Beyond dashboards: on the many facets of metrics and feedback in agile organizations

Olivier Liechti
University of Applied Sciences and Arts Western Switzerland
Yverdon-les-Bains, Switzerland
olivier.liechti@heig-vd.ch

Jacques Pasquier
Fribourg Univesity
Fribourg, Switzerland
jacques.pasquier@unifr.ch

Rodney Reis
Avalia Systems
Yverdon-les-Bains, Switzerland
rodney@avalia.systems

***Abstract*—One of the key principles in agile development is that the team should use continuous feedback cycles to maintain awareness both about the product, the process and the involved people. In healthy agile environments, people see a lot of value in the collection of metrics and in their collaborative analysis to support a continuous improvement process. In pseudo agile environments, metrics are often perceived as a threat and they make people uncomfortable. In theses conditions, they can do more harm than good. In this paper, we argue that the acceptance of a data-driven continuous improvement process is heavily influenced by the nature of the channels through which the metrics are published. We argue for the need to go beyond typical dashboards and propose additional engagement channels. We discuss the notions of *awareness* and *story telling* and illustrate them with various experiments conducted with agile teams.**

***Keywords*-agile; metrics; feedback; continuous improvement; awareness; retrospective meetings**

I. Introduction

The agile mindset embraces the fact that there is no perfect, nor stable development process. Every team operates in a different context and this context changes constantly for human, business and technical reasons. Accepting this reality, effective teams should be engaged in a continuous introspection and improvement exercise. They should be eagerly looking to adjust their engineering and collaborative practices to improve quality, lead time and satisfaction.

There is nothing new here: practices such as retrospective meetings have been promoted since the early days of the agile movement. Yet, they are harder to do than it would appear at first glance. Over the last 15 years, we have worked with many development teams. The majority did not perform them consistently and with a sincere purpose. In some cases, the culture of the company did not truly encourage transparent and honest communication. In other cases, time pressure was presented as a reason to skip these types of activities. Even when the team took the time to meet at the end of an iteration, it was quite rare to see a candid discussion with a genuine desire to change and experiment. The reality of many development teams is quite far from textbook examples and case studies presented at agile conferences.

On the bright side, we have worked with teams who had a true agile culture and who were eagerly looking for weaknesses in their practices. In these cases, we saw clear and tangible benefits offered by continuous introspection. It did not only make the product and the team better. It also reinforced the culture of trust and it boosted engagement and satisfaction. Another encouraging element is that we see a growing awareness and willingness to adopt this continuous improvement mindset. Ten years ago, many teams only started to adopt extreme programming practices such as unit testing and continuous integration. Today, these practices have become the norm. We are optimistic that the adoption of introspection practices will follow a similar adoption curve. To foster this adoption, we argue that research in software engineering should focus on approaches, techniques and tools that support a *holistic data-driven continuous improvement process*.

In the remaining sections, we start by defining this notion in more details. We use Pink's motivation theory [1] to frame the discussion. We explain how *autonomy*, *mastery* and *purpose* are key factors for the acceptance and effectiveness of such a process. We then consider different types of feedback channels used to engage the team. We make the point that it is worth going beyond the usual dashboards, which can be perceived as a threat to *autonomy*. Hence, we argue for the need to complement them with other types of channels. We first consider the role of *awareness* and its impact on *mastery*. We illustrate the discussion with a case study, where a team had set the goal to improve its testing practices. We then consider the role of *story telling* and its impact on the *sense of purpose*. For illustration purposes, we consider a case study where we mined various software repositories to reveal the 10-years long history of a software system. We explain how we created a visual representation of the software story and how this engaged the team in introspective discussions. To conclude, we present a model for a holistic data-driven continuous improvement process and summarize the questions that need to be addressed during its implementation.

II. Data-driven continuous improvement

Agile methodologies put a strong emphasis on short feedback cycles, in all aspects of the development activity. Think about the following agile practices and their relationship to a different type of feedback:

- *on-site customer*: feedback about product usage
- *review meetings*: feedback about product evolution
- *automated testing*: feedback about product quality

- *continuous integration*: feedback about convergence
- *daily meetings*: feedback about progress
- *code reviews*: feedback about quality
- *task boards and charts*: feedback about progress
- *retrospective meetings*: feedback about team practices

In all of these examples, the idea is to regularly assess the impact of individual and collective activities. It is then to make this assessment visible to the stakeholders so that they can take the appropriate actions to stay on track. The rationale is that when issues and flaws are discovered early, they are easy to fix. This equally applies to technical issues (e.g. design flows, coding guidelines violations), to product usage issues (e.g. misunderstanding of customer needs, usability problems) and to team issues (e.g. bottlenecks in the development process, inefficient handovers, latent conflicts).

