Computer Support for Agile Human-to-Human Interactions with Social Protocols

Willy Picard

Poznań University of Economics,
Department of Information Technologies,
ul. Mansfelda 4, Poznań, Poland
Willy.Picard@ue.poznan.pl,
WWW home page: http://www.kti.ue.poznan.pl/

Abstract. Despite many works in CSCW, groupware, workflow systems and social networks, computer support for human-to-human interactions is still insufficient, especially support for agility, i.e. the capability of a group of human beings, collaborators, to rapidly and cost efficiently adapt the way they interact to changes. In this paper, requirements for computer support for agile H2H interactions are presented. Next, the concept of social protocol is proposed as a novel model supporting agile H2H interactions. A social protocol consists of an extended social network and a workflow model.

Key words: social protocols, workflow, adaptation, social networks, human-to-human interactions

1 Introduction

Computer support for Human-to-Human (H2H) interactions has a long history in computer science: from early visionary ideas of Douglas Engelbart at the Stanford Research Institute's Augmentation Research Center on groupware in the 60's, through CSCW and workflows in the 80's, and with social network sites in the 2000's. However, computer support for agile H2H interactions is still insufficient in most collaborative situations.

Among various reasons for the weak support for H2H interactions, two reasons may be distinguished: first, many *social elements* are involved in the H2H interaction. An example of such a social element may be the roles played by humans during their interactions. Social elements are usually difficult to model, e.g. integrating hierarchical relations among collaborators to collaboration models. A second reason is the *adaptation capabilities* of humans which are not only far more advanced than adaptation capabilities of software entities, but also are not taken into account in existing models for collaboration processes.

Agility in H2H interactions refers to the capabilities of a group of human beings, collaborators, to rapidly and cost efficiently adapt the way they interact to changes. Changes may occur:

– within the group: e.g., a collaborator may be temporary unavailable or he/she may acquire new skills,
– in the environment of the group: e.g., a breakdown of a machine may occur, weather conditions may prevent the realization of a given task.

In this paper, we present a model which provides support for agile H2H interactions based on the concept of social protocols. In Section 2, requirements for a computer support for agile H2H interactions are presented. Next, the concept of social protocols supporting agile H2H interactions is detailed. Then, the proposed solution is discussed. Finally, Section 5 concludes the paper.

2 Requirements for Support for Agile H2H Interactions

2.1 A Model of the Social Environment

A first requirement for support for agile H2H interaction is the modeling of the *social environment* within which interactions take place. H2H interactions imply the involvement of at least two collaborators, each of them having her/his own social position. By social position, we mean a set of interdependencies with entities (generally individuals or organizations): e.g. a collaborator has a set of colleagues, works in a given company, belongs to a family.

Agility during H2H interactions implies a rapid adaptation of the collaboration group to new conditions. The social environment is a core tool in the adaptation process as it provides information about available resources collaborators are aware of:

– within the group: e.g., if a collaborator is temporary unavailable, another person in the social environment may substitute for the unavailable collaborator,
– in the environment of the group: e.g., if weather conditions prevent the realization of a given task, new collaborators which were not initially involved in the realization of the cancelled task may be needed to overcome it.

A partial answer to the question of modeling a social environment may be found in popular in the last five years social network sites, such as LinkedIn [17], MySpace [18], Orkut [19], Friendster [10] and Facebook [8], to name a few. Boyd and Ellison [6] define social network sites as "web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system." The second and third points of this definition illustrate a key feature of social network sites, i.e. social network sites allow users for an easy access to information about persons they know (friends, colleagues, family members) and potentially about contacts of these persons.

However, the model of social environment adopted in social network sites captures only interdependencies among individuals or organizations. The interdependencies with information systems, e.g. web services, are an important element of the landscape of H2H interactions: while individuals represent the "who"

part of H2H interactions, information systems usually represent the "how" part. A collaborator (the *individual*) performs some activity with the help of a tool (the *information system*). Therefore, we claim that a model of the social environment for H2H interactions should integrate both interdependencies among collaborators and interdependencies among collaborators and information systems.

