MORE HANDOVER SOLUTION PATTERNS

KEI ITO, Waseda University
HIRONORI WASHIZAKI, Global Software Engineering Laboratory, Waseda University
JOSEPH W.YODER, The Refactory, Inc.,
YOSHIAKI FUKAZAWA, Global Software Engineering Laboratory, Waseda University

The lifecycle of large software systems inevitably includes personnel changes. Most business people are familiar with the concept of a handover, but issues with handovers became apparent in Japan when many people from the Baby Boomer Generation retired simultaneously in 2007. Although effective handovers are crucial for seamless business operations during personnel changes, the preferable elements for a handover are ambiguous and not well researched, motivating our research. By analyzing anti-patterns, we note handover solution patterns to mitigate problems. However, our solution patterns may not be complete. Since our pattern language is familiar to many business people, we held workshops to find potential pattern seeds. The workshop participants suggested 22 new handover solution patterns. Three of which were suggested by multiple groups: *Ability check*, *Pair working*, and *Matching the knowledge*. *Ability check* evaluates the successor's ability to prevent a handover failure due to lack of ability of the successor. *Pair working* is used to avoid handover altogether by assigning more than two people to the same job. *Matching the knowledge* avoids misunderstanding knowledge by successor by filling up the gaps between predecessor's explanation and successor's understanding

Categories and Subject Descriptors: **• Social and professional topics~Project and people management • Software and its engineering~Maintaining software • Software and its engineering~Software verification and validation**

General Terms: Pattern Language

Additional Key Words and Phrases: Handover, Pattern Language, People Management, Maintaining Software

ACM Reference Format:

Ito, K., Washizaki, H, Yoder, J.W., and Fukazawa, Y. 2017. More Handover solution patterns. HILLSIDE Proc. of Conf. on Asian Pattern Lang. of Prog. 12 (March 2017), 9 pages.

1. INTRODUCTION

Handovers are very important for a business to maintain superior knowledge inside the organization. A handover is the process of transforming responsibilities and knowledge from the predecessor to the successor [1] [2]. Because handovers occur frequently within most organizations, they can be a persistent problem in business, affect critical stages of the software lifecycle [3]. In fact, the lifecycle of large software systems inevitably includes personnel changes. Despite its importance, handover problems are not well studied [1], planned or understood. One study mentioned that insufficient knowledge of the successor is one of the main problems that happens during a handover [1], while another study found that the information sharing process contains complex problems [4].

We undertook this research to explore concrete solutions for handover issues. In order to define and present our handover solutions, we decided to use the concept of a pattern language. A pattern language is used to provide desirable solutions based upon known best practices. They help guide and present a solution given the problem with a given context. Although the concept of pattern languages has primarily been applied to architecture and software fields, it is applicable to other fields such as cooking [5] and child development [6] as well as introducing new ideas into an organization [7]. Herein it is used to present desirable handover solutions.

Previously, we identified handover problems using anti-patterns. We presented three handover anti-patterns at Asian PLoP 2016 (*Unsupported to review*, *Background is unclear*, and *Necessary knowledge is omitted*) [8], which were confirmed by questionnaires sent to businesses. Analyzing the anti-patterns revealed their origins. From this analysis, we noted many handover solution patterns and outlined them as "A pattern language for handover" at PLoP 2016 [9]. In the paper, we introduce three concrete solution patterns (*Spread of the knowledge*, *Handover in a different room*, and *Firewall for the handover*). These patterns are intended for anyone who involved in a handover process and many people empathized about the utility of these solution patterns.

However, our pattern language has potential for further development. Experienced business people may have additional ideas for handover solution patterns. These ideas are indispensable for the development of our pattern language. To generate new ideas, we opened our pattern language to the outside to obtain new opinions using two workshops about the pattern language for handovers as well as via a Wiki-site (https://www65.atwiki.jp/handover/). The workshops revealed 22 possible new patterns.

In this paper we outline three new handover patterns which were identified in our workshops and advocated by many groups. These patterns are written as a mixture of the Alexandrian Pattern and Fearless Change Pattern forms [7] [10] [11]. The context, problems, solutions, and forces are written with stories. We begin each pattern with a quote followed by a brief description of the pattern. We then use a scenario to describe the context, problem, forces, and the solution. This storytelling form, which is similar to that presented in the Fearless Change patterns, can be easily understood. To describe concrete stories, we assume a fictitious company, Waseda Co., Ltd.

