Patterns for Managing Remote Software Projects

MICHAEL WEISS, Carleton University

The trend towards working remotely has been accelerated significantly by the COVID-19 pandemic, in particular in software development. In this paper, we describe patterns for managing remote software projects. These patterns were mined from the literature by conducting a literature review and synthesizing the findings using a research synthesis framework developed in design science (van Burg & Romme, 2004) in combination with the holistic pattern mining approach by Iba & Isaku (2012), but using topic modeling instead of visual clustering to identify themes. The current paper describes two of the patterns: Information Flow and Shared Mental Model. The target audience for the patterns are managers of software-intensive organizations transitioning to a remote work mode, as well as students and researchers.

Categories and Subject Descriptors: D.2.9 [**Software Engineering**]: Management—*Remote Software Projects*

General Terms: Patterns

Additional Key Words and Phrases: Remote software projects, pattern mining, topic modeling

ACM Reference Format:

Weiss, M. 2020. Patterns for Managing Remote Software Projects. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 27 (October 2020), 8 pages.

1. INTRODUCTION

The COVID-19 pandemic has triggered vast changes in how we live and work. A major change has been the shift to remote work. For example, in mid-2020, Google announced that they will let their employees work from home until at least the summer of 2021.[1] Although there has been a general trend towards working remotely over the past few years, witness the rise of companies like Automattic (the makers of WordPress) and GitLab, who operate as remote companies by default, this trend has been accelerated significantly by the pandemic.

Remote work here describes the situation where at least some of the members of a team work in different locations or time zones. They could also be working for different companies or individual contributors to an open source project. The main challenges facing remote workers are communication (it is difficult to get a hold of other team members), awareness of the work of others, and missing social interaction. If people are working from home, they encounter additional issues that include creating a working environment at home, maintaining a boundary between work and life, and the distractions of family life (Ford et al., 2020).

In this paper, we describe patterns for managing remote software projects. These patterns were mined from the literature using a research synthesis framework developed in design science (van Burg & Romme, 2004; Pilbeam et al., 2019) in combination with the holistic pattern mining approach by Iba & Isaku (2012), but using topic modeling (Blei et al., 2003) instead of visual clustering for identifying underlying themes.

[1] https://edition.cnn.com/2020/07/27/tech/google-work-from-home-extension/index.html

Author's address: M. Weiss, Carleton University, 1125 Colonel By Dr, Ottawa, Ontario, K1Y 0Z3; email: michael_weiss@carleton.ca

The target audience for these patterns are managers of software-intensive organizations who are transitioning to a remote work mode, as well as students and researchers in project management. The following sections first describe the method used to mine the patterns in greater detail, followed by an example of using the method, then the patterns themselves, and finally a conclusion and outlook at future work.

2. METHOD

The patterns were mined using a three-step process:

(1) Literature review and research synthesis to produce a collection of design principles that describe practices for managing remote software projects. The research synthesis follows an approach from design science (van Burg & Romme, 2004; Pilbeam et al., 2019) that produces design principles. Each of these design principles describes how a mechanism produces a certain outcome in a particular context.
(2) Clustering of the principles using topic modeling to identify underlying themes.
(3) These themes were then developed into patterns by extracting further details from the articles reviewed. These themes capture underlying (also referred to as *latent*) aspects of the practices and do not necessarily correspond to already established categories (for example, project management activities).

The intuition behind this approach is that the themes help us understand how the principles are connected. While the principles themselves are a reflection of the problems addressed by the different categories of papers, the underlying themes cut across categories and, thus, problems. The outcome of this process will be a set of patterns that, we argue, capture, at some deeper level, the reason for why those principles work. The patterns thus discovered can be said to represent the *essence* of our knowledge, rather than particular practices.