A model of social environment integrating interdependencies among collaborators and among collaborators and information systems would allow collaborators to react to new situations not only by changing the set of collaborators, but also by changing the set of tools. Additionally, such a model would allow collaborators for agility with respect to changes related with information systems: e.g., if an information system is unavailable, collaborators may seek for an alternative in their social environment.

2.2 Structured H2H Interactions

Supporting agile H2H interactions requires guidance for collaborators about tasks they may perform at a given moment of time. Such a guidance allows collaborators for *focusing on appropriate tasks* that need to be fulfilled at a given moment of time, in a given collaboration situation, instead of facing all potential tasks that they may perform.

The tasks that a given collaborator may perform depend also on the *role* he/she is playing within a given group. Therefore support for agile H2H interactions implies the mapping between collaborators and roles they are playing within a given group.

Additionally, H2H interactions are often structured according to collaborative patterns [3, 23]. In similar situations, in different groups, collaborators perform activities whom successiveness is identical among the various groups: e.g., a brainstorming session consists usually of five phases:

1. the chairman presents the problem,
2. every participant presents his/her ideas,
3. the chairman classifies the ideas,
4. every participant may comment any idea,
5. the chairman summarizes the brainstorming session.

In the former example, each phase may be decomposed as a sequence of activities to be performed, with activities associated to roles. H2H interactions could therefore be structured with the help of a *process* and an associated *process model* specifying the sequences of activities, the association between activities and roles, and the mapping between collaborators and roles.

Results of studies in workflow technology and process modeling [9, 15, 11, 5] provide a strong foundation for support for structured H2H interactions based on the concepts of workflow and process models.

2.3 Layered Interaction Models

The concept of process model presented in the former subsection as a mean to structure H2H interaction has to be considered at three levels of abstraction:

- *abstract process model*: a process model is abstract if it defines the sequence of activities to be potentially performed by collaborators playing a given role, without specifying neither the implementation of activities, nor the attribution of roles to collaborators. As an example, an abstract process model for a brainstorming session may specify that, first, a chairman presents the brainstorming session problem, next, participants present their ideas. Neither the implementation of the presentation of the problem and participants' ideas, nor the group collaborators are defined in the abstract process model.
- *implemented process model*: a process model is implemented if it defines the implementation of activities defined in an associated abstract process model. As an example, an implemented process model based on the brainstorming abstract process model formerly presented may specify that the presentation of the brainstorming session problem will be implemented as the sending of an email to all participants, while the presentation of ideas will be performed as posts to a forum.
- *instantiated process model*: a process model is instantiated if the attribution of roles to collaborators for a given implemented process model has been set. Additionally, an instantiated process model, referred also as *process instance*, keeps trace of the current state of the H2H interactions. As an example, the former implemented process model may be instantiated by specifying who plays the chairman role and who are the participants. Additionally, the process retains its current state which may for instance be "participants are presenting ideas".

The following analogy with object-oriented programming illustrates the three levels of abstraction presented above:

- abstract process models are similar to interfaces or abstract classes. An abstract process model does not rely, nor provide an implementation of activities, as an interface does not provide an implementation of methods;
- implemented process models are similar to classes. An implemented process model provides an implementation of activities, as a class provides an implementation of methods.
- instantiated process models are similar to objects. An instantiated process model rules the H2H interactions according to a given implemented process model and has its own state, as an object behaves according to its class and has its own state too.

The separation of these three levels of abstraction leads to *process model reuse*. By separating the logical structure of H2H interactions from its implementation, an abstract process model may be reuse in various contexts, IT environments, groups of collaborators. As a consequence, a group of collaborators facing some unpredicted situation may identify an already defined abstract or implemented

process model allowing them to solve their problem. Then, the group may react rapidly by just (eventually implementing and) instantiating the process. The brainstorming process presented above is an example of an abstract or implemented process that may be reuse by various groups of collaborators to interact in an agile way.

