

The Problem of Knowledge Decoupling in Software Development Projects

Yutaka Yamauchi, Jack Whalen, Nozomi Ikeya and Erik Vinkhuyzen

Palo Alto Research Center

3333 Coyote Hill Rd

Palo Alto, CA 94304

{yutaka.yamauchi, jack.whelen, nozomi.ikeya, erik.vinkhuyzen}@parc.com

ABSTRACT

In our ethnographic investigation of software integration projects a recurrent pattern emerges. The detailed understanding leaders have of the design and development decreases over time as they become busier and busier attending meetings, creating documents, and resolving issues and thus cannot spend much time on design or development work. As a result, their leadership becomes increasingly decoupled from the work of the project. We discuss various dimensions of this problem.

Categories and Subject Descriptors

D.2.9 [Software Engineering]: Management – *programming teams*; K.6.1 [Management of Computing and Information Systems]: Project and People Management - *management techniques*

General Terms

Management.

Keywords

Project management, knowledge decoupling, leadership.

1. INTRODUCTION

As much as the technical aspects of software development are stressed in the literature, the social and organizational aspects are often considered as key. Projects often fail not because of technical difficulties but rather human ones: lack of leadership, broken communication, unclear work distribution, and the like. We focus on this side of software development. Particularly, we stake a case study approach to problems in project organization and management or “leadership.”

We have been doing ethnographic fieldwork in two software projects. These are both system integration efforts by a firm’s software business organization. One project aims to develop a system for financial transactions among various investors, banks and other financial companies. The estimated size of the project was about 3,500 person-months. We started our observation in the

basic design phase. The second project is chartered to develop an infrastructure system for mediating transactions between back-end systems that manage master databases and front-end systems with which users interact. The estimated size of this second project was about 600 person-months. We conducted fieldwork two to three days a week at the project rooms for about four months.

In our fieldwork we repeatedly came across one particularly intriguing issue. Leaders’ knowledge of the details of the design work that the team is engaged in decreases over time. We call this problem *knowledge decoupling*. In this paper, we describe the features of this problem as we have experienced it in these two cases and discuss its larger significance for software project organization as well the direction future research should take.

We limit our discussion on team leaders and group leaders. In a typical project organization we studied, there is a project manager and a few members to support the project manager. Beneath them are group leaders. A group consists of two to five teams. There is a team leader for each team that typically consists of three to ten members. Although we believe that project managers face a similar, if not the same, problem, we do not have enough data on project managers.

2. Knowledge Decoupling Challenge

2.1 The problem *in situ*

The challenge of large-scale software development is how to manage and coordinate work that can grow in complexity and volume through the project life cycle. As we began our field observations in the two software projects, we expected to find team leaders becoming more knowledgeable and ‘literate’ about their project’s technical requirements, the responsibilities and day to day work of team members, and the customer relationship. But as we spent more time in the field, we noticed that something quite different occurred. Instead of team leaders’ increasing their understanding of the system design as it became more concrete, they started to lose contact with the many of its important details. Moreover, while they participated early on in the meetings externally with end-users and internally with project developers and seemed to have a good sense of how the design was being specified, as that design continued to develop the leaders could not attend some important meetings, did not read critical specification documents, and therefore slowly lost track of what each developer was up to.

To make the matter worse, as the design was gradually specified in detail, a number of unexpected problems emerged. Coordination needs intensified when pieces of the specified design had to work together. For instance, when a group of

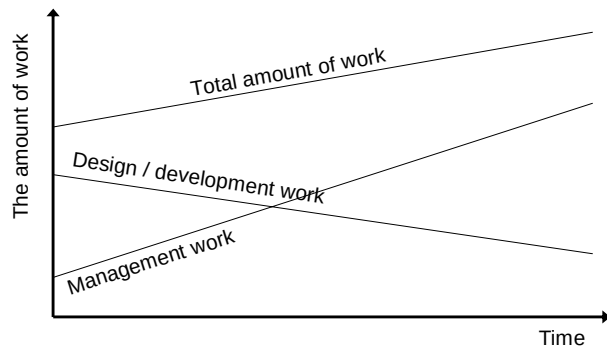


Figure 1: Apparent tendency of project life

developers met with a customer and reached agreement that specific variables should be used to identify one data record, this affected other teams who needed to make changes to their design in accordance with the agreement. Similarly, when a front-end system needed to have all the data in a particular format so that it could be easily shown on the screen, engineers of the back-end system needed to implement a new data handling component. Simply put, details required more coordination. But in both our cases the project leaders were not able to successfully manage these coordination needs. Their time was increasingly taken up by other management tasks, and the total amount of work for them increased dramatically (See Figure 1).