Depending on the context, the way to make the assessment and to give the associated feedback varies a lot. In some cases, everything happens in the course of a face-to-face discussion and there is no need for automation or tool support. In other cases, however, it is wise to drive actions based on facts and quantitative metrics, rather than on mere impressions. This is where the notion of *data-driven continuous improvement process* becomes relevant. Such a process should allow a team to set specific improvement goals, to define how the progress towards these goals can be measured, to monitor the actual progress and to evaluate the broader impact of the improvement. To illustrate this last point, think about a team who sets the goal to improve test coverage. A proper improvement process should not only indicate whether test coverage is going up and down. It should also indicate the impact of a better test coverage on customer feedback, team satisfaction, savings, etc. Going beyond the first level metrics is not trivial, but it is of uttermost importance when it comes to give a *sense of purpose* to contributors.

A process should incorporate additional capabilities. Firstly, it should not only allow the team to maintain awareness about the current situation. It should also allow the team to easily compare the present with the past (i.e. it should allow the team to analyze how the situation has evolved over time). It should also allow the team to make projections about the future (i.e. it should allow the team to analyze how the situation is likely to evolve going forward). Finally, it should allow the team to compare itself with other teams, inside and outside the organization. Going back to the example of test coverage, it would be helpful for teams to know how they perform compared with other teams in the company, in the same industry and in reference open source communities.

A. Autonomy, mastery and purpose

In truly agile organizations, using metrics to drive a continuous improvement process should not be controversial and should be welcome by the team. However, many organizations are still in transition towards true agility. It is relatively easy to introduce engineering practices and process ceremonies. It is much harder to change the mindset and the perceptions of people. In many environments, introducing a measurement program is sensitive and can be threatening if it is perceived as an external control mechanism. The good news is that done properly, it is also a great way to promote cultural values and to boost motivation.

Pink has proposed a theory for motivation in the workplace [1] which aligns very well with agile values. Drawing from research in social psychology, he argues that while most enterprises rely on *extrinsic motivators*, these are not the most efficient. Extrinsic motivators are typically financial incentives supposed to drive certain behaviours and results. Pink then describes three ingredients that contribute to the *intrinsic motivation* of employees: *autonomy*, *mastery* and *purpose*. With these ingredients, employees are inherently driven by the satisfaction of doing their job, which turns out to be more effective. Pink's theory provides a useful framework to study factors that can hinder and facilitate the adoption of a data-driven improvement process.

1) Autonomy: for an agile team, *autonomy* is the freedom to organize itself in order to reach the goals for which it has taken responsibility. The opposite of autonomy is *control*, where employees are not asked to deliver results, but are told what to do, how to do it and are constantly watched over. Autonomy does not mean that the team works in a vacuum and in secrecy. The best autonomous teams value transparency and are happy to expose their progress, as well as the issues they are facing.

2) Mastery: Pink describes mastery as the urge to get better and better at something that matters. Mastery is the very essence of the continuous improvement mindset. The implementation of a data-driven improvement process should help people monitor their level of mastery. This suggests that mastery needs to be defined and measured, which is not trivial. Furthermore, we believe that both *individual* and *collective mastery* should be considered. For a software engineer, to see progress in the ability to use a technology is rewarding. For the same engineer, to be part of a team and to see the increasing ability of the team to handle all aspects of releasing software is perhaps even more rewarding.

3) Purpose: people have a sense of purpose when they realize that their actions do not only address an immediate need, but also serve a greater cause. A data-driven continuous improvement process can increase the sense of purpose, by making the larger impact of individual and collective practices explicit and measurable. It should not only measure output, but also outcome. Going back to the previous example, tracking the evolution of the test coverage is valuable, but measuring the impact on customer satisfaction provides a much stronger motivation.

We argue that paying attention to *autonomy*, *mastery* and *purpose* increases the chances of success for a data-driven improvement process. We further argue that these motivation factors are best served by different types of feedback mechanisms. In the following paragraphs, we look at the relationship between *analytics dashboards* and *autonomy*, between *awareness channels* and *mastery* and finally between *story telling* and *sense of purpose*.