These patterns are intended for anyone involved in the handover process. The scenarios assume a systems department, but we expect that even those without a systems' background will find the paper useful since a handover is a common business concept, independent of domain. Concrete scenarios are used to help the reader understand the use and interactions of the patterns. In this paper, the scenario used to illustrate the patterns is when someone transfers within a company. However, these patterns should also be applicable to other contexts such as retirement and job changes. These scenarios assume that there is time for a formal handover (e.g., a few weeks or a few months) with the person transferring within the company.

The rest of the paper is organized as follows. Section 2 introduces concrete handover scenes using the example of a fictitious company to illustrate our handover patterns. Section 3 summarizes our workshops. Section 4 presents three handover solution patterns (*Ability checks*, *Pair working*, and *Matching the knowledge*) in detail. Section 5 concludes this paper and outlines future work.

2. SCENE

In this section, we outline a concrete scene that we will use to describe our patterns. Each pattern uses a scenario from this scene to explain its contexts, forces, problems, and consequences. We found these concrete scenarios aided in the comprehension of a pattern and can be used to evaluate effectiveness [12]. (Note that this paper uses the same scene presented in our previous paper at PLoP 2016.)

2.1 Example Scenario

Hiro works at Waseda Co., Ltd., one of the largest companies in Japan. Its business focuses on systems auditing and consulting. Hiro belongs to the internal systems department and is responsible for the development, operation, and maintenance of internal systems. He has been in charge of the financial system for a long time. However, Hiro is being transferred overseas. Kei, a mid-level employee in another department, has been appointed as Hiro's successor. Before leaving, Hiro has to handover knowledge of the financial system to Kei.

The financial system was developed in Java about ten years ago. It is one of the biggest internal systems in Waseda Co., Ltd. Hiro has been in charge of the system since its inception. The system outputs the settlement of accounts for external auditors and the government for tax purposes. An accountant in charge inputs accounting data. The system must be revised when the tax law and other requirements change. Additionally, scheduled maintenance also causes downtime of the system.

2.2 Characters

In the following examples, the handover consists of the activities of three actors (predecessor, successor, and third party), but there are four characters: predecessor (Hiro), successor (Kei), and the third party's role is shared by two people (Joe and Yoshiaki). Each character plays a unique role.

Hiro, who is a veteran worker that has been in charge of the financial system for a long time, is being transferred overseas. Although Hiro is staying within the organization, he is moving to a different department, and possibly in another location. He will not be directly involved with the financial system after the handover, but he can answer questions from the successor. Kei is a mid-level employee, who has been appointed as Hiro's replacement. She has been an employee of Waseda Co., Ltd for a couple of years, but in a different department. Joe is a veteran worker and a colleague of Hiro. He also works for the internal systems department, but is in charge of a different system. Yoshiaki is the project manager of the financial system and Hiro's boss.

3. SUMMARY OF OUR WORKSHOP

To get a better understanding of handover patterns and mine for other possible patterns, we held two workshops about pattern language for handovers: one of these workshops was held at Waseda University and we also held a Focus Group at PLoP2016 [13] (see pictures below).

Figure 1. Workshop in Waseda University

Figure 2. Workshop in Focus Group at PLoP

The primary goal of the workshops was to find the sequences of our 25 handover solution patterns [9] as well as to elucidate new handover solution patterns. The sequence of patterns is an index of the patterns [14], which details a path through a pattern language and the process to build something [15]. There is much discussion and dialog when finding the sequences of patterns, which often leads to new patterns for your language. Therefore, as our participants discussed the sequences of our 25 patterns, they mined 22 possible new solution patterns outlined in Table 1.

Table 1. List of new handover solution patterns

	Pattern name	**Summary of the pattern**	**Number of groups**
1	Simplify the system	Keep system simple, allowing knowledge to be easily handed over.	1
2	Automate a way the need for handover	Automate the work to avoid a handover.	1
3	Pay a person to maintain now	Keep the predecessor inside the organization.	1
4	Relief pitcher	Find temporary substitutes from the company.	1
5	Ability check	Check the ability of the successor before the personnel change.	3
6	Vacation chaos monkey	Randomly give someone vacation to see what happens.	1
7	Find another successor	If the test period does not work well, find another successor.	1
8	Work as a team for a while	Before the predecessor leaves, work as a team to aid in a seamless handover.	1
9	Pair working	For critical tasks, avoid handovers by using a pair programing.	3
10	Cross training	Work with a rotation between more than two people and share their work.	2
11	Make key person dependencies visible	Clarify the ownership of knowledge.	1
12	Develop a review process for docs	For example, create a review manual.	1
13	Matching the knowledge	Successor takes a note to confirm the understanding is right.	3
14	Prioritize	Prioritize knowledge when there is insufficient time for a comprehensive handover.	2
15	Handover evaluation	Evaluate and visualize risks for handover failure.	1
16	Get motivated	Recognize the successor's contributions to increase amount input.	1

