Structural Considerations in Defining Executable Process Models

Borislava I. Simidchieva, Leon J. Osterweil, and Alexander Wise

Laboratory for Advanced Software Engineering Research (LASER)
Department of Computer Science
University of Massachusetts Amherst
Amherst MA, 01003 USA
{bis,ljo,wise}@cs.umass.edu

Abstract. This paper examines the question of how to structure the representation of a process in order to assure that the representation is effective in supporting such diverse activities as process understanding, communication among process participants, and process execution. The paper uses the example of a negotiation process to demonstrate that one process structure (which we refer to as the narrative form) seems to be quite effective in supporting understanding and communication, but then indicates that this structure seems problematic in supporting process execution. The paper indicates that a different structure (which we refer to as the role-oriented form) seems much more appropriate and effective in supporting execution, but may be lacking at supporting communication. In addition to serving different purposes, the two structures seem to represent different underlying models–a static process model, and a similar, but more complex, execution model. The properties of these two complementary structures are then analyzed and evaluated. The paper then uses these observations to raise questions about the underlying needs for effective process representation, suggesting in particular that a single process representation may not be a suitable basis for supporting the range of needs that process representations are expected to address.

1 Introduction

In other papers we have noted that there are many uses to which people wish to put representations of processes [1,2]. Among these uses are the facilitation of communication and coordination, the identification of defects and deficiencies in processes, and the automation of processes. We have previously noted that these differences in motivation have given rise to different notations with which to model and represent processes. Thus, for example, box and arrow diagrams have been popular and successful as devices for facilitating coordination of the efforts of process participants by communicating to them a sense of the juxtaposition of the various roles of process performers. On the other hand, there is growing evidence that the more formal process modeling notations could be effective as bases for supporting process defect detection and process automation [3,4].

Our recent experience indicates that different motivations and prospective uses for process representations seem to suggest not only the value of different process representation notations, but also the value of using different process representation architectures, even when one process representation notation may seem suitable for meeting multiple uses. In particular, this paper describes our experiences in discovering that a process representation architecture that seemed quite suitable for supporting process communication posed problems for supporting desirable infusions of automated support into the performance of the process.

To be specific, the paper will describe the development of a “narrative model” of a specific dispute resolution process, and summarize our experience in using it to help process participants understand their roles and consider improvements. The narrative model emphasizes the intermixing of the activities of the various process participants, thereby elucidating various coordination issues. The paper then goes on to describe the difficulties that we encountered when we attempted to use this representation as a guide in inserting the use of automation in order to address some of the coordination issues. The difficulties we encountered caused us to completely refactor our process model in order to structure it around specifications of the different roles played by different types of process performers. The “role-based model” of the dispute resolution process has been effective as the basis for supporting the desired automated support. On the other hand, the role-based model seems significantly less useful than the narrative model in supporting communication among the process participants.

The paper begins by presenting a summary of the process to be modeled and the initial narrative model used to describe it. The paper then indicates the problems we encountered when we attempted to use this model as the basis for automation. The paper then presents the role-based model resulting from the refactoring of the narrative model. Finally, the paper ponders various issues raised by this experience, some of which indicate the possibility of an inherent need for different process languages and architectures in order to effectively meet various user needs.

2 The Process—Online Dispute Resolution

In earlier papers we have described our efforts to use process definition and analysis technology to facilitate the resolution of disputes [5,6]. In those efforts, we collaborated with the US National Mediation Board (NMB) to develop a process definition of the approach that NMB uses to mediate disputes in the US transportation industries. Our goals in doing this were various, and included 1. helping the NMB to understand their process so that they might improve it, 2. enabling the NMB to involve disputants in arriving at mutually agreeable approaches to the mediation of their disputes, 3. helping the NMB to train new mediators, and 4. supporting NMB’s process with automation in order both to create efficiencies and to create novel mediation approaches made possible through automated support. In pursuing this last goal we were aiming at the development of systems that are commonly described as Online Dispute Resolution (ODR) systems. This term is used to describe systems that exploit computer and communication technologies to facilitate the resolution of disputes.