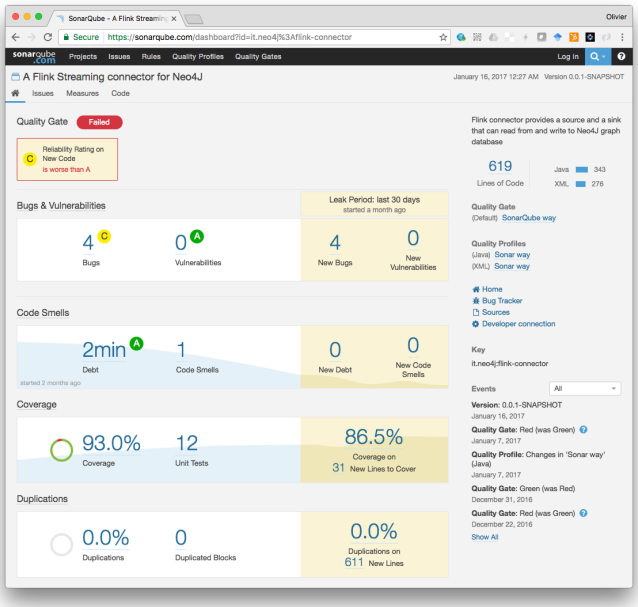

Fig. 1. Code metrics in the Sonar dashboard

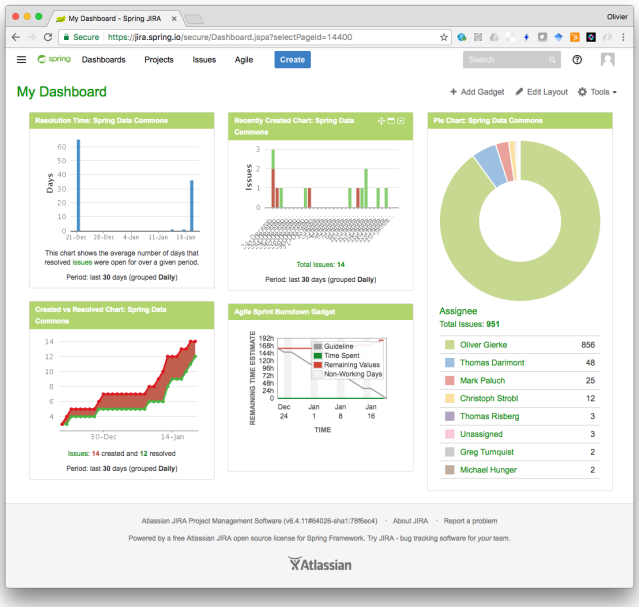

Fig. 2. Process metrics in the Jira dashboard

III. Dashboards and their ambivalent impact on autonomy

The first user interface that comes to mind when talking about a *data-driven* approach is most likely a dashboard, with a collection of textual and visual widgets. In other domains, analytics dashboards help people make data-driven decisions to improve their sales process or their energy consumption. In our context, a dashboard presents software metrics and helps people make decisions about the development process. SonarQube and Jira are two popular tools that include a dashboard interface. As shown in Figure 1, SonarQube focuses on code metrics: complexity, unit test coverage, duplications, etc. As shown in Figure 2, Jira focuses on process metrics: issue resolution time, team velocity, etc.

Software analytics dashboards are definitely helpful and important when implementing a data-driven continuous improvement process. However, we argue that they need to be introduced with care and for the right motives. In organizations that do not have a sincere agile culture, it is tempting for management to introduce a measurement program to track quality and productivity from the outside. Managers who operate in command-and-control mode often love the idea of using dashboards and KPIs to watch over the team. For obvious reasons, this approach does more harm than good. Many people will interpret it as a lack of trust and their motivation will suffer. Besides, they will then often find ways to game the measurement system. For example, you can easily imagine what is likely to happen if individual productivity is naively measured by the number of lines of codes added to the source. We have witnessed cases where a code coverage ratio was imposed on the team without a shared purpose. Targets were achieved, but many tests did not make a single assertion.

