Practice collection for scrum development

Version 1.0

August 15, 2018

Practice	Context	Issues	Aim	Method	Outcome
e.g. practice name	**Team configuration or project environment**	**Issues faced by team**	**What outcome was hoped for?**	**Execution method of practices**	**Results of the practices (was the aim achieved?)**
Pair programming	* Seventh-year scrum master and development team containing two second-year programmers, one third-year programmer and one collaborator * Scrum master and collaborator had strong technical skills * The second-year and third-year programmers had business knowledge but did not have much development experience	* The code created by the technically skilled team members lacked business considerations and many areas had to be rewritten * The code created by the team members with strong business knowledge also had to be rewritten in many areas because of issues from a technical standpoint that were flagged during reviews * All of the members' code was eventually of a high quality as a result of the reviews, but the quality was not high initially * The technically skilled team members were able to write code, but their lack of business knowledge created a bottleneck, which meant that tasks such as investigation could not be done * The team members with strong business knowledge wanted to write code, but performed tasks such as investigation	* Filling the gap in technical skills * Reducing the time needed for code reviews * Elimination of situations where only certain people have certain business knowledge or the ability to perform a certain task	* After tasks are assigned in the daily scrum, each programmer decides which tasks will be carried out by pair programming * Pairs must be different from the previous day if at all possible * It is not necessary to perform all tasks in pairs (difficult business or technical tasks should be prioritized)	* The quality of the code was improved * Technical skills and business knowledge is now shared among the team, eliminating situations where only certain people can do certain tasks * Team members can now handle a wider range of tasks
Pair investigation	* Seventh-year scrum master and development team containing two second-year programmers, one third-year programmer and one collaborator * The third-year programmer and one second-year programmer had business knowledge, but the other members did not have much * The third-year programmer and second-year programmer with business knowledge tended to lead investigation tasks such as those related to operations.	* Investigations increase team members knowledge of specific areas, but this was mainly done by the team members in charge of execution, resulting in situations where only certain people knew how to do certain tasks * In cases where knowledge gained during investigations needed to be applied to a task, there were often cases where only certain people could do those tasks or where reviews were done by team members who did not have the required knowledge, causing a decrease in quality * Only certain people knew about certain resources, which led to time being wasted as team members got on the same page with each other or searched for those resources	* Elimination of situations where only certain people have certain business knowledge or the ability to perform investigation tasks * Improvement of business knowledge within the team * Improvement of quality of deliverables	* Team members with business knowledge work together with those who do not have this knowledge during investigations (applying the principles of pair programming to investigations) * It is not necessary to perform all investigation work in pairs (difficult business work or work that only certain people can do should be prioritized)	* There are fewer gaps in business knowledge that result in situations where only certain people can do certain tasks * There are now more members who can do investigation tasks, reducing delays in post-investigation tasks * Deliverables are now of a higher quality
Design by all members of Sprint Planning (Dept. 2)	* One scrum master and four development team members * Each team member was assigned a story to be executed in the sprint, and individual design work and tasks were planned and decomposed * There was a large imbalance in the business knowledge and technical skill level of the team members	* When design or planning and decomposing task for a story was assigned to any particular team member, that team member became the "leader" of that story, and would carry out development with only that story in mind * The size of task units varied significantly, making it difficult to carry out tasks * Misunderstandings occurred between the team member who was assigned the story and the other team members, resulting in many issues flagged during reviews and areas that needed to be rewritten	* Elimination of misunderstandings at the design and planning and decomposing task. * Enabling all team members to carry out tasks for all stories	* The facilitator stands at a whiteboard and writes about the design there * The other team members work on the design together and suggest other methods, ask questions, etc. * Measures such as a printing function are used so that the information on the whiteboard can be output on paper and attached to the Redmine wiki	* Misunderstandings no longer occur at the design and planning and decomposing task. * Team members now carry out tasks for multiple stories instead of just one, eliminating situations where only certain people can do certain tasks * Team members who do not know how to implement something at the design stage now receive immediate follow-up