2.4 Adaptability

Adaptability is a core requirement of support for agile H2H interactions. Adaptability refers in this paper to the capability of a group of collaborators to modify *at run-time* the process model ruling their interactions.

In typical workflow management systems, two parts may be distinguished: a *design time* part allows for definition of workflow schemas while the *run-time* part is responsible for execution of workflow instances. A main limitation of typical workflow management systems is the fact that once a workflow schema has been instantiated, the execution of the workflow instance must stick to the workflow schema till the end of the workflow instance execution. This limitation is not an issue if the lifespan of workflow instances is short in comparison with the time interval between two requests for changes of the workflow schema. When the lifespan of workflow instances is long in comparison with the time interval between two requests for changes of the workflow schema, a high number of workflow instances has to be executed with an "incorrect" workflow schema (i.e. that does not take into account required changes) or cancelled. As a consequence, typical workflow management systems are not flexible enough to support collaborative processes in two cases: highly dynamic, competitive markets/environments and long lasting collaboration processes.

In the case of highly dynamic, competitive markets/environments or long lasting collaboration processes, there is a strong need for the possibility to modify a workflow instance at run-time. Such modifications are usually needed to deal with situations which have not been foreseen nor modeled in the associated workflow schema. Adaptation refers to the possibility to modify a running instantiated process model to new situations which have not been foreseen and modeled in the associated abstract/implemented process model.

3 Social Protocols

Computer support for agile H2H interactions requires novel models to support requirements presented in Section 2. The solution presented in this paper is based on the concept of *social protocol*. This concept has been presented first in 2006 [22], based on the concept of *collaboration protocol* [20]. An extended version of the concept of social protocol, including elements related with the modeling of the social environment, is presented in this paper.

3.1 Abstract Social Protocols

An abstract social protocol, SP_a, consists of two parts:

- an *abstract social network*: a direct graph modeling interdependencies among *abstract resources*. An abstract social network models the social environment required for a particular collaboration pattern.
- an *abstract interaction protocol*: a direct graph modeling interdependencies among *abstract activities*. An abstract interaction protocol models the sequence of activities in a particular collaboration pattern.

In an abstract social network, vertices represent abstract resources that may support or be actively involved in the collaboration process, such as a collaboration role or a class of information systems. Edges represents relations between resources associated with social interaction types, such as "works with", "has already collaborated with" among roles, or "is the owner", "uses" between a role and a class of information systems. Labels associated with edges are not predefined, as the concept of social protocol should be flexible enough to encompass new types of interdependencies among resources. Therefore, new labels may be freely created at design time.

An example of an abstract social protocol for brainstorming sessions is presented in Figure 1.

Formally, an abstract social network, SN_a, is a directed graph $< R_a, SI_a >$ where R_a is a finite set of nodes, each node referring to an abstract resource, SI_a the social interdependencies relation $SI_a \subseteq R_a \times R_a \times SIT$, with SIT: a set of social interaction types.

In an abstract interaction protocol, vertices represents:

- abstract activities that may be performed during the collaboration process, such as "present the brainstorming problem" or "present an idea". Activities are associated with a given role, e.g. only the chairman may present the brainstorming problem;
- states in which the group may be at various moments of the collaboration process, e.g. the group may be "waiting for ideas".

Edges run between activities and states, never between activities nor between states. Edges capture the potential activities in a given state, or states after the execution of a given activity. One may recognize in abstract interaction protocols the concept of Petri nets, where states are places and activities/roles pairs are transitions.

Formally, an abstract interaction protocol, IP_a, is a 3-tuple (T_a, S_a, E_a), where T_a is a finite set of abstract transitions, S_a is a finite set of states, and E_a is a multiset of arcs $E_a : (T_a \times S_a) \cup (S_a \times T_a) \rightarrow \mathbb{N}$. An abstract transition $t_a \in T_a$ consists of an abstract activity $a_a \in A_a$ and a role $\rho_a \in Roles_a$.

