

A Study Case: Evolution of Co-location and Planning Strategy

Amy Law and Allen Ho

TransCanada

450 – 1st Street SW

Calgary, Alberta

T2P 4K5, Canada

amy_law@transcanada.com and allen_ho@transcanada.com

Abstract

Agile practices can and should be evolved throughout a project. This paper focuses on the evolution of two agile practices, namely co-location and planning strategy, in a software development project at TransCanada. From inception to conclusion, the evolution contributes to the successful delivery of an in-house developed system and leads to change in organizational culture.

1. Introduction

At TransCanada, our network of about 39,000 kilometers (24,200 miles) of pipeline transports most of Western Canada's natural gas production to the fastest growing markets in Canada and United States. We also own, control or are constructing more than 4,700 megawatts of power – enough to meet the needs of about 4.7 million average households. To support these industries, the IS department has delivered many successful in-house software solutions for specific business needs.

1.1. Land System

One of the software solutions developed in TransCanada is called the Land System. The goal of the Land System was to replace a legacy application with a Java web-enabled enterprise application. The Land System is responsible for managing land titles, landowners, and payment processes.

As continuous improvement, the Land project manager looked for change in development strategy. She funded the team to take a software development course on "Scrum and Agile". With the influence of the course, the team decided to apply a hybrid approach of Scrum and XP in the Land System project.

In this report, we will focus on the evolution of two agile practices, namely co-location and planning strategy.

2. Co-location

The idea of co-location is to physically move the whole team into a common area. It facilitates team interaction and communication. When TransCanada Tower was built in December 2000, each developer was assigned a cubicle in the same location. The cubicle was smaller than six square feet with a partition over four feet tall. Six of these cubicles were grouped into a "Honey Comb" cubicle as shown in Figure 1.



Figure 1: "Honey Comb" Cubicle

2.1. Honey Comb Cubicles

In October 2001, the Land System project team was formed and consisted of fifteen people including customers, project manager, system analysts, and developers. The seating plan was arranged so that the

whole team was co-located in Honey Comb cubicles. The close proximity encouraged team interaction.

In addition, the customers were seated just twenty feet from the team on the same floor. They were responsible for deciding and communicating user requirements. They were also the product users. The customers, being on the same floor, encouraged their continuous involvement with the team.

Despite all of these advantages from the Honey Comb cubicles, the team had to improve the seating strategy to accommodate their programming practices.

The team applied pair programming in which two people worked together using the same computer. The positive effect of pairing was improved software quality, source code consistency, and cross-training opportunity. However, our cubicles were too small. The furniture inside the cubicle took over the limited space, and it was hard to see the monitor, awkward to fit two chairs side-by-side, and impossible to move around inside the cubicle, see Figure 2.

In addition to pair programming, the team applied continuous integration to support the source code integration, build, and test cycle. Source code changes were frequently integrated to the repository to ensure the latest changes would be available to the entire development team. With the rapid growth of the source code, the duration of each build increased to forty minutes. Hence, we had to optimize continuous integration so that more than one pair could check-in the changes at the same time to better facilitate the efficiency of our build. The pair had to walk around the cubicles to notify everyone about changes that were checked in. The physical movement from one cubicle to another interrupted the flow of work.



Figure 2: Interaction in Honey Comb Cubicles

Team communication is one of the key criteria in a successful software development team. It is important

to involve the whole team to discuss the planning, design, development, and testing strategies. However, the Honey Comb cubicle layout became a physical constraint. In any discussion, our developers had to stand up or walk around the cubicles. This discouraged effective team communication.

2.2. War Room

To address these problems, the team considered alternatives. There were two conference rooms immediately next to the Honey Comb cubicles. They had a white board, network connections, and round table as shown in Figure 3.



Figure 3: War Room

The team decided to take ownership of these rooms in December 2001. Our customers supported this idea because everyone wanted the project to succeed and all agreed that anything that impeded progress along the way would be eliminated.