Workshops for new areas	* Seventh-year scrum master and development team containing two second-year programmers, one third-year programmer and one collaborator * There was an imbalance in the technical skill level of the members	* There was a gap in technical skill between the team members, and some members could not picture how things needed to be implemented in technical areas that were different from those they had worked on before (e.g. the difference between screen implementation and the HTTP communication level) * Tasks were divided into small units no larger than one hour each, but work took longer than expected because team members could not picture how things needed to be implemented * A lot of time was spent on investigations	* Improvement of the team's technical skills * Reducing the gap between the time estimated for each task and the actual time taken * Reducing the time needed for investigations by each team member	* Team members who are technically skilled or the only person performing a task are asked to create samples and hold workshops on implementation based on these * Samples can include information such as how to write tests	* The team's technical skills have improved * Investigation times are now shorter and closer to the estimated time as samples can be referred to during implementation * At first, one technically skilled team member made most of the samples, but other team members are now actively involved in creating samples, which has made the team as a whole more productive
Outputting new information to the wiki	* The team was not outputting information on errors and how to solve them at the time the errors occurred * The team did not accurately remember how to solve errors because the information was not being output in a timely manner	* When an error occurred during development, it turned out to be an error that other team members had encountered before * It took time to carry out investigations to fix the error	* Reducing time needed to resolve errors * Facilitating the sharing of more tips within the team	* Errors and similar are written in the Redmine wiki as early as possible * Resources such as new materials used in investigations are also added to share information	* Errors are resolved faster as more specific information is available than that found by researching online, etc. * Tips shared among the team are not forgotten as easily
Definition of time boxes	* Seventh-year scrum master and development team containing two second-year programmers, one third-year programmer and one collaborator * Only the scrum master was knowledgeable about scrums	* Daily scrums were expected to be finished in 15 minutes but after 30 minutes, the team still had not decided what to work on that day, and issues were being discussed with team members who were not involved in those issues * Sprint retrospectives were expected to be finished in 1.5 hours, but the team spent four hours trying to find solutions to every problem, and there were no suggestions that could be acted on in a concrete way * Too much time was spent on each event, which cut into the time for executing tasks	* Staying within time boxes so that there is more time for executing tasks * Making each team member aware of what needs to be done for each event so that discussions and activities are more productive	* Introduction of a timekeeper * Sharing of the time boxes for each event among the team * Strictly enforcing end times	* Less time is needed for each event, which has increased the amount of time available for executing tasks * The team is now aware that when discussions take a long time with no decision made about the next specific action, this is due to insufficient information. They now take action by gathering the information that is needed. * Team members are now conscious of staying within the time boxes and wasting less time during their daily work
Implementation of chat tools	* The tasks to be done each day were decided at the daily scrum in the morning * Periodic reports issued after the daily scrum were not covered until the next day's scrum * Information was mainly shared by email	* There was no environment where team members could easily share what they were working on * There was no environment where team members could easily share the URLs of websites that provided technical information or were otherwise useful, or ask simple questions during their work	* Making it easy for team members to determine who is doing what * Making it easier to share information, without spending time on points such as greetings in standard email formats	* A chat environment that can be accessed by team members using devices has been built * A chat room is created for the timeline of each team member, where the team member posts the status of their tasks and any errors	* The team members now communicate more actively * There are fewer hurdles involved in status updates and information sharing * Tasks and errors can now be assessed easily as they are mentioned in each member's timeline

Communication of CI results by chat	* CI was carried out by pushing the source code to the development branch * Members checked CI results in the CI results screen when they had time	* The team did not know when CI would be finished, so team members had to confirm when each task was finished * Team members forgot to confirm the results if they were busy with other tasks, which meant that errors went unaddressed	* Reducing time needed to confirm CI results * Enabling errors to be detected faster	* CI is linked to chat tools and the results are posted to the timeline in the chat program * Only errors are communicated, because there will be too much information to read if both success and failures are posted to the timeline.	* Less time is wasted on unnecessary confirmations of CI results * Errors in CI are now being addressed more quickly * Errors caused by other team member's tasks can now be detected, so team members now communicate with each other to eliminate errors
Adoption of clean architecture	* Development of a new mobile app (iOS) * Architecture and design policies were also not decided	* Architecture such as MVC, MVVM and clean architecture was possible for mobile apps, but the team were unsure which to use * The team knew that the app would be expanded in future, but there were no predictions about how big it would become	* To adopt architecture that has plenty of potential for expansion and enables easy testing of loosely integrated components, as apps are expected to become large * To incorporate DDD elements in design	* Clean architecture is used * Information on clean architecture is available at the links below * https://qiita.com/koutalou/items/07a4f9cf51a2d13e4cdc * https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html	* The responsibilities at each layer are clearly distinguished, making design easier * As the layers are separate, it is easier for team members to be on the same page about the work that needs to be done for each layer * As the layers and responsibilities are clearly separated, it is clear which areas need to be fixed during development * The business logic is maintained well, as variance does not occur in the code