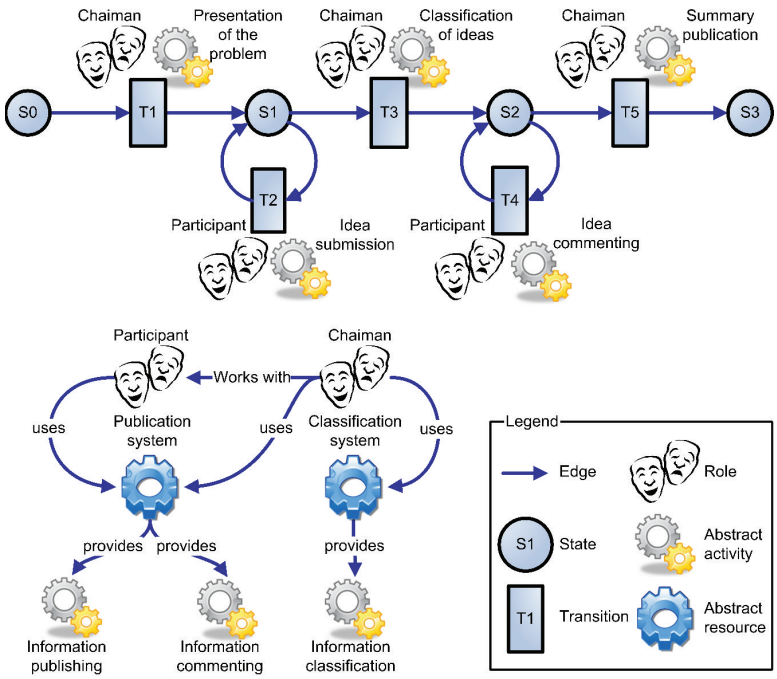

Fig. 1. An example of an abstract social protocol. At the top, the abstract interaction protocol of a brainstorming session. At the bottom, the abstract social network.

3.2 Implemented Social Protocols

Similarly to the relation between implemented process models and abstract process models presented in Section 2.3, an implemented social protocol defines the implementation of abstract activities associated with an abstract social protocol.

Therefore, an implemented social protocol consists of three parts:

- an abstract social protocol,
- a mapping of *abstract resources* associated to with abstract activities to *implemented resources*. For instance, the abstract resource "Publication system" of the former example may be mapped to a forum system on a given server.
- a mapping of *abstract activities* to *implemented activities*. For instance, the abstract activity "presentation of the problem" of the former example may be mapped to the URL of the form used to post information on the formerly mentioned forum system.

These two mappings may be built based on a pre-existing social environment defining interdependencies among resources (abstract and implemented). Additionally, the pre-existing social environment may be extended by the addition of missing resources. Therefore, on the one hand, the implementation procedure may take advantage of the social environment, on the other hand, the social network may benefit from the implementation procedure.

Formally, an implemented social protocol, SP_i, is a 5-tuple $(SP_a, R_i, A_i, \mathcal{R}_\downarrow, \mathcal{A}_\downarrow)$, where SP_a is an abstract social protocol, R_i is a finite set of implemented resources, A_i is a finite set of implemented activities, $\mathcal{R}_\downarrow : R_a \rightarrow R_i$ is a mapping function, such that $\forall r_a \in R_a, \exists r_i \in R_i, \mathcal{R}_\downarrow(r_a) = r_i$, and $\mathcal{A}_\downarrow : A_a \rightarrow A_i$ is a mapping function, such that $\forall a_a \in A_a, \exists a_i \in A_i, \mathcal{A}_\downarrow(a_a) = a_i$.

3.3 Social Processes

Similarly to the relation between instantiated process models and implemented process models presented in Section 2.3, a social process defines the implementation of abstract roles associated with an implemented social protocol, as well as keeps trace of the state of the H2H interactions.

Therefore, a social process consists of three parts:

- an implemented social protocol,
- a mapping of *abstract resources* associated with roles to *collaborators*. For instance, the abstract resource "brainstorming chairman" is mapped to collaborator "John Smith".
- a *marking* of active states.