2.1 Literature review and research synthesis

The literature review and initial research synthesis were conducted as part of a course on managing software projects taught by the author in Summer 2020. The literature reviewed was based on 36 course readings grouped into six themes (management challenges, dealing with external contributors, understanding customers, working in teams, planning and execution, and architecture and organization). These articles were selected based on their relevance to remote software project management, except for a small number of foundational articles. The articles for the course were obtained by searching Google Scholar for articles on managing remote projects, reviewing the abstracts and then retaining representative articles for each of the themes of the course. The majority of the articles were published in 2015 or later. Careful reading of the articles produced 180 design principles.

2.2 Clustering the principles using topic modeling

To cluster the principles, we created a topic model (Blei et al., 2003). We created models with different numbers of topics and obtained the best results for 10 topics. For each topic, we collected the top keywords associated with the topic. As topic modeling is a probabilistic technique and produces slightly different results with each run, we controlled for topic stability by combining the result of four runs into an aggregate topic model. To produce topics that identify underlying themes in project management, the list of stopwords included common words like *project*, *practice*, or *software* that appear in most principles but provide little information about a topic.

Following the approach in (Bailetti & Tanev, 2020), we recorded the top-10 keywords for each run and retained keywords that were repeated across at least 3 out of 4 runs. We similarly collected the design principles associated with each topic whose topic weights were at least 40% for 3 out of 4 runs. Those principles were considered most representative of the topic. The process for running the topic models and extracting the keywords and design principles was automated by a tool developed by the author.[2]

[2]https://github.com/michaelweiss/topic-model-explorer

Table I shows the list of topics, the top keywords and the number of design principles associated with each topic (this indicates the support the topic has in the corpus). For example, take topic 0. The words *team*, *reduce*, *effort* and *competency* were consistently associated with this topic. Based on these keywords and a close inspection of the strongly associated design principles, we can assign the label *information flow* to topic 0.

2.3 Developing themes to patterns

The top three topics, in terms of the number of supporting principles, are: *requirements discovery*, *synchronization*, and *shared mental models*. Two topics only have weak support: *structure* and *knowledge sharing*, and we may want to combine them with other topics. We can, furthermore, group the topics into overarching themes based on their similarity. Figure 1 shows the result of manually clustering the topics.[3] There are three groups of themes: *sharing*, *coordination [early]*, and *dealing with change*.[4] Managing remote projects successfully thus hinges on sharing knowledge and resources, early coordination of activities, and being able to deal with change.

We then develop themes into patterns. Patterns can be formed from islands of meaning (Iba & Isaku, 2012). Here, we choose the topics identified in Table I, or the leaves of the tree in Figure 1, respectively, as the appropriate level of abstraction for individual patterns. To generate the pattern descriptions, we can start with the associated principles. As we elaborate the patterns, we will also consult the articles from which those principles were extracted for more information about the context, problem, forces, solution, and known uses of each pattern.

3. EXAMPLE OF CREATING A PATTERN

We next show how we can use the method to create a pattern from the design principles associated with a topic and drawing on the corresponding papers for information on context, problem, forces, solution, and known uses of the pattern. Consider topic 0, which we labeled *information flow*. The design principles associated with this topic that made the cut-off threshold are listed below:

(1) Optimize IT infrastructure to enable efficient collaboration between internal and offshore teams (E25; Ebert et al., 2016)
(2) Good information flow in teams is relevant, timely and clear (T10; Westrum, 2014)
(3) Treat all team members (even if there are co-located) as remote (T3; Lous et al., 2018)
(4) Form small teams focusing on product features and core competencies to achieve coordination between multiple teams and communication efficiency (A15; Sutherland et al., 2019)

Here, the labels in parentheses are identifiers assigned to the design principles. The first letter of each label indicates the category (E = *dealing with external customers*, T = *working in teams*, and A = *architecture*), and the number that of the design principle in this category. We also record the source of the principle, that is, the paper from which it was extracted. The principles are sorted by their strength of association with the topic.

This topic does not correspond to just one of the categories used to select the articles, but cuts across three of them: *dealing with external contributors*, *working in teams*, and *architecture*. For all of these project management activities, good information flow between stakeholders is essential. The first and third principles are, furthermore, specific to remote software projects. They explicitly reference terms like *internal* and *offshore*, or *co-located* and *remote*. The remaining principles apply to software projects, in general. The second principle refers to *information flow*, and the fourth to *coordination* and *communication*.