Implementation of KPTA	* Retrospectives (KPT) are done for each sprint * During each retrospective, try points are decided and executed in the next sprint	* KPT was used in retrospectives, but many of the improvements were vague ("Improve X", etc.), so no actual improvements have been made	* To ensure that improvements proposed in the retrospective of each sprint are made	* A (Action) is added to KPT * Actions are not vague points such as "Improve X"; they include specific, actionable points such as "Do Y" * The person/people who will carry out the action (a specific team member, the whole team, etc.) are designated to ensure that the action is carried out	* Improvements are reliably carried out as actions are now specific * It used to take time to identify specific actions
Creation of working agreement	* Things like rules for the team were tacit	* There have been misunderstandings between team members because the team rules were not documented * Disagreements in the team tend to result in lengthy discussions as there is no consensus about how to resolve such disagreements	* Determining rules for the team, increasing unity and enabling the team to function as an autonomous group * Documenting rules to eliminate misunderstandings between team members * Determining rules for decision-making so that discussions about disagreements can proceed smoothly	* Rules are decided among the team, written on imitation paper and placed where the team can see them * The rules are periodically reviewed and constantly updated * Any rules agreed upon by the team can be written (how to conduct daily scrums, rules to be complied with during development, etc.)	* Misunderstandings no longer occur easily as rules are documents * Team members take the initiative to follow the rules because they created and agreed on the rules themselves * Disagreements are now settled smoothly as the working agreement now contains rules on making decisions
Assignment of tasks in 30-minute units during daily scrums	* At sprint planning meetings, tasks are assigned in small units (several hours to a day at maximum) * The Kanban method is used as a task management tool	* It is difficult to assess how much work remains for tasks that were not completed that day when reporting on work at evening scrums	* Visualizing completed and uncompleted tasks for each day in detailed units	* The tasks for the day that are assigned in the sprint plan are broken down further into small tasks around 30 minutes long (this is done by each member before the daily scrum) * Tasks are assigned using a task management tool based on the Kanban method and shared at the morning and evening scrums	* Completed and uncompleted tasks for each day are now visualized in detailed units * Team members are now able to get on the same page about the specifics of procedures and how to carry out tasks at the daily scrum
Designation of tasks other than implementation that will take time	* Each sprint was a two-week cycle * Meetings with the client (including reviews of deliverables) were held weekly	* Allocation of tasks at script planning meetings only covers tasks required to complete implementation of the functions for each story, and preparation for demonstrations at meetings with the client has not been done.	* Ensuring that demonstrations can be performed as soon as a story is completed	* At sprint planning meetings, the content and methods to be shown during demonstrations are included in the goals for each story, and methods for accomplishing these are assigned as tasks	* Preparation work for demonstrations can now reliably be carried out as tasks for each story rather than as meeting preparation tasks * The goal of each story is now clearly defined
Setting of maximum patch size for code reviews	* Development team consists of one seventh-year programmer, one fourth-year programmer and one new programmer (all full-time) * All programmers work on development and all except the new programmer perform reviews	* Reviews that cover a large volume of content, such as those where the whole of a function needs to be reviewed at once when implementing the function, are a lot of work for reviewers and it is easy for points to be overlooked (based on experience with previous projects)	* Lightening the load of reviewers * Creating realistic reviewing conditions so that reviews are not conducted less stringently	* The following maximum SLOCs are set for each review and the developers adjust the scope of the content to be reviewed and the time at which they submit this so that the maximum SLOC is not exceeded ** Java+SQL+JSP: 600 ** JavaScript+SCSS: 300 ** Documents: 600	* The load of reviewers has been lightened * It is now possible to prevent situations where large amounts of backtracking are needed, as minor feedback about code can be given
Involvement of client in daily scrums	* The development team works in a different location from the business experts (PO and client) * This means that there is a physical distance between the development team and business experts, including the PO * There are no business experts in the development team	* The initial response is late if it is done via email or QA. * If misunderstandings occur during this correspondence, getting on the same page is a lengthy and laborious process that wastes man-hours	* Immediately resolving business points that are unclear to members of the development team * Preventing misunderstandings between the team and the PO and other business experts and facilitating more communication	* The PO and other business experts participate in daily scrums * The team phones these parties, using speaker phone so that the whole team can hear	* Team members became to ask simple questions easily, which eliminates the need for time-consuming Q&As and emails, and frees up more time for development tasks. * There are fewer misunderstandings than before as team members can discuss matters to make sure they are on the same page * The quality of information from the PO and other business experts is now higher (as information is now being provided in a way that makes it easy for the development team to understand what the PO and other business experts have in mind)

