A Pattern Language for Inter-team Knowledge Sharing in Agile Software Development

VIVIANE SANTOS, ALFREDO GOLDMAN, University of São Paulo
EDUARDO GUERRA, National Institute for Space Research
CLEIDSON DE SOUZA, Vale Institute of Technology and Federal University of Pará
HELEN SHARP, The Open University

Inter-team knowledge sharing is an important aspect of agile software development because it allows agility to be scaled to an entire organization. However, achieving inter-team knowledge sharing is not easy because the practices that allow it to happen are poorly reported in the literature. As a consequence, many agile software organizations find it difficult to share experiences and best practices in their contexts, as well as improve employees' skills and expertise. Results from our previous qualitative studies enabled us to generate a pattern language that aims to help agile software organizations to adopt practices for fostering interaction among agile teams in order to share knowledge across teams and create collective knowledge. This paper contributes to the agile software development and knowledge management fields with practical guidance to organizations aiming those improvements. Because of space limitations, we describe only five patterns of the pattern language in this paper, which are briefly presented as follows. `Open Workspace` helps to stimulate face-to-face conversations across teams. `Rotation of Teams' Members` is a practice about transferring of professionals to other teams in order to spread technical, methodological and management solutions in a sustainable way. `Pair Programming among Different Teams` are specially adopted to level technical knowledge throughout the company. `Collective Meetings` foster inter-team communication and alignment about company's projects and goals. Finally, `Technical Presentations` stimulate continuous learning and knowledge sharing behaviour. It is important to note that the adoption of the pattern language is affected by forces, such as organizational culture, environment, and top management and leadership support. These forces need to balanced to facilitate and/or reinforce the patterns.

Categories and Subject Descriptors: **D.2 [Software Engineering]; D.2.9 [Management]**: *Programming teams; Software process models (e.g., CMM, ISO, PSP)*; **K.6.1 [Project and People Management]**: *Systems development*; **I.2.6 [Learning]**: *Knowledge acquisition*; **H.5.3 [Group and Organization Interfaces]**: *Computer-supported cooperative work*; **K.3.1 [Computer Uses in Education]**: *Collaborative learning*

General Terms: Human Factors, Management

Additional Key Words and Phrases: Inter-team Knowledge Sharing, Agile Software Development, Knowledge Management, Organizational Learning

ACM Reference Format:

Santos, V., Goldman, A., Guerra, E., Souza, C. R. B., Sharp, H. 2013. A Pattern Language for Inter-team Knowledge Sharing in Agile Software Development. 20th Conference on Pattern Languages of Programs (PLoP), Allerton Park in Monticello, IL (October 2013), 21 pages.

1. INTRODUCTION

Intra-team tacit knowledge sharing is fairly promoted by agile methods through face-to-face conversations, strengthening of relationships, cross-functional and self-organizing teams, adoption of agile practices, and customer involvement (Karlsen et al., 2011). However, this is only the first step for creating an enabling context for knowledge sharing in the company. The other aspect is how to scale and reuse local knowledge across teams in an agile company. In fact, inter-team knowledge sharing enables the creation of collective knowledge that consequently improves organizational quality, performance, and innovation. Thus, agile software organizations (as well as non-agile software organizations) may use practical guidance on inter-team

This work was supported by FAPESP project no. 2009/16354-0 and CNPQ project no. 476661/2010-2. We are grateful to all organizations and members participating in the study. Special thanks to Allan Kelly, his valuable feedback and advices during the shepherding process helped us to improve our paper a lot.
Author's address: Department of Computer Science, Rua do Matão, 1010, Cidade Universitária, São Paulo, SP, 05508-090, Brazil; e-mail: vsantoa@ime.usp.br; e-mail: gold@ime.usp.br; Laboratory of Computing and Applied Mathematics (LAC), P.O. Box 515 – 12227-010 São José dos Campos, SP, Brazil; e-mail: guerraem@gmail.com; Tv. Dr. Moraes, 78, Nazaré, Belém, Pará, 66035-080, Brazil; e-mail: cleidson.desouza@acm.org; Walton Hall, Milton Keynes, Buckinghamshire MK7 6AA, UK; e-mail: h.c.sharp@open.ac.uk

knowledge sharing to enhance competitiveness. Despite its importance, there are only a few recommendations in literature about how to achieve inter-team knowledge sharing.

To address this gap, this paper presents a pattern language that is based on our previous qualitative studies (Santos et al., 2012) (Santos et al., 2013b) conducted in eight agile software organizations for about 2 years. These studies resulted in a conceptual model that captured the main aspects of inter-team knowledge-sharing mechanisms.

Figure 1 illustrates our final model. The studied organizations apply practices for socialising knowledge in order to build their organizational knowledge and competitiveness. The investigated companies empower the interactions among teams as a way to avoid feeling vulnerable by not making explicit most of their knowledge.

Inter-team knowledge sharing in agile contexts is triggered by different stimuli, such as problem-situations, common interests and/or needs, incentives, and sustainable pace. The adoption of practices for fostering interaction among teams is facilitated and reinforced by enablers, e.g., top management and leadership support, agile adoption, communication flow and channels, and integration among teams. Agile teams often select the practices according to purposes, which are the reasons for sharing knowledge, for instance to identify knowledge owners.

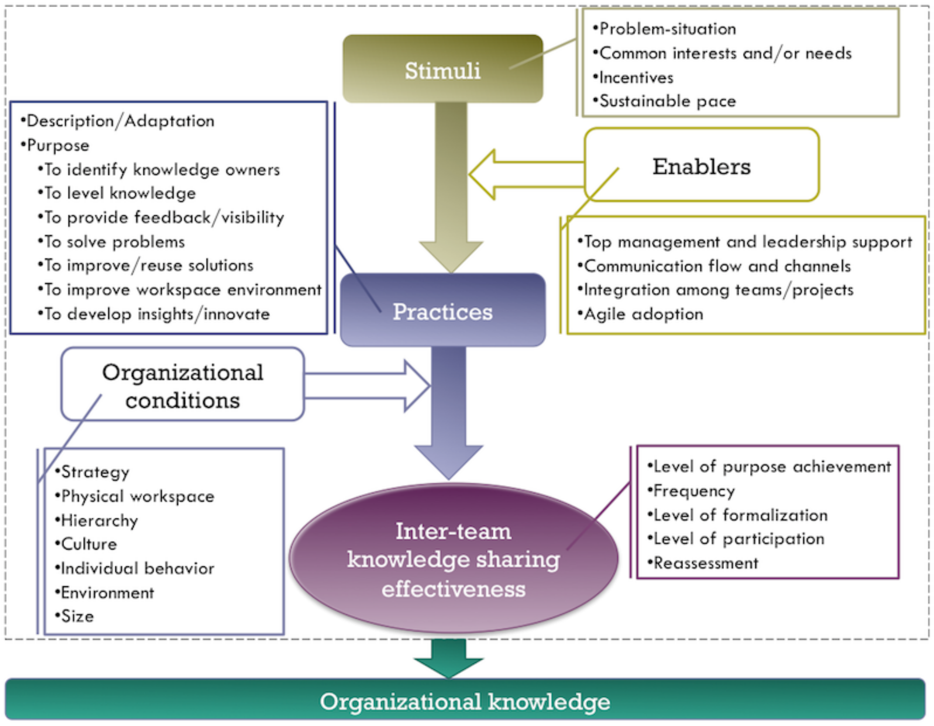
The effectiveness of each practice is measured by five components, which encompass the level of purpose achievement, frequency, level of formalization, level of participation, and reassessment of the practice.

We observed in our studies that great deal of attention is devoted to create and sustain adequate organizational conditions, such as organizational culture, strategy, and physical workspace, as these deeply influence inter-team knowledge sharing effectiveness. In summary, through our model it is possible to clarify what an agile software organization needs to consider when scaling knowledge sharing initiatives to create organizational knowledge. In this paper this model is presented as a pattern language that aims to help the adoption of knowledge sharing practices across agile teams.