17	Official project by sponsor	Recognize a handover as an official project, to secure the time and human resources.	1
18	Order of transfers	The most knowledgeable and experienced person leaves last when most of a team transfer.	1
19	Meeting together	Have the successor and predecessor meet prior to a handover.	1
20	Professional questioner	Include a person who asks insightful questions to avoid loss of tacit knowledge.	1
21	Create new knowledge	Recreate lost knowledge.	2
22	Accept reduced capacity	If the knowledge is lost, accept the reduced capacity.	2

We recruited people interested in handovers. About 25 people participated in our workshops in Tokyo and Illinois. These focus groups were intended for anyone involved in the handover process. We divided participants into groups and distributed pattern cards that briefly described our 25 patterns (see Figure 1 and Figure 2).

Then each group brainstormed the pattern sequences, priorities, new issues, and solutions not on the cards. Based on the discussion, the participants created a pattern map by identifying the sequence of the pattern cards. In addition, we asked the participants to write down new patterns on blank cards. Finally, each group shared their pattern maps and any issues they noted. These workshops led us to find the sequence of the patterns. We outlined 5 pattern maps and 22 possible new patterns. We do not limit participants of our workshops to IT companies and Japanese companies. Therefore, these new patterns can be used in fields other than IT and in countries other than Japan.

The “Number of groups” shows the number of groups that proposed the same pattern. According to the table, eight patterns were proposed by multiple groups, suggesting that they might have some universality. The higher the Number of groups, the more people are suggesting, and has used by many people. Therefore, we chose the pattern with the most proposers. However, some patterns proposed by a single group were evaluated as ingenious and interesting; for example, some participants had a positive impression of "Evaluation of the handover". Overall, we received excellent feedback from the workshops, and the participants might have gained insight on how to achieve better handovers. We outline three new solution patterns, *Ability check*, *Pair Working*, and *Matching the Knowledge*. These patterns were proposed by the largest “Number of groups” and are presented in this paper.

4. NEW HANDOVER SOLUTION PATTERNS

In this section, we introduce in detail the three new handover solution patterns mined from our workshops. By using *Ability check*, the capacity of the successor can be verified in advance. *Pair working* avoids the handover, mitigating the impact of the personnel change. *Matching the knowledge* is a pattern to match the views of both the predecessor and the successor so the successor is better prepared for the explanation by the predecessor.

There are some other ways to help with handover or mitigate handover problems such as sharing knowledge ahead of time. For example, it is common as part of the agile to share the knowledge to others. By sharing knowledge and having more people understand issues related to the tasks, it helps reduce issues related with handover. The appendix outlines three class diagrams, which describe how the solution patterns solve a problem.

4.1 Ability check

"Seeing is believing" — Western saying
"One eye-witness is better than many hearsays" — Japanese saying

Before the actual change, check the successor's ability by making the successor work on behalf of the predecessor. This will help determine the successor's ability.

Hiro has been in charge of the financial system for a long time, but he will soon be transferring overseas. His successor Kei, is a mid-level employee of Waseda Co., Ltd. She is currently providing technical support in the sales department and has never worked on the financial system. Kei also lacks some of the technical background that Hiro has.

***

It is important to assess Kei's ability to determine if she is the right person to be Hiro's successor. Otherwise, important knowledge and technology that Hiro has might be lost due to her lack of ability. How can Yoshiaki quickly determine if Kei will be a good successor to Hiro?

Since the financial system is very important, it is necessary to avoid any troubles after Hiro transfers overseas. Understanding the financial system can be difficult as the system is very complex. The knowledge and experiences of the person in charge is a key factor for smooth system operations. However, nobody of the internal system department has worked with Kei, and her ability is unknown.

There is limited time before Hiro takes on his new position. Also, there is not much support staff or people to assist with the handover or finding good people. It is important to quickly find out who can be a successful successor before much time is committed towards the handover.