Thus, the leaders tend to become busier and busier as the project proceeded and because they could work only so many hours in a day, they had to spend less time on meetings related to the substantive design and development work because members could continue that work by themselves. Project members can discuss design issues without the leaders. Because the knowledge of the system is contextual in that it can be gained most effectively through the actual practices of designing and developing, the leader who is distant from the practices become less knowledgeable [1]. And so over time leaders' knowledge of the design and development decreased and their knowledge was then decoupled with their leadership. Although the tension between technical work and the administrative work was recognized even in the early days of software engineering [e.g., 3, 4], the problem has not been much studied.

2.2 Causes of Knowledge Decoupling

2.2.1 Meeting-Centric Coordination

Software engineers have to spend so much time on meetings. This is primarily because of the coordination intensive nature of the work. The developers we observed had two to three internal regular meetings (e.g., regular team meetings, regular project progress meetings and regular design meetings) and two to four meetings with the customer and other vendors (e.g., regular progress meetings, regular meetings on a sub-system, and occasional steering committee meetings).

On top of these, they have frequent meetings with the customer to specify design and many meetings with other teams. Whenever a problem occurs, leaders need to spend time on reporting both to the customer and to the management how the problem happened and how it is being dealt with. Typically, customers tend to ask for more meetings when the projects are in trouble. Managers of the IT organization also tend to demand more meetings to closely

monitor high-risk projects. In these problematic projects, the leaders and members all need to work harder simply because they need to put the projects back on track. The leaders and members become even busier to deal with extra meeting obligations.

2.2.2 Document-Centric Coordination

The leaders we observed had to spend time to prepare documents for each meeting. As the design gets specified more in detail, the length of the report increases. Indeed, it was extremely difficult for us to get any time from the leaders. They were almost always in rush preparing documents for meetings. After they come back from the meetings, their job is not over. They need to update the documents as new things are suggested in the meetings. For example, as new requests, new issues, and new meetings are discussed in the meeting, the leaders need to update the issue management sheet and the team's schedule and inform the team members of the change.

On top of the regular meetings, when the customer asks a question on the design, the leaders need to create a document for explanation. In one of the projects, the customer wanted to know how much storage the system would need. A leader was then asked to create a document that summarized the storage requirements for various scenarios. He asked several members to prepare drafts and put together the document. This kind of document creation is a common occurrence. For each issue to discuss, somebody has to create a document to explain. Of course, typically team members create documents and leaders review them. Yet, members are busy and lack the context of the issue. Leaders need to spend much time on reviewing and modifying the documents.

Mostly, they do not meet with the customer without documents. This is probably because of the complicated nature of the topic. They need to create tables and figures to summarize the content. For the storage capacity, they needed to create various tables that listed estimated capacity required. We have never seen meetings without documents. Each document takes considerable time to create. As the design develops, more issues come up; issue management sheets grow rapidly. In fact, they make those sheets simply because there are too many issues to keep track of without them. Therefore, the amount of documents to create increases as issues are added.

2.2.3 Coordination with Other Teams

The more detailed the design becomes, the more coordination developers need to have among themselves. Even small issues require coordination and therefore a series of meetings and documents associated to them. For example, a web server was being built on a UNIX server. Because of the combination of the operating system, Java SDK and the web server, they had to use a certain character coding. However, the character coding was not consistent with other sub-systems—access logs were supposed to be written in another coding. A series of discussions and some investigation of what solution they can choose were then initiated.

Coordination among teams is often problematic because teams often become defensive of their work. The teams tend to avoid additional tasks as every team is busy just processing what they are originally told to do. This problem is confounded by the fact that teams are often formed around subcontractors—one subcontractor takes care of some specific functionality. As subcontractors work is specified in fixed price contracts, they are

reluctant to take up more tasks. Yet, somebody has to deal with the issues. Often, issues remain unresolved. When such an issue manifests itself, it will certainly create more work.

