AN ORGANISATIONAL MODEL FOR SIMPLIFYING THE COMPLEXITY OF MANAGING SOFTWARE PROJECT[1]

Mourad ZELLOUF, Patrick PREVOT, Régis AUBRY
Laboratoire LISPI
INSA de Lyon, Bat 502
20, Avenue Albert Einstein,
69621 Villeurbanne Cedex. FRANCE
E-mail: zell@ifhpserv.insa-lyon.fr
Facsimile: 72 43 85 18

Abstract: The project success depends to a large degree on communication and coordination among team members. But project management systems and models do not support cooperative group work. Their basic philosophy is rather hierarchical and centralistic. It is the intention of this paper to present an Organisational Model for simplifying the complexity of managing software project, both at software project modelling aspect and the aspect of communication and coordination among project team members. In particular, it focuses on the following points: an activity is the main component of project management, the constraints between activities and resources must be established. A role defines a group of duties and responsibilities. The activity related communication proceeds by the exchange of products between roles. While the constraints coordinate the products flow between roles. The benefit of the practical use of the model is to reduce the coordination effort required of project team members, and thus to increase the productivity in software development project.

Keywords: Software Project Management, Organisational Model, Activity Model, Group Communication and Coordination, Computer Supported Cooperative Work (CSCW), Petri-Nets.

1. Introduction

With the advance of an information-oriented society, the need for software development has increased. The software products are becoming larger and more complex and that project teams are, and will continue to be, the dominant way of organising software production [6]. Therefore, co-operation between teams also becomes increasingly crucial for the project success. In [1] is suggested that software projects is a typically a collective human activity that involve communication among participants during development project. However, many systems have been developed to support project management and many models have been suggested to represent the structure of a project. These models focus mainly on the structure of work, as in the Work Breakdown Structure but do not support cooperative group work. Their basic philosophy is rather *hierarchical* and *centralistic*.

The aim of this work is to develop an *Organisational Model* which meets the requirements of software project management, and communication and coordination of a software project team. Over the last few years there has been a considerable amount of research devoted to the subject of Computer Supported Cooperative Work (CSCW). Many of area have been addressed within this field, ranging from the development of underlying technologies to support CSCW, through studies of group working in general, to the proposal of abstract frameworks for modelling group communication using computers. Our analysis is based on the two following projects that have addressed this issue:

[1] The reported work is part of TELECO3 project (which is under way) sponsored by Rhone-Alpes region as part of the Telepresence program.

☛ The AMIGO Advanced project [3] involved several European institutions, and was funded by the Cost-11ter programme. It concentrated on describing the applications of their AMIGO Activity Model (AAM) in designing concrete group communication activities.

☛ The Activity Model Environment (AME) [5] was developed by the MacAll II group at Nottingham University, and sponsored by Digital. The project aimed to develop a model of group working within an organisation based on the concept of an activity.

In addition, the study of others CSCW models and applications that support and coordinate communication in groups, presented in [3], helped us to improve our understanding of the requirements and needs of group communication. This paper is organized as follows: section 2 gives a description of the framework of the proposed *Organizational Model*. Section 3, outlines the existing implementation of the model, by using Cap Gemini's *Process Weaver* toolset. A conclusion and area for future work will be given in section 4.

2. The conceptual framework

The conceptual framework of the model encompasses the requirements of a software project management as well as some specific requirements of the existing CSCW models. We considered the following eight components of the framework:

✓ ***Activity:*** An activity is the work through which a worker uses and creates the products, such as documents, codes, etc. The formal description of an activity details the start-state and end-state(s) as well as the roles and products that will be involved in its performance. Any activity includes also the attributes which store the data required for project management.

✓ ***Product:*** A product is in input and output of an activity. It may contain informations produced by one person which is to be consumed by the receiving person(s). Products are also used to measure the progress of the activity in a quantitative way. A product should carry enough information to enable the receiving user to recognize what s/he is expected to do in order to continue the activity, i.e. to realize the requested manipulation and/or to produce the requested output.

✓ ***Resource:*** A resource is an entity that performs an activity or support it. It includes person resources and non-person resources such as tools and machines.

✓ ***Role:*** A role defines a set of duties and responsibilities that may be taken on by one or more people. Examples of roles are: project managers, designers, programmers, quality assurance managers, etc.

✓ ***Project:*** A project includes the activities, products and resources of the project. Project has also attributes, which are used for retrieving similar projects in existence.

We have adopted the *Activity Model* for software project modelling [7]. Thus, the activities are main components for project management. The structure of the project is described as *sets of activities*, whose partial ordering can be calculated from product items that are produced as output from certain activities and needed as input to others. Further *constraints* on how activities can be scheduled are introduced by used resources.

✓ ***Constraint:*** we have insisted in [7] that activities as well as the resources which perform and support activities should be specified and that the relationships between them must be *dynamically* established in order to represent the characteristics of a software project. Thus, four constraints on the relationships between an activity and its associated components have been identified. They are shown as follows:

☒ An activity has some constraints on its products which must be meet before the execution of that activity. We call these *'pre-condition'*.

☒ An activity has some constraints on its products which must be meet after the execution of that activity. We call these *'post-condition'*.
The pre-conditions and post-conditions of each activity can be compared to determine the relationships among the various activities, e.g. whether an activity is preceded by another activity, succeeded by another activity, etc.