The vehicle we used for defining NMB’s dispute resolution process was the Little-JIL process definition language [7]. The features of this language are addressed in numerous other papers, and for that reason (and because of space limitations) we omit

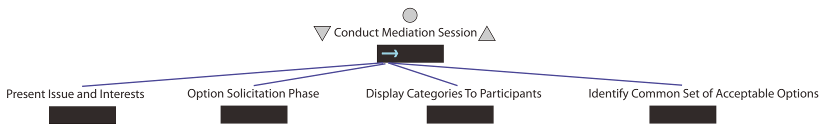
Conduct Mediation Session
Present Issue and Interests
Option Solicitation Phase
Display Categories To Participants
Identify Common Set of Acceptable Options

Fig. 1. Narrative flow of process

descriptions of most of those features here. A central feature of Little-JIL is that it supports the definition of processes as steps (represented as black bars in Figures 1-4) that are organized hierarchically (with child steps connected to the left half of their parent's step by edges), and where each step is annotated with a specification of the type of agent that is responsible for the step's performance. In the case of the NMB's dispute resolution process, some steps are annotated as being the responsibility of the mediator, while other steps are annotated as being the responsibility of a disputant. The NMB's process assumes that there are two sides in every dispute, and that each side may be represented by more than one disputant. In some cases a step may be annotated to mandate that it is to be performed by a disputant from a side that is either the same, or opposite, of the side of a disputant that performed some other step.

As is often the case in creating a model of a process, elicitation of the process became an issue of central importance. Experience in several domains [3,8] has suggested to us that one coherent overarching view of the process is often unavailable, and must be synthesized by putting together different views of the process, each being elicited from a different participant in the process. In the case of the NMB dispute resolution process, however, our elicitation efforts were considerably aided by a domain expert who was a very senior, very experienced mediator. This domain expert was the source of both the high level view of this process, and many of the details needed to support formulation of a coherent view of the process. Our domain expert also indicated places and ways in which different mediators and mediation situations dictated the desirability of creating variants from the baseline mediation process.

From a very high level point of view the process definition that emerged can be viewed as the orchestration of a carefully structured multi-person conversation. The structure specifies first the elucidation of the issues underlying a dispute by the mediator, then the injection of ideas and views by the disputants, then the summarization of these by the mediator, and then the iteration of suggestions for a resolution by the mediator, with responses by the disputants. The process is defined to iterate either until agreement on a resolution has been reached, or until it is agreed that agreement cannot be found. This description of the process strongly suggests that showing the interactions of the mediator and the participants (both individual disputants and their parties) would seem to capture of essence of this process.

Figure 1, for example, shows a small portion of the Little-JIL definition of this process in which the mediator and disputant activities are interleaved (note that the right arrow in the root step's step bar indicates sequential execution of its child steps), with the mediator presenting the issue and interests, the disputants contributing possible options that might address the issue, the mediator then categorizing and presenting the contributed options, and the disputants identifying a common set of acceptable options. Figure 2 shows another portion of the process in which the mediator, after having the

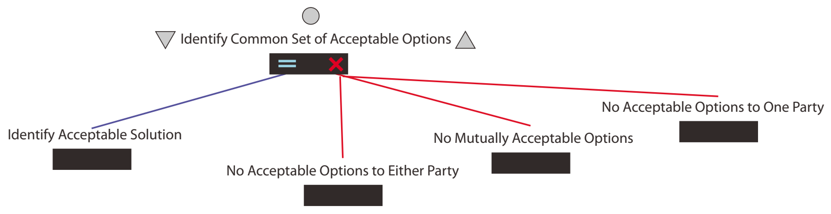
Identify Common Set of Acceptable Options
Identify Acceptable Solution
No Acceptable Options to Either Party
No Mutually Acceptable Options
No Acceptable Options to One Party

Fig. 2. Interruption of normative flow

disputants attempt to identify a common set of acceptable options, decides that it is necessary to interrupt the interactions among the disputants in order to deal with the fact that the process has not identified any mutually acceptable options. Our experience suggests that the process model depicting these interactions helped the mediator to gain a better understanding of the nature of the process, and was indeed useful to him and to the NMB in the training of new mediators. Further, preliminary experience has suggested that disputants should be better able to accept the process because this view helped them to understand why they were being asked to structure their participation as mandated (e.g. why from time to time it is desirable for the mediator to interrupt ongoing discussions). Thus this experience suggested that this model of the process was of considerable value in addressing goals 1, 2 and 3, outlined above.

When we moved on to address goal 4, the provision of automated support for the process, difficulties became apparent.