In observance to the space limitation, we only describe five patterns from our inter-team knowledge sharing pattern language. Following we provide a brief description of these patterns:

- `Open Workspace`: When the company needs to stimulate face-to-face conversations across teams, create open workspaces for the whole company to facilitate spontaneous and informal conversations;
- `Rotation of Teams' Members`: When you need to prepare team members to work on different projects throughout the company and to improve overall motivation, employ rotation among different teams[1] and projects in order to spread technical, methodological and management solutions to other teams in a sustainable way;
- `Pair Programming among Different Teams`: When overall collective code ownership and internal software quality need to be enhanced in the company, adopt pair programming among different teams involving people knowledgeable in specific issues to level knowledge throughout the company;
- `Collective Meetings`: When you need to foster inter-team communication and awareness about company's projects, adopt collective meetings to improve knowledge sharing throughout the company and to provide wider organizational alignment;
- `Technical Presentations`: When the company needs to stimulate continuous learning and knowledge sharing behaviour, employ periodic technical presentations in the company.

[1] Different teams mean teams that work for different projects or customers.


•Problem-situation
•Common interests and/or needs
•Incentives
•Sustainable pace
Stimuli
•Description/Adaptation
•Purpose
•To identify knowledge owners
•To level knowledge
•To provide feedback/visibility
•To solve problems
•To improve/reuse solutions
•To improve workspace environment
•To develop insights/innovate
Enablers
•Top management and leadership support
•Communication flow and channels
•Integration among teams/projects
•Agile adoption
Practices
Organizational conditions
•Strategy
•Physical workspace
•Hierarchy
•Culture
•Individual behavior
•Environment
•Size
Inter-team knowledge sharing effectiveness
•Level of purpose achievement
•Frequency
•Level of formalization
•Level of participation
•Reassessment
Organizational knowledge
Fig. 1. Conceptual model on inter-team knowledge sharing in agile environments.

In Section 2 we briefly describe how the pattern language was mined and documented. Section 3 presents the pattern language and explains five patterns under the Alexandrian pattern writing style (Alexander et al., 1977). Section 4 presents our final conclusions.

2. TOWARDS A PATTERN LANGUAGE FOR KNOWLEDGE SHARING ACROSS AGILE TEAMS

We conducted a qualitative study that aimed to understand how companies shared knowledge across its agile software development team. Table 1 below presents a summary of eight Brazilian agile companies. The last column (Nº OF AGILE TEAMS) presents a subset of all teams within the respective company, including developers, testers, systems analysts, and managers. In the first four companies we gathered data from interviews, questionnaires and observations in four iterative cycles of data collection and analysis with people playing different roles, such as managers, product owners, scrum masters, and developers, as described in (Santos et al., 2012). More precisely, we conducted a grounded theory study at these companies.

Then, in order to validate our resulting conceptual model, we selected additional companies considered by their employees as favourable to knowledge sharing. By favourable, we mean levels of collaborative attitudes, respect, mutual trust and camaraderie in the organizational context. After selecting and obtaining acceptance of the companies participating in the study, we employed another qualitative study with four companies to refine the conceptual model and provide a final contribution to the research field.

Table 1 Details about the studied organizations

COMPANY ALIASES	SIZE	BUSINESS DOMAIN	AGILE METHOD(S)	EXPERIENCE ON AGILE METHOD(S)	Nº OF AGILE TEAMS
Acme Inc	>1000	Content and services portal	Scrum and XP (some practices)	4 years	8
B-simple Inc	80	Local search platform	Scrum and XP (several practices)	3 years	3
Creative Inc	>60	Software development and training	Adaptation of Scrum, XP and Kanban	6 years	4
Dyna Inc	>100	Software development and consulting	Adaptation of Scrum, XP and Kanban	> 10 years	5

Eagle Inc	>80	Software development and training	Adaptation of Scrum, XP and Kanban	5 years	15
Factory Inc	>150	Software development	Scrum	6 years	25
Goal Inc	40	Software development and training	Adaptation of Scrum, XP and Kanban	> 5 years	3
Maps SA	200	Product development	Adaptation of Scrum, XP and Kanban	3 years	>10

The IBQ approach can help to determine the subjectively crucial characteristics of an image and therefore to give weights to objective and computational measures.

We applied constant comparison method (Corbin and Strauss, 2007), pattern-matching mapping (Yin, 2000), and pattern mining techniques (Rising, 1998) in order to discover and document the pattern language presented in the following section.

3. INTER-TEAM KNOWLEDGE SHARING PATTERN LANGUAGE

Figure 2 proposes a pattern language for improving knowledge sharing across agile teams. This language is composed of eight different patterns that emerged from eight agile software organizations, each one with different contexts and concerns about organizational knowledge creation. Each node represents a different pattern, while edges indicate the relationship between the nodes. For instance, we have a pattern called Open Workspaces that can be used to address the goal of "To stimulate face-to-face conversations across teams". In addition, an edge from Open Workspaces to Pair Programming Among Teams means that conversations in open workspaces may raise awareness on technical problems of different teams that can be solved by adopting pair programming across teams.

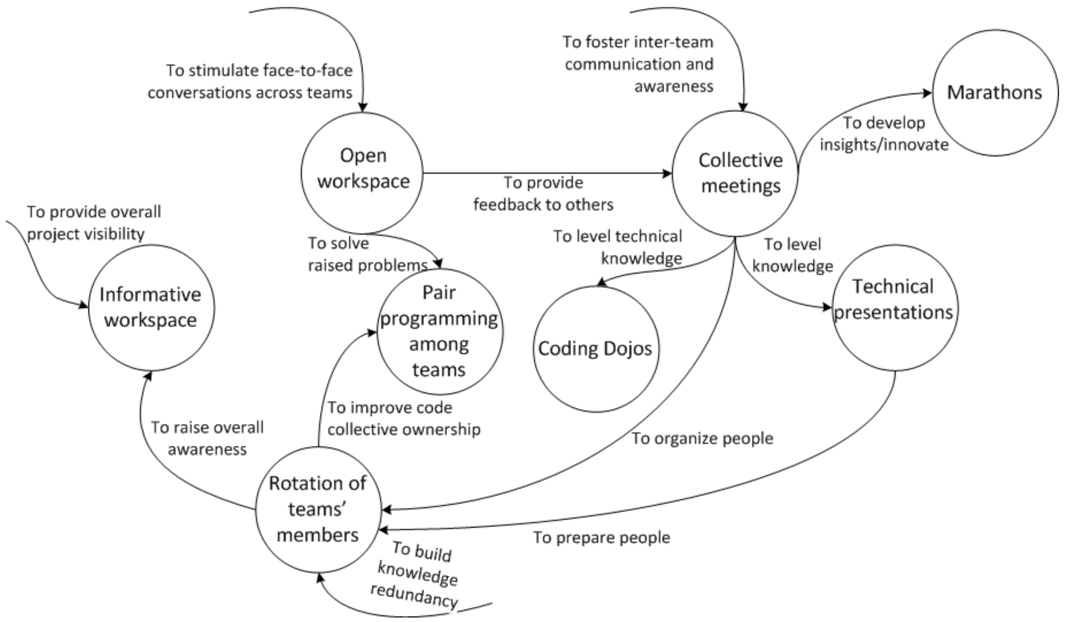

To stimulate face-to-face conversations across teams
To foster inter-team communication and awareness
Marathons
To develop insights/innovate
Open workspace
To provide feedback to others
Collective meetings
To provide overall project visibility
To solve raised problems
To level technical knowledge
To level knowledge
Informative workspace
Pair programming among teams
Coding Dojos
Technical presentations
To improve code collective ownership
To raise overall awareness
To organize people
Rotation of teams' members
To prepare people
To build knowledge redundancy

Fig. 2. Inter-team knowledge sharing pattern language.

3.1 Open Workspaces **

At Creative Inc people can sit wherever they want, next to anyone else. The developers themselves chose to have this freedom instead of their own tables or seat. As all members are together, the development room encourages making questions anytime. Anyone can answer and sometimes a debate will arise. Also, at Eagle Inc they state, "We try to encourage exchange and there are many things that people reuse. (...) the environment, the physical workspace, by having the video game room, coffee room, sofa room, we end up having [information] exchange[s] more naturally and we try to encourage some things."

Agile practices encourage face-to-face conversations within the team. However, face-to-face conversations among different teams' members are rare. Because of that, team members may face difficulty in identifying people outside their teams that are knowledgeable in specific topics to help solving problems, in providing feedback and visibility to other teams, in reusing specific technical, methodological and management solutions, and also in improving the workspace environment.

* * *

It is important to share information about techniques, practices and technologies among teams. How to stimulate face-to-face conversations across teams?

