A PROCESS PATTERN LANGUAGE FOR COORDINATED SOFTWARE DEVELOPMENT

Chintan Amrit, René ter Haar, Mehmet N. Aydin and Jos van Hillegersberg
Department of IS&CM, University of Twente

Keywords: Organizational and process patterns, social networks, software engineering, globally distributed, software development.

Abstract: In distributed and collocated teams we often find problems in the organizational process structures. Though process patterns have been around for many years, there has been little research in categorizing the different solutions to various problems dealing with coordination, for easy access by practitioners. This study aims to describe a way to use the emerging idea of a pattern language to deal with problems related to coordination in software development. The patterns are a result of conclusive statements in the information systems and software engineering field and a pattern language is used to develop these patterns. We propose a technique to convert the knowledge base in IS and CS research on coordination into process patterns which are more accessible to practitioners.

1 INTRODUCTION

While there are many ways to describe a patterns, Christopher Alexander who originated the notion of patterns in the field of building architecture described patterns as a recurring solution to a common problem in a given context and system of forces (Alexander et al., 1977). In Software Engineering patterns are attempts to describe successful solutions to common software problems (Schmidt et al., 1996). Software Patterns reflect common conceptual structures of these solutions and can be used repeatedly when analyzing, designing and producing applications in a particular context. Patterns represent the knowledge and experience that underlie many redesign and re-engineering efforts of developers who have struggled to achieve greater reuse and flexibility of their software. The different types of patterns are:

Design Patterns: Are simple and elegant solutions to specific problems in object oriented design (Gamma et al., 1995).

Analysis Patterns: Capture conceptual models in an application domain in order to allow reuse across applications (Fowler, 1997).

Organizational Patterns: Describe the structure and practices of human organizations (Coplien & Harrison, 2004).

Process Patterns: Describe the Software Design Process (Coplien & Schmidt, 1995).

Patterns are most generally represented in natural language and are typically published in printed catalogues. Pattern presentation is generally loosely structured and consists of a series of fields each having a meaning introduced via an informal definition or description. An example of such a structure representing patterns can be found in Table 1.

Table 1: The Pattern framework(based on (Coplien, 1994)).

Field Explanation/Definition
Name: Ideally a meaningful name that will be part of the shared design vocabulary. Many existing patterns do not satisfy this requirement for historical reasons.
Problem: A problem growing from the Forces
Context: The current structure of the system giving the context of the problem
Forces: Forces that require resolution
Solution: The solution proposed for the problem
Resulting Context: Discusses the context resulting from applying the pattern. In particular, trade-offs should be mentioned
Design Rationale/Related patterns: The design rationale behind the proposed solution. Patterns are often coupled or composed with other patterns, leading to the concept of pattern language.

Amrit C., ter Haar R., N. Aydin M. and van Hillegersberg J. (2007).
A PROCESS PATTERN LANGUAGE FOR COORDINATED SOFTWARE DEVELOPMENT.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - HCI*, pages 298-301
DOI: 10.5220/0002385102980301

Though a lot of literature exists on coordination in software development (Crowston, 1997; Herbsleb & Grinter, 1999; Kraut & Streeter, 1995; Parnas, 1972), there is no place where both researchers and practitioners can look up solutions to known problems dealing with coordination in software development. This study aims to provide a framework by which we can bridge the gap in literature, dealing with problems of Coordination in Software Development. In this research we have tried to convert information systems knowledge (especially those dealing with social networks of teams and their tasks) into organizational patterns that can be used for solving problems related to coordination in software development. The newly developed organizational patterns of this study are related to social networks and processes within organizations.

The rest of the paper is organized as follows. Section 2 gives a brief overview of the literature on Coordination. The pattern language is described in section 3. In section 4 we discuss and summarize the results of this study and mention some recommendations for further research.

2 COORDINATION IN SOFTWARE DEVELOPMENT

Coordination can be defined in the following ways:

"integrating or linking different parts of an organization to achieve a collective set of tasks" (Ven et al., 1976)

In software development, it means that different people working on a common project agree to a common definition of what they are building, share information, and mesh their activities. To build software efficiently, they must share detailed specifications and information about the progress of software modules. In sum, they must coordinate their work so that it gets done and fits together, so that it isn't done redundantly, and so that components of the work are handed off expeditiously (Kraut & Streeter, 1995)

Coordination also focuses on managing interdependencies among multiple individuals or activities involved in the overall task (Crowston, 1997).

