Architecture of an ERP System Supporting Project-Oriented Management

Willy Picard, Grzegorz Wojciechowsk
Department of Information Technology
The Poznań University of Economics
ul. Mansfelda 4
60-854 Poznań, Poland
{picard,gwojc}@kti.ae.poznan.pl
http://www.kti.ae.poznan.pl/

Abstract. Existing ERP systems provide an IT solution to the management of enterprise resources based on the function-oriented management approach. With an increasingly wide adoption of the project-oriented management, new models are needed for ERP systems to support the management of enterprise resources in a project-oriented manner. This paper presents an architecture of an ERP system supporting project-oriented management. Two characteristics of the project-oriented management are integrated in the proposed architecture: first, social protocols are used to model interactions between actors (humans or software agents) within a given group. Second, the concept of group actions is detailed as a way to integrate group dynamics to social protocols.

1 Introduction

Enterprises are increasingly using ERP systems in all areas of their business activity to improve their business processes. The main reason for that is to achieve value-added differentiation over competitors , to ensure brand awareness, and client satisfaction. ERP systems aim at providing an integrated solution to the management of resources of the enterprise. Current ERP systems aspire to support all tasks required to achieve operational goals of the enterprise.

In management theory, two approaches to management of operational goals may be distinguished: function-oriented and project-oriented management.

The function-oriented management is usually used in environments where a set of relatively simple tasks are frequently performed. The function-oriented management implies that tasks are handled in a routine manner where each employee has his/her own function in achieving operational goals. A manager does not coordinate the execution of tasks for each goal, employees just react on the incoming documents, phone calls, etc., by completing tasks they are assigned. In existing ERP

systems, the function-oriented management is supported via *data-flow engine*. The data-flow engine, which is the core of ERP systems, is responsible for managing the co-peration of ERP modules by providing modules with appropriate data, potentially from other modules.

The project-oriented management is usually used when the achievement of operational goals required the coordinated interactions of various persons possessing different skills. In a project-oriented management approach, a project manager usually supervises the work being done. The project-oriented management implies that tasks are performed within groups where employees are cooperating to reach a common goal. Within a given group, employees are usually assigned with various roles depending on the skills and/or the position of a given employee. Depending on their role, employees may perform different tasks.

As a consequence of the increasing complexity of business interactions, enterprises are moving from function-oriented management to project-oriented management. The lack of support for project-oriented management in ERP systems is currently a real obstacle to a wide adoption of project-oriented management by enterprises, and therefore an obstacle to their efficiency and competitiveness.

Existing ERP systems improve business processes especially by supporting employees to perform single tasks effectively. Due to many years' enhancement of data-flow engines, employees may execute tasks in an efficient way. However, the solutions applied in data-flow engines should also be available in ERP systems supporting project-oriented management.

In project-oriented management a project manager needs to coordinate activities performed by employees and software agents. In regard to ERP systems, they support coordination and orchestration of business process activities but not sufficiently to entirely take advantage of project-oriented management. Workflow solutions[5][3] or business process execution solutions such as BPEL[1] to automate business processes are often offered but a critical element of project-oriented management remains missing, i.e. support of *collaboration with group management*.

In our opinion, the following areas concerning business process improvement are crucial to implement project-management support in ERP systems: activity efficiency which is human-to-machine interactions; coordination and orchestration called machine-to-machine interactions; and collaboration which is human-to-human interactions (Figure 1).