3 Supporting Execution of the ODR Process

Little-JIL's semantics are rigorously defined by means of finite state machines. The semantics define the behavior of a step to be quite similar to the behavior of a procedure invocation. Thus, each step definition includes a specification of input and output parameters that function in a way that is similar to that of the formal parameters of a procedure. The edge that connects a Little-JIL step to its parent can carry arguments that are bound to the child step's formal parameters at run time. As noted above, each step is annotated with a specification of the type of agent that is to be responsible for the performance of the step, and at runtime a resource manager searches a repository of available resources to identify and then bind a resource instance that matches the step's agent type. After this has been done, the step is ready to be executed.

All that being the case, the execution of a Little-JIL step is left to the resource instance that has been bound as the step's agent. Little-JIL's runtime system allocates to each agent an agenda, which is a list of the steps to which the agent has been assigned. The step's input arguments are passed to the agent by placing them in the agenda item that corresponds to the step. Similarly, once the step has been executed, the resulting output arguments are placed by the agent into the agenda item that corresponds to the completed step, and execution then proceeds.

From this point of view it can be seen that execution of a process defined in Little-JIL is centered largely upon the manipulation of the agendas of the various agents that are participating in the process. It is particularly important to note that there may be many

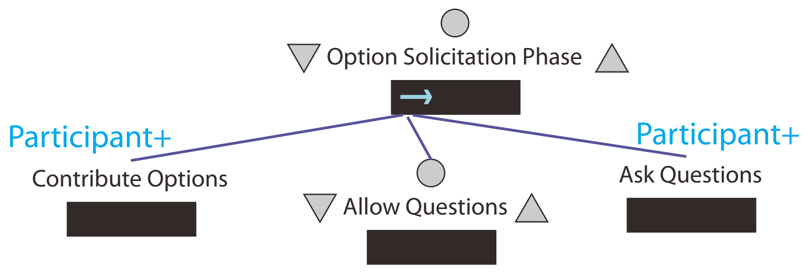

Fig. 3. Option solicitation phase

resource instances of the same type participating in the execution of a process. Thus, for example, there will be many resource instances of type participant (disputant) in an ODR process. Each of these resource instances must have its own agenda, containing agenda items that are the specific steps to be executed by that resource instance. Thus, for example, when a specific resource instance, say Participant1 is assigned the task of submitting a comment, only Participant1 can carry out that task. Moreover, a reply intended for Participant1 must be delivered only to Participant1, not to any participant who might be available.

The need to treat each resource instance individually raised problems in our efforts to use the narrative version of our ODR process as the basis for execution of the process, as the narrative version of the process is essentially a static structure of step types (e.g. steps that are to be bound to any resource instances of a specified type), but the execution of the process results in a more complicated, dynamic structure that requires the management of individual step instances (i.e., the specific steps that each of the resource instances is charged with carrying out).

To be specific, note that Figure 3 defines the way in which the option solicitation phase, one of the bottom level steps referenced in Figure 1, is to be carried out. First, the disputants are asked to contribute options (the Participant+ notation on the step's incoming edge indicates that the subprocess Contribute Options, whose details are not shown here, is to instantiated once for each agent of type Participant), then, after a certain number of options have been suggested, the mediator can choose to allow participants to ask questions by executing the Allow Questions step, after which participants can submit clarifying questions about identified options in the Ask Questions subprocess.

Difficulties with the narrative architecture become clearer when considering the case when only some participants who have submitted options are allowed to ask questions. Since the two subprocesses that are executed by disputants, Contribute Options and Ask Questions, are instantiated once per each participant separately, there is no way of knowing whether a participant for whom Ask Questions is instantiated has actually submitted any options. In order to solve this problem and also to be able to account for disputants on a per-instance basis, the process could be restructured so that the entire Option Solicitation Phase subprocess is instantiated once for each participant. Although this addresses the original concern, it would also result in the mediator having to execute the Allow Questions step multiple times, to allow each single disputant to proceed. This is highly undesirable because it introduces unnecessary work for the mediator and is, moreover, error-prone since it might result in a disputant being overlooked inadvertently.