Involvement of client in daily scrums	* The development team is posted on the client's premises * The client themselves understands and participates in scrums	-	* Ensuring fast course correction by making the status of the scrum team's activities known * Sharing what the product owner has in mind	* The product owner is asked to participate in daily scrums	* The aims were achieved * There are fewer misunderstandings and there is more sharing of context (communication has been smooth because the premise is understood without needing much explanation in emails or similar) * On the other hand, the competence of the scrum master is being questioned more because tasks that were not included in planning are sometimes requested
Active use of whiteboards	* The majority of the development team are from China and there are times when it is not possible to get the point across in Japanese * There are also many new technical concepts, and each member cannot clearly picture what they are supposed to do	* Misunderstandings occur easily in executive tasks * Documenting the vision for each goal and the nature of each task is a lengthy (and thus costly) process, which is not desirable for the client either	* Efficient sharing of visions for goals	* Business questions and technical issues are explained on the whiteboard each day * The whiteboard is also used to get on the same page with the product owner about points such as the functions and screens that will be created during sprint planning	* The aims were achieved
Establishment of completion conditions for individual tasks	* Tasks are managed in Redmine. After being divided up in sprint planning, they are registered as tickets.	* Completion conditions of tasks varies from one individual to another (when the completion reports of a task are checked, it is found that the actual work that was needed for the task has not been completed) * Tasks are allocated during sprint planning and there tend to be cases where the goal is not reached even when all of the tasks are completed (due to misunderstandings about the completion conditions of each task)	* Ensuring that all team members are on the same page about what needs to be done for each task * Ensuring that the development team can work on tasks with a unified understanding of the goals of each task	* Completion conditions for all to-do tickets representing tasks are indicated	* There has been positive feedback that the work that needs to be done is clearly defined and easy for team members to act on * If tasks are broken down into units that are too small, it takes a lot of work to write up each task based on its completion conditions →The team plans to try executing operations by note rather than by ticket and providing supplementary information on completion conditions through communication to reduce the load
Elimination of evening scrums	* The client and all team members (up to 16) participated in daily scrums * The purpose of evening meetings was to assess progress and any issues or bottlenecks * Each team member was asked to report on the work they had done that day and any issues they had encountered	* Participating in daily scrums is a lot of work	* Reducing the work involved in participating * Concentration on work where the development team needs to generate value (verification work)	* Evening scrums were eliminated after discussion with the client * Bottlenecks are now shared in daily communication or at morning scrums * Status updates are now shared by Skype or email	* The aims were achieved * As the client is in the same office and the scrum team is right nearby, information can be shared without a venue such as an evening scrum
Pair programming	* There is a significant gap in technical skills and experience among the team members * The end client, who does not have practical experience with Java, is also on the development team * Our newer team members also lack knowledge and practical experience with Java	* Implementation tasks cannot be assigned to members who do not have much experience or knowledge	* Improvement of team members' knowledge and experience * Improvement of performance throughout the team	* Pair programming tasks are assigned explicitly	* Visible results are yet to be seen (as it has not been long since this practice was adopted)
Involvement of product owners in creation of quotes (planning poker)	* The product owner themselves is a highly skilled engineer (and can therefore oversee all stages up to implementation)	* The product owner does not seem satisfied about what can and cannot be accomplished in each sprint	* Ensuring that the product owner is satisfied with the scope of each sprint	* The product owner is asked to participate in planning poker * Information on points such as the tasks and coordination required and the risks involved is discussed and shared	* The product owner has accepted the scope
Creation and use of problem/try votes in sprint retrospectives	* KPT is used for sprint retrospectives	* Opinions are not expressed openly during KPT (there was a concern that team members did not feel invested)	* Making team members feel more invested * Making team members aware that they can improve their own environment	* Paper is distributed before retrospectives and team members are asked to write down problems and things to try before being asked to share their opinions * A vote is taken among individual team members before adopting points to try	* Team members feel more invested * It is also easier to assess each team member's opinion * This has helped to make communication smoother