Moreover, Yoshiaki cannot spare much time in judging Kei's ability. Yoshiaki must quickly determine her ability and decide whether she is suitable person for Hiro's successor or not. So he must take the most effective way to access her ability to know if she is the right person to replace Hiro or not.

Therefore:
Have Yoshiaki request that Kei come to the internal system department for few days to assess her ability. **To access the ability of the successor, Yoshiaki will ask Kei and Hiro to meet and work together. Based upon this work and dialog, an assessment of Kei can be done to get a feel to see if Kei can be Hiro's successor.**

Yoshiaki requests from Kei's current boss if Kei could come to the internal systems department for a few days in order to determine if she is a good fit as Hiro's successor. With her current boss approved, Yoshiaki set up an ability check and uses this opportunity to validate Kei's ability. With this check, Yoshiaki is able to know how much to support Kei after the change. During the ability check, Kei performed the job without any trouble. Yoshiaki is confident that that Kei is the appropriate person to be Hiro's successor.

If there are any problems, Yoshiaki can determine if they need to train Kei more or follow up Kei's ability. But if they cannot do so or do not believe Kei is a good successor, they should find another potential successor.

***

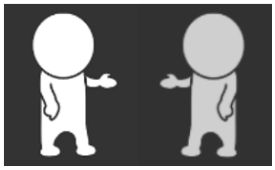
In Japan, new employees in many companies have three to six months trial employment period. However, it might be too long of an ability check for a handover. Also, even if the person has worked in the company for over six months, they still might need to be evaluated if they are the best person for taking on the new responsibilities. Thus, the third party should determine the appropriate duration of the Ability check considering the timeframe of the personnel change. Note that the person in charge of the financial system during the ability check is still the predecessor, Hiro. Yoshiaki must not push responsibility or have too high of expectations for Kei since she is not yet the person in charge. Some Japanese business people said this pattern is desirable to be executed, but it is difficult due to chronic staffing shortage.

By using the *ability check*, *apprentice* [9] is automatically executed. The *ability check* is one method to realize *successor as capable as predecessor* [9]. If the successor is unable to fulfill the expected role during the *ability check*, then *find another successor* should be used (Table 1). However, if time or other reasons prohibit considering another successor, the third party must be prepared for *support developing successor* and *reunion for old members* [9] to fulfill the successor's lack of ability. *Vacation chaos monkey* (Table 1) is a variation of this pattern.

4.2 Pair Working

"Save up for a rainy day" — English saying

It is impossible to reduce the risk of loss of knowledge in a handover to 0%. To help mitigate the risk, handovers can be minimized by having more than two people do the same job.



The financial system is one of the most important systems in Waseda Co., Ltd., and it must be stable in any situation. Yoshiaki chose Hiro as the person in charge of the system as he was a project member during the system development and has much experience. Also, he proved his value in maintaining the financial system. However, Hiro now has the opportunity to transfer to a new division and given his transfer, a handover cannot be avoided.

***

How can Yoshiaki avoid the risk of loss of knowledge due to a handover as part of the critical financial system?

The financial system is a critical system. It cannot lose data or stop business. Hiro surely can operate the system stably, but the risk caused by his personnel change cannot be avoided unless Yoshiaki can create a situation that help minimize handover problems.

In case Hiro is transferred, many people that are part of the internal section is busy and they do not have enough time to follow up and help the successor who has just joined the department and is not accustomed to how the financial system works. It is important to reduce the risk of personnel changes.

Therefore:
Appoint Joe and Hiro as the person in charge of the financial system to ensure a backup of the knowledge. Have them pair together and share important knowledge about the working of the financial system.

There are other members in the internal systems department besides Hiro and Yoshiaki. Joe is one of them. He is a veteran employee and has worked with Hiro for a long time. It is possible for Hiro and Joe to work together. Hiro and Joe have worked on the operation of the financial system together, sharing knowledge and technology with each other. Several years later, Hiro is transferred to overseas, but Joe has knowledge of the financial system. Thus, knowledge is not lost when Hiro transfers. Joe also shares work with Kei, the successor of Hiro. Even if there are personnel changes, because there are two persons who understand the system, one remains, ensuring that knowledge is never lost.