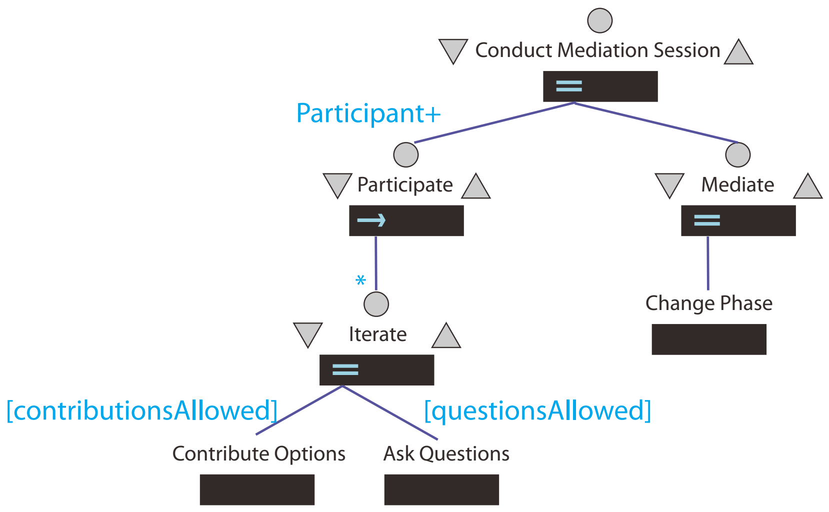

Fig. 4. Role-based architecture

This difficulty is only one example of the way in which the narrative form of the process turned out to be quite unsuitable for specifying important forms of agent interactions clearly, precisely, and tersely. As indicated above, an obvious way in which to address this sort of difficulty is to define such a process in such a way that there is a separate process subtree defining this interaction for each resource instance of type Participant at every point in the process where an interaction between mediator and participants was defined. Ultimately, as noted above, the process definition structured in this way must become large, complex, and increasingly difficult to understand.

Our solution to this problem was to refactor the process definition as a role-based process, in which each of the activities of each of the resource instances was modeled as a separate Little-JIL process tree, with all such trees defined to be executing in parallel with each other. Figure 4 shows the portion of the process needed to represent the process structure from Figure 3 (note that the parallel execution of these trees is denoted by the = sign in the step bar for the step that is the parent of all of these instances). In this process, the left branch (Participate) is performed by each participant, and the right branch (Mediate)–by the mediator. Participation in this process is an iterative activity in which for each iteration, the predicates associated with Contribute Options and Ask Questions consult the process state, which is specified by the mediator's use of Change Phase (and in the case of Ask Questions by the participant's previous actions) to determine which actions can be performed currently.

This role-based process architecture made it straightforward to define the actions of each resource instance, and to separate the actions of the different agents of the same type because this architecture mimics the dynamic execution model closely, unlike the narrative process architecture, which is based on the static model. On the other hand, the role-based process made it correspondingly difficult to indicate the necessary coordination of the activities of the different agents. In the role-based process, a message-passing channel construct in Little-JIL is used to define the transfer of messages and information between the mediator (for example) and each of the separate participants (for example). This use of channels is effective in defining appropriate coordination, but at the expense of the clarity that is a feature of the narrative form of the process definition. Thus we see that the need for executability (goal 4) has given rise to the need for a process definition architecture that does indeed support executability, but at the expense of goals 1, 2, and 3.

4 Discussion

Our experience in developing the executable form of our ODR process has called into question our previous belief that there was a single representation of a process that was equally effective in supporting all of the many desired uses for process representations. Previously, we had believed that it was essential to identify a process definition language that was sufficiently clear, precise, and broad in semantic scope. Although we still believe that this is essential, we now also believe that even such a process definition language may need to be used in different ways in order to represent a process in ways that are effective in supporting different process uses.

This experience highlights particularly clearly that the need to actually execute a process definition raises a set of issues and requirements that are different from the issues and requirements that seem to be foremost in supporting process understanding and communication. One key difference seems to be that process execution requires the creation and maintenance of the dynamic state of the executing process. One particularly important aspect of the dynamic state of a process is its specification of the precise activities that each of the performers of the process is engaged in at any time, and which precise artifacts and other resources are being used in order to carry out these activities. This dynamic model can become very different from the static process definition. Especially in cases where resource instances of the same type are performing activities at the same time as each other, it becomes clear that a structure such as the narrative process, which is a structure of types of resources, lacks an important dimension, namely facilities for specifying the different items of information relevant to each of the different instances. This need to address the different characteristics, activities, and artifact utilization for each different resource instance is met far more successfully and effectively by the role-based process structure exemplified by Figure 4. As noted above, however, this process structure seems notably less clear for the purposes of understanding than the narrative form.