Co-located agile teams usually work nearby. Because of this physical proximity, several face-to-face conversations occur naturally. These conversations are important because they foster knowledge sharing, facilitate work coordination and help to build trust and camaraderie. However, some organizational conditions may affect the effectiveness of the overall face-to-face conversations. Thus, it is important to consider a physical structure with few barriers, an organizational culture that values openness and freedom of inquiry; organizational size, because the larger and more distributed, the more difficult to facilitate overall face-to-face conversations; and an organizational strategy that shows commitment to knowledge sharing.

The characteristics of the physical workspace impact on the way people share knowledge across teams. Physical workspace composed by rooms, walls, and partitions, considered as traditional offices, end up acting as barriers to informal conversations. As a consequence, the practice becomes less effective.

Organizational hierarchy also impacts on the way people establish their relationships and on the way people share knowledge across teams. A rigid organizational hierarchy with several levels of depth creates distance and barriers to informal conversations among people from different teams.

Even having strong agile teams with a culture of commitment, transparency, reciprocity, self-organization, solidarity, mutual trust, tolerance and freedom to admit mistakes, etc., the organizational culture may prevail on the team agile culture, which ends up as a barrier to the knowledge sharing behaviour.

Therefore:

Create open workspaces for the whole company to facilitate spontaneous and informal conversations.

Open workspaces can be achieved by providing offices with very few walls. The furniture needs to be flexible and characterized by plain tables without partitions, and chairs without arms to facilitate proximity. Place plain tables in the middle of the office, so the walls are used for informative workspaces. Employees work with laptops (not desktop computers) to facilitate their mobility and freedom to sit wherever they want in the workplace. This also facilitates the adoption of pair programming. There are also couches, tables and chairs for sitting around informally, and a projector area for presentations and virtual video-conferences with distributed teams.

Open workspaces allow employees to work in the same space, as illustrated in Figure 3. People's mobility is stimulated throughout the workspace and employees become easily aware of what others are doing because they can overhear conversations and even jump in when it is relevant for them. Overall face-to-face conversations might reach not only team's members, but also other teams throughout the entire organization.

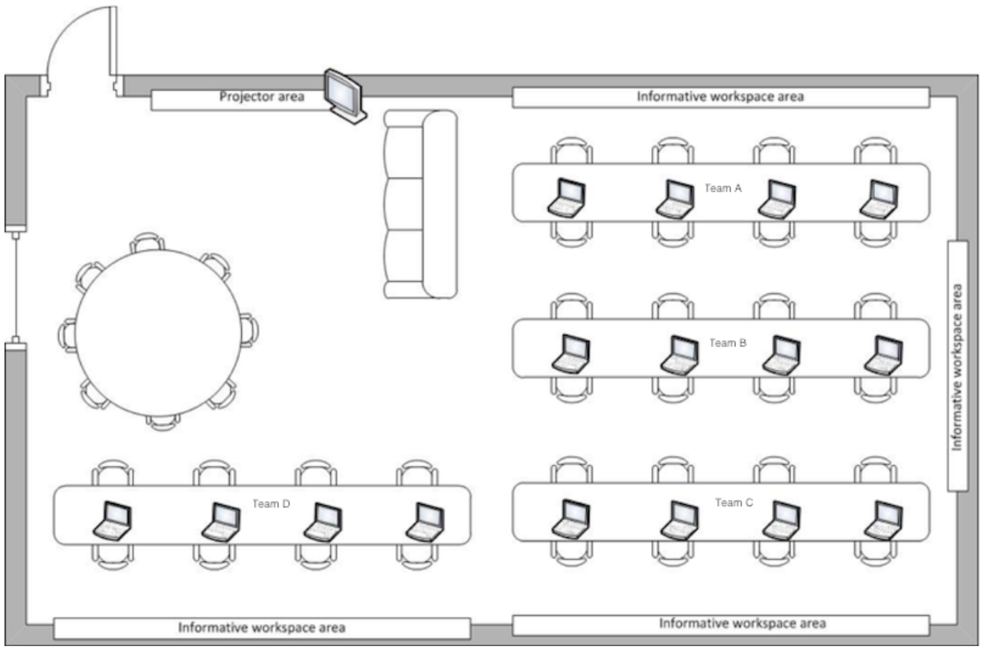

Projector area
Informative workspace area
Team A
Team B
Informative workspace area
Team D
Team C
Informative workspace area
Informative workspace area

Figure 3 - An example of open workspace.

Open spaces also impact on the organizational hierarchy as it demonstrates equality among all in the company and allow for easy access to others. Fewer levels and flexibility in organizational hierarchy promotes more effective knowledge sharing across teams, because it provides easier access to people and to teams' work processes and technical solutions. So, people become more aware of the latest issues of the company, interact more easily, and feel part of something bigger than just their own project or their own team. As a consequence, it reinforces individual behaviour and organizational culture.

Even when it is not possible to have open workspaces for the entire company, you can consider other relevant solutions. Create **semi-open workspaces**, which means open workspaces within rooms for a subset of teams, as presented in Figure 4. By doing so, you can place sub-teams[2] to foster more face-to-face conversations across these teams. Close to these semi-open workspaces, try to create **spaces for integration**, like shared rooms to bridge everyone in the organization, such as coffee, game, or resting rooms.

If it is difficult to change the traditional office settings in the company (e.g., it might not be possible to eliminate walls or desktop partitions for the whole company), consider at least to place **sub-teams close to each other** to foster inter-team face-to-face conversations at some point (Figure 5). Also consider creating spaces for integration, as described before.

[2] Teams that are interrelated and work for the same product or customer.

It is important noting that this pattern might be considered redundant by countries where it is normal to locate people in open workspaces, e.g., in European countries where the pattern is highly applicable. However, for instance, in the USA, employees can be highly resistant to the idea of giving up their own space. Since, we did not analyse this aspect deeply by country, we only outline that it is important to take into consideration the national norms and preferences about office settings.

Even assigning a great value to open workspaces, there is also a need for considering the trade-offs of this approach. Teams' members still need to coexist with the problem of noise, because this type of physical workspace environment may eventually harm some people's ability to focus. A general difficulty is to moderate people's noise, however, in this case, it is suggested to raise awareness of people regarding noise in the workspace and also provide some quiet areas, so that people can eventually work in a silent place, when needed.

Another negative consequence to consider when adopting Open Workspaces is the possible increase in office costs by investing on different furniture and settings. It is also important to consider the national norm on this, since costs might become lower in European countries, as open workspaces are considered a common practice for office settings. Another concern might be to ask people to move from their desks, but with the concept of open workspaces, there are no desk owners, people are free to sit wherever they need or want.

In addition to identifying open workspaces in our studies, we also observed that this pattern is recommended by the knowledge management literature. For instance, the term 'ba', introduced by Nonaka and Takeuchi (1995), consists of a space where knowledge is shared, created and utilized since the knowledge needs a context to exist. The main goal of 'ba' is to create interaction and it needs to be "energized" (encouraged) to become active and allow the construction of shared meanings (Berger and Luckmann, 1966) (Crossan et al., 1999) in the workspace.