One room was kept for meetings and the other was labeled the war room and kept for pair programming. The developers were using laptops; therefore, they were not tied to a specific location. The mobility of the laptops helped the team to be self-sufficient and to move into the war room without going through a formal and expensive move request process.

The war room provided sufficient office space for pair programming and allowed for easy communication during continuous integration and discussion in general. The communication inside the war room was convenient, and it was especially effective during the design phase because not everyone had the same knowledge of the business domain and technical architecture. The team could freely discuss the design utilizing the white board. The team also used the war room for the daily standup meeting.

At first, only a single pair moved into the war room. The team soon started to see the benefits of working together in the war room, and more developers gradually moved to the war room. But, as popularity increased, it became obvious that space was an issue and we would need to move away from the war room.

The war room was a perfect solution for team interaction and communication, but it was too small to accommodate the entire team. It reached the maximum limit of eight developers. The overcrowded room reintroduced the old problem in which the pairs were not able to work comfortably as shown in Figure 4.



Figure 4: Overcrowded War Room

Another issue was the team members who were not working in the war room were isolated from their team resulting in a communication gap.

The hot temperature in the war room also became a problem. Although there was air conditioning, the temperature could become very high when the room had five laptops operating side by side.

The war room was located in the center of the 21st floor and therefore had no windows or natural sunlight. We felt a bit claustrophobic. Although the developers fell in love with the war room initially, they did not wish to settle down there for the life of the project. After a year of working in the war room, the developers wanted to change the environment.

A logical alternative would be to find a larger war room. However, those larger rooms were in high demand. We could not always keep them, and they were typically located further away from the customers.

Our TransCanada Health and Safety Department ruled out continuing to work in war room because of ergonomic concern. When a new developer joined our team, she had an ergonomic assessment in the war room. The result indicated that improvement was

required from a health and safety perspective. The round table was designed for meeting, and not for pair programming. It was too tall and not adjustable. The chair's armrests had to be adjusted to accommodate the table defeating the purpose of the arm support. The ergonomic concern was the driving factor to move the team out of the war room.

2.3. Open Environment

In May 2003, the team worked with building services to come up with a new seating plan to address the ergonomic concerns and accommodate the team practices. Building services gathered our requirements and understood our agile practices. They came up with several seating alternatives ranging from the changes on the partition to the relocation of the entire cubicle. In the end, a new seating plan was introduced. The partitions between cubicles were lowered to allow the team to talk without the physical barriers. Some partitions were removed to open up the office layout and bring natural sunlight into the common area. The walkway between cubicles was minimized to reduce any physical movement from one cubicle to another. The file cabinets and side desks were moved elsewhere.

The new seating plan allowed us to pair comfortably and facilitated effective interaction. This was perfect for anyone on the team to ask and answer quick questions and no one needed to get up and walk around. This prevented any interruption to the flow of work and supported easy spontaneous discussion among team members, see Figure 5.



Figure 5: Pairing in the Open Environment

The success of the open environment drove change in our organizational culture. Specifically, it gained attention from different departments in TransCanada.

Several other teams started to look at our open environment and copied the office layout to fit their practices.

As the project was approaching the final phase in April 2004, the project manager invited our key customer to move to our open environment. She performed QA testing, answered questions on system requirements, and prioritized feature sets. Her presence had instant benefits to the project. For example, our team discovered an issue in the production environment. Since the customer was just a step away, we conveniently invited her for the discussion. We then realized that this was a problem in data integrity limited to one record in the database. The same day, another customer was ready to spend time working on the problematic record. Our key customer notified the others, in effect, saving their time and avoiding unnecessary investigation and side tracking. This ultimately increased customer satisfaction.

3. Planning Strategy

While our co-location approach was changing, our strategies for iterative development planning were also changing.

3.1. Monthly Planning

The first planning session took place on the last day of the “Scrum and Agile” course in October 2001. The course instructor helped the team to start Sprint #1, where a sprint is a fixed period of time in which a development team worked toward a common goal. Since the team was new to this lightweight process and did not know the appropriate length of a sprint, the course instructor suggested each sprint to be one month long. In addition, the team chose to use the standard sprint backlog template provided in the course.