IV. Awareness channels and their positive impact on mastery

Awareness is a generic term and has different meanings depending on the context. Saying that agile teams should be self-aware is the same thing as saying that they should observe and reflect on their practices and behaviours. In this paragraph, however, we look at awareness as it has been studied in the Computer Supported Cooperative Work and Human Computer Interaction literature [2]. One of the early definitions [3] for awareness is that it is *an understanding of the activities of others, which provides a context for your own activity*. A common characteristic of awareness systems is that they often operate at the periphery of the users attention. In other words, users do not have to focus their attention on a user interface and still have a sense of what is happening. Many such systems use ambient user interfaces [4] embedded in the physical environment. Typically, awareness systems capture some form of activity in a physical or digital space, before generating representations of this activity in another physical space.

Interestingly enough, the idea to use the overall physical environment as a channel to broadcast information has long been recommended by agile experts. What is the motivation for sticking user stories, burndown charts and other artifacts on the walls? It is make activities, progress and feedback effortlessly visible to everyone. Status information does not have to be pulled out of a system by developers, it is pushed onto them. The concept of *information radiator* [5] has been proposed to describe this approach. An information radiator is a large display that makes some aspect of the software, the process or the team publicly visible. Some information radiators are *low-tech* (paper signs on the wall), others are are *high-tech* (large LCD displays). Some information radiators

use concrete representations (numbers, line graphs), others use abstract representations (trafic lights, lava lamps, smileys).

A. Make progress tangible with information radiators

When we introduced the notion of *mastery*, we explained that people are motivated when they feel that they are getting better at what they are doing. We also stated that in agile development, mastery relates to both individual and collective practices. Motivation comes from personal progress, but also from the team progress in mastering professional skills.

Information radiators are a very interesting vector for displaying progress indicators. They publicly convey this information, without people having to ask for it. Some of them have a playful dimension, which can have a significant emotional impact. Think about a physical bell in the middle of the office, which rings every time a critical bug has been fixed. Think about a visual representation, which represents the team as a garden, where bees on flowers represent developers committing on certain code repositories. These types of representations are interesting, because they are good at prompting the curiosity and interest of people with low cognitive overhead. When unusual or interesting events are noticed, then people can transition towards more concrete and detailed representations if they wish to know more.

B. Case study: improve agile testing practices

A couple of years ago, the first author was leading a team of about 20 engineers in a software startup. There was a strong culture in the company distributed across Switzerland, Romania and Singapore. From the early days, the team had been driven by agile values: trust, commitment, transparency, collective responsibility, having the right to make mistakes and a sincere desire to continuously learn and improve. This does not mean that everything was perfect. But the key point is that it was ok not to be perfect, as long as there was a desire to understand why and to propose adjustments. The team was not systematically using metrics to track product and process performance. The reason was not fear, nor a lack of interest. The reason was rather the perceived complexity and effort required to put a measurement program in place, combined with the urgency of the product development and lack of time. As a side note, this shows the need for tools that make the process of collecting software engineering data easier. Such tools provide the foundation on top of which data-driven improvement processes can be implemented.

At some point, the team faced product quality issues. A combination of factors had led to this situation, but the team agreed that not enough effort had been put on automated testing. At that time, there was agreement that automated testing was a key quality driver. But there was also still a reluctance to do it systematically. Testing was an activity decoupled from coding and was not perceived as very interesting.

This was a perfect situation to experiment with our early ideas in the area of data-driven continuous improvement. Our goal was to change behaviours and to grow a testing culture within the team. To do that, we started by making the improvement goal explicit and very clear. We got shared agreement that we had to get better at testing, at all levels, and that the expected impacts would be a shorter time to release new features and an increased developer satisfaction (releasing buggy features after many back-and-forth between development and QA is not satisfying for anyone). We then designed and implemented a basic platform, which would allow collect the results of all test executions, across category (unit, integration, performance, etc.), system component, programming language and environments (developer machines, build pipelines). With this data, we would be able to know who was creating new tests, who was running tests and how the test suite was evolving. Finally, we had two ideas for adding a fun dimension to the feedback cycle. The first idea was to introduce gamification features: who would be the test champion of the week? Did the team beat its previous record during this iteration? We implemented a web page to answer these types of questions. The second idea, which we only implemented at a later stage [6], was to make the progress perceptible via ambient interfaces, in realtime.