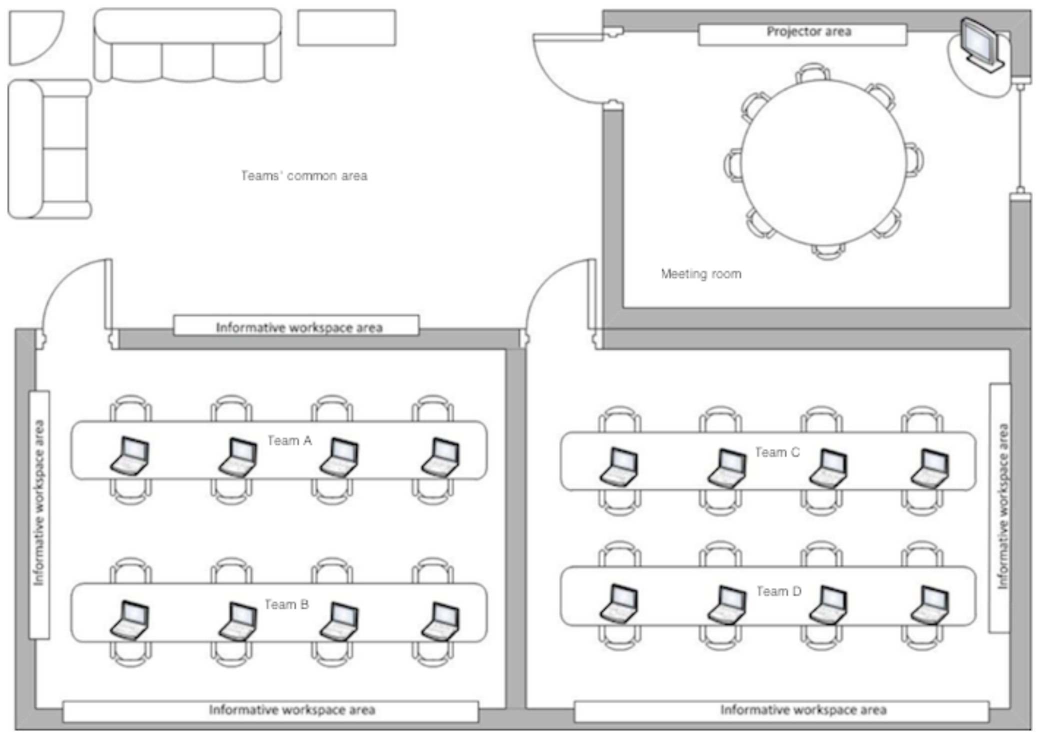

Projector area
Teams' common area
Meeting room
Informative workspace area
Informative workspace area
Team A
Team B
Team C
Team D
Informative workspace area
Informative workspace area
Informative workspace area

Fig. 4. An example of semi-open workspace with spaces for integration.

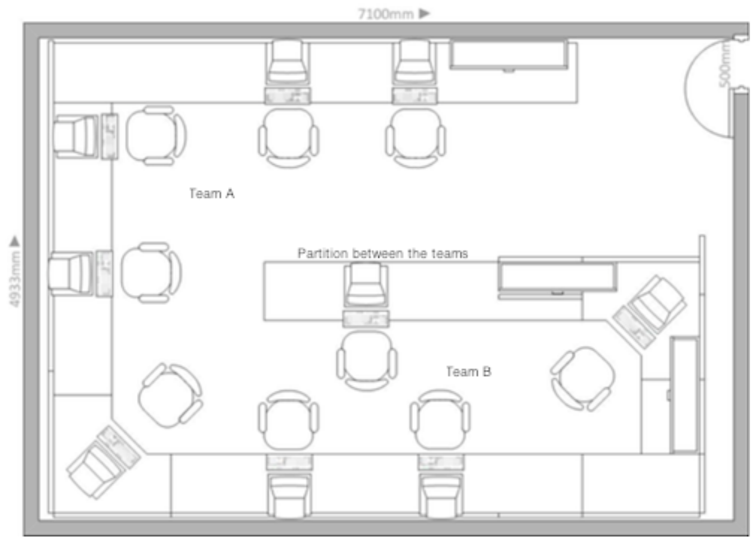


Fig. 5. Sub-teams nearby in traditional offices with spaces for integration.

In this perspective, the company physical structure is considered crucial to trigger interactions through informal conversations in the workspace. The more the physical workspace is conducive to facilitate interactions, the more people are willing to interact with others, sharing experiences, skills, and expertise to different organizational levels.

Von Krogh et al. (2000) also reported that open workspaces reflect the new logic of knowledge-intensive organizations, facilitating continuous exposure to all types of operations of the organization.

To improve the expected effectiveness of this practice, the company should analyse their actual status-quo to make the adequate improvements. For companies that adopt face-to-face conversations within the team and expect to achieve moderate effectiveness, they need to choose which level they wish to foster face-to-face conversations, such as across teams or whole organization, and then work for it.

* * *

`Organization Follows Location` (Coplien and Harrison, 2005) creates structures in the organization or in the workspace that encourage the communication connections that support other patterns.

`Hallway Chatter` (Coplien and Harrison, 2005) moves team members physically as close to each other as possible. Situate people with outer roles near the central roles. This pattern extends the adoption of War Rooms (Cockburn, 2004). A war room is a type of physical structure that places team members in the same room in the company. Cockburn has recommended this to provide more osmotic communication, which means information flow into the background hearing of members of the team that makes people pick up relevant information as though by osmosis. We suggest that it should be scaled to the company, or, at least, scaled to some inter-related development teams.

With osmotic communications questions and answers flow naturally. This makes the cost of communications low and feedback rate high, so that knowledge is disseminated quickly. For instance, people learn the project priorities and who holds what information; pick up new programming, design, testing, and tool handling tricks; and catch and correct small errors before they grow into larger ones.

Regarding noise, Cockburn (2004) explains that osmotic communication generates its own hazards, most commonly noise and a flow of questions to the team's most expert developer. People usually self-regulate themselves by requesting less idle chit-chat or more respect for think time. Thus, he recommends to set up a workplace or even working hours (agreed with all) with no interruptions at all and extremely limited communications with the team, as he termed `Cone of Silence`.

Beck (2004) also recommends providing a workplace with areas for communication or for concentration, remembering the pattern `Cave and Commons`, where an open bullpen should have little cubbies around the outside of the space. Cubbies would be used by team members to keep their personal items, make phone calls, and spend time at them when they don't want to be interrupted.

Bricout et al. (2004) also describe the pattern termed as `Prepared Workspace` to accommodate visiting team members and virtual meetings for managing distributed development teams. They suggest flexible furniture to easily move and reconfigure, as a way to convert a meeting area to workspace for the visitors.

`Personal Safety` (Cockburn, 2004) means to build mutual trust, in which requires exposure, e.g. revealing one's ignorance, a mistake, one's incapability on an assignment. When trust is built-up in a team, the leader and the team as a whole act to support their exposure.

By having an open workspace, the adoption of `Pair Programming among Different Teams`, spatial `Rotation of Teams' Members` are facilitated. It also helps dealing with information flow through `Public Character` and with change in `Conway's Law` (Coplien and Harrison, 2005). This pattern also contributes to `Watercooler` (Coplien and Harrison, 2005), which encourages social structures that are unrelated to workplace structures and that will likely cut across the formal partitioning of the organization. For instance, strong coffee culture that revolves around coffee machines, where one can find small groups congregating mid-morning and mid-afternoon.

Pierce (2011) in his blog article has stated, "The physical space we work in impacts on the way we feel, interact and how productive we are. (...) From an agile perspective, office spaces have an impact on a number of key factors:

- *How it makes you feel.*
- *How it encourages positive social interactions.*
- *How it enable communication.*
- *How it manages distractions.*
- *How it enables productive development.*

Lots of communication and a high level of development productivity are often in tension and balance needs to be struck."

3.2 `Rotation of Teams' Members **`

A development manager at Eagle Inc declared, "We apply rotation on projects to encourage the [information] exchange for a while. We try to make people rotate, 'not to spend too much time on the same project' and go to another project, to take things that are going on the projects. I think that even though there are quite different things happening, there are many tools and techniques in common that you will see several projects using, because we get this exchange of knowledge there. (...) It's difficult to say what it brings in a practical way, but I'm sure if they had not been in touch with what's going on out there, we would not be as competitive and offer a quality service that we can." And a project manager at this company added, "(...) it only requires preparation in advance. Sometimes you take a person of a project, who is key, but, before that, you need to prepare another one to temporarily substitute that person, so you can transfer the person to another project. It's something that has to be considered while composing existing teams. (...) we need to define a project that will be able to adjust the dates to fit with another project, because the rotation should not affect the project." At Acme Ince, they also conduct rotation as a successful practice to either improve knowledge redundancy and motivation, "(…) we have people who are in the same project since joining the company. So, sometimes people feel discouraged, it is good to do this rotation."

In an agile team, knowledge sharing is promoted through agile practices. However, companies, in general, need to take into account ways to scale knowledge within teams to other teams as well. Furthermore, depending on the company workspace and teams' proximity, face-to-face conversations might not be enough to disseminate knowledge throughout the company. Also, depending on the age the team has been composed, team members might feel discouraged by working in the same project for a long time.

* * *

How to bring solutions and practices used on one team to another without stopping the software development?

As agile teams have a great focus on product and delivery of value to customers (i.e. short-term strategies), business domain conditions or customer needs end up affecting the way companies deal with knowledge sharing initiatives. Since agile teams often complain about lacking time to stop the software development to promote trainings and workshops with their employees, when it comes to share knowledge across teams, this needs to be feasible endeavour in this context.