Interestingly, the desire to gain process understanding and communication through process elicitation leads to a similar conclusion. The case described above was unusual in our experience in that a single domain expert had a clear understanding of the role of all involved in the process being described. It is far more usual to find that in processes involving multiple types of agents, each contributer has a clear view of his or her own participation, but a less clear view of what others contribute. Thus each contributor group (corresponding to a type of agent) can be helpful in contributing information needed to define a different parallel branch of the process structure. The role-based process structure is often the logical starting point in dealing with such processes, and it then becomes important to refactor this role-based structure to synthesize a narrative structure from it for communication purposes.

In retrospect, our view that processes are a type of software should have suggested far sooner the need for these types of structures. If a process definition is analogous to the text of an executable program, then the structure of an executing process should be expected to be analogous to the internal structure of an executing program, which is clearly very different from its source text. Indeed a process that integrates and coordinates the actions of different types of participants–and multiple instances of different participant types–seems quite analogous to a software system composed out of components each of which supports multiple execution threads. A visualization of such complex parallel, multi-threaded systems should not be expected to bear much resemblance to the source text of such systems, thus presaging the situation described here.

Indeed, as in the case of complex software systems, different needs dictate the need for two representations. A narrative structure–a static representation that is a structure of types of activities, artifacts, and resources–seems necessary to aid understanding. A role-based structure–a dynamic representation, which is necessarily a structure of instances–however, serves a different set of needs. Although our initial expectation was that the instance structure needed to support representation of the process dynamic state might be patterned closely after the static structure, the experiences in this paper now strongly suggest that this may not be a realistic expectation. The analogy of processes to programs further suggests that this expectation is probably unrealistic.

4.1 Future Directions

The preceding discussion suggests a number of directions for future research. Most immediately, we note that the role-based structure of a process is not itself an actual representation of the dynamic state of an executing process, but rather a suggestion of at least part of its underlying structure. We propose to take the suggestion and use it as the basis for creating just such a clear representation of the dynamic state of a process. We will do this with a recognition of the probable analogy to the problem of representing the state of an executing concurrent, multi-threaded program. Accordingly, we expect to draw upon work from that domain, while also expecting that our work in the process domain might have applicability to the domain of concurrent program visualization.

In previous work [9], we discussed the benefits of considering closely-related processes as variants of one another, and proposed an approach for reasoning about a collection of such variants by defining process families. We described a process family as a group of processes that are the same, or sufficiently similar, at a high level of abstraction, but may exhibit differences, or variations, at lower levels of abstraction. This definition rested on the assumption that all variants within a family share a common process core, and the variations are different elaborations of this common core.

According to this definition, the narrative and the role-based process architectures described in this paper are not members of the same process family because they are not elaborations of a common process core. It is apparent, however, that these two architectures have a lot in common. Contrary to our previous experience with process variants, these two process architectures share low-level functionality (e.g. a disputant contributing an option, or the mediator presenting an issue statement), but have completely different orchestration of events at the high level. They might even be considered to be different projections of the same underlying model since, ultimately, they both define the same process functionality. If these two process architectures can be construed as different views of the same model, they must be members of the same process family.

This clearly indicates that further investigation is needed to determine if the narrative and role-based versions of the negotiation process are variants of one another, and whether our initial definition of a process family needs to be reassessed to accommodate architecture differences.

Finally, in undermining the expectation that there should be only one form of a process representation that supports all possible uses, this work also raises the question of how many different process representation structures may be needed in order to support all of the many varied uses of processes. Indeed, such an understanding of the basic structures needed to support different uses may lead to clearer understandings of the notations best suited to supporting these different representations. This may in turn, then, help to shed new light on the ongoing question of which process representation notations are best suited for which needs.

5 Related Work

There are many approaches to representing processes, and moreover, many of the approaches to representing software systems are also quite applicable to representing processes as well [10,11]. Most approaches, such as UML module interaction diagrams [12] and IDEF diagrams [13] are similar in goals and design to the narrative form described in this paper. Others, such as UML message sequence diagrams (or “ladder charts”) [12] are more similar in goals and design to the role-based form described here. It is also interesting to note that scientific workflow systems such as Kepler [14] are more in the style of the narrative form of description, but it has been noted that this form is increasingly inadequate as the scientific processes that it describes are to be used to support process execution [8]. Finally, we note that work on Viewpoints [15] also recognizes the value of representing a system as a collection of the different views of the system by the different participants and observers of the system.