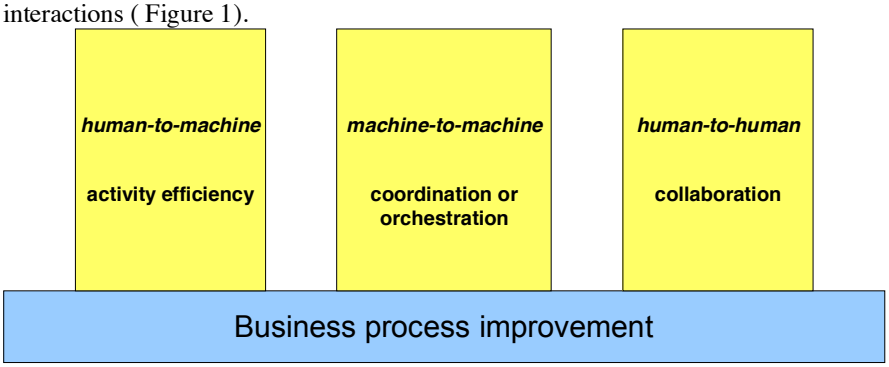

Fig. 1. Pillars of business process improvement

As it has been already stated, existing ERP systems concentrate their efforts on improving human-to-machine and machine-to-machine interactions. The proposed architecture of an ERP system supporting project-oriented management is built on the architecture of existing ERP systems. Therefore, the solutions of human-to-machine and machine-to-machine interactions are integrated into the presented architecture. The remaining problem of collaboration with group management is addressed by social protocols and group actions concepts.

The rest of this paper is organized as follows. In Section 2, the concept of *social protocol*, used to model collaboration processes, is presented. In Section 3, group management in collaboration processes is described. Next, the architecture of ERP system supporting project-oriented management is presented. Finally, Section 5 concludes the paper.

2 Social Protocol Definition

In the project-oriented management approach, all tasks required to achieve operational goals are performed within groups, as a result of interactions among actors, humans or software agents. Efficiency of the work may be improved by structuring interactions among actors. Therefore, a model for structured interactions is required. A main contribution of this paper is the concept of *social protocols* which are used to model structured collaboration processes.

A social protocol is a formal definition of possible actors-to-actors interactions. A social protocol has to capture characteristics of human-to-human, machine-to-machine, and human-to-machine interactions. Interactions are strongly related with social aspects, such as the role played by actors. The proposed model integrates the concept of role, which may explain the choice of the term “social protocols”.

Roles need to be integrated to social protocols as various actors may play different roles. Depending on its role, an actor may perform different tasks. For instance, during the decision-making concerning a delivery date for an order, two humans may collaborate: one of them plays the role of a shipping company representative, while the second person plays the role of a worker responsible for servicing client’s order. A role r is a label which is assigned to an actor. Let’s denote R the set of roles existing in a given social protocol.

Interactions among actors are modeled with the concept of *action*. An action a is an execution of a software entity. The software entity is a web service used to call an external program. The execution of actions is a part of the common knowledge of the group, i.e. all actors are aware of the execution of an action by one of the members of the group. Depending on the fact that an actor executes actions or not we differentiate two types of actors: *passive actors* which only monitor the execution of the social protocol or *active actors* which perform actions. A passive actor may be a client of the company which observes the decision-making concerning the production date of ordered goods. Let's denote A the set of actions available in a given social protocol.

Each action may be associated with *metadata*. Metadata m are information about their associated action. Metadata consist of two parts: a metadata type and a metadata content. The two-fold aspect of metadata found its origin in the speech act

theory by John Searle[14]. In the speech act theory, an utterance consists of both a propositional content and an illocutionary force. The illocutionary force of an utterance specifies the purpose of the actor. Similarly, metadata for social protocols consist of a metadata content and a metadata type which explicitly specifies the purpose of the actor.

Two kinds of metadata may be distinguished: unstructured and structured metadata. Unstructured metadata are metadata with unstructured metadata content. Unstructured metadata are typed with a content potentially written in natural language. Unstructured metadata are adapted to human-to-human communication. An example of an unstructured metadata could be an explanation for why ordered goods can not be produced at a specific date. In this metadata, "explanation" may be the metadata type, while "our plants are overbooked till the end of the month" may be the unstructured content. Such unstructured metadata may cause the worker responsible for servicing client's order to propose a new delivery date. Structured metadata are metadata with structured metadata content. Structured metadata are typed with a structured content. Structured metadata are adapted to machine-to-machine communication. Let's denote M_t the set of metadata types available in a given social protocol.