We derive the context for the pattern from the common overarching theme of the three topics *information flow*, *shared mental model*, and *knowledge sharing*. These topics all have in common that they deal with *communication*.

[3]Note that we could cluster the topics automatically based on the keywords associated with each topic and their weights.
[4]The square brackets behind the name of a theme further qualify the theme: *coordination [activity]* refers to coordination of activity, whereas *coordination [early]* refers to coordination at the early stages of a software project.

Table I. Results from topic modeling

Topic	Label	Keywords	Design principles
0	Information flow	team, reduce, effort, competency	4
1	Requirements discovery	requirement, user, process, time, factor, cost, establish, concern	16
2	Resource sharing	resource, early, problem, portfolio	4
3	Shared mental model	communication, model, developer, synergy, mindset, goal	11
4	Synchronization	provide, frequent, work, build, distributed, stakeholder, remote, party	12
5	Focus	development, create, focus	9
6	Knowledge sharing	sharing	2
7	Structure	system, dsm	3
8	Managing uncertainty	multiple, idea, people	9
9	Agility	change, agile, trust	8

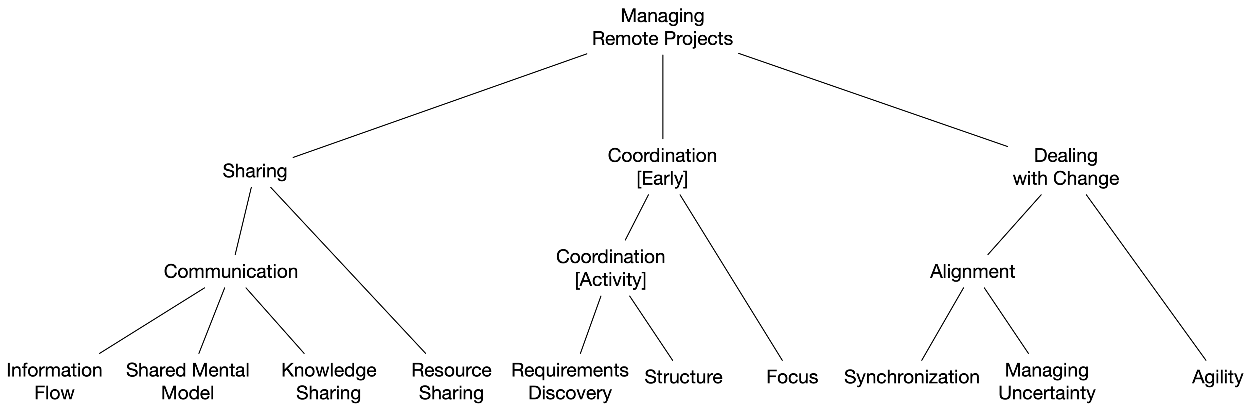


Fig. 1. Topics grouped into overarching themes.

Thus, for now, the context of the pattern becomes: effective communication is essential to team performance. This context does not, as yet, include information about other patterns that may be applied before this one.

We consider *information flow* to be the central concept that links the principles. The principles deal with how information flow is supported by technology; the qualities of information flow; ensuring inclusiveness, so that all team members, who need to, receive the information; and the efficiency with which the information flows. Therefore, we base the problem statement on Westrum's (2014) study of information flow in organizations. He notes that "[o]rganizations function on information. Stop the information and the organization stops, too."

The second principle provides the seed of the solution: enabling information flow that is relevant, timely and clear. To implement this solution, we need to put collaboration technology in place (although technology, on its own, can never be the solution to a social problem; this leads us to a force), treat co-located and remote team members the same (again, technology can facilitate this), and structure the team around product features and core competencies. Three of the principles also refer to the consequences of the recommended solution:

(1) Enable efficient collaboration between internal and offshore teams
(2) Good information flow
(3) NA
(4) Achieve coordination between multiple teams and communication efficiency

Thus far the information we can extract directly from the principles; the articles provide more details on the solution and the other elements of the pattern (context, problem, forces, solution, and known uses).