Prior to the planning session, the team did not know what they would work on during the sprint. All decisions were made during the planning session. Therefore, each planning session typically took the whole afternoon.

During the sprint planning, we worked with our customers to determine available resources, identify scope, and prioritize tasks for the sprint.

In the sprint backlog, we recorded the goals of the sprint, tasks, originators, developers or analysts to work on the tasks, time estimates, and hours of work remaining until completion.

Throughout the sprint, the system analysts would work with customers to write decision documents and white papers. The team chose to have analysts, rather than developers, gather requirements because our analysts had strong backgrounds in the business acumen and technical expertise to guide the customers towards the most effective solutions. The analysts tended to stay with the project team from the beginning. They could describe the evolution of requirements in addition to documentation.

The decision documents contained the problem statement, current situation, desired outcomes, potential solutions, and explanation of why the solution was the most appropriate to the situation. Senior management was the target audience of decision documents.

The white papers contained the problem statement, current situation, desired outcomes, proposed functionality, assumptions, database changes, and detailed user requirements. Developers were the target audience of white papers.

As developers typically moved from one project team to another, the decision documents and white papers became value-added to the project.

The developers would gather requirements by reading the white papers and having discussions with the system analysts. They implemented the functionality based on these requirements. When the requirements were complex, the developers consulted the customers for further clarification. On a daily basis, the team recorded the “hours of work remaining until completion” against each task in the backlog.

At the end of the 30-days, we had a review session in which the senior management, customers, and the project team met and reviewed the sprint results. The review session usually occupied the whole morning.

During the review session, we discussed the project progress, budget forecast, requirement changes, estimate deviation, and risks. By reviewing what happened, we learned to better estimate.

In some cases, we described the evolution and growth of the data model, business scenarios, and technical architecture so that the customers were aware of the development progress in the parts of the software that weren’t visible from the user interface.

To increase visibility, we demonstrated the latest working software. For that reason, we always focused on what could be demonstrated. We did not wish to over-commit. We enjoyed the process because customers truly needed the functionality to perform their jobs effectively.

The first release of the Land System was successfully delivered into production in February 2003. It was followed by a retrospective session. The

purpose of the session was to identify the “good, bad, and the ugly” team practices and to drive improvements in the subsequent phases.

During the retrospective session, two main issues were identified. We found that the customers were not able to provide timely feedback on the working software until thirty days after the planning session. This did not help us to respond quickly to business changes, which could take place anytime during the sprint.

Another lesson learned was that our estimates needed improvement. We underestimated the time effort for completing our feature sets. In some cases, the actual time ballooned to as much as triple the original estimates. Some underestimates were caused by business changes, whereas others were caused by underestimation of the complexity of the tasks. Although, the team recorded the “hours of work remaining until completion” on the daily basis, it did not track the actual effort to complete the task. This defeated the use of the backlog as a benchmark to forecast future tasks. It was a challenge to forecast the time effort when the requirements changed over the duration of the sprint.

3.2. Bi-weekly Planning

One way to address the moving target was to change the planning interval from monthly to bi-weekly. The strength of this evolution was that the feature sets were demonstrated in shorter intervals enabling timely customer feedback and incremental improvement. As a result, the customers’ feedback was made in time to have an impact on the subsequent development. The shorter interval also helped us to estimate because we tended to break the feature set into finer tasks in order to fit all of them into a two-week interval. The team found that it was easier to forecast smaller tasks.

In March 2003, the team decided to tailor the planning strategy to once every two weeks, and the session took only two hours, rather than the whole afternoon.

The product review session still took place once every thirty days to accommodate the busy schedule of senior management.

The sprint backlog template was changed. In addition to what we had, we introduced the concept of tracking the actual time. On a daily basis, the developers recorded the actual effort spent on the task and the remaining effort to complete the task. This enabled the project manager to monitor project progress with respect to the actual effort to complete a

feature set and effort remaining to complete the sprint. This also enabled the team to develop a benchmark.