The role-collaborator mapping may be built based on the pre-existing social environment. Additionally, the pre-existing social environment may be extended by the addition of missing resources, by the addition of collaborators. Therefore, on the one hand, the instantiation procedure may take advantage of the social environment, on the other hand, the social network may benefit from the instantiation procedure.

Formally, a social process, π, is a -tuple $(SP_i, C, \mathcal{C}_\downarrow, M)$, where SP_i is an implemented social protocol, C is a finite set of collaborators, $\mathcal{C}_\downarrow : Roles_a \rightarrow C$ is a mapping function, such that $\forall \rho_a \in Roles_a, \exists c \in C, c \in \mathcal{C}_\downarrow(\rho)$, and $M : S_a \rightarrow \mathbb{N}$ is a marking assigning a number of tokens to each state.

The creation of a social process requires: 1) the choice of an implemented social protocol, 2) the attribution of roles to collaborators, and 3) the creation of an *initial marking*. The initial marking contains a set of active states, referred as initial states.

Next, the social process rules the interactions according to the associated social protocol:

- a collaborator may trigger a transition t if, 1) he/she plays the associated role, 2) all states from which at least one arc leads to t contain a token, i.e. the marking contains all states from which at least one arc leads to t.
- when a collaborator triggers a transition, 1) he/she performs the associated activity, 2) the marking is updated, i.e. the tokens from input states (those states from which at least one arc leads to t) are removed, and new tokens are created in output states (those states to which at least one arc comes from t).

3.4 Meta-Processes

The concept of *meta-process* is our answer to the adaptation requirement. During the execution of an instantiated social protocol, collaborators may identify a need for modification of the process instance they are involved in. As a consequence, collaborators need to interact to decide how the process should be changed. A meta-process is a social process associated with another social process π allowing collaborators of π to decide in a structured collaborative way how the process π should be modified.

Formally, a meta-process μ is a pair π, π^{α}, where both π, the to-be-adapted process, and π^{α}, the adaptation process, are social processes, share the same set of collaborators C. The social process π^{α} rules H2H interactions concerning changes to be performed in π.

Depending on the type of changes provided during the meta-process, five situations (summarized in Table 1) may be distinguished:

- Level 1 changes – *role attributions*: a meta-process may result in changes in the role-collaborator mapping. For instance, "Susan Doe" may replace "John Smith" as the brainstorming chairman. Such changes have an influence only on the instantiated process model;
- Level 2 changes – *activity implementation*: a meta-process may result in changes in the mapping of abstract activities to implemented activities. For instance, instead of publishing new ideas on a given forum, collaborators may decide to publish their ideas using a mailing list at a given address. Such changes imply not only modifications of the instantiated process model, but also modifications of the associated implemented social model;
- Level 3 changes – *structural simplification*: a meta-process may result in changes towards the simplification of the structure of the interaction protocol, with activities, states, transitions, roles or edges removed from the interaction protocol. For instance, collaborators may decide that the summary of the brainstorming session is not required. Such changes imply modifications of the instantiated, the associated implemented, and the abstract social protocol. However, no new implementation has to be provided.
- Level 4 changes – *structural modification*: a meta-process may result in changes towards the modification of the structure of the interaction protocol such that no new activity, state, or role have to be defined. Edges and transitions can be freely modified to reflect a new organization of states, activities and roles. For instance, collaborators may decide that any participant may summarize brainstorming session, and not only the chairman. Such changes imply here also modifications of the instantiated, the associated implemented, and the abstract social protocol. However, no new implementation has to be provided.
- Level 5 changes – *structural extension*: a meta-process may result in changes towards the extension of the structure of the interaction protocol by the addition of new activities, states, or roles. Edges and transitions can be freely modified to reflect a new organization of states, activities and roles. For instance, collaborators may decide that the classification of the ideas should

be accepted by an "observer". This change implies the creation and the attribution of the role of "observer", the creation of two new activities "accept classification" and "reject classification", as well as the choice of the implementation of these activities, and finally the creation of appropriate transitions. Such changes imply here also modifications of the instantiated, the associated implemented, and the abstract social protocol. Implementation of newly added activities has to be provided, and the role-collaborator mapping has to be redefined for newly created roles.