Even considering an Open Workspace, face-to-face conversations might not be enough to create organizational knowledge. For instance, scale the collective code ownership not only to the team, but to the company is important to empower the reuse or improvement of proven solutions in the company context. Because of that, knowledge sharing activities must have a low cost and a sustainable pace to not affect the agile way of delivering value to customer. Not only agile companies, but software development companies in general have difficulty in sorting these problems out.

Therefore:

Employ rotation among different teams and projects in order to spread technical, methodological and management solutions to other teams in a sustainable way, and to improve overall motivation.

Rotation is the transfer of professionals between teams whether in existing projects or for composing new projects. This practice relies on a clear organizational commitment to freely adopt it throughout the company, in which also needs top management support, leaderships' alignment, willingness to share knowledge by team members, and organizational culture of collaboration, respect, and tolerance and freedom for questioning and committing mistakes.

Thus, it is important to make **temporary rotation** of members from different teams, in an organized and clear way to everyone in the company, i.e., establish appropriate frequency, formalization and reassessment according to the company's context. For instance, every month try to stimulate this type of rotation with 5% of the company members. Then, in the long-term, several professionals would have the opportunity to rotate among different teams. This endeavour should be properly planned with leaders.

Rotation of teams' members influence the way people have access to different team tacit knowledge and help on increasing collective and organizational knowledge, as people become aware of diverse ways to deal with project, management, methodological issues. The access to different projects also enhances people's motivation.

Following we describe pattern variations. Another type of approach is the **rotation of seniors**, who are professionals respected for their technical expertise and their exemplary attitudes in the company. These seniors, often, can promote learning and reflection in teams as well as contribute to solving technical problems or conflicts.

Consider also performing the rotation of members between teams in **mentoring** sessions to help with specific issues, such as disseminating test-driven development or the use of certain technical solutions (e.g., reuse of a component or a framework).

In teams coordinated by the same manager, such as sub-teams from the same project or client, or sub-teams, the manager can handle the **sub-teams' members rotation** according to the situation of each team/project.

Depending on the company's physical and technological infrastructure, **spatial rotation** is enabled. This practice consists of changing teams' position in the open workspace. Just by moving teams around the open workspace, team' members might learn from listening the informal conversations of different teams.

The benefits of knowing lessons learnt and best practices from other teams are perceived mostly in the long-term, that is why an organizational commitment with long-term strategies is needed to consider knowledge sharing as an important resource for achieving the organizational goals. Depending on the strategy adopted by the company, it needs to be consistent with short-term and long-term company goals. This means establishing a general awareness that knowledge is part of the work processes and establish a clear and long-term process to achieve its goals.

As a consequence of applying `Rotation of Teams' Members`, it is important to consider that it might affect the balance of team morale. The newcomer might bring emotional stress to the team, for instance, by speaking less tactfully, which would impact the team environment. So, prepare the newcomer and the team regarding an overall culture of respect, mutual trust and commitment. Also, team leaders need to be prepared with proper techniques for conflict-resolution, if necessary.

Another consequence to consider is the impact of a newcomer affecting the team iteration delivery. That is why it should be carefully planned by the team leaders, such as (1) analysing the beginning and context of teams' iteration, (2) fostering appropriate attitudes and learning culture to the newcomer, (3) preparing the newcomer with the minimum needed skills to work in the team, (4) prepare the team to receive the newcomer, and (5) evaluate the newcomer's development during the iteration.

On the other side, both consequences promote disturbance in teams, which is valuable to foster reflection on how to improve their dynamics. As declared by Rolloff et al. (Easterby-smith and Lyles, 2011), multiple team membership helps to cross-fertilize (Nonaka and Takeuchi, 1995) learning among teams and build organizational learning. So, this trade-off has to be balanced, so the organization can benefit from this practice.

* * *

This pattern contributes to the creation of `Diverse Groups` when assembling a team. Rotation also helps to `Moderate Truck Number` by fostering a culture of shared knowledge and by not becoming overly dependent on a small number of individuals (Coplien and Harrison, 2004).

`Collective Meetings` with leaders help to organize and prepare people to properly employ this practice. It is needed to consider proper people to the team with attitudes and skills required, the iteration start and duration, and type of delivery. It was raised by a member of Eagle Inc, *"What we have is a meeting between the project managers. They share information and make decision on teams' members allocation (rotation)."*

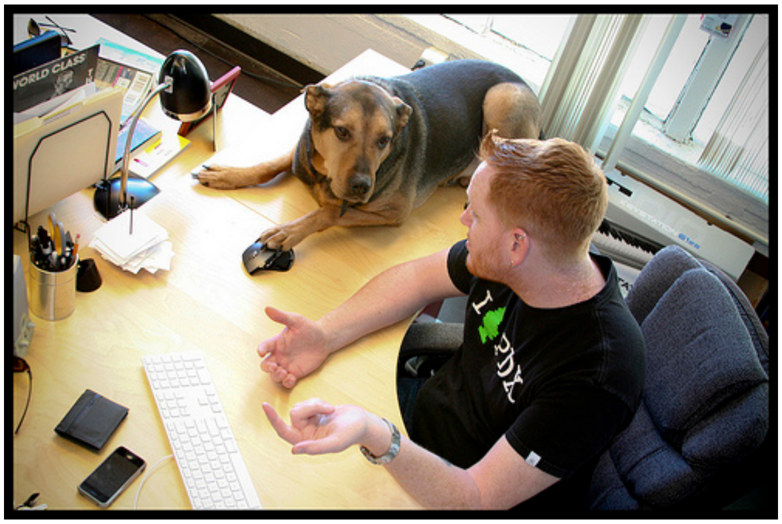
In larger organizations, Moe et al. (2009) recommend to assign people to one project at a time and let them focus on it. These authors outline that resources must be coordinated, and management, not the team members, must decide when other projects or support requests should get resources. If the team members decide which project to work on, they will yield to the one making the most noise. This will damage the self-managed team's potential.

`Technical Presentations` also help to prepare people on the required skills before conducting rotation. *"For instance, security was a subject that we felt it was important to arrange a workshop. We discussed at the roundtable and saw that it was nice to do a workshop for people, to disseminate more. It usually happens once a month at the amphitheatre and there is a person responsible for organizing the workshop, it's not a fixed person, it always changes."*

Associating `Rotation of Teams' Members` with `Pair Programming Among Different Teams` help on levelling the required knowledge to work on the project. Moe et al. (2009) suggest pair programming for the whole company to achieve organizational knowledge redundancy along with job rotation.

In agile teams, according to their level of integration among teams and projects, job rotation promotes multiple team membership and increase knowledge redundancy (Faegri et al., 2010). These authors add that job rotation enables overlapping product experience among developers and requires open and trustful discussion among all involved stakeholders, as it incurs in collective cost that must be amortized and legitimized by the organization.

3.3 `Pair Programming among different teams *`



At Creative Inc someone from one project might join another team for a couple of hours for any reason he judges necessary. They have created a PP matrix, in which they have awarded the member with the maximum number of developments in pairs. At Dyna Inc they employ PP across teams of the same customer, "Pair programming is a rule, you rarely see someone working alone here (...) At the end of the day we are exhausted, but we know that we worked intensively."

Agile software development teams, in which pair programming is an ordinary practice within the team, usually have similar or interrelated activities during their iterations, and also have to deal with software integration. These lead to create common technical solutions, such as frameworks or components, and to work jointly with other teams to deliver an overall integrated solution.

Even when employing other practices to share knowledge across teams, such as face-to-face conversations in Open Workspace and Rotation of Teams' Members, it is possible to not have enough detail of a solution, but only an overview of it.

Spreading and scrutinizing the details of a solution is sometimes difficult. Likewise, it is hard to achieve overall collective code ownership and internal software quality. As a consequence, teams might develop the same solution in different ways rather than reusing this from other teams.

* * *

How to enhance overall internal software quality and collective code ownership?

Software developers, specially the ones that work more isolated (even within the team), need to move out of their comfort zone, so that they can reflect on their own code. Internal code quality is related to software design, which involves (1) knowledge on the problem domain for assuring generality and concision, (2) writing code with good clarity, cohesion and minimum of interdependencies/interrelation between objects to ensure simplicity, (3) rigorous testing, and (4) improving the code with refactoring.

