Views on the Agile Retrospective

Bjørn Dølvik, Studentid:723690

November 29, 2014

Introduction The agile retrospective, also known as postmortem, is a process used within agile development to bind people together and open opportunities for team to improve on their current development practices. It is a practice that are conducted in several different ways, by many different people. Through this article I will look at the agile retrospective through different perspectives and connect this to existing literature. I'll be focusing on three main topics: Participation, knowledge and group awareness. But first I'll go a little bit deeper into the agile retrospective.

The agile retrospective As previously mentioned the agile retrospective is a process used within agile development. Usually it takes place in the end of a development iteration, often called sprint within scrum. The retrospective are performed in different ways, but there are some common techniques used as KJ-sessions, root cause analysis and fishbone diagrams. KJ-sessions are done by separating all participants, make them write opinions on the last iteration of the project, and when all are done they get together and each participant then explains their comment to the rest of the group. Root cause analysis can then be performed on these comments were all the participants group the different comments together and discusses their causes. Fishbone diagrams may also be applied for this reason were the participants takes each big cause and find subcauses for each cause until they all agree that they found the root problems. When the retrospective activities are finished the group find a set of the issues identified, which they will try to work on for the next iteration. Through these activities development teams are able to improve on they development practices.

The participants in an agile retrospective varies from team to team. Some

teams include the customer, others include management and someone only includes the team of developers. Including different participants depends on what kinds on practices one wanna improve. If one want to become better at including customers in the project development it could be a benefit to include them in the retrospective as well. However if one want to improve on the internal practices of a development team it might be wise not to include the customers. By including different kinds of participants one might get different results from performing an agile retrospective.

Agile retrospectives is a practice that supports sharing of knowledge. Through the agile retrospective participants are required to share their experiences through verbal communication, thus making tacit knowledge explicit. This can later be recorded for sharing with other teams or in a company. This may also support communities of practice as participants gets experience in converting tacit knowledge into explicit. By performing agile retrospectives one trains participants in knowledge conversion from tacit to explicit.fa

Through agile retrospectives group awareness is increases as a result of practice. As all the participants are presenting in KJ-sessions and contributing in cause analysis it raises awareness to the others participating. The other participants may then acquire these experiences and such become aware of what others are thinking of the current practices used. Thus it increases the awareness as everyone shares their opinions on the current state of the project.

Existing literature Following I will be examining literature related to the topics participation, knowledge and group awareness.

Participation In the article "Investigating the design process: participatory design in agile software development?" Karlheinz Kautz [4] identifies some different aspects of participation by stakeholders outside of development team. He differentiate between two types of outside stakeholders that can participate in a development process: Customer and user. The customer are the one that has ordered the information system, who will pay for it, and whose requirements are initially specified. The user on the other hand is the one who will use the information system. Further Kautz describes three types of roles a stakeholder can have:

- Informative: The stakeholder provide information about their domain knowledge or are subject to observation.
- Consultative: The stakeholder are asked to comment on preset design solutions.
- Participatory: The stakeholder actively participate in decision-making regarding the solution and in the design process.

He uses this roles through a case study to identify which role one external stakeholder have in different situations during a development process.

His findings reveals that during preparation of an agile iteration the onsite customer is in both the participative role and the informative role. He participates during the design approval and the requirement prioritization and acts informative when the requirements list is specified. The users that are involved during preparation of an iteration also acts informative in the same way as the onsite customer.

During the development in an iteration the onsite customer acts in all the three roles, while the operational users contribute through an informative and consulting way. The onsite customer contributes during daily informal visits, weekly feedback meetings and during working software presentations. The operational user are involved through trials and working software presentations.

In the evaluation phase of an agile iteration Kautz finds that the onsite customer acts participative and consultative while the operational user acts only consultative. The participation from the customer is during the preparations of a performance and acceptance test. The consultative contribution from both the onsite customer and the operational user is during the performance and acceptance test.

Kautz summarizes his findings and discusses some issues that was observed during the case study. The close contact between the stakeholders and the developers brought with them some false expectations. The stakeholders assumed that developers would be able to change requirements on a day to day basis as the development process was agile and during the daily informal meetings came with new or changed requirements. The developers also asked

a lot of questions to the stakeholders to get a better understanding of what was need of the system. This lead to the stakeholders believing that the developers didn't bring any new smart ideas to the system. The close contact also lead to positive issues. The user stories become more accurate through user information. And the change rate was high during the iterations as to the close contact.

Finally Kautz concludes that during agile development participatory design can be a vital element and that creating an integrated framework for participations is important to get the most out of it.