2.3 Inability to Lead Design and Development

Software development needs strong leadership. Design integrity is one of the key elements to make software development successful. Without the leadership, individual teams create parts of the system in the way they think is the best, which is not necessarily the best for the entire system. For example, one team was working on a communication sub-system. The other sub-system that it was supposed to communicate with was not working as it should be. The team modified their sub-system to absorb the deficiency in the other sub-system instead of coordinating with the other team. It was easier for them to simply change their sub-system than to ask the other team to correct their sub-system. However, if the communication is not implemented according to the specifications other parts may be affected. Also, if the sub-system ever must be modified it may cause confusion.

Leaders may not be aware of this kind of detail. In this particular case it was identified by an external reviewer who was called in to assess the quality of the system. If leaders are not aware of the details, the problem can be left unnoticed.

2.4 Indirect Coordination

When a leader loses contact with the design details, he or she cannot play as a representative of the team to discuss design. When leaders are a coordination hub, the project can work efficiently. Whenever they have an issue leaders can get together and solve it. Design consistency can also be managed efficiently because a small number of leaders can work out the design. Yet, if the leaders lack knowledge to do so, some team members now need to be involved in solving the issue.

One leader was particularly distant from the substantive design work his team was carrying out. He participated in a review meeting, in which a new issue regarding transaction messages came up. Other team leaders could explain their team's design and the assumptions on which they were basing their design, however, he could not and had to call one of his team members. We often saw this leader call his members right in the middle of meetings to clarify details. Another team leader of this project said to us that because some leaders did not understand the content of the teams' work, he had to talk to members directly. He suggested that the level of leadership was pushed one level down to members. That is to say, some members start to play the leader's role.

2.5 Inability to Manage the Team

Although we have discussed the design work and the management work separately, in practice the two kinds of work are hard to distinguish. The management work is dependent on the leader's knowledge of the design work. When they know the substantive design work that is going on in the team, they do not even have to ask the members how much progress they have made. Yet, when they lack the understanding of the context of the work, they always need to ask members for information.

Rather than managing based on a full understanding, such managers often resort to managing the project plan. They assess member's progress by counting the number of screens that

members have completed designing, the internal reviews user reviews completed and so on.

One team leader realized that he became decoupled when his team members had a disagreement in a review meeting that was attended by the project manager. The team leader thought of this incident as manifestation of his inability to manage his team and next day declared to his members that he would work harder to take stronger leadership. He became careful in communicating with members of the necessary information, reinstated meetings that he was supposed to run but could not for lack of time, and spent more time with his members.

2.6 Inability to Negotiate with Customer

Leaders are also supposed to interact with the customer. When the leader lacks the knowledge of details of the design and development work, they cannot make good decisions when meeting with the customer.

In one of the projects, the customer indicated that the master database definitions were not consistent with the requirement definition document. The leaders present agreed that there was the problem and that they would prepare a document to explain the situation next day. The leaders called for a meeting with team members as soon as they came back. In the discussion, members pointed out various misunderstandings on the part of the customer. One member said that the master database was clearly classified and everybody else in the meeting agreed. The leaders did not know this. Members also suggested that the customer might not understand that the master entries were base elements and a new set of data had to be created to make transactions possible. Leaders could not explain to the members what the customer's understanding was. It became clear that the developers had various reasonable explanations for why the master database was the way it was presented to the customer. Yet, because the leaders could not explain it to the customer, they came back with the agreement to create a document to explain it. If they could have answered to the customer on the spot, the members did not have to spend time on discussing the issue and creating the document. More important, the customer would have been more confident on the design team.

2.7 Leaders Who Avoid Decoupling

We have seen a few leaders who do not let their leadership get decoupled from the content of the project work. At first sight, we thought that they were excellent leaders. But, as we investigated more, we started to see a problem hidden underneath the apparent success. These leaders who kept close touch with the team's substantive design and development work could do so because they stuck closely to defined scope of their work. But as they did so, they often neglected to coordinate closely with other team leaders. Without coordination nobody takes up new issues and inconsistent and incomplete systems can result.

One leader managed his team's technical work very well. His team was located at a remote site and was developing a tool for other teams. He came to the main project room once or twice a week. A tension arose between him and the other team leaders because he lacked the attitude to work with the other teams and rather tried to define specifications in a formal manner and developed the tool only minimally. He could keep the level of management and coordination overhead low largely because he was not collocated and limited his time in the project room. Hence,

he could spend more time on working with his team. He complained that the other team leaders did not have thorough knowledge of what their teams were doing and spent too much time in meetings.

So for leaders there is no easy solution; coordination activities are very important and require their time and attention, yet they can get in the way of understanding the details of the technical work.