In their paper Kraut and Streeter (1995) mention scale of software projects, inherent unpredictability of software specifications and tasks as well as the Interdependence of software components as some of the factors that lead to the necessity of efficient co-ordination between the different work groups involved in the development process.

Practical experience and organizational theory suggest that previous efforts in software engineering have not solved the coordination problems in software engineering. The combination of large size, uncertainty and interdependence requires special coordination techniques that may not be necessary in more routine production environments (Kraut & Streeter, 1995).

Traditionally, most project management approaches for improving software development coordination have emphasised on one of the following three methods of technical innovations:

Development of new and enhanced methods and tools (Andres & Zmud, 2001; Crowston, 1997; Kraut & Streeter, 1995). Modularisation both technical (Object Oriented Programming) or managerial such as the organizational separation of requirements, coding and testing functions, to encapsulate the behaviour of program elements and individual software professionals, and thereby reduce the needs for coordination (Kraut & Streeter, 1995).

Formal procedures, both technical, such as version control software, case tools, and specification languages such as test plans, delivery schedules and requirements documents to control communication among development personnel. (Crowston, 1997; Kraut & Streeter, 1995)

While these techniques contributed to a modest increase in software productivity over the past twenty years, they only partially address the problem of coordination. (Kraut & Streeter, 1995)

A more recent approach has involved improved project management practices applied to software development process. These practices focus on improving task decomposition, task assignment and work group coordination, which are considered important issues in the context of Coordination (Andres & Zmud, 2001; Crowston, 1997; Kirsch, 1996). Malone and Crowston (1994) define *Coordination Mechanism* as the additional activities that the firm must perform to overcome this coordination problem. These coordination mechanisms may be specific to a particular setting, such as a code management system to control changes to software, or general, such as hierarchical or market mechanisms to manage assignment of activities to actors or other resources (Malone & Crowston, 1994). In this paper we concentrate on the aspect of coordination related to social networks (teams) and their tasks.

3 THE PATTERN LANGUAGE

A pattern language is a language that comprises patterns and the rules to put patterns together in meaningful ways, in a certain sequence(Coplien & Harrison, 2004). Coplien's Organizational Patterns (1994) provides a process pattern language for growing organizations; it doesn't concentrate on one aspect such as coordination. Further, the patterns mentioned here are based on papers taken from top IS journals. Hence, these patterns are backed by extensive and elaborate empirical validation.

This section represents the description of nine patterns related to team and tasks, and team performance. Each of the patterns is introduced by a short association to which the pattern is related to. The patterns are elaborated on basis of the concerning references in the descriptions. All of the patterns are successively discussed below.

1. **Hierarchical structure in projects with complex and non-routine tasks** (based on (Cummings & Cross, 2003))

2. **Core-periphery structure in projects with simple and routine tasks** (based on (Cummings & Cross, 2003))

3. **Group leader and structural holes** (based on (Cummings & Cross, 2003))

4. **Interdependence and conceptual tasks** (based on (Stewart & Barrick, 2000))

5. **Team self-leadership and conceptual tasks** (based on (Stewart & Barrick, 2000)),

6. **Interdependence, team self-leadership and behavioural tasks** (based on (Stewart & Barrick, 2000))

7. **Alignment between design interfaces and team interactions** (based on (Sosa et al., 2004))
Problem

8. **Alignment in design interfaces and its interrelated components** (based on (Sosa et al., 2004))

9. **Allocating Tasks in a Virtual Network** (based on (Ahuja et al., 2003)) , Managing shared resources

4 DISCUSSION AND CONCLUSION

In this research we have tried to convert information systems knowledge into organizational patterns, which can be used for solving problems related to coordination in software development. The newly developed organizational patterns of this study are related to social networks and processes within organizations, and especially related to social networks of teams and their tasks. Many of the patterns suggested by Coplien (1994) can be added to this coordination language, for example, the Conway's Law pattern, Code Ownership pattern, GateKeeper pattern, Bufallo mountain pattern, etc. We have left them out for the purposes of this paper in order to describe the process of extracting patterns from existing information systems literature.