Another article talking about participation is "Postmortem reviews: purpose and approaches in software engineering" by Torgeir Dingsøyr [2]. He talks about who one should include in postmortem reviews. Dingsøyr looks at different practices used in postmortems reviews and summarizes that one would want to include as many participants as possible as all that are involved in the project can contribute knowledge. He also states that it will broaden existing communities of practice, especially if people from new projects also are invited. However Dingsøyr suggests that customer and users should not be included in the postmortem meetings. By including external stakeholders one moves the focus of internal improvement to customer relations. He states that it will become difficult to blame stakeholders if they are present. To summarize, Dingsøyr suggest including as many participants for postmortem reviews except external stakeholders as it will change the focus of the meeting.

Group Awareness The article "Group Awreness in Distributed Software Development" by Carl Gutwin, Reagan Penner and Kevin Schneider [1] talks about the group awareness in a distributed software development setting. They interview people from three big open source software projects about how they maintain group awareness across distance between each developer in a team. They find that there are mainly two kinds of awareness that are present: general awareness and specific awareness.

The general awareness are the part where the developers get an overview of the project. This includes what are being done, what are plans for the project as a whole and such. This is done textual aids such as mailing lists,

commit logs and text chat. Through mailing lists the developers gets information about what has been done and what is about to happen. It also gives and oversight of who is active with a particular task. Keeping the mailing lists however requires a strong culture of making things public and an additional communication effort. Through commit logs other developers are able to get an indication of people's activity levels and area of work. It can however become time consuming to keep oneself updated on all the commit logs as these can be numerous as up to over a hundred per day. Lastly text chat is used as an informal way to maintain a general awareness. According to the authors this kind of informal chat provides similar awareness as the watercooler in an office landscape, by getting information one might not have been looking for and thus getting awareness. Text chat has a potential drawback as it can replace the mailing lists, but if this danger is indicated one can post summaries of the chat back to the mailing list.

Specific awareness are the awareness that provides an oversight of whom one can contact if one has a specific issue. This is useful one has a bug in a specific part of a system or need an introduction to some detailed part of a code. The authors were presented with several techniques used by the projects. Maintainers field indicates who has the codes ownership at the moment and if changes are wanted who to contact. This technique however only works if the information is up to date. Code repository is another way of obtaining specific awareness where one are able to see what is done to specific parts of the project. Issue and bug trackers helps provide an oversight over which issues are worked and the communication related to these. Social networks are also used to find other more senior developers if one need some information one are not able to retrieve oneself. This requires a strong organizational culture that allows and promotes contact. Finally Project documentation is used to get specific details about any part of the project. Project documentation, as well as other techniques requires that it is up to date.

By identifying these techniques the authors of the article discovers that distributed teams are able to maintain awareness through textual aids. And they conclude that the developers maintain general awareness of the entire project and a more specific awareness of the parts they are planning to work with in the future. They also state that the projects are able to maintain a good awareness as they have a strong organizational culture in promoting

behaviour to share and maintain information.

Knowledge Two articles talk about the nature of knowledge and communities of practice. As indicated above Dingsøyr [2] talks about how a postmortem review are able to contribute to stronger communities of practice.

Dingsøyr tells about how through a postmortem review one do externalization of knowledge. Through meeting together participants are forced to share their knowledge and thus make their own tacit knowledge explicit. One also can discover new knowledge as the participants together analyzes their issues, according to Dingsøyr.

In his article he also talks about how one can use the postmortem reviews to share knowledge. He suggest that if a company has the wish to share knowledge one should after each postmortem review write a report codifying the knowledge, thus making what being discussed explicit. He recommends that larger companies should use more time than smaller companies codifying the knowledge to documentation as they are more dependent of codified knowledge. Smaller companies shouldn't use that much time on it as it is an expensive task.

The other article talking about knowledge and communities of practice is "Knowledge Management: The Benefits and Limitations of Computer Systems" by Geoff Walsham [3]. The focus of Walsham's article is how Information and Communication technologies (ICT) can support knowledge management. I'll not dwell much of the ICT part, but rather look at the knowledge management.

Walsham starts of by talking about knowledge in general and how humans use sense-reading and sense-giving to interpret knowledge. Sense-reading is the process of giving meaning to what others try to share with us. While sense-giving is the process of giving meaning with what we share to others. Further it is said that what a person A transfers to person B will not give the same meaning to B as it did to A as both are individuals.

Following Walsham describes community of practice. Community of practice

is a gathering of individuals that creates a shared understanding of some area of practice. This gives knowledge sharing as the individuals in the community are required to share their knowledge so they all together can create a shared understanding. This sounds pretty great, but there is a challenge: Sharing knowledge between communities and doing this in an effective way. Walsham describes some organizational behaviours that can help this. These changes include facilitation through mentor relationships, special interests groups or reward systems. Walsham then continues presenting the benefits and limitations of ICT system as a solution to this challenge, which earlier mentioned i wont dwell on now.

