ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/220711688

A Process Pattern Language for Coordinated Software Development.

Conference Paper · January 2007

Source: DBLP

CITATIONS
0

READS
52

4 authors:

Chintan Amrit
University of Twente
67 PUBLICATIONS **303** CITATIONS
SEE PROFILE

René ter Haar
Marel Meat
4 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Mehmet N Aydin
Kadir Has University
40 PUBLICATIONS **359** CITATIONS
SEE PROFILE

Jos van Hillegersberg
University of Twente
159 PUBLICATIONS **2,170** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

DevOps View project

The Governance of Cloud-Based Collaboration View project

All content following this page was uploaded by Chintan Amrit on 27 May 2014.

The user has requested enhancement of the downloaded file.

A Process Pattern Language for Coordinated Software Development

Chintan Amrit, Rene ter Haar, Mehmet N. Aydin and Jos van Hillegersberg

University of Twente

The Netherlands

c.amrit@utwente.nl

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.

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

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.

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.

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

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 ten 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))

Problem

Which group structure can be best used in projects with complex and non-routine tasks where interdependence among group members is high?

Context

Group performance is inferior in projects with complex and non-routine tasks.

Forces

Team hierarchy can be based on status and prestige

Solution

In projects with complex and non-routine tasks a flat hierarchical structure (the concept of a hierarchical structure characterizes the extent to which relations are ordered, such as those determined by status or prestige, (Krackhardt, 1994)) is appropriate to improve group performance.

Resulting context

The use of a flat hierarchical structure in projects with complex and non-routine tasks improves group performance.

Design rationale

Choosing the right group structure in projects with complex and non-routine tasks.

This pattern is related to the 'core-periphery structure' and the 'group leader and structural holes' patterns.

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

Problem

Which group structure can be best used in projects with simple and routine tasks where interdependence among group members is low?

Context

Group performance is inferior in projects with simple and routine tasks.

Forces

Periphery members have a smaller contribution in the group outcome then core members do.

Solution

In projects with simple and routine tasks, and a lower interdependence a core-periphery structure (typically in this type of structure there exists a dense, cohesive core with a sparse, unconnected periphery. Though there may be multiple cores, it is often the case that a smaller subset of the total population participates more actively than the rest, (Borgatti & Everett, 2000)) is appropriate to improve performance. This structure is suitable since it holds the potential to improve speed and flexibility in diffusing information within a group.

Resulting context

The use of a core-periphery structure in projects with simple and routine tasks improves group performance.

Design rationale

Choosing the right group structure in projects with simple and routine tasks, can improve group performance.

This pattern is related to the 'flat hierarchical structure' and the 'group leader and structural holes' patterns.

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

Problem

How to improve group performance in projects with complex and non-routine tasks in which a group leader maintains an incorrect structure that consequently leads to structural holes (structural holes are

weaker connections between individuals in the social structure which create a competitive advantage for an individual who spans the holes, (Burt, 1992)), bad communication and a weak group performance?

Context

Projects with complex and non-routine tasks that are under control of a group leader dealing with structural holes and a bad group performance.

Forces

A group leader may choose to maintain an inappropriate group structure in a project. This leader may choose to create a structural hole within a group and enjoy the concurrent informational and power benefits from non-redundant ties.

Solution

Group members working on complex and non-routine tasks should regularly interact with each other and share their information. In this way potential structural holes (possibly caused by a group leader) can be avoided and may result in a fewer number of steps and chances of misinformation. As a consequence of this group performance will be improved.

Resulting context

The number of structural holes within a group in projects with complex and non-routine tasks is reduced and group performance is improved.

Design rationale

Choosing the right group structure in projects concerned with complex and non-routine tasks that are under control of a group leader, can improve group performance.

This pattern is related to the 'flat hierarchical structure' and the 'core-periphery structure' patterns.

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

Problem

What form of interdependence (the extent to which team members cooperate and work interactively to complete tasks. High interdependence occurs when team members interact cooperatively and depend on each other for information, materials, and reciprocal inputs, (Campion et al., 1993; Emery & Trist, 1969)) can be used for teams that have a conceptual focus (such as planning, deciding and negotiating) in order to reach a high team performance?

Context

Teams engaged in projects with a conceptual focus that are dealing with a poor team performance as a result of a less optimal amount of interdependence.

Forces

A moderate level of interdependence is applied in projects with primarily conceptual tasks.

Solution

Extremes in interdependence are appropriate for teams that are primarily engaged in conceptual tasks (such as planning, deciding and negotiating). An either low or high level of interdependence is related to both open communication and less conflict among team members. These processes are in turn associated with higher team performance.

Resulting context

Team performance is optimal in projects with a conceptual focus when either low or high levels of interdependence are applied

Design rationale

Choosing the right level of interdependence influences the team performance in projects with a conceptual focus.

This pattern relates to the 'Team self-leadership and conceptual tasks' and 'Interdependence, team self-leadership and behavioural tasks' patterns

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

Problem

What type of team self-leadership (the extent to which teams have the freedom and authority to lead themselves independent of external supervision. Teams with high self-leadership decide how tasks should be carried out, as well as what should be done and why. They are given responsibility and authority for their behaviour, and team members rather than supervisors make decisions and organize work processes, (Stewart & Barrick, 2000) can be used for teams that have a conceptual focus (such as planning, deciding and negotiating) in order to reach a high team performance?

Context

Teams engaged in projects with a conceptual focus that are dealing with a poor team performance as a result of a less optimal amount of interdependence.