Scaling collective code ownership to other teams and to the whole organization is important because it forces a share of responsibility for internal code quality and allows more people to be able to make the necessary changes in code.

The more the teams and/or projects are interrelated (sub-teams)[3], the more they need to enhance internal code quality and collective code ownership for the organizational level.

Internal code quality and collective code ownership on organizational level is challenging because there is a lack of overall visibility and commitment to force joint work across teams, as agile teams often get into a maddening pace for delivering to their customer and forget to consider a wide picture of the domain problem that affect their long-term competitiveness.

Therefore:

Adopt pair programming among different teams involving people knowledgeable in specific issues to level knowledge throughout the company.

Establish pair programming as a practice to be freely adopted throughout the company in order to improve internal code quality and collective code ownership, along with pair rotation within the team. Developing in Pairs is a fundamental premise of this pattern, which is described by Coplien and Harrison (2004).

On the other side, however, there is a fundamental issue of organizational acceptance to pair programming. There is scepticism about PP that two people doing one person's work will deliver more than twice as much. This is quite discussed in literature (Hannay et al., 2009) (Aristholm et al., 2007) (Williams and Kessler, 2002). In a meta-analysis of pair programming conducted by Hannay et al (2009), the authors stress the need to consider moderating factors, such as programmer expertise, system complexity, duration, effort, and correctness, in order to perceive the effects of pair programmming. They suggest that pair programming is faster than solo programming when programming task complexity is low and yields code solutions of higher quality when task complexity is high. The higher quality for complex tasks comes at a price of considerably greater effort, while the reduced completion time for the simpler tasks comes at a price of noticeably lower quality.

As a consequence, this pattern helps the emergence of reflections on internal code quality and collective code ownership in the organizational level. More people get in touch with other parts of the code in order to identify bad software designs (such as code smells), bad testing, and needs for improving code with refactoring. On the other hand, this pattern might disturb team dynamics, because it involves human relationships, there is a need to select compatible pairs and deal with conflicts, if necessary.

Even nowadays, pair programming remains a challenging extreme programming practice (Beck, 2004). Companies characterized by coming from a traditional software development (e.g., waterfall software development methodology) generally need to employ deep changes, such as considering specific furniture, task complexity, and selection of 'good' pairs to perceive their benefits in the long-term (Dyba et al., 2007).

In our previous research, the investigated companies characterized by having a traditional background, they state that mandating the adoption of practices is often disapproved by the programmers and tends to resign. Then, what those companies did was a bottom-up strategy, making people experience pair programming through influencing its adoption in the company (e.g., reactively in mentoring sessions to help solving specific practical issues). Consequently, programmers would perceive its value and request the change on their attitude to pair programming, which implies the frequency of adoption, the right furniture, and establishing effective ways to PP within the company context (considering management support, task complexity, expertise, and sensitive aspects like the selection of 'right pairs').

This practice also relies on a strong culture of respect and collaboration to work well. People need to be willing to teach and keen to share knowledge and understand the other's difficulties to level knowledge. Likewise, personal affinity is another crucial factor that might affect the effectiveness of this practice (Williams and Kessler, 2002).

[3] e.g., work for the same product/customer; adopt the same technologies, frameworks or components; have common areas of overlapping and integration etc.

* * *

This pattern contributes to minimize the number of key experts with unique knowledge - `Moderate Truck Number` (Coplien and Harrison, 2004), but in an organizational level.

It is important to consider the overall relationships in the company - `Diverse Groups`. As other previous studies have stated (Williams and Kessler, 2002) (Dyba et al., 2007), there is also a need for improving pair selection. Coplien and Harrison (2004) suggest the assemble of compatible pairs across groups if the pairing is done across team and organizational boundaries.

Another important issue is that enhancing overall collective code ownership and internal software quality demands interaction among teams in order to cross boundaries. Without that teams will stay isolated and their design will reflect that. As a consequence, Architecture is a copy of Organization, which means `Conway's Law` (Coplien and Harrison, 2004). This pattern helps in aligning organizational intentions and actions to improving code and quality.

`Open workspaces` also facilitate movement of developers to help the adoption of this practice.

When the organizational settings are not favourable to overall pair programming, Scott (2011) suggests to use technology to overcome this problem, such as remote pairing involving screen sharing on similarly sized monitors and a good audio connection, which is described by `Real Distributed Pairing`.

3.4 `Collective meetings **`

At Maps SA they promote company workshops as a way to enhance company alignment and integration. At Eagle Inc they adopt roundtable once a month, "We set topics and vote on the topic for the next meeting. There is always a person arranging it, which is not fixed as well, a project manager leads it [It is important to state that in this company there is no Scrum master role, but technical project leaders and project managers (as well as developers), which are enthusiastic people in aspects of the company culture and in agile methods. They are aware of the importance of knowledge sharing practices and strive to adopt them sustainably]. Other project managers select team representatives to attend the roundtables. Then, the project manager responsible for the roundtable sends an e-mail to the project managers asking who will attend the meeting, sets the date by consensus and facilitates the meeting as a moderator by letting everyone speak for their projects. At the end, a meeting report is generated as a record of what was discussed and the actions to be taken within the projects."

Agile methods provide several types of meetings, such as daily (or stand-up) and planning meetings, that foster much intra-team communication and awareness about the project. However, agile teams in organizations often have common objectives, interrelated activities, and also have to deal with software integration during their iterations that concern the company's projects.

Organizational alignment is an important aspect that is usually not taken into account when adopting agile methods. Few meetings across teams are reported in agile contexts specifically to provide organizational alignment.

* * *

How to involve members of different teams in discussions of specific themes that concern the company's projects in general?

As software development teams keep more focused in achieving their own goals, consequently they become less aware of what is going on in other teams or projects. However, teams are part of something bigger than their own teams, like departments and the whole company.

Software development companies in general face the need to reach cross-team or organizational strategies, so other practices need to be adopted to reach specially organizational alignment and other strategies. Face-to-face conversations and rotation among teams help in knowledge sharing, however, may not have a specific common objective.

Also inter-related projects need to establish an effective communication channel to manage and share knowledge across teams. For instance, there may be user stories that will be developed by one team that might interest or affect other teams, because of interdependencies or areas of overlap.

Therefore:

Adopt collective meetings focused on specific themes to improve knowledge sharing throughout the company and to provide wider organizational alignment.

Analyse the inter-dependencies or areas of overlap with other teams. Also, assess the way face-to-face conversations are undertaken in the entire company and also consider the company size. If it is a large or medium-sized company and there are overall communication (not localized communication), try to employ the approaches described below with teams' representatives. However, if it is a small company with overall communication, try to employ the approaches below with everyone.

In the studied companies, specific agile roles like Scrum master and Product owner were not strictly followed. For instance, the Scrum master role was removed in most companies, as they were sharing the knowledge and responsibility of this role together (with all team members) and trying to become a self-organized team. They adapted their existing roles, such as technical leader role as an agile coach focusing on following the company values and principles, which were aligned with the agile ones.

Conduct **agile meetings**, such as daily meetings, retrospectives, reviews, planning meetings, etc. with participation of all members or only representatives of sub-teams of the same product or customer. In Figure 6, it is illustrated a 10 minute demo presentation of two sub-teams of a product, where a representative of each sub-team (not a project manager) presents their weekly achievements towards the iteration delivery. The role of the representative can be volunteer or can change every week, so that everyone in the team, except for the project manager, would have the chance to present the demo.

Fig. 6. Sub-teams 10 minutes demo presentation at Maps SA.

Depending on the number of members and of sub-teams, only representatives would present about their respective sub-team. These meetings serve to get awareness of teams on the several aspects (like user stories and issues) that they have been or are being or will be through, and to share relevant knowledge of each sub-team to others. These meetings are also useful to bring issues about interdependencies and to foster alignment among sub-teams.

Try practices described further, such as **brainstorming or brainwriting**, **Open Space Technology**, and **Birds of a Feather** conducted by enthusiastic people on the topics chosen or by people acknowledged by the company members.

Also involve employees in biannual or annual **organizational meetings** for discussion of further strategies (the picture of this pattern vignette is a company workshop at Maps SA). At these meetings the short-term, medium-term and long-term strategies would be communicated to all to get commitment and understanding of the organization's goals.