***

The pair combination must be carefully selected to ensure pair compatibility. Using this pattern is expensive as it doubles the personnel cost. Consequently, this pattern should be used only for critical work. If this pattern is cost prohibitive, another option is *Cross-training* (Table 1). *Cross training* that makes multiple persons in charge share their work has the same effect as *pair working*, and it does not cost than *pair working*. Because of that, *cross training* is commonly used in Japan.

Some organizations or practices focus on this sharing of knowledge as part of their corporate culture. For example, eXtreme Programming (XP) [16] and Mob Programming [17] continuously shares information by pairing and rotating tasks that people work on. For all production code, there is always at least two people that participate in maintaining and evolving the system. *Spread of knowledge* [9] shares the knowledge of the predecessor to a third party before handover. In contrast, this pattern always tries to share knowledge. *Pair working* is a development of the *Spread of knowledge*. This pattern is similar to *Pair programming* [18] because this pattern focuses the effect of *Pair programming* in a handover.

4.3 Matching the knowledge

"Change before you have to "— Jack Welch, (American businessman)

Sometimes the explanation and the document created by the predecessor are difficult for the successor or third party to understand. In such a case, the successor should actively replace them with documents to match the predecessor's knowledge and the successor's understanding.

As soon as Hiro accepts the overseas transfer, he begins his handover with his successor Kei. Since knowledge about the financial system is very important, Kei tries to understand Hiro's explanation. But Hiro's explanation is not good, and Kei cannot understand his telling. What Hiro wants to convey and what Kei understands do not match.

***

Hiro is not very good at explaining, and he cannot successfully communicate his knowledge to Kei. Hiro's knowledge and understanding of Kei are totally different. What can Kei do to match Hiro's understanding and her understanding?

Although it is important to create and transfer documents to a successor in a personnel change, these internal documents are rarely referred to. Often issues arise because the predecessor's communication skills are insufficient or the predecessor tends to postpone creating such documents [18]. In such situation, discrepancies tend to occur between the understanding of successors and the knowledge of their predecessors.

Because Kei is motivated to learn about the financial system, she takes the handover seriously. She takes notes of Hiro's explanations and later organizes her notes to make the knowledge easier to understand.

Therefore:
Kei visualizes her understanding with notes and let Hiro check her notes. Hiro can easily validate Kei's understanding and correct her notes. By doing so, the knowledge of Hiro matches Kei's understanding and avoid misunderstanding.

Kei took notes during Hiro's verbal explanations as well as when she read the handover documents. Note taking and supplementary explanations from self-learning help Kei grasp the knowledge. Moreover, these materials allow Kei to ask Hiro if her understanding is correct. As a result, Kei can match her understanding and Hiro's knowledge and successfully understand the financial system.

***

The important point of this pattern is that the successor should confirm his or her understanding of the knowledge with the predecessor to avoid misunderstanding the handover knowledge. Even when the explanation and any documents by the predecessor are easy to understand, this pattern provides utility as taking notes and writing supplementary explanations are great help in understanding knowledge.

Immediate question at the unclear spot [9] is used to solve the successor's question. *Matching the knowledge* is used when *Immediate question at the unclear spot* cannot cover all the questions. This pattern is reinforced by *Incremental handover* [9] since it increases the opportunity to confirm with the predecessor. Motivation of the successor is a key factor in this pattern. The third party should enhance the successor's motivation for the handover by using *Evaluation of the handover* and *Get motivated* (Table 1).

5. CONCLUSION AND FUTURE WORK

To improve our pattern language, we opened a Wiki-site and held workshops to increase feedback about our pattern language. Herein we discuss 22 new patterns discovered in our workshops, which are effective methods for pattern mining. Additionally, we describe three of these new patterns in detail (*Ability check*, *Pair working*, and *Matching the knowledge*). To better understand these patterns in relation to the handover issues, we used concrete scenarios.

We envision four future works for a pattern language for handovers. First, our pattern language defines the third party as an indirect person for the handover. In reality, there are various kinds of indirect persons for handovers. We need to define the roles of these other persons. Second, our pattern language does not address important questions as part of the handover process, including, "What should be shared verbally and what should be formally documented?" Third, the sequences of the new handover solution pattern and our handover solution patterns have yet to be defined. We should think about sequences. Finally, some patterns remain to be discovered.

ACKKNOWLEDGEMENT

We thank all the respondents of the questionnaire. We would also like to thank our shepherd Hongyu Zhang for his valuable feedback during the shepherding process. Finally, we thank our workshop members in Asian PLoP, Ayaka Yoshiakawa, Yuma Akado, and Yu Chi. Cheng.