Thus the observations described in this paper are not inconsistent with work that has been done previously in the area of software systems. Our paper, however, indicates that the need for these two different structural approaches is also present in representing processes. Moreover, our work suggests that the desired application may have a particularly important role to play in deciding which structural approach to use. The role-based structure seems particularly useful and necessary in supporting execution, while clear communication of coordination issues seems to indicate the use of the narrative form. Our ongoing work is attempting to determine whether a process family approach may indicate how the two structures may be considered views of a more fundamental form. If so, then this work should also be interesting and applicable in the domain of software system modeling and representation.

Acknowledgments

The authors wish to express gratitude to Stanley M. Sutton, Jr, Barbara Staudt Lerner, Lori A. Clarke, and members of the Laboratory for Advanced Software Engineering Research at the University of Massachusetts Amherst who have participated in this research, and clarified the points made in this paper.

This material is based upon work supported by the US National Science Foundation under Award Nos. CCR-0427071, CCR-0204321 and CCR-0205575. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of The National Science Foundation, or the U.S. Government.

References

1. Zhu, L., Osterweil, L.J., Staples, M., Kannengiesser, U., Simidchieva, B.I.: Desiderata for languages to be used in the definition of reference business processes. International Journal of Software and Informatics 1(1), 37–65 (2007)
2. Osterweil, L.J.: Unifying microprocess and macroprocess research. In: Li, M., Boehm, B., Osterweil, L.J. (eds.) SPW 2005. LNCS, vol. 3840, pp. 68–74. Springer, Heidelberg (2005)
3. Clarke, L.A., Avrunin, G.S., Osterweil, L.J.: Using software engineering technology to improve the quality of medical processes. In: ACM SIGSOFT/IEEE 30th International Conference on Software Engineering (ICSE 2008), pp. 889–898 (May 2008); Invited keynote address by Lori A. Clarke
4. Chen, B., Clarke, L.A., Avrunin, G.S., Osterweil, L.J., Henneman, E.A., Henneman, P.L.: Analyzing medical processes. In: ACM SIGSOFT/IEEE 30th International Conference on Software Engineering (ICSE 2008), pp. 623–632 (May 2008)
5. Osteweil, L.J., Katsh, E., Sondheimer, N.K., Rainey, D.: Early lessons from the application of process technology to online grievance mediation. In: 2006 National Conference on DIgital Government Research (2005)
6. Osterweil, L.J., Clarke, L.A., Gaitenby, A., Gyllstom, D., Katsh, E., Marzilli, M., Sondheimer, N.K., WIng, L., Wise, A., Rainey, D.: A process-driven tool to support online dispute resolution. In: International Conference on Digital Government Research, pp. 356–357. ACM Press, New York (2006)
7. Wise, A.: Little-JIL 1.5 Language Report. Technical report, Department of Computer Science, University of Massachusetts, Amherst, MA (2006)
8. Osterweil, L.J., Clarke, L.A., Podorozhny, R., Wise, A., Boose, E., Ellison, A.M., Hadley, J.: Experience in using a process language to define scientific workflow and generate dataset provenance. In: ACM SIGSOFT 16th International Symposium on Foundations of Software Engineering (FSE16), pp. 319–329 (2008)
9. Simidchieva, B.I., Clarke, L.A., Osterweil, L.J.: Representing process variation with a process family. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2007. LNCS, vol. 4470, pp. 109–120. Springer, Heidelberg (2007)
10. Osterweil, L.J.: Software processes are software too. In: 9th International Conference on Software Engineering (ICSE 1987), pp. 2–13 (March 1987)
11. Osterweil, L.J.: Software processes are software too, revisited. In: 19th International Conference on Software Engineering (ICSE 1997), pp. 540–548 (September 1997)

12. Object Management Group: OMG Unified Modeling Language (OMG UML) Superstructure. Technical Report formal/2007-11-02, Object Management Group, Version 2.1.2 (November 2007)
13. US Air Force: ICAM architecture. part II, functional modeling manual (IDEF0). Technical Report AFWAL-TR-81-4023, Materials Laboratory, Wright-Patterson Air Force Base (1981)
14. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J., Zhao, Y.: Scientific workflow management and the kepler system. Concurrency and Computation: Practice & Experience 18(10), 1039–1065 (2006)
15. Nuseibeh, B., Kramer, J., Finkelstein, A.: Expressing the relationships between multiple views in requirements specification. In: Proceedings of the 15th International Conference on Software Engineering, pp. 187–196 (May 1993)