Walsham concludes his article by saying that as of today computer system isn't the solution for improving knowledge sharing. He states they don't replicate or replace the deep tacit knowledge of humans. However he also says that they can help improve communication of already existing relationships, and if one are to use computer system for this one should be assessing their benefits and limitations.

Discussion As I've now have described the three different views and the literature connected to them, I'll connect this up against the agile retrospective.

First the issue on who should participate in an agile retrospective. Kautz finds that including the customer can be very beneficial throughout the agile development. Dingsøyr on the other hand recommends that the customer is left out of the retrospective. Why could it be wise to leave the customer out of the retrospective? We could assume that if a customer would take a participative role as is required by all the participants in the retrospective. He would almost definitely bring more knowledge to meeting as a new set of eyes. Let's use Kautz findings that the customer expected the development team to change requirements on a daily basis. If the customer where to attend a meeting where this was an issue the worry from the developers about changing requirements might never be spoken. This might happen as the developer could be afraid he would ruin the customer relationship. And such the issue wouldn't be resolved. However if the customer didn't attend the meeting the developer would probably raise this issue and the participants could then resolve the issue. This is just one example of how

customer participation in an agile retrospective could be destructive. It also fits well with Dingsøyr's assumption that if the customer were to participate the focus would change from improving development practices to customer relations. This does not mean you should leave the customer out during the whole development process. If your goal is to improve development practices you want an open discussion which the participation of the customer might destroy.

As to the awareness in a project the retrospective may contribute some additional information to the techniques described by Gutwin, Penner and Schneider. The cases that were studied was all distributed development projects. The techniques they all used to get a general awareness provided them with information about what was going on in the project and what was about to happen. It gives however no regards to how things are done. Using mailing lists as an example the developers get information on which parts of a system is being worked on. It does not tell who are pleased with the progress of the project. Thus not giving any feedback to any of the developers. As the purpose of retrospectives is to improve development practices this naturally comes with developers raising issues about the project. If the projects were to adapt retrospectives they would be able to gain additional awareness. The retrospective information could be produced into a report and be distributed throughout their techniques. However performing a retrospective might prove a challenge as all the developers are distributed. The retrospective cannot replace all the other techniques as the primary source for awareness. The specific awareness is not covered by the retrospective as it doesn't necessarily contain information about who is doing specific parts of the project. It can however implicit raise the specific awareness as the contact between participants are more personal and thus reveal specific details. This cannot replace other techniques described by Gutwin, Penner and Scheider as the specific awareness raised is more coincidental. The retrospective raises general awareness and can contribute to other awareness techniques with new information.

Walsham describes about how humans do sense-reading and sense-giving when they process knowledge. Through the agile retrospective tacit knowledge is externalized into explicit according to Dingsøyr. This process is done through discussion and analysis and thus much sense-reading and sense-giving occur among the participants. Dingsøyr claims that retrospectives

broadens communities-of-practice as part of the process. Through analysis and discussion the participants creates a common understanding of how the development process should be and thus broadens communities of practice. The challenge Walsham presented in being able to share knowledge between communities effective is possible through the retrospective. The retrospective itself isn't very costly as it in most cases only lasts for a short duration. The knowledge obtained from the retrospective can be written down in a report, as it is already externalized through discussion. However Walshams challenge still holds merit as the retrospective only applies to the knowledge field on apply a retrospective on. Holding a retrospective broadens communities-of-practice and have the ability to share the knowledge externalized through reports.

Summary Through this article I have looked at the practice agile retrospective through different views of participation, awareness and knowledge. Connecting this to literature I have experienced how the agile retrospective is able to broaden understanding of different issues related to software development. Participation in agile retrospectives are a matter to consider as different participants can give different results. Awareness is increased in development projects through the agile retrospective especially related to how things are done. Tacit knowledge is externalized through the agile retrospective and thus broadens communities-of-practice and makes software development practices easy to share.

References

[1] Gutwin C., Penner R., and Schneider K. Group Awareness in Distributed Software Development. *CSCW'04*, 2004.

[2] Torgeir Dingsø yr. Postmortem reviews: purpose and approaches in software engineering. *Information and Software Technology*, 47(5):293–303, March 2005.

[3] Walsham G. Knowledge Management: The Benefits and Limitations of Computer Systems. *European Management Journal*, 19(6):599–608, 2001.

[4] Kautz K. Investigating the design process: particpatory design in agile software development. *Information Technology and People*, 24(3), 2001.