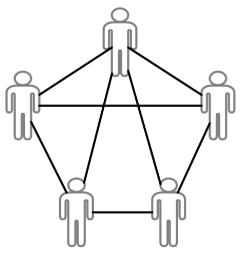
To extract forces, we derived focal questions from the principles, then read the articles associated with each principle to answer those questions:

(1) What makes collaboration with remote team members difficult?
(2) What prevents good information flow?

We build the solution description around the recommended actions in the principles. First, we draw on Westrum's (2014) study of how organizational structure impacts information flow. Then, we incorporate the observations from Lous et al. (2018) on ensuring the cohesion of distributed teams, and the lessons from Conway's Law (Sutherland et al., 2019). Finally, we include Ebert et al.'s (2016) discussion on the role of collaboration infrastructures. Even though the corresponding principle appeared first in the list of principles associated with the topic, we feel is it appropriate to deemphasize the role of technology in ensuring information flow.

4. PATTERNS

4.1 Information Flow



. . . effective communication is essential to team performance.

∗ ∗ ∗

Teams operate on information. If you stop information from flowing, the team stops working, too.
Organizational structure and politics can get in the way of effective information sharing; some organizations hoard information or do not encourage the sharing of "bad" news (Westrum, 2014). Additionally, since co-located team members often communicate face-to-face, remote team members may often feel "left out" of the conversation (Lous et al., 2018). Communication among team members should follow product features and core competencies; yet, cross-cutting links across feature and competency groups are also required, as work can never be perfectly partitioned (Sutherland et al., 2019). Finally, while information flow is not foremost a tool issue, poor collaboration infrastructure makes sharing information with remote team members less efficient (Ebert et al., 2016).
Therefore,
Enable information flow that is relevant, timely and clear.
Westrum (2014) has studied the impact of organizational structure on information flow. Pathological and bureaucratic organizations, on one hand, withhold information or share it only after putting their "spin" on it. They are also "good news" organizations and threaten their members from sharing information that makes them look "bad". In these organizations, information flow is prevented from flowing freely.
Generative organizations, on the other hand, are driven by their mission, that is, by what they are supposed to do (Westrum, 2014). In such an organization, information flows to the people who need to receive it, is shared immediately, and is communicated clearly to enable the receiver to act on it. Generative organizations can effectively respond to reported problems when they arise and fix them or their underlying causes.
To ensure that all team members, whether local or remote, receive the information they need, treat everybody, even those who are are co-located, as remote (Lous et al., 2018). In the startup studied by Lous et al. (2018),

even meetings between co-located team members are conducted using web conferencing or messaging tools. The results of task allocation are also shared via electronic tools like whiteboards for everyone to see.

While you want everyone to have access to the information they need, uncontrolled information flow can be too much of a good thing. It can inundate team members with irrelevant information. Therefore, you want information to flow mostly in subteams formed around product features and core competencies (Sutherland et al., 2019; *Conway's Law*). However, since work can never be partitioned perfectly, you also need cross-cutting links that bridge subteams to ensure coordination between them, while keeping communication efficient.

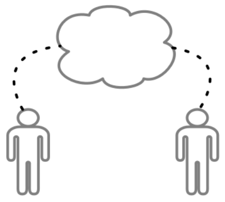
While good information flow is not primarily a tool issue, it is essential to have a good collaboration infrastructure in place for sharing information with remote teams (Ebert et al., 2016). This includes tools like web conferencing, instant messengers, code repositories, and collaborative wikis. Through an effective collaboration infrastructure, all team members will have access to the same information (Lous et al., 2018).

∗ ∗ ∗

At Debitoor, the startup studied by Lous et al. (2018), the default way for team members is to communicate virtually. For example, in the daily stand-up meetings, developers sit at their desks wearing headsets.