Table 1. The effects of various change types on the social process, the implemented social protocol, and the abstract social protocol, and the potential need for a redefinition of activities implementation

Level	Type	Instance	Implemented	Abstract	Implementation
1	role attributions	X			
2	activity implementation	X	X		X
3	structural simplification	X	X	X	
4	structural modification	X	X	X	
5	structural extension	X	X	X	X

4 Discussion

Some interesting works have been done in the field of electronic negotiations to model electronic negotiations with the help of negotiation protocols. In [16], it is stated in that, in the field of electronic negotiations, "the protocol is a formal model, often represented by a set of rules, which govern software processing, decision-making and communication tasks, and imposes restrictions on activities through the specification of permissible inputs and actions". One may notice the similarity with the concept of social protocol. The reason for this fact is that the model presented in this paper was originally coming from a work on protocols for electronic negotiations [21]. However, to our knowledge, none of the works concerning negotiation protocols provides support for the modeling of the social environment. Moreover, these works are by nature limited to the field of electronic negotiations which is just a subset of the field of H2H interactions.

As process modeling is concerned, many works have already been conducted in the research field of workflow modeling and workflow management systems. Many works [1, 2, 25, 12, 13] have focused on formal models and conditions under which a modification of an existing – and potentially running – workflow retains workflow validity, the ADEPT2 project[7] being probably the most advanced one. However, to our best knowledge, current works concerning workflow adaptation focus on interactions, and the importance of social aspects are not or insufficiently taken into account by these works.

Sadiq and al.[24] have proposed an interesting model for flexible workflows, where flexibility refers to "the ability of the workflow process to execute on the basis of a loosely, or partially specified model, where the full specification of the model is made at runtime, and may be unique to each instance." However, support for flexibility does not ensure support for adaptability, as flexibility, as proposed by Sadiq and al., implies that the workflow designer has specified at design time frames and boundaries to possible modifications of the workflow.

5 Conclusion

While many works are currently done on modeling collaboration processes in which software entities (agents, web services) are involved, modeling collaboration processes in which mainly humans are involved is an area that still requires much attention from the research community. Some of the main issues to be addressed are the social aspects of collaboration and the adaptation capabilities of humans. In this paper, the requirements of computer support for agile H2H interactions are presented. Additionally, the concept of social protocol, combining social networks and workflow models, is proposed as a model supporting agile H2H interactions.

The main innovations presented in this paper are 1) the requirements for agile H2H interactions, 2) the refinement of the concept of social protocol by the addition of the concept of social network as a way to model the social environment, and 3) the three-layer view on social protocols – abstract, implemented, and instantiated – and the concept of meta-process.

A prototype, based on Dyng [14], is currently under implementation to validate the model presented in this paper. Among future works, methods to update the social network to reflect H2H interactions performed in a given process are still to be proposed.

References

1. van der Aalst, W. M. P.: The Application of Petri Nets to Workflow Management. J. of Circuits, Systems and Computers. 8(1), 21–66 (1998)
2. van der Aalst, W. M. P., Basten, T., Verbeek, H.M.W., Verkoulen, P.A.C., Voorhoeve, M.: Adaptive Workflow: On the Interplay between Flexibility and Support. In: Filipe, J. (ed) Proc. of the 1^{st} International Conference on Enterprise Information Systems, vol. 2, pp. 353–360. Kluwer Academic Publishers (1999)
3. van der Aalst, W.M.P., van Hee, K.M., van der Toorn, R.A. Component-Based Software Architectures: A Framework Based on Inheritance of Behavior. BETA Working Paper Series, WP 45, Eindhoven University of Technology, Eindhoven, `http://wwwis.win.tue.nl/~wvdaalst/publications/p108.pdf` (2000)
4. van der Aalst, W. M. P., Weske, M., Wirtz, G.: Advanced Topics in Workflow Management: Issues, Requirements, and Solutions. J. of Integrated Design and Process Science. 7(3), 47–77 (2003)