REFERENCES

[1] Ahmad Salman Khan., Mira Kajki-Mattsson. 2010. "Core handover Problems," in Proc 11th International Conference on Product Focused Software 2010 (PROFES'10), Limerick, Ireland, 2010., 135-139

[2] Ahmad Salman Khan., Mira Kajki-Mattsson. 2010. "Taxonomy of Handover Activities," in Proc 11th International Conference on Product Focused Software 2010 (PROFES'10), Limerick, Ireland, 2010., 131-134

[3] Thomas Volleman. 1990. "Transitioning From Development to Maintenance," in Proc Software Maintenance, 1990, Proceedings., Conference on 1990, San Diego, USA, 1990., 189-199

[4] T. Grandon Gill., Eli Cohen. 2008. "Research themes in complex Informing", Informing Science: the International Journal of an Emerging Transdiscipline (2008, Vol. 11) 1)., pp. 147-164

[5] Yuma Akado., Shiori Shibata., Ayaka Yoshikawa., Akimitsu Sano., Takashi Iba. 2016. "Cooking Patterns: A Pattern Language for Everyday Cooking," in Proc 5th Asian Conference on Pattern Language of Programs 2016 (Asian PLoP 2016), Taipei, Taiwan, 2016.

[6] Alice Sasabe., Taichi Isaku.,Tomoki Kaneko., Emi Kubonaga., Takashi Iba. 2016. "Parenting Patterns: A Pattern Language for Growing with your Child," in Proc 5th Asian Conference on Pattern Language of Programs 2016 (Asian PLoP 2016), Taipei, Taiwan, 2016.

[7] Manns, M. L., and Rising, L. Fearless Change: Patterns for Introducing New Ideas. Addison-Wesley Professional, 2004.

[8] Kei Ito., Hironori Washizaki., Yoshiaki Fukazawa. 2016. "Handover anti-patterns," in Proc 5th Asian Conference on Pattern Language of Programs 2016 (Asian PLoP 2016), Taipei, Taiwan, 2016.

[9] Kei Ito., Hironori Washizaki., Joseph W. Yoder., Yoshiaki Fukazawa. 2016. "A Pattern Language for Handover," in Proc 23rd Conference on Pattern Language of Programs 2016 (PLoP 2016), Illinois, U.S., 2016.

[10] Joseph W. Yoder., Rebecca Wirfs-Brock. 2016. AsianPLoP Pattern Writing Bootcamp, http://pl.csie.ntut.edu.tw/asianplop2016/files/AsianPLoP2016BootcampPatternWriting.pdf

[11] Manns, M. L., and Rising, L. More Fearless Change: Strategies for Making your Ideas Happen. Addison-Wesley Professional, 2015.

[12] Joseph W. Yoder., Rebecca Wirfs-Brock., Ademar Aguiar. 2014. "QA to AQ Patterns about transitioning from Quality Assurance to Agile Quality," in Proc 3rd Asian Conference on Pattern Language of Programs 2014 (Asian PLoP 2014), Tokyo, Japan, 2014.

[13] Kei Ito., Hironori Washizaki., Joseph W. Yoder. 2016. PLoP Focus Group, https://www.hillside.net/plop/2016/index.php?nav=program#focusgroups

[14] Christopher Alexander., Sara Ishikawa., Murray Silverstein. 1977. A Pattern Language: Towns, Buildings, Construction. Oxford University Press, Inc

[15] Neil B. Harrison., James O. Coplien. "Pattern sequences" in Proc 6th European Conference on Pattern Languages of Programs 2001 (Euro PLoP 2001), in Kloster Irsee in Bavaria, Germany, 2001.

[16] Kent Beck., Cynthia Andre. 2004. Extreme Programming Explained: Embrace Change, 2nd Edition (The XP Series). Addison-Wesley; 2nd edition

[17] Woody Zuill., Kevin Meadows. 2016. Mob Programming. Leanpub

[18] James O. Coplien., Neil B. Harrison. 2005. Organizational Patterns of Agile Software Development. Pearson Prentice Hall

APPENDIX

We describe the status before and after using the patterns by class diagrams and object diagrams to express the results of the patterns. We used UML, to help readers understand where the problems exist, how a pattern works, and what problems are resolved. These diagrams help readers understanding the mechanism of the problems and how the patterns work. Herein we introduce three figures to describe *"Ability check"*, *"Pair working"*, and *"Matching the knowledge"*.