Effective information flow also depends on having a mental model shared among team members. A *Shared Mental Model* allows the sender to match the information to the needs of the receiver.

4.2 Shared Mental Model



. . . effective communication is essential to team performance. You are using *Information Flow* and have identified team members who should receive the information. How do you communicate the information to them?

∗ ∗ ∗

When a task is complex or has not been encountered before, team members need to collaborate. However, they often lack a common understanding of the task.

When project tasks are interdependent, team members need to work together (Kude et al., 2019). This requires an understanding of the project's goals and awareness of the skills and knowledge of other team members. However, in a remote context, awareness of other team members is often limited (Mantelli et al., 2012; Avritzer et al., 2010). There are also less opportunities for team members to gain shared experiences. How project decisions, such as project goals or design decisions, are made is, in many cases, also not transparent (Knauss et al., 2018; Weiss, 2009). There can also be cultural differences between different parts of a project team and language issues that prevent effective communication (Carmel & Agarwal, 2001). This can be particularly challenging in large projects that involve multiple locations, time zones, and organizations (Paasivaara & Lassenius, 2003).

Therefore,

Foster a shared mental model among team members.

A shared mental model is the "shared, organized understanding and mental representation of knowledge" among team members (Kude et al., 2019). It enables team members to collaborate closely on interdependent tasks. It includes a shared understanding of project goals, methods and skills (so team members could substitute for one another), and awareness of other team members (in particular, those with unique skills). In a remote team, special effort needs to be made so that team members can create a common understanding. This can be achieved by creating a system metaphor, through greater transparency, sharing of people across teams, adapting practices common in a co-located settings to the distributed context, and the use of social network analysis tools.

Establishing a common architectural vision (referred to as the “system metaphor” in the agile practice of extreme programming) at the beginning of a project creates a common language among team members (Yu & Petter, 2014). The system metaphor helps developers develop a shared understanding of how the system will work, as well as which team members are responsible for each part of the system. It is the role of the lead architect or other high-level managers to communicate the system metaphor to the team members.

Transparency can be enabled by stand-up meetings across different sites (Yu & Petter, 2014). Open source projects create transparency by conducting all communication in the open (Knauss et al., 2018). Since the communication is also archived, new team members can access the project history and understand how past decisions have been made (Weiss, 2009). Team norms and work practices are also documented in written form in a code of conduct and in procedures (Weiss, 2017).

Cultural differences between team members – which can be organizational or national in nature – can be overcome by introducing a liaison role to bridge between the cultures (Carmel & Agarwal, 2001). A liaison is somebody who is at home in both cultures and can help team members from different cultures understand each other. Such a role is also known as boundary spanner, broker, or gatekeeper (Mantelli et al., 2012). More generally, sharing people across sites can help establish common ways of thinking within a team.

When large teams are involved, inter-organizational communication can be further improved by establishing peer-to-peer links between collaborators at all organizational levels (subcontracting managers, project managers, and developers) (Paasivaara & Lassenius, 2003). These roles establish points of contact between the organizations and provide a standard way of sharing information between them. Misunderstandings can also be due to language issues. Those could be reduced through language training (Carmel & Agarwal, 2001).

Skills and knowledge can be distributed across team members through the practice of pair programming. In pair programming, a pair of developers jointly work (“pair up”) on a complex or novel task. Pair programming can thus help create a shared mental model between team members (Kude et. al., 2019). This also means that these team members can now substitute (“backup”) for one another if necessary.

Better communication, access to expertise, and increased team awareness can be achieved by using tools that visualize communication patterns among team members (Manteli et al., 2012). Such tools are based on social network analysis, which can help extract the coordination and communication structures of teams from project artifacts, for example, by analyzing code contributions. The communication needs in a remote software project can also be anticipated at an early stage through architectural modeling (Avritzer et al., 2010).

* * *

Kude et al. (2019) surveyed developers at a large enterprise software firm that had recently adopted Scrum. Their results show that pair programming help create shared mental models between team members. When faced with high degrees of task novelty, these shared mental models allow developers to backup for one another.