Triplets *(role, action, metadata type)* are called *behavioral units*. The concept of behavioral unit comes from the idea that the behavior of an actor is to a large extent determined by the role he/she plays, the actions she/he may perform and the type of metadata she/he may send. Therefore, roles, actions, and metadata types have to be associated to determine the behavior that an actor playing a given role should expose. Let's denote BU the set of potential behavioral units. Formally, $BU=R\times A\times M_t$.

One may say that a behavioral unit is executed. A behavioral $bu=(r,a,m_t)$ is said to be executed if an actor playing the role r executes action a, while sending a metadata with type m_t. It should be noticed that only actors playing the role r can execute the behavioral unit $bu=(r,a,m_t)$. Examples of behavioral units that may be executed during the decision-making process concerning the establishment of a delivery date for ordered goods are:

bu = (worker responsible for servicing client's order, propose date, $\varnothing$)
bu = (shipping company representative, accept date, information)
bu = (shipping company representative, reject date, counter-offer)

A state s is a label associated with a given situation in a collaborative process. Let's denote S the set of states that may occur in a given social protocol.

A transition t is a triplet $(bu, s_{source}, s_{destination})$. Let's denote T the set of transitions that may occur in a given social protocol. Formally, $T=BU\times S\times S$.

A ***social protocol*** p is a finite state machine consisting of $\{ S_p, S_p^{start}, S_p^{end}, Tp\}$ where S_p is the set of states, $S_p^{start} \subset S$ is the set of starting states, $S_p^{end} \subset S$ is the set of ending states, $S_p^{start} \cap S_p^{end} = \varnothing$, and T_p is the set of transitions from states to states.

Following a given social protocol, actors are "moving" from state to state via the execution of behavioral units. In other words, the execution of behavioral units are transition conditions. As mentioned before, a behavioral unit may be executed only by an actor playing

the appropriate role. The conditions that protocols have to fulfill to be valid, both structurally and semantically have already been presented in [9].

The last concepts related with social protocols are *role-to-actor mapping* and *social protocol instance*. A social protocol is a model for a class of collaboration process. A given collaboration process may be structured according to a given social protocol on the condition that the following additional data are known:

- the current state in which the collaboration process is,
- the role-to-actor mapping which associated at least one actor with each role specified in the social protocol.

The role-to-actor mapping is related to the current state, as actors may play various roles during the collaboration process. A ***social protocol instance*** is a triplet $(P, RAM, S_{current})$, where P is a given social protocol, RAM is the role-to-actor mapping, and $S_{current}$ is the current state.

3 Group Management in Collaboration Processes

A social protocol models interactions among actors within a given group. However, in the project-oriented management approach, the work related with the achievement of the operational goals is usually performed within many groups. Moreover, the interactions taking place within these groups are different, as roles, actions and metadata types may be different from group to group. For instance, brainstorming and negotiations are two classical techniques used during the realization of projects. Therefore, various social protocols may be involved in the realization of a single project, with some actors playing potentially many different roles depending on the group they are participating to at a given moment of time. The possibility for actors to modify protocols during the realization of a project has been presented in previous works [10,11].

Since interactions between humans take place in groups and various groups are created, modified, and destroyed during the realization of a project, social protocols have to support group management. Group management has to be designed to be interoperable with social protocols. The integration of group management and social protocols is required to be able to specify social protocols in which group creation, modification, and deletion may be seen as transitions from a given state to another one.

The proposed solution is based on the used of specific actions, called *group actions*. Group actions are actions – usable in social protocols – responsible for group management. Therefore, all actions that may be used to modify the set of groups related with the realization of a given project are group actions.