The improvement was seen immediately as we received customer feedback more often leading to less rework. Another software development team heard the benefits and requested a sample of our backlog. This led to change in organizational culture.

Several releases were made to production during the year of 2003. A major feature delivery was made to production in January 2004 followed by another retrospective session.

In this retrospective session, the team found that the problem of poor estimation persisted because we started to work on new business requirements and the analysts were not as experienced in this new domain. They had to gather system requirements from an offsite customer who typically came to the office only three times a week. Therefore, the analysis took longer and could span across multiple sprints. In some cases, developers would run out of work because the analysis on system requirements was not completed to the point where developers could start implementation. Without knowing the system requirements, it was a challenge for the team to divide the feature set into tasks efficiently and to provide estimates in the planning session.

3.3. Planning Cycle

To improve estimates, the team decided to modify the planning cycle in February 2004. Rather than marrying the analysis and development of feature sets into a sprint, we separated them into two sprints. We had pre-planning sessions prior to planning session, see Figure 6.

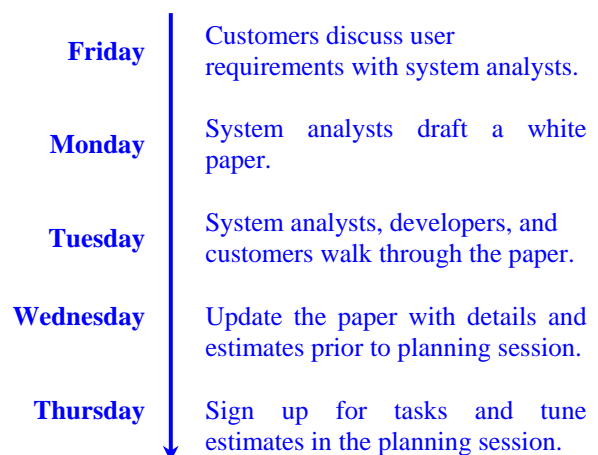


Figure 6: Pre-Planning Cycle

The customers first went over the user requirements with the system analysts. The system analysts drafted a white paper. The entire team then walked through the white paper in detail. The customers would also be present to address any concerns. As a result of this discussion, the white paper could be enhanced with more technical details. The development team provided estimates during the walk through. The pre-planning session provided enough time for the system analysts and developers to understand the requirements and hence led to better estimates.

In the planning session, the developers were aware of the system requirements and time estimates. Therefore, they simply signed up tasks and fine tuned estimates based on the new findings and comfort level. With the help of the pre-planning session, we were able to complete the planning session in about an hour, rather than two hours.

At the same time, our project manager was changed. The new project manager continued all of the agile practices with a couple minor modifications in the sprint backlog template.

If new tasks were identified midway through the sprint, they were added with a color identifier (the “blue” color) on the sprint backlog. Thus, we could visually see how much additional work was added after the planning session. Moreover, these new tasks were typically left unassigned until the high priority tasks were completed to ensure the sprint feature sets were delivered as planned. Once the team completed the original assigned tasks, they would sign up for the new tasks to accommodate the changes in the sprint.

In addition to tracking the development and analysis time, we started to track the time against our daily meetings, planning sessions, and review meetings on the sprint backlog. Therefore, we were able to monitor the time used for different types of meetings.

Although tightening of control sounds like micromanaging developer and analyst time, the evolution helped the team to improve time tracking, project management, and resource management. The total meeting time was significant enough that the project manager decided only a few members would attend the review meeting. The development team would not miss information because the project manager reviewed the meeting highlights with the team. As a result, the development team gained focus leading to good working product.

4. Reflection

Evolution began with the appearance of new practices that incorporated behavior changes enabling

unprecedented benefits. While trying these new practices, we responded with changes to increase their effectiveness. The time line in Figure 7 shows the changes over time.



Figure 7: Evolution Time Line

4.1. Co-location

We learned several lessons during the evolution of co-location. In order to gain the benefits of co-location, organizational support plays a significant role. In our case, our customers supported the idea of the war room. TransCanada building services worked with us to come up with the concept of the open environment. Our project manager encouraged the idea of co-located customer.