Two of the companies studied by Paasivaara & Lassenius (2003), both with multiple teams across Europe and North America, used links at three levels of the organizational structure (at the management, project, and team level), while the remaining teams maintained links at least at the project manager level.

Two related patterns are *Vision* (Sutherland et al., 2019) and *Product Pride* (Sutherland et al., 2019).

5. CONCLUSION

In this paper, we experimented with a new way of mining patterns using topic modeling and applied it to mining patterns for remote work. The topic model helped us discover patterns (for example, the need for shared mental models) that underlie visible practices (such as pair programming). These patterns were at a more fundamental level than the practices themselves and are “anchored” around qualities of well-functioning remote teams.

Acknowledgment

I would like to thank my shepherd Tiago Boldt Sousa for his insightful comments. My thanks also go to the members of my writers' workshop at PLoP 2020: Neil Harrison, Lise Hvatum, Hélène Finidori, and David Haselberger.

REFERENCES

Avritzer, A., Paulish, D., Cai, Y., & Sethi, K. (2010). Coordination implications of software architecture in a global software development project. *Journal of Systems and Software*, 83(10): 1881–1895.

Bailetti, T., & Tanev, S. 2020. Examining the Relationship Between Value Propositions and Scaling Value for New Companies. *Technology Innovation Management Review*, 10(2): 5-13. http://doi.org/10.22215/timreview/1324.

Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3: 993–1022.

Carmel, E., & Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development, *IEEE Software*, 18(2): 22–29.

Ebert, C., Kuhrmann, M., & Prikladnicki, R. (2016). Global software engineering: Evolution and trends. *International Conference on Global Software Engineering (ICGSE)*, 144–153, IEEE.

Ford, D., Storey, M. A., Zimmermann, T., Bird, C., Jaffe, S., Maddila, C., Butler, J., Houck, B. Nagappan, N. (2020). A tale of two cities: Software developers working from home during the COVID-19 pandemic. *arXiv preprint*, arXiv:2008.11147.

Iba, T., & Isaku, T. (2012). Holistic pattern-mining patterns: A pattern language for pattern mining on a holistic approach. *Conference on Pattern Languages of Programs (PLoP)*.

Kude, T., Mithas, S., Schmidt, C., & Heinzl, A. (2019). How pair programming influences team performance: The role of backup behavior, shared mental models, and task novelty. *Information Systems Research*, 30(4): 1145–1163.

Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., Kuhrmann, M., & Ebdrup, A. (2018). Virtual by design: How a work environment can support agile distributed software development. *International Conference on Global Software Engineering (ICGSE)*, 97–106, IEEE.

Manteli, C., Van Vliet, H., & Van Den Hooff, B. (2012). Adopting a social network perspective in global software development. *International Conference on Global Software Engineering*, 124–133, IEEE.

Paasivaara, M., & Lassenius, C. (2003). Collaboration practices in global inter-organizational software development projects. *Software Process: Improvement and Practice*, 8(4): 183–199.

Sutherland, J., Coplien, J. & The Scrum Patterns Group (2019). *A Scrum Book: The Spirit of the Game*, Pragmatic Bookshelf.

Van Burg, E., & Romme, A. (2014). Creating the future together: Toward a framework for research synthesis in entrepreneurship. *Entrepreneurship Theory and Practice*, 38(2): 369–397.

Weiss, M. (2009). Performance of open source projects. *European Conference on Pattern Languages of Programs (EuroPLoP)*, CEUR, 566. http://ceur-ws.org/Vol-566/A5_PerfOpenSource.pdf.

Weiss, M. (2017). Patterns for regulating behavior in innovation communities. *Conference on Pattern Languages of Programs (PLoP)*, 15: 1–12.

Westrum, R. (2014). The study of information Llow: A personal journey. *Safety Science*, 67: 58–63.

Yu, X., & Petter, S. (2014). Understanding agile software development practices using shared mental models theory. *Information and Software Technology*, 56(8), 911–921.