, 2011.
4. S. C. Christov, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, and E. A. Henneman. A benchmark for evaluating software engineering techniques for improving medical processes. In *Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care*, SEHC '10, pages 50–56, New York, NY, USA, 2010. ACM.
5. Z. Diskin and U. Wolter. A diagrammatic logic for object-oriented visual modeling. *Electronic Notes in Theoretical Computer Science*, 203(6):19 – 41, 2008. Proceedings of the Second Workshop on Applied and Computational Category Theory (ACCAT 2007).
6. H. Ehrig, R. Heckel, G. Rozenberg, and G. Taentzer. *Graph Transformations*. Springer Berlin/Heidelberg, 2008.
7. D. J. France, S. Levin, R. Hemphill, K. Chen, D. Rickard, R. Makowski, I. Jones, and D. Aronsky. Emergency physicians behaviors and workload in the presence of an electronic whiteboard. *International Journal of Medical Informatics*, 74(10):827 – 837, 2005. Supporting Communication in Health CareSupporting Communication in Health Care.
8. K. Göser, M. Jurisch, H. Acker, U. Kreher, M. Lauer, S. Rinderle, M. Reichert, and P. Dadam. Next-generation process management with ADEPT2. In *Proceedings of the BPM Demonstration Program at (BPM'07), Brisbane, Australia, 24-27 September 2007*, volume 272 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
9. Y. Lamo, X. Wang, F. Mantz, W. MacCaull, and A. Rutle. Dpf workbench: A diagrammatic multi-layer domain specific (meta-) modelling environment. In *Computer and Information Science 2012*, pages 37–52. Springer Berlin Heidelberg, 2012.
10. W. MacCaull and F. Rabbi. NOVA workflow: A workflow management tool targeting health services delivery. In *Foundations of Health Informatics Engineering and Systems - First International Symposium, FHIES 2011, Johannesburg, South Africa, August 29-30, 2011. Revised Selected Papers*, volume 7151 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2011.
11. F. Mantz, G. Taentzer, Y. Lamo, and U. Wolter. Co-evolving meta-models and their instance models: A formal approach based on graph transformation. *Sci. Comput. Program.*, 104:2–43, 2015.
12. O. M. G. (OMG). Business process model and notation (bpmn) version 2.0. Technical report, jan 2011.
13. J. Preece, Y. Rogers, and H. Sharp. *Interaction Design: beyond human-computer interaction*. Wiley, 4th edition, 2015.
14. F. Rabbi, Y. Lamo, I. Yu, and L. Kristensen. A diagrammatic approach to model completion. In *Proceedings of the 4th Workshop on the Analysis of Model Transformations co-located with MODELS 2015), Ottawa, Canada, September 28, 2015.*, volume 1500 of *CEUR Workshop Proceedings*, pages 56–65. CEUR-WS.org, 2015.
15. F. Rabbi, Y. Lamo, I. C. Yu, and L. M. Kristensen. WebDPF: A web-based metamodelling and model transformation environment. In *In the 4th International Conference on Model-Driven Engineering and Software Development: February 19–21, 2016, Rome, Italy*, 2016.
16. A. Rutle. *Diagram Predicate Framework: A Formal Approach to MDE*. PhD thesis, Department of Informatics, University of Bergen, Norway, 2010.
17. A. Rutle, W. MacCaull, H. Wang, and Y. Lamo. A metamodelling approach to behavioural modelling. In *Proceedings of the Fourth Workshop on Behaviour Modelling - Foundations and Applications*, BM-FA '12, pages 5:1–5:10. ACM, 2012.
18. J. Sánchez-Cuadrado, J. de Lara, and E. Guerra. *Model Driven Engineering Languages and Systems: 15th International Conference, MODELS 2012, Innsbruck, Austria, September 30–October 5, 2012. Proceedings*, chapter Bottom-Up Meta-Modelling: An Interactive Approach, pages 3–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
19. D. M. M. Schunselaar, F. M. Maggi, N. Sidorova, and W. M. P. van der Aalst. Configurable declare: Designing customisable flexible process models. In *On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, Sept 10-14, 2012. Proceedings, Part I*, volume 7565 of *LNCS*, pages 20–37. Springer, 2012.
20. W. van der Aalst and A. ter Hofstede. Yawl: yet another workflow language. *Information Systems*, 30(4):245 – 275, 2005.
21. K.-P. Yee. *User interaction design for secure systems.* Springer, 2002.