The following group actions have been identified:

Join – adds at least one collaborator to the set of collaborators of an existing group. Formally:

Join: $RAM \rightarrow RAM'$, where $\exists_{RAM'' \subset RAM'} : RAM = RAM' - RAM''$

Quit – removes at least one collaborator from the set of collaborators of an existing group. Formally:

Quit: *RAM*→*RAM'*, where $\exists_{RAM'' \subset RAM} : RAM - RAM'' = RAM'$

Split – splits an existing group in two or more new groups and the union of the set of collaborators of the created group equals the set of collaborators of the existing group. Formally:

Split: *RAM*→*RAM*$_1$, *RAM*$_2$, …, *RAM*$_n$, where $RAM = (R, A_c)$, with A_c denoting the set of actors involved in a given social protocol instance, $\forall_k RAM_k = (R_k, A_{c,k})$, and $A_{c,1} \cup A_{c,2} \cup ... \cup A_{c,n} = A_n$

Merge – creates a new group consisting of the union of the set of collaborators of at least two groups. Formally:

Merge: *RAM*$_1$, *RAM*$_2$, …, *RAM*$_n$ →*AA*, where $RAM = (R, A_c)$, $\forall_k RAM_k = (R_k, A_{c,k})$ and $A_{c,1} \cup A_{c,2} \cup ... \cup A_{c,n} = A_n$

Create – creates new group. Formally:

Create: $\varnothing \rightarrow RAM$

End – deletes an existing group. Formally:

End: $RAM \rightarrow \varnothing$

ChangeRole – change role of at least one collaborator in an existing group. Formally:

ChangeRole: *RAM*→*RAM'*, where $RAM = (R, A_c)$ and $RAM' = (R', A_c')$, $A_c = A_c'$, and $\exists_{(r', a_c') \in RAM'} : \forall_{(r, a_c) \in RAM} (r', a_c') \neq (r, a_c)$

The presented list of group actions is not exclusive. Other group actions may be defined. However, the group actions proposed above address the most common actions related with group management.

4 ERP System Architecture

The proposed architecture of an ERP system supporting project-oriented management is based on the concepts of social protocol and group actions. A diagram presenting the proposed architecture is given in Figure 2.

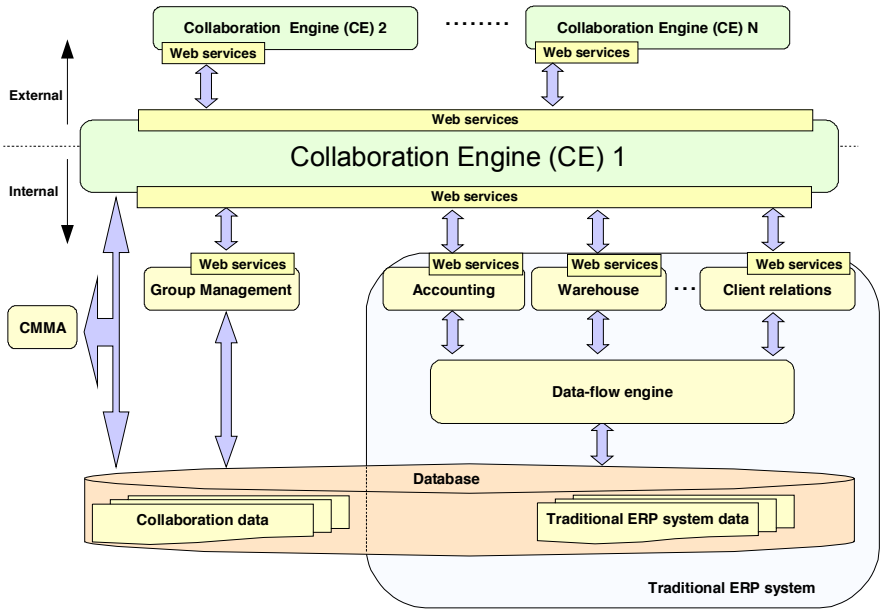

Fig. 2. Architecture of an ERP system supporting project-oriented management

The proposed architecture consists of the following elements:

Database – stores data of a traditional ERP system necessary to manage an organization. Additionally the database stores data specific to collaboration, i.e. group management data (existing groups, history of groups), social protocols defined in an ERP system, running instances of social protocols and history of finished instances of social protocols.