The relationships between activities and resources have the following constraints:

☒ An activity may have some requirements about the qualifications and skills necessary for performing the activity. On the other hand, persons resources have the qualifications and skills (i.e. roles). When the project manager assigns an activity to someone, the qualifications and skills required by the activity and possessed by the person are compared and checked for consistency. This constraint is represented by the relationship *'engaged in'*.

☒ When tools and machines are used to execute an activity, this constraint is represented by the relationship *'supported by'*.

✓***Function:*** A function is performed by roles and products as part of an activity. Examples of functions are: send, receive, fill in, reply, evaluate, etc.

✓***Workspace:*** A workspace is a conceptual work area associated with a particular role (so a person playing several roles would use a different workspace for each role). It contains a collection of products, tools and access to external resources that are needed for successful performance of an activity. Conceptually, the workspace for each role is shared by all persons playing the similar role. Persons playing other roles may see the contents of the workspace, but they may not access them.

Workspace constitutes the basis concept of *cooperation modelling* among project team members. The activity related *communication* proceeds by the exchange of products between roles (via workspace), on which they may perform functions. The functions which roles must perform on products are not modeled and stay within the roles' responsibility. Thus privacy of the participants is not affected. The constraint component *coordinates* the flow of products between roles associated with an activity. The consequent part of the constrainte specifies *what* is to be done and *who* is to carry out the structuring works, that parts of it may be executed *concurrently* or must be executed *sequentially*. The persons can be addressed by their role name. The duties, rights and responsibilties of role should be expressed in the role specification. We assume that each team member may have more than one role, and that one role may be held by more than one team member, and that it is possible to *change* associations of roles to persons during project development, e.g. in case of illness.

3. Implementation

The proposed model has been implemented on a HP Workstation using the Cap Gemini's *Process Weaver* toolset **[2]**. The project structure is described as hierarchy of activities. An activity is based on process centred modelling. The process representation is done using *Petri-Nets* formalism, which models the organisation of work between the various roles performing activity. We used the *Process Weaver Cooperative-Shell (Co-Shell)* language for constructing scripts which are attached to conditions (i.e. pre-conditions and post-conditions) and actions. During the life of a process, in addition to linear action sequences which may occur, concurrent or alternative action bundles may also take place.
The *Co-Shell* language includes functions for broadcasting or catching events on the network, matching patterns of events and exchanging of objects (such as products, tools, etc.) among workspaces by using a Broadcast Message Server (BMS for short), invoking the standards UNIX Shells, etc. Workspace is modelled as a *work context* (using a work context graphical editor) which consists of a description of the activity to be done. The model has been implemented centrally on one workstation, however it can not be called a centralistic approach. Each *Petri-Net* for the activities can be processed on different workstations.

4. Conclusion

In this paper, we have proposed the suitable *Organisational Model* for simplifying the complexity of managing software project. The structure of project has been described as sets of activities. The constraints between activities and resources have been established. Roles are differentiated from people, thus allowing a separation of responsibilities between a role and the person playing it. The activity modelling is based on *Petri-Nets* formalism, which aims at describing work on organisation and team levels. The coordinated communication between roles is governed by the constraints. They define who should execute what, when, and by which means throughout the whole activity. These requirements were applied to the representation of the *Organisational Model*, which leads to an effective organisation of both individual and team work. For the team, it ensures better *coordination* and *just in time development*. We have described in **[8]** the multimedia courseware development project through the proposed model. Because this example project is typical feature of work distribution, and the degree of coordination of the skills and efforts of the courseware team is relevant factor affecting the courseware quality as well as the efficiency of production **[4]**. Thus, this modelling of group working will facilitate the subsequent development of the computer system, to support collaborative authoring and courseware development, which is under development.

References

1. ACM. 1993. Orchestrating Project Organization and Management. *Commun. ACM*, 36, 10 (93).
2. Cap Gemini Sogeti. 1994. Process Weaver. *User's Manual, Version PW 2.0.* Cap Gemini Sogeti.
3. Pankoke-babatz, U. 1989. *Computer Based Group Communication: The AMIGO Activity Model.* Ellis Horwood.
4. Prévôt, P. 1992. Un Tuteur Intelligent pour la Formation Indutrielle, Application à l'Intégration d'un Didacticiel Cimentier. *Workshop de Thunder Bay.* Ontario. pp.20-39.
5. Smith, H.T, Hennessy, P.A, Lunt, G.A. 1991. An Object-Oriented Framework For Modelling Organisational Communication. *In Studies in Computer Supported Cooperative Work*: Theory, Practice and Design, Eds J.M. Bowers & S.D. Benford, Elsevier. pp. 145-158
6. Ying, L. 1994. Treating Interactions Systematically: a Theme to Cope with an Aspect of the Complexity of Managing Life-Cycle Software Production. *Software Engineering Journal.* pp. 67- 82
7. Zellouf, M., Prévôt, P., Aubry, R. 1995. Groupware and Peopleware Aspects in Software Project. Will appear *in IASTED International Conference on Modelling and simulation*, Pittsburg, Pennsylvania, USA, April 27-29, 1995, 5 pages.
8. Zellouf, M., Prévôt, P., Aubry, R. 1995. Computer Supported Communication and Coordination in Collaborative development of Courseware. Will appear *in IEEE WESCANEX Conference on Communications, Power and Computing*, May 15-16, 1995. Winnipeg, Manitoba, Canada, 6 pages.