Although these patterns have been tested as propositions in the papers they have been taken from, a more thorough testing of the patterns themselves could improve their reliability. When this has been done it would be possible to use them in development projects. While this study describes some patterns, more research is needed on patterns in order to develop a larger pattern language related to team and tasks, and team performance. Much IS literature is already available on these topics that can be translated into useful patterns.

Future research can work on extending this language with more useful and tested patterns in the field of coordination (related to team and tasks) in software development.

REFERENCES

Ahuja, M. K., Galletta, D. F. and Carley, K. M. (2003) Individual centrality and performance in virtual r\&d groups: An empirical study. *Manage. Sci.* 49 (1), 21-38.

Alexander, C., Ishikawa, S. and Silverstein, L. A. (1977) A pattern language. New York.

Andres, H. P. and Zmud, R. W. (2001) A contingency approach to software project coordination. *Journal of Management Information Systems* Vol. 18 (Issue 3), p41.

Borgatti, S. P. and Everett, M. G. (2000) Models of core/periphery structures. *Social Networks* 21, 375-395.

Burt, R. (1992) *Structural holes*. Harvard University Press, Cambridge, MA.

Campion, M. A., Medsker, G. J. and Higgs, A. C. (1993) Relations between work group characteristics and

effectiveness: Implications for designing effective work groups. *Personnel Psychology* 46 (4), 823-850.

Coplien, J., O (1994) A development process generative pattern language. *Proceedings of PLoP/94*, Monticello, Il., pp 1--33.

Coplien, J., O. and Harrison, N., B. (2004) Organizational patterns of agile software development. Upper Saddle River, NJ, USA.

Coplien, J. O. and Schmidt, D. C. (1995) Pattern languages of program design. New York, NY, USA.

Crowston, K. (1997) A coordination theory approach to organizational process design. *Organization Science* 8 (2), 157-175.

Cummings, J. N. and Cross, R. (2003) Structural properties of work groups and their consequences for performance. *Social Networks* 25, 197-210.

Eisenstat, R. A. (1990) *Fairfield coordinating group.* San Francisco.

Emery, J.-B. F. L. and Trist, E. L. (1969) *Socio-technical systems.* Penguin, London.

Fowler, M. (1997) Analysis patterns: Reusable object models. Reading MA.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) Design patterns: Elements of resuable object oriented software. MA.

Herbsleb, J., D. and Grinter, R., E. (1999) Architectures, coordination, and distance: Conway's law and beyond. *IEEE Softw.*, Los Alamitos, CA, USA, pp 63--70.

Kirsch, L. J. (1996) The management of complex tasks in organizations: Controlling the systems development process. *Organization Science* 7 (1), 1-21.

Krackhardt, D. (1994) Graph theoretical dimensions of informal organizations. In *Computational organization theory*, pp 89-111, Lawrence Erlbaum Associates, Inc.

Kraut, R., E. and Streeter, L., A. (1995) Coordination in software development. *Commun. ACM*, New York, NY, USA, pp 69--81.

Malone, T. W. and Crowston, K. (1994) The interdisciplinary study of coordination. *ACM Comput. Surv.* 26 (1), 87-119.

Mcgrath, J. E. (1984) *Group interaction and performance.* Prentice-Hall, Englewood Cliffs, NJ.

Parnas, D. L. (1972) On the criteria to be used in decomposing systems into modules. *Commun. ACM*, New York, NY, USA, pp 1053--1058.

Pearce, J. A., Iii and Ravlin, E. C. (1987) The design and activation of self-regulating work groups. pp 751-782.

Schmidt, D., Fayad, M. and Johnson, R. E. (1996) Software patterns. *Commun. ACM*, pp 37-39.

Sosa, M. E., Eppinger, S. D. and Rowles, C. M. (2004) The misalignment of product architecture and organizational structure in complex product development. *J Manage. Sci.* 50 (12), 1674-1689.

Stewart, G. L. and Barrick, M. R. (2000) Team structure and performance: Assessing the mediating role of intrateam process and the moderating role of task type. *The Academy of Management Journal* 43 (2), 135-148.

Trist, E. L. (1981) *The sociotechnical perspective: The evolution of sociotechnical systems as a conceptual framework and as an action research program.* Wiley, New York.

Ven, A. H. V. D., Delbecq, A. L. and Koenig, R., Jr. (1976) Determinants of coordination modes within organizations. *American Sociological Review* 41 (2), 322-338.