Adopt technical leaders' meetings (e.g., Scrum of Scrums (Schwaber and Beedle, 2002)) to map common problems, interests or needs among teams and decide on actions across teams. First of all, establish the objectives and agenda of the meeting discussion. During the meeting follow steps: (1) present the actions resolved since the last meeting (e.g., if the solution suggested in the last meeting sorted out the problem and how was the experience for the team?), (2) share relevant knowledge apprehended by the team since the last meeting that would also interest other teams, (3) each leader presents: what my team has done since the last meeting? What my team plan to do until the next meeting? What are the impediments?, (4) perform action planning, and (5) close the discussion and present a summary of actions with people responsible.

To enhance effectiveness on collective meetings, the company members need to consider the ability to interpret and change the environment for continuous improvement. The higher the adaptive capacity of the company, more effective is this practice.

A clear organizational strategy, integration between projects and teams, and commitment to join all people helps to enhance the effectiveness of collective meetings to the organizational level as people conduct these meetings and make decisions more properly. These meetings require a moderator to organize the sessions, assign actions to people and monitor the actions until they are resolved.

Also there are other important factors to consider, such as interaction among people, organizational culture, individual behaviour, and self-organization. All the above factors may act as barriers or facilitators to the effective knowledge sharing.

As a consequence, this practice might cause clash of opinions and affect people's morale and time to work on their teams' goals. That is why it is important to clearly establish principles, such as respect, trust, and willingness to learn. The meetings should be focused and well-organized, to have well-defined goals to be clarified in the beginning of the collective meetings, and consider conflict-resolution techniques, if necessary. Time for agile teams is usually scarce, so it is important to make better use of collective meetings by adding value to the participants and being acknowledged by the company.

In the agile context, Scrum of Scrums is an initiative for collective meeting to coordinate scrum sub-teams. These meetings usually involve senior people or common technical areas (Schwaber and Beedle, 2002) to solve critical problems or to discuss projects integration and overlapping areas (Maranzato et al., 2011) (Srinivasan and Lundqvist, 2009) (Lindvall et al, 2002). However, to work well, project members, especially, Product Owners, and Scrum Masters need to have a broad view of the interrelated projects.

In the knowledge management and organizational learning fields, practices such as brainstorming (Dierkes et al., 2003), open space technology (Dierkes et al., 2003) and communities of practice (CoP - a mechanism for knowledge sharing in informal communities within the organization with common goals or interests) (Wenger et al., 2002) (Brown and Duguid, 1991) are reported as collective meetings to support learning across organizational boundaries.

* * *

Collective meetings to work well require organizational Unity of purpose and an effort to create a Community of Trust (Coplien and Harrison, 2004). This pattern increases the perspective of Reflection workshop (Cockburn, 2004) as it promotes collective reflection across agile teams. Book study groups (Kelly, 2008) are also a very good practice to cause collective reflection and understanding.

In agile context, we examined the use of open space technology for sharing knowledge across teams of companies (Santos et al., 2013a). Open space technology (OST) is used to identify solutions to problems in a project, such as specific architectural and technical solutions. It also serves to show results for clients, develop new ideas across teams and provide new solutions for the indicators to improve the company. As this meeting focuses on results, it also requires a facilitator who ensures that all steps (1-5) are met and that allows everyone to give their opinions, as can happen in some people dominate the meeting. OST consists of five steps: (1) planning/invitation, (2) setting the session agenda, (3) developing focal group sessions, (4) closing the session and action planning, and (5) implementing and monitoring. This meeting is also adopted by several agile conferences[4] to spread knowledge, improve discussions and gather feedback from attendees.

Brainwriting is also suggested to improve the development of ideas (Kua, 2013). This practice consists of writing teams' issues on story cards, which are shared among different teams. Each team provides solutions to the issues, contributing to the choice of final solutions. This practice was adopted by agile teams in an agile academic environment to explore its benefits across teams (Santos et al., 2012). After conducting this practice, most teams' members stated their projects were positively affected by the solutions proposed or raised during the Brainwriting. This practice can be used to increase knowledge about the possibilities to solve similar problems with perspectives from other people (who may have a new look to the problem), to assist in the projects' inception, search for solutions to problems for different projects, and help define the project scope. This meeting requires a facilitator who engages in organizing the session and allows everyone to participate.

Some authors have reported the use of CoP in agile environments (Mestad et al., 2007) (Kähkönen, 2004) and stated that, considering the dynamic and people-centered approach of agile methods, the standard form of CoP has to be adapted to be suitable for agile teams, such as circles of dynamic skills with voluntary participation and freedom to participate in several communities.

De Diana et al. (2010) describe meetings involving architects from different teams to organize the architecture group's work and keep track of work being done at the moment. Another known use of CoP are the Birds of a Feather (BoF) sessions promoted in several computing conferences, including agile conferences[5] and also by

[4] http://xp2013.org/program/,http://xp2012.org/program/features/open-space

http://brasil2013.agilealliance.org/about-open-space/

[5] http://www.agilecambridge.net/ac2012/programme.php

http://www.alm-summit.com/bof/Session

agile consultants (Santos et al., 2013a). BOF also requires a facilitator to organize the session, because the steps are (1) defining the theme and invitation, (2) defining the topics to be discussed within the theme, (3) discussion of topics in certain time intervals and action planning (if any), (4) closing the discussion and summary of actions with people responsible (if any).

3.5 Technical Presentations **

At Maps SA lightning talks are employed in their open workspace to foster intra and inter-team discussions and reflections on improvements. At Creative Inc, because of their business domain related to teaching and innovation, brown bag seminars are employed every week as a way to increase learning in the workspace.

Agile teams or team members often have interesting subjects to share with other teams or with members of the entire company, but few initiatives to create cross-team or organizational knowledge are promoted in this context.

Not only agile companies, but software development companies in general have difficulty in creating a continuous learning and knowledge sharing behaviour among their collaborators. Training programs and courses are often scarce and little encourages in software development companies.

* * *

How to promote a continuous learning and knowledge sharing behaviour in spontaneously?

Software development companies face the need to reach collective knowledge, then a continuous learning and knowledge sharing behaviour must be encouraged in order to become part of teams routine and create the necessary discipline for learning and sharing continuously.

All kinds of incentives need to be considered to accomplish this goal. Organizational commitment and strategies must be defined to improve expertise and projects' solutions. Acknowledged practices must be adopted repeatedly, since learning and sharing knowledge in the company improve collective knowledge. Because of these organizational factors, incentives are somewhat hard to introduce and maintain.

Other important concerns encompass building a culture of openness to discuss, balancing cost, time and scope between project work and knowledge sharing activities. Individual factors, such as stimulating people to leave their comfort zone and to improve their presentation skills, also need to be considered.

Therefore:

Encourage developers to study and to perform periodic technical presentations in the company.

Promote **internal seminars** to disseminate knowledge on specific subjects, usually chosen to be of common interest or need, such as software development with TDD, best practices, teams' successful stories, mature solutions, innovations for the entire company or department, or knowledge apprehended at conferences attended by teams' members. These seminars stimulate teams' members to study topics of interest and generate collective knowledge during work time and at the workplace.

This practice should be adopted to the entire company. If it is not possible to the whole company, try to adopt it among sub-teams of the same department, product or customer.

Make presentations on **complex technical solutions** for more experienced people from other teams in order to evaluate it and/or improve it.

Adopt short presentations, known as **Lightning Talks**, of no more than 7 minutes on a topic to share with others (e.g., present a solution, a technology, a technique, show different designs for others etc.). At the end of the presentation, it is important to have a small range for questions or discussions. There must have a facilitator to look after the session to be very focused and bring the most value to people.

Promote **technical lunches** (also known as Lunch and Learn), which consist of presentations at lunchtime, where the company provides lunch / snack. It is advisable to conduct a retrospective at the end of the presentation to uncover collective learning, seek improvements for further sessions and decide the next topic to be presented.

In Lightning talks and technical lunches, try to involve presenters that are enthusiastic about the chosen topic (her(is) project, technology, a retrospective report, company strategy or marketing).

Perform **formal training** or hire outside consultants to leverage knowledge on specific issues within the company.

This requires a level of questioning and reconsideration of the company work processes. Company need to analyse their processes, procedures and routines to make continuous improvements that promote the adaptive capacity of the company.

This practice relies on organizational commitment and consistent orientation towards knowledge, i.e., organizational strategy must be clear for the effective adoption of this practice. Inconsistent strategies may negatively impact the effectiveness of the practice.