Group management module– provides features related with the interactions between the collaborative engine and the social protocols stored in the database;

ERP system modules – supply the ERP system functions as actions to social protocol;

Collaboration Engine (CE) – parses definitions of social protocols and executes instances of social protocols. In the ERP system, some special actions (for example ordering goods by a client) may trigger the creation of a new instance of a social protocol. Despite that the engine takes care of persistence, queues, and other execution details.

Collaboration Management, Monitoring and Analysis (CMMA) – supports management of social protocol definitions and instances, supports traceability of actual instances to find bottlenecks, supports analysis of finished instances of social protocols to detect friction points, enabling modifications of social protocols to improve the execution of future instances.

The communication between Collaboration Engine and all modules relies on web services to facilitate interoperability and integration. There are two types of internal communication, i.e. communication that take place inside one ERP system: first, communication may take place between the CE and ERP modules used to call services of an ERP system functions. Second, communication may take place between the CE and the Collaboration module used to call group actions of a social protocol.

The CE may also communicate with external ERP systems, specifically CE 1 may call actions of external CEs as it is shown in the Figure 2. The external communication is especially applicable in a case of collaboration among actors from two groups of different ERP systems. The 'join' and 'split' group actions are used to start and finish collaboration respectively.

Actions invoked by an actor in a social protocol may be executed in either a synchronous or an asynchronous way. In the synchronous scenario, results of the execution of an action are immediately returned as an output message of invoked service. In the asynchronous scenario, the architecture has to support the push model and/or the pull model for asynchronous communication. In the push approach, the Collaboration Engine looks up in the database for the results of former action executions at a given interval of time. The push approach implies a high load of the network because of the polling of the database, but ERP system modules do not need to know about existence of the Collaboration Engine. In the pull approach, the Collaboration Engine has to expose a callback interface (marked as web service interface in Figure 2). Once the results are available, they are returned from an ERP system module via callback invocation on the Collaboration Engine. The usage of the pull model minimizes a network load but ERP system modules have to know how to return action results. A pull model is a solution well adapted to the case of asynchronous communication between different CEs and may be implemented using WS-Addressing[15] standard or a mechanism similar to the correlation sets concept used in the BPEL specification.

5 Discussion

As process modeling is concerned, many works have already been conduced in the research field of workflow modeling and workflow management systems. Paul Buhler and Jose M. Vidal [2] proposed a mechanism allowing for enacting workflows in an adaptive way using multi-agent systems (MAS). Robert Müller and al. presented in [8] various mechanisms for adaptation of workflows to deal with exception occurrences in running workflow instances, with an application to medical treatments. However, to our best knowledge, current works concerning workflow adaptation focus on interactions among software entities. Characteristics of interactions between humans, such as the importance of social aspects, are not or insufficiently taken into account by these works. Moreover, these works are lacking support for group management.

Some interesting works have been done in the field of electronic negotiations to model electronic negotiations with the help of negotiation protocols. In [7], 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 [12,13]. However, these works are by nature limited to the field of

electronic negotiations which is just a subset of the field of human collaboration, and may not be applied directly to ERP systems.

6 Conclusion

While the function-oriented management is currently well supported in ERP systems, the project-oriented management lacks support in ERP systems. In this paper, an architecture for ERP systems supporting project-oriented management is presented. The proposed architecture is based on the concept of social protocols. The concept of social protocol aims at being a start of the answer to the question of computer support for social collaboration. The introduction of group actions allows to support group dynamics, i.e. structured collaboration "spread" in a dynamic way within many groups.

The main innovations presented in this paper are 1) the concept of social protocols, integrating social aspects with roles, communication aspects with metadata, and structured collaboration based on the use of behavioral units as transitions in a finite state machine, 2) the concept of group actions which allows to integrate group creation, modification, and deletion to social protocols, 3) an architecture for ERP systems integrating support for social roles and using web services as an interoperability mean. The proposed concepts are currently under implementation as extensions to the *DynG* protocol [6], a social protocol-based platform.

The next steps will include a refinement of the concept of role, so that relationships between roles, e.g. specialization, compositions, may be integrated to the presented model. Another area to be investigated is the adaptation of social protocols, so that actors may modify a social protocol to tailor it to their own needs at run-time.