The experiment was very successful. We did not use a scientific method to evaluate its results, but we believe that the following observations are worth sharing. They are important lessons that can facilitate the implementation of an improvement process in other contexts. The first observation is that because the initiative emanated from the team, there was little concern about being controlled and judged. Only one developer showed some initial resistance, but gradually understood the rationale of the approach and later became one of the most rigorous players. The second observation is that the fun dimension brought by the gamification served as an initial trigger for behaviour change. It engaged all team members, who had this initial motivation to write more tests. The awareness widgets, which not only showed the status of the build but also the growth of the test suite conveyed a sense of mastery. Even more interestingly, once the team got into the routine of writing tests, this activity naturally became an integral part of the development activity. Very quickly, the team saw the impact of the shift and at this point, the gamification features became a secondary motivation factor. The process improvement had an impact that was visible to everybody and that gave a lasting exhilarating feeling to people. It gave them a real sense of mastery. More details about the platform that was developed to support the team are described in [7].

V. Channels that enable story telling and their positive impact on the sense of purpose

One function of any dashboard is to present an overview of a system state. The aggregation of visual indicators constitutes a model of the system. As such, it is a simplification of the reality and has to be used with some caution. We already hinted at this issue before, when we described naive attempts to measure developer productivity. If a dashboard only displays low-level metrics, such as the number of lines of codes, of git commits or of open issues, then it will fail to provide

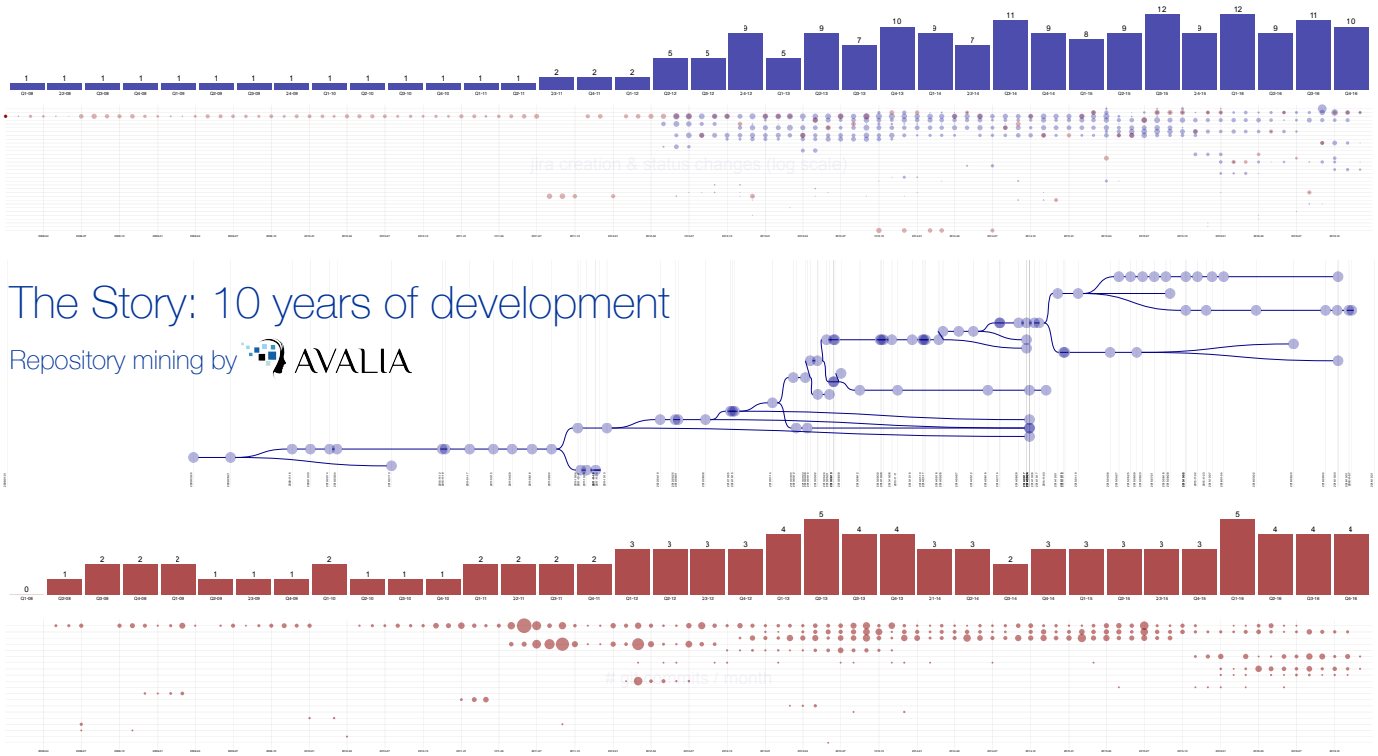

Fig. 3. Using visualizations to trigger introspective discussions

actionable insights to the audience. Worse, the indicators could be misinterpreted and lead to all sorts of problems.