Likewise, a culture of openness to discuss, improve, and learn from past mistakes is needed. Also it is important to establish a discipline of continuous search for learning and improvement in the company. As a consequence, the spread of this learning and sharing behaviour stimulates a feeling of personal relevance to the company and career development.

On the other hand, time devoted to this practice might affect intra-team iteration goals. That is why it should be aligned to the company goals, values and principles, and be properly planned with the participants. As technical presentations expose people, it is important to consider that it might affect their morale. Because of this, it is important to consider conflict-resolution techniques, such as compromising (acceptable solution which partially satisfies both parties), if necessary.

Several authors from the knowledge management field has recommended presentations on diverse topics by members of the company to enhance the opportunity of others becoming aware of related knowledge and experiences (Dierkes, et a., 2003), as well as reinforcing the behaviour in seeking continuous learning in the workplace that involves learning from practical experiences and re-examining assumptions, values, methods, policies, and practices (Easterby-smith and Lyles, 2011).

* * *

`Technical Presentations` contribute to preparing people to employ `Rotation of Teams' Members` and to enhancing `Domain Expertise in Roles` (Coplien and Harrison, 2004) related to core competencies in the company. This pattern also helps an `Apprenticeship` program in order to turn new hires into experts (Coplien and Harrison, 2004).

Maranzato et al. (2011) have explained that once or twice a month, usually on Fridays, they have some presentations about training or conferences that people attend, relevant features or best practices that they consider important for everyone. They also share experiences from some teams to the others.

At universities and private companies, brown bag seminars are employed to share knowledge to the attendees in a voluntary and informal setting during lunch break (http://en.wikipedia.org/wiki/Brown_bag_seminar). At Acme Inc and Eagle Inc, this is a practice that was abandoned after some editions. As a member at Acme stated, *"Sometimes, it is really cool to bring up new things, but sometimes we are so overloaded with work, then you have to do a presentation for two weeks on a new technology. So, it ends up just not working".*

We found out the use of Lightning Talks for sharing diverse knowledge across teams of agile companies (Santos et al., 2013a) and also in agile conferences[6].

4. FINAL REMARKS

This paper provided practical guidance to organizations striving at sharing knowledge across teams to foster organizational knowledge. We presented a pattern language composed by eight patterns observed in eight Brazilian agile software organizations.

These patterns and their relationships were described in Figure 2. In this paper, due to space constraints, we described five of those patterns, namely: Open Workspace, Rotation of Teams' Members, Pair Programming among Different Teams, Collective Meetings, and Technical Presentations. We expect that the details and variations provided by the patterns help agile software organizations to successfully achieve inter-team knowledge sharing.

It is important to point out that the adoption of the pattern language needs special attention on the organizational enablers, such as organizational culture, environment, and top management and leadership support, which facilitate and reinforce their effects.

REFERENCES

Alexander, C., Ishikawa, S., Silverstein, M. 1977. A Pattern Language: Towns, Buildings, Construction, Oxford University Press.

Aniche, M. and Silveira. G. 2011. Increasing learning in an agile environment: Lessons learned in an agile team. In Agile'11: The Agile Conference, Salt Lake City, UT. pp. 289–295.

Beck, K. 2004. Extreme Programming Explained: Embrace Change. Pearson. ISBN: 978-0321278654.

Berger, P. L. and Luckmann, T. 1966. The social construction of reality: a treatise in the sociology of knowledge. Doubleday & Co Inc., 1. ed..

Bricout, V., Heliot, D., Cretoiu, A., Yang, Y., Simien, T., Hvatum, L. B. 2004. Patterns for Managing Distributed Product Development Teams. In the proceedings of the EuroPLoP.

Cockburn, A. 2004. Crystal Clear, A Human-Powered Methodology for Small Teams. Addison-Wesley Professional, ISBN 0-201-69947-8, pages 336, October.

Coplien, J. O. and Harrison, N. B. 2004. Organizational Patterns of Agile Software Development. ISBN 978-0-13-146740-8, July.

Corbin, J. and Strauss, A. C. 2007. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, 3rd edition.

Crossan, M., Lane, H. W. and White, R. W. 1999. An organizational learning framework: from intuition to institution. Academy of Management Review, vol. 24, no. 3. 522-537.

DeDiana, M., Kon, F., and Gerosa, M. A. 2010. Conducting an Architecture Group in a Multi-team Agile Environment. Brazilian workshop on agile methods (WBMA), pp. 137-149.

Dybå, T., Arisholm, E., Sjoberg, D. I. K., Hannay, J. E., and Shull, F. 2007. Are Two Heads Better than One? On the Effectiveness of Pair Programming. IEEE Software, vol. 24, issue 6, pp. 12 - 15.

Easterby-Smith, M. and Lyles, M. A. 2011. Handbook of organizational learning and knowledge management. Wiley, 2nd edition.

Hannay, J. , Dybå, T., Arisholm, E., Sjberg, D. 2009. The effectiveness of pair programming: A meta-analysis. Information and Software Technology 51(7):1110–1122.

Karlsen, J. T., Hagman, L. and Pedersen, T. 2011. Intra-project transfer of knowledge in information systems development firms. Journal of Systems and Information Technology 13(1):66–80.

Kelly, A. 2008. Changing Software Development: Learning to Become Agile. Wiley, 1 ed., February.

Von Krogh, G., Ichijo, K. and Nonaka, I. 2000. Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation. Oxford University Press, USA.

Kua, P. 2013. The retrospective handbook. E-book available at: https://leanpub.com/the-retrospective-handbook.

Maranzato, R. P., Neubert, M. and Herculano, P. 2011. Moving back to scrum and scaling to scrum of scrums in less than one year. In the Proceedings of the SPLASH '11: The ACM international conference companion on Object oriented programming systems languages and applications companion. ACM, New York, NY, USA, pp. 125-130.

Moe, N. B., Dingsøyr, T. and Røyrvik, E. A. 2009. Putting Agile Teamwork to the Test – An Preliminary Instrument for Empirically Assessing and Improving Agile Software Development. P. Abrahamsson, M. Marchesi, e F. Maurer (Eds.). In: XP 2009, LNBIP 31, páginas 114–123. Springer-Verlag Berlin Heidelberg.

Moe, N. B., Dingsøyr, T. and Dybå, T. 2009. Overcoming barriers to self-management in software teams. IEEE Software, 26 (6) 20–26.

Nonaka, I. and Takeuchi, H. The knowledge-creating company: How Japanese companies create the dynamics of innovation. New York, NY: Oxford University Press, 1995.

Pierce, J. 2011. Great Agile Workspaces: The Physical Environment. Available at: http://lunatractor.com/2011/05/11/great-agile-workspaces-the-physical-environment/

[6] http://www.xp2013.org/

http://www.xp2011.org/

http://agile2013.agilealliance.org/

Rising, L. 1998. The Patterns Handbook: Techniques, Strategies, and Applications. SIGS Reference Library.
Santos, V., Goldman., A. and Souza, C. R. B. 2012. Fostering Effective Inter-team Knowledge Sharing in Agile Software Development. Paper submitted to the Empirical Software Engineering (Awaiting second assessment).
Santos, V., Goldman, A. and Roriz, H. 2013a. The influence of practices adopted by agile coaching and training to foster interaction and knowledge sharing in organizational practices. In HICSS'13: The 46th Hawaiian International Conference on Systems Sciences, Agile/Lean Startup Organizations, pp. 4852-4861.
Santos, V., Sharp, H., Goldman, A. and Souza, C. R. B. 2013b. Tacit knowledge socialization in agile software development: an enhanced inter-team knowledge sharing conceptual model. Paper submitted to the Information and Software Technology journal, special edition on human and social aspects (Awaiting first assessment).
Schwaber, K. and Beedle, M. 2002. Agile Software Development with SCRUM. Prentice-Hall.
Scott, R. 2011. Patterns of Pair Programming. Available at: http://www.rallydev.com/community/engineering/patterns-pair-programming
Williams, L., and Kessler, R. 2002. Pair Programming Illuminated. Addison-Wesley Professional; 1st edition.
Yin, R. 2000. Case Study Research: Design and Methods. SAGE Publications, Thousand Oaks, CA.

Received June, 2nd 2013; revised August, 4th 2013; accepted August, 13th; final version March 30th, 2014.