References

1. BEA, IBM, Microsoft, SAP AG, and Siebel Systems, *Business Process Execution Language for Web Services*, BPEL4WS V1.1 specifications (2003) ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

2. P. Buhler and J. M. Vidal, "Towards Adaptive Workflow Enactment Using Multiagent Systems", *Information Technology and Management Journal, Special Issue on Universal Enterprise Integration*, **6** (1), 61-87 (2005).

3. J. Cardoso, R. P. Bostrom and A. Sheth, "Workflow Management Systems and ERP Systems: Differences, Commonalities, and Application", *Information Technology and Management* **5**(3/4), 319-338 (2004).

4. H. Chen, T. Finin and A. Joshi, An Ontology for Context-Aware Pervasive Computing Environments, *Knowledge Engineering Review, Special Issue on Ontologies for Distributed Systems*, **18**(3), 197-207 (2003).

5. L. Da-You, Z. Zhao-Hui, L. Wei-Jiang, L. Zhan-Shan, and S. Cheng-Min, An improved workflow management system for ERP, in: *Proceedings of 2005 International Conference on Machine Learning and Cybernetics* (IEEE, Piscataway, 2005), pp. 3618-3623.

6. T. Huriaux and W. Picard, DynG: a Multi-protocol Collaborative System, in: *Proceedings of the 5th IFIP International Conference on e-Commerce, e-Business, and e-Government (I3E): Challenges of Expanding Internet: e-Commerce, e-Business and e-Government, Poznan, Poland, October 26-28, 2005*, edited by M. Funabashi and A. Grzech (Springer, Berlin and Heilderberg, 2005), pp. 591-605.

7. G. E. Kersten, S. E. Strecker and K. P. Law, Protocols for Electronic Negotiation Systems: Theoretical Foundations and Design Issue, in: *Proceedings of the 5th Conference on Electronic Commerce and Web Technologies (EC-Web04), Sarragoza, Spain, Lecture Notes in Computer Science volume 3182* (Springer, Berlin, Heidelberg, 2004), pp. 106- 115.

8. R. Müller, U. Greiner, and E. Rahm, AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation, *Data and Knowledge Engineering* **51**(2), 223-256 (2004).

9. W. Picard, Towards Support Systems for Non-Monolithic Electronic Negotiations. The Contract-Group-Message Model, *Journal of Decision Systems, Electronic Negotiations - Models, Systems and Agents*, **13**(4), 423-439 (2004).

10. W. Picard, Adaptive Collaboration in Professional Virtual Communities via Negotiations of Social Protocols, in: *Proceedings of the 7th IFIP Working Conference on Virtual Enterprises (PRO-VE 2006)*, Helsinki, Finland, September 25-27, 2006, forthcoming.

11. W. Picard, Computer Support for Adaptive Human Collaboration with Negotiable Social Protocols, in: *Proceedings of 9th International Conference on Business Information Systems in cooperation with ACM SIGMIS (BIS)*, Klagenfurt, Austria, May 31 – June 2, 2006, forthcoming.

12. W. Picard, Modeling Structured Non-monolithic Collaboration Processes, in: *Collaborative Networks and their Breeding Environments. Proceedings of the 6th IFIP Working Conference on Virtual Enterprises (PRO-VE 2005), Valencia, Spain, September 26-28, 2005*, edited by L. Camarinha-Matos, H. Afsarmanesh and A. Ortiz, (Springer, Boston, MA, 2005), pp. 379-386.

13. W. Picard and T. Huriaux, DynG: A Protocol-Based Prototype for Non-monolithic Electronic Collaboration, *CSCW in Design 2005, Lecture notes in Computer Science volume 386,* edited by W. Shen et al., (Springer, Berlin, Heidelberg 2006), pp. 41-50.

14. J. R. Searle, *Speech Acts - An Essay in the Philosophy of Language*. (Cambridge University Press, Cambridge, 1969).

15. Web Services Addressing (WS-Addressing) http://www.w3.org/Submission/ws-addressing/