2.8 Team Members

Team members are not outsiders to the decoupling problem. Members cannot continue working in the same way when leaders cannot spend much time with them. Typically, one of the members emerges as a practical leader of the team. Other teams talk to this unofficial, emergent leader on technical issues. In one team we observed, members started to take leadership. They constantly discussed technical issues among themselves and worked out the design. The official leader could rely on the members. In a team in another project, one member became a hub of the design and even started to be involved in discussions of other teams. Members should not assume that because the leaders do their job, they can just focus on their own tasks.

3. Implications

Knowledge decoupling problem is the touchstone of successful software project organization. Uneven distribution of knowledge in the project is a major issue in software engineering [5]. If the project can avoid or minimize the decoupling problem, the coordination can be efficient, and design integrity can be achieved, customers can have more confidence, and so on. On the other hand, if leaders lose contact with the substantive design and development work, issues can be left unnoticed, members may hide important information, members are given more overhead work to deal with uninformed leaders, and the like.

Any software engineering methods need to come to terms with this problem. Agile software engineering methodology [e.g., 2] appears to address this problem to some extent. The amount of meetings for the purpose of reportage and the amount of documents to create may be reduced if the customer relationship is based on mutual trust and the work can be organized in the Agile way. Yet, large-scale projects within limited time and cost require some kind of formal organization. Large system requires more than a small team. Projects face enormous time pressure and are forced to use a large number of developers in a short time period. If the team becomes large, the knowledge decoupling is always an issue.

Brooks [3] advocated the director and producer model as a form of leadership in software development. The director takes care of the substantive design and development work while the producer supports him or her by taking care of the management and other work. In this approach, however, decoupling between the director and the producer is still an issue. The question is how the producer can do the management work if he or she does not know the substantive work. This is an empirical question.

Another important implication is that we should not take the project organization as fixed. Although this issue is recognized [e.g., 6], no specific guideline is provided to monitor and change the organization. Knowledge decoupling is a key area to monitor. One interesting way to address the decoupling problem is "peeling." If a leader becomes distant from the design work of the team, the team should be peeled once by promoting the member

who knows the design work but is ready to take the leadership role. This is actually what people do in projects out of necessity and somehow in an unofficial manner. Project managers, leaders and members need to understand this problem of knowledge decoupling and constantly reflect on their knowledge and leadership to adjust and redefine the project organization.

4. Future Directions

Our goal is to deepen our understanding of the knowledge decoupling problem by collecting more data. We will collect more ethnographic data to understand how this problem manifests itself in the everyday work of leaders and developers. Our data are still limited in part because we did not start collecting the data with this particular problem in mind. We will collect our data more systematically and want to study project leader's time allocation. Longitudinal data of this kind can reveal the process. Combined with the ethnographic data, we can understand the problem in a more systematic way.

In this study we focused mostly on team leaders who lead teams rather than project managers who lead the entire project and members who work in the teams. Project managers may well face the same problem. We saw some project managers involved in details of design in the basic design phase while other managers were occupied with the management work and communication with customers. Members' work should not be constant throughout the project, either. They may exhibit similar or different dynamics of their knowledge. Particularly, we have seen in various occasions that decoupling on the part of leaders is accompanied with the accumulating knowledge in some of the members. Some members become a focal point of coordination as they are heavily involved in the design and development work. They then appear to become more knowledgeable [See also 5]. We seek to collect data on these two groups of people.

Finally, we need to move from understanding the problem to creating solutions. We do not believe that a technical solution alone can solve this problem. We need to reconsider the ways in which the software development work is organized in projects. Any solutions should be grounded in the reality of work within projects, which we seek to reveal in our fieldwork.

5. REFERENCES

- [1] Barley, S.R. Technicians in the Workplace: Ethnographic Evidence for Bringing Work into Organizational Studies. *Administrative Science Quarterly*, 41 3 (1996). 404-441.
- [2] Beck, K. *Extreme programming explained: embrace change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [3] Brooks, F.P. *The mythical man-month : essays on software engineering*. Addison-Wesley Pub. Co., Reading, Mass., 1995.
- [4] Bruggere, T.H., Software engineering: Management, personnel and methodology. in *International Conference on Software Engineering*, (1979), 361-368.
- [5] Curtis, B., Krasner, H. and Iscoe, N. A field study of the software design process for large systems. *Commun. ACM*, 31 11 (1988). 1268-1287.
- [6] SEI. Capability Maturity Model Integration (CMMI), Version 1.1, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, 2002.