Ability check

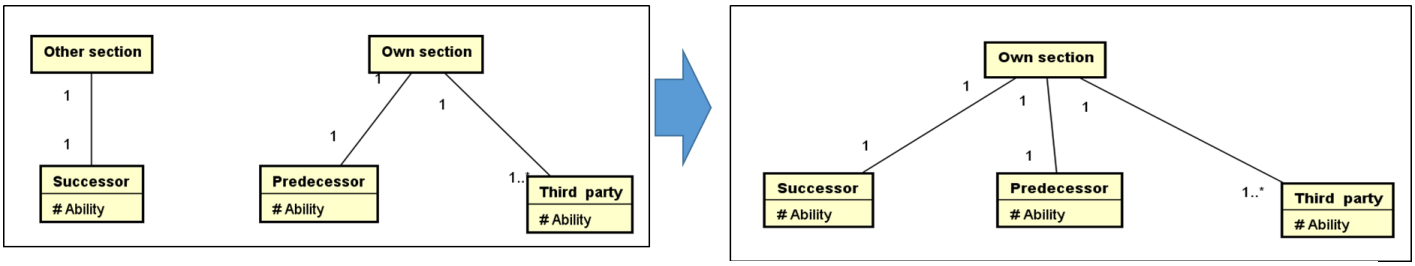


Figure 1. Ability check

As seen in Fig. 1, the capacity of each individual can only be referenced within the same section. However, the successor is in a different section from the predecessor and the third party. Thus, they do not know the successor's capacity. However, by using *Ability check*, the successor will belong to the same section as the predecessor and third party, allowing the third party to access the capacity of the successor.

Pair working

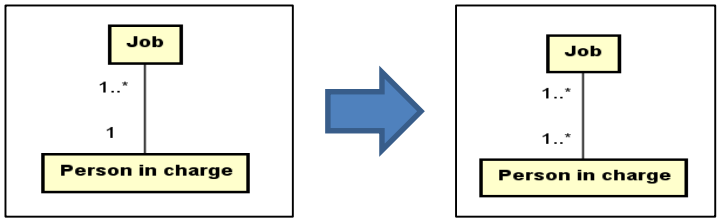


Figure 2. Pair working

Before using *pair working*, a job and the person in charge is a multiple-to-one relationship. In this state, there is a great risk that the person in charge will be changed due to personnel changes, making it impossible to avoid a handover. *Pair working* makes a job and the person in charge a multiple-to-multiple relationship. Even if one person in charge is changed, there are other persons in charge, greatly reducing the impact of a personnel change. Particularly important job should be handled by more than one person.

Matching the knowledge

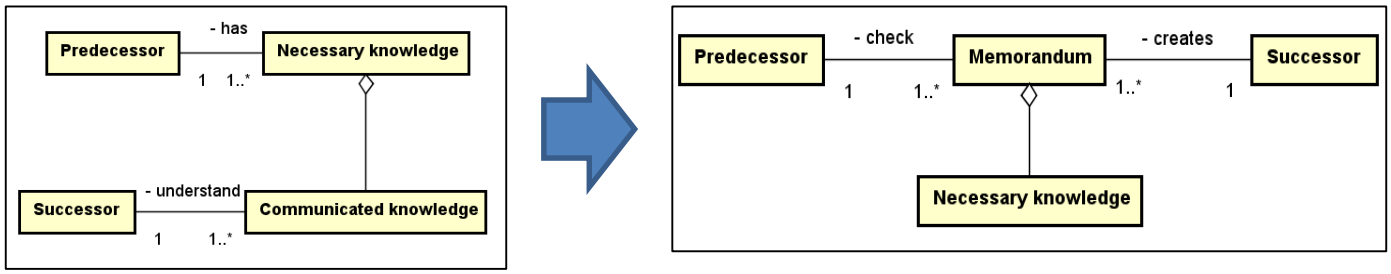


Figure 3. Matching the knowledge

Before using this pattern, the predecessor and the successor see the different knowledge, creating a gap between the knowledge that the predecessor and the successor. However, a memorandum created by the successor, makes the predecessor and the successor see the same item, and they can match their knowledge. Thanks to a memorandum (*matching the knowledge*), the successor acquires necessary knowledge.