We believe effectiveness of co-location in terms of team interaction and communication lead to the success of our system delivery. The open environment was complementary to our agile practices, and other departments decided to copy the office layout resulting in change to organizational culture.

Co-location enables communication, but it does not cause communication. The essence of communication relies on the team itself and the mutual trust between one another in the team. In our case, the team consisted of professionals from TransCanada, ThoughtWorks, ClearStream, and RIS independent consultants. The team clearly benefited from this type of dynamic. For example, ThoughtWorks brought in the practical strategies for applying agile practices.

ClearStream introduced the automated user acceptance testing. The RIS independent consultants provided the business domain knowledge. The willingness to share is the reason for good communication. Co-location just helped create the environment where easy communication could thrive.

The open environment worked well for us, but it might not be favorable to everyone in their project. Some people might prefer privacy and dislike the openness. Some might even find the openness distracting which could impact their performance. Changes in office layout required financial support from senior management. Each formal office move typically costs TransCanada over five hundred dollars per person. As a result, it would be up to the team to decide whether the open environment is beneficial for the project.

4.2. Planning Strategy

We learned several lessons during the evolution of our planning strategy. We see that changes in our planning strategy are the result of incremental improvements. In each retrospective review, our team tailored the planning process to improve estimates and speed up sprint planning. We strongly recommend holding a retrospective session on a regular basis. In addition to our formal retrospective sessions, we have mini-lesson learned sessions throughout the year. This gives the team opportunity to tune the planning process to accommodate changes in the project environment.

Our backlog template showed the actual and remaining time to complete. Tracking the actual effort created a benchmark. By referring to a benchmark, our time estimates improved progressively. For instance, if our estimate was 1 day, but it actually took 1 ½ days to complete the task, we would add 50% to our future estimates when similar requirements surface in the future. As a result, they became a knowledge base of the organization providing benefits to other IT projects.

The remaining time to complete gave project manager a sense of project progress, so she could better manage resources and customer expectation to ensure the project was successfully delivered on time, within budget, and meeting quality specifications.

We also recommend that each project team should have a coach. The coach guides and reminds the team to keep the sprint backlog up-to-date with the latest estimates, actual efforts, and any new information

discovered throughout the sprint. Otherwise, the sprint backlog could become meaningless.

Regardless of which specific planning approach is adopted, it is important to ensure that the planning process helps information, such as sprint goals, priorities, resources, and task estimates, spread around the team. Planning leads to four key results: clear goals and priorities, good time estimates, allocated resources to task completion, and team communication.

5. Conclusion

In conclusion, it is much easier to determine where we are going once we see where we have been. As described in different literature, there is no silver bullet [3]. No single methodology will fit all software development projects. The process must be evolved and optimized for the team and environment it's in. Looking back, the results of our choices or practices allowed us to adjust what we needed in order to improve productivity, reliability, and simplicity. We at TransCanada continuously evolve our team practices to accommodate different business needs and system requirements to increase customer satisfaction and operational excellence.

Acknowledgements

We would like to thank Frank Maurer, Rick Coutts, Karen Brown, Jeff Patton, Laurie Shafer, Eric Liu, Jennitta Andrea, Robert Purdy, Teresa DeMarco, Lynne Ralston, David Shellenberg, Tom Kuntz, Robbins Choi, Brent Sprecher, Brad Marlborough, Richard Hurst, Nam Nguyen, and Shanmei Mao, for their comments, guidelines, and encouragement.

References

- [1] Kent Beck, "Extreme Programming Explained – Embrace Change", Addison Wesley Professional, 2000.
- [2] Ken Schwaber and Mike Beedle, "Agile Software Development with Scrum - Series in Agile Software Development", Prentice Hall, 2002.
- [3] Fredrick Brooks, "No Silver Bullet – Essence and Accident in Software Engineering", IEEE Computer Vol. 20, No.4.