For the collected data to be really useful, it needs to be placed in context, analyzed and augmented with human explanations. There is a big difference between showing raw data and *telling a story* that puts this data in perspective. Digital storytelling [8][9][10] is the art of combining data analysis, interactive visualization and narratives to communicate knowledge. Journalism and education are currently its main application domains, but the approach is applicable to any data-driven initiative. In particular, we believe that fills a gap in the context of an agile software process improvement. When we talked about the sense of purpose, we explained that it is important for people to understand and measure the larger impact of their actions. We claim that it is difficult to get a powerful impression only by looking at a dashboard. A complete story, with explanatory narratives and rich visalizations is better suited to communicate the broader outcome of individual and collective practices.

A. *Case study: trigger discussions and document insights*

We have recently worked with a software company, which is developing a sophisticated software platform since a bit more than ten years. The company is growing fast and many aspects are changing. The company is using agile methods, but it not using a lot of metrics to monitor and tune them. Because of the increasing complexity, the team is facing challenging questions: are we as quick and efficient as we were ten years ago? If not, what are the reasons that could explain the slower pace? If the pace is indeed slower, then did we gain anything in terms of quality or sustainability? Are there practices that we should introduce or adjust to improve certain aspects? These are complex questions to tackle. Without data and facts, it is difficult to have an informed and constructive discussion. Without data and facts, it is very hard to assess the impact of any action taken to improve the situation.

The case study is interesting, because a lot of historical data was available when we started to work with the company. There was data in the version management system, in the issue tracker, in the code review platform, etc. There is growing interest from the research community to apply data mining and visualizations techniques to data collected from such repositories and the term *software analytics* [11] is increasingly used. Our activities fit in the domain of software analytics, with the specific goal to help agile teams understand and improve their processes. We want to help them uncover valuable information, communicate about them and finally document and share the gained insights.

In order to apply the concept of data-driven continuous improvement in the context of the software company, we went through the following process:

1) We applied the Goal-Question-Metrics [12] method to define a specific goal for analysis, to derive a number of related and concrete questions and to define what measurements can be made to answer these questions. The goal selected for the first iteration was to assess how the pace of delivery had evolved over time.
2) We got access to the software repositories of the com-

pany: the version management system, the issue tracker, the agile planning tool and the code review platform. We used our own software analytics platform to extract data from these repositories, to feed a unified model and to create a collection of data visulizations.

3) We then organized a workshop to engage the team in a discussion about the evolution of the software and of the organization. To support the discussion, we printed some of our visualizations on large paper posters (see Figure 3 for an example that shows team activity across repositories and the frequency of releases). In addition, we projected interactive visualizations on the presentation display. During the workshop, participants walked in the room, looked at the data presented in the visualizations and naturally engaged in small group discussions. We observed these participants and took notes to keep a record of discussion points.
4) We integrated the factual insights gained during the data analysis and the perspective captured during the workshop in an interactive report. Our software analytics platform makes it possible to produce interactive stories, which combine narratives and data visualizations. These stories are always written for a specific audience in mind. A first example would be someone in the team might want to write a story to make a point about the value of test-driven development and target new developers joining the team. A second example would be the CTO writing a story to support his interactions with the management team. A third example would the team writing a story to celebrate the successful adoption of a new practice and the evidence of its positive outcome on quality or lead time.

From this experiment, we have learned a number of interesting things. A key learning was the impact of the data visualizations and of the medium we used during the workshop. Somehow, the posters on the wall played a key role to engage the participants. We strongly believe that the impact of a slide deck would not have been the same. Seeing what had happened over the last ten years in a visual form even had an emotional aspect on the participants, which was the initial trigger for many discussions. Another learning was a confirmation that the storytelling format offers and effective way to communicate a message and thereby is a valuable complement to a dashboard. The two interfaces serve different purposes, but they can be build on top of the same software analytics platform. With the appropriate user interface design, this makes it possible to seamlessly transition between explanatory and monitoring phases.