5. van der Aalst, W. M. P., van Hee, K.: Workflow Management: Models, Methods, and Systems (Cooperative Information Systems). The MIT Press (2004)
6. Boyd, D.M., Ellison, N.B.: Social Network Sites: Definition, History, and Scholarship. J. of Computer-Mediated Communication. 13(1), 210–230 (2007)
7. Dadam, P., Reichert, M.: The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support. Technical Report. Fakultät für Ingenieurwissenschaften und Informatik, Ulm, http://dbis.eprints.uni-ulm.de/487/1/Reichert_01_09-2.pdf (2009)
8. Facebook, http://www.facebook.com/
9. Fisher, L.: 2007 BPM & Workflow Handbook. Future Strategies Inc. (2007)
10. Friendster, http://www.friendster.com/
11. Harrison-Broninski, K.: Human Interactions: The Heart And Soul Of Business Process Management: How People Really Work And How They Can Be Helped To Work Better. Meghan-Kiffer Press (2005)
12. ter Hofstede, A.H.M., Orlowska, M.E., Rajapakse, J.: Verification Problems in Conceptual Workflow Specifications. Data Knowledge Engineering. 24(3), 239–256 (1998)
13. ter Hofstede, A.H.M., Orlowska, M.E., Rajapakse, J.: Verification Problems in Conceptual Workflow Specifications. In: Thalheim, B. (ed.) Conceptual Modeling - ER'96, 15^{th} International Conference on Conceptual Modeling, Cottbus, Germany, October 7-10, 1996. LNCS 1157, pp. 73–88. Springer (1996)
14. Huriaux, T., Picard, W.: DynG: a Multi-protocol Collaborative System. In: Funabashi, M., Grzech, A. (eds.) Proc. of the 5^{th} IFIP International Conference on e-Commerce, e-Business, and e-Government (I3E 2005), pp. 591–605, Springer (2005)
15. Jeston, J., Nelis, J.: Business Process Management, Second Edition: Practical Guidelines to Successful Implementations. Butterworth-Heinemann (2008)
16. Kersten, G.E., Strecker, S.E., Lawi, K.P.: Protocols for Electronic Negotiation Systems: Theoretical Foundations and Design Issue. In: Proc. of the 5^{th} Conference on Electronic Commerce and Web Technologies (ECWeb04), pp. 106–115, IEEE Computer Society (2004)
17. LinkedIn, http://www.linkedin.com/
18. MySpace, http://www.myspace.com/
19. Orkut, http://www.orkut.com/
20. Picard, W.: Modeling Structured Non-monolithic Collaboration Processes. In: Camarinha-Matos, L., Afsarmanesh, H., Ortiz, L. (eds.) Collaborative Networks and their Breeding Environments, the 6th IFIP Working Conference on Virtual Enterprises PRO-VE 2005, pp. 379–386, Springer (2005)
21. Picard, W., Huriaux, T.: DynG: A Protocol-based Prototype for Non-monolithic Electronic Collaboration. In: CSCW in Design 2005. LNCS 3865, pp. 41–50 (2006)
22. Picard, W.: Computer Support for Adaptive Human Collaboration with Negotiable Social Protocols. In: Abramowicz, W., Mayr, H.C. (eds.) 9th International Conference on Business Information Systems BIS 2006. Lecture Notes in Informatics, vol. P-85, Gesellschaft fur Informatic, pp. 90–101 (2006)
23. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P. Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, http://www.workflowpatterns.com/documentation/documents/Resource%20Patterns%20BETA%20TR.pdf (2004)
24. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Information Systems. 30(5), 349–378 (2005)
25. Sadiq, S.W., Orlowska, M.E.: Analyzing process models using graph reduction techniques. Information Systems. 25(2), 117–134 (2000)