Forces

Authority (like an external supervisor) has a negative impact on the team's self-managing capacity and therefore on team performance

Solution

Teams primarily engaged with conceptual tasks, should have greater team self-leadership (the extent to which teams have the freedom and authority to lead themselves independent of external supervision, (Stewart & Barrick, 2000)) in order to reach a high team performance.

Resulting context

Team performance is optimal in projects with a conceptual focus when a greater team self-leadership is applied. Greater team self-leadership corresponds with decreased shirking because employees have greater commitment and feelings of personal ownership (Pearce & Ravlin, 1987), increased member flexibility by allowing members to learn from one another (Trist, 1981), and effective management of conflict among members (Eisenstat, 1990).

Design rationale

Choosing the right type of team self-leadership influences the team performance in projects with a conceptual focus.

This pattern relates to the 'Interdependence and conceptual tasks' and 'Interdependence, team self-leadership and behavioural tasks' patterns.

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

Problem

What level of interdependence (the extent to which team members cooperate and work interactively to complete tasks. High interdependence occurs when team members interact cooperatively and depend on each other for information, materials, and reciprocal inputs, (Campion et al., 1993; Emery & Trist, 1969) and type of team self-leadership (the extent to which teams have the freedom and authority to lead themselves independent of external supervision. Teams with high self-leadership decide how tasks should be carried out, as well as what should be done and why. They are given responsibility and authority for their behaviour, and team members rather than supervisors make decisions and organize work processes, (Stewart & Barrick, 2000) can be used for teams that have a behavioural focus (dealing with overt, physical behaviour, with the execution of manual and psychomotor tasks, (McGrath, 1984)) in order to reach a high team performance?

Context

Teams engaged in projects with a behavioural focus that are dealing with a poor team performance as a result of a less optimal amount of interdependence and wrong type of team self-leadership.

Forces

Extreme cases of interdependence (either high or low) are not good in projects with behavioural tasks

Team self-leadership is not appropriate in project with a behavioural focus.

Solution

Teams engaged in behavioural tasks must have moderate levels of interdependence and greater external leadership in order to have an effective team performance.

Resulting context

Team performance is optimal in projects with a behavioural focus when external supervision and a moderate level of interdependence is applied.

Design rationale

Choosing the right level of interdependence and type of team self-leadership influences the team performance in projects with a behavioural focus.

This pattern is related to the 'Interdependence and conceptual tasks' and 'Team self-leadership and conceptual tasks' patterns.

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

Problem

What level of strength of a design interface (in a design interface component i's functionality is affected by component j, (Sosa et al., 2004)) leads to alignment between these design interfaces and team interactions (technical information which flows from team j to team i, Sosa et al., 2004) in complex product development (projects with a large number of both physical components and design participants involved in the process, (Sosa et al., 2004))?

Context

Design interfaces and team interactions in complex product development projects are misaligned with each other.

Forces

The unexpected case of unmatched design interfaces (design interfaces that are not addressed by direct team interactions, (Sosa et al., 2004)) occurs.

The unexpected case of unmatched team interactions (communication between teams whose components do not explicitly share design interfaces (Sosa et al., 2004)) appears.

Solution

In complex product development there is a greater degree of alignment between design interfaces and team interactions when these design interfaces are strong (those that involve several types of design dependencies and have high impact on the functionality of the other component.) than when they are weak (design interfaces as those which involve few types of design dependencies and have low impact upon the functionality of the other component, (Sosa et al., 2004)). Although, this pattern is based on complex products development on basis of studies in the automobile and aerospace industry as described by Sosa et al., (2004) it can also be applied in the software engineering field.

Resulting context

Alignment appears in design interfaces and team interactions in complex product development projects.

Design rationale

This pattern is used to overcome the misalignment between team interactions and design interfaces.

This pattern is related to the 'Alignment in design interfaces and its interrelated components' pattern.

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

Problem

In what way can unmatched design interfaces (in a design interface component, when component i's functionality is affected by component j) between corresponding design teams of two interrelated components be avoided?

Context

Misalignment occurs between design interfaces of interrelated components in complex product development projects.

Forces

The unexpected case of unmatched design interfaces (design interfaces that are not addressed by direct team interactions, (Sosa et al., 2004)) of interrelated components occurs.

Solution

When corresponding design teams communicate without indirect team interaction (technical information flow that takes place between two teams through an intermediary design team) it is more likely that there is alignment in design interface (in a design interface component i's functionality is affected by component j, (Sosa et al., 2004)) of interrelated components.

Resulting context

Design interfaces of interrelated components in complex product development projects are aligned with one another.

Design rationale

This pattern is used to overcome the misalignment between design interfaces of interrelated components.

This pattern is related to the 'Alignment between design interfaces and team interactions' pattern.

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

Problem

How should one allocate tasks in a virtual network?

Context

The virtual organization has a clear structure (degree of hierarchy, centralization, and hierarchical levels), clear task characteristics, and a clear network performance.

Forces

If there is a high level of centralization in a group attempting to accomplish a non- routine task this adversely affects the performance. If there is a low level of centralization in a group attempting to fulfill a routine task this negatively influences the performance.

Solution

In a virtual organization, high (low), organizational task routines coupled with a high (low) degree of hierarchy, centralization, and hierarchical levels will lead to high network performance.

Resulting Context

The success of this pattern depends on finding a good fit between the type of tasks and the degree of centralization or hierarchy in authority and/or centralization or hierarchy in information

Design Rationale

If a task is very routine and people in the network do not work structured in a rigid manner the task may not be performed in time. Conversely if the task is not routine and requires creativity it is not a good idea to structure the network rigidly.

This pattern is related to patterns 1, 2 and 4.

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. </pre> </body> </html>.

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.

View publication stats