VI. The Implementation of a Continuous Data-Driven Process

The implementation of a continuous data-driven process is always done in a context that is specific to an organization. Nevertheless, its overall structure and the questions that need to be considered are recurring. Figure 4 provides a visual representation for the process that we propose. Figure 5 places

Fig. 4. Data-driven continuous improvement process

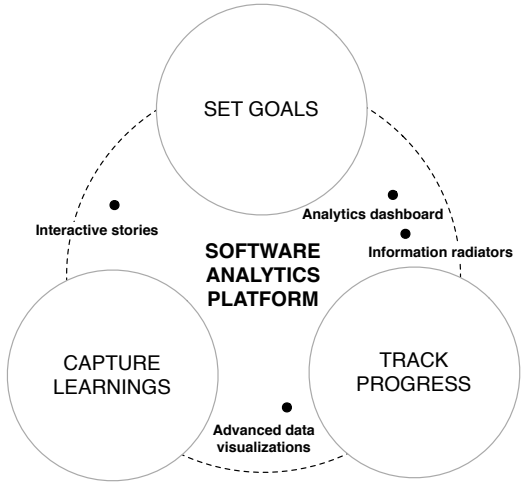

Fig. 5. Feedback channels in the context of the process

the feedback channels described earlier in the context of the process:

1) The process starts at the end of an iteration, typically during a retrospective meeting. The team follows the Goal-Question-Metrics method [12] to define an improvement *goal* for the next iteration, to derive concrete *questions* from this goal and to identify the *metrics* needed to answer these questions in quantitative ways.
2) Before the iteration starts, the team has to make sure that the raw metrics can be collected during the iteration. The means to collect the data can be a mix of automated and manual techniques. For instance, scripts can be written to extract data from the bug management system. Also, surveys can be prepared to gather other metrics from the team. Before the iteration starts, the team must also decide how the raw metrics should be processed and made visible to the team. This typically involves the

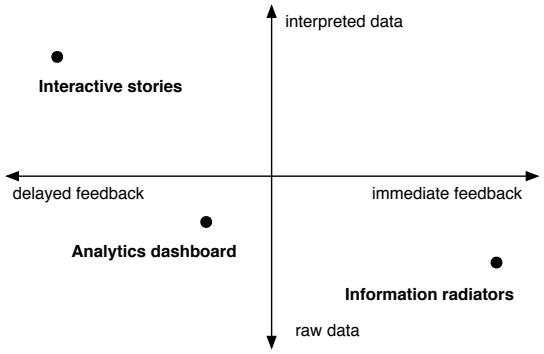

Fig. 6. Multiple feedback channels

configuration of information radiators and of analytics widgets in the team dashboard.

3) During the iteration, the team pursues its activities and benefits from the feedback channels. The information radiators give them a continuous sense of their progress. If things go well, they should get a feeling of mastery and be motivated to keep going with their efforts. At times, the team can also use the team dashboard to get more concrete information.
4) At the end of the iteration, the team should be provided with more advanced visualizations to analyze what has happened in details. This is when the team starts to assess the impact of the process changes. We argue that the tools that present interactive visualizations should have a strong social dimension: they should trigger discussions and engage the team in a collaborative introspection exercise. This can happen *synchronously* (the workshop in the case study, where we used the poster, is a good example), or *asynchronously*. For instance, think about a tool where users can annotate visualizations and use them as a context for conversation [13].
5) Finally, from time to time, the team should take the time to document its learnings and its progress. This information is valuable within the organization (e.g. to facilitate conversations between technical and business stakeholders, to share experience between teams, to onboard people, etc.) and outside. This is where authoring tools can be used to create interactive stories to deliver a specific message to a target audience, with a combination of data, visualizations and narratives.

VII. Conclusion

Agile teams thrive when they are given *autonomy*, when they have a feeling of *mastery* and when they have a sense of *purpose*. These three dimensions need to be considered when introducing a data-driven continuous improvement process in the organization. If the process is imposed to the team, then it is likely to be detrimental. Collecting metrics and presenting them in a dashboard designed primarily for outsiders will be perceived negatively, because it will be seen as a control mechanism that reduces *autonomy*. In the contrary, if the ownership of the same infrastructure is given to the team, then it becomes a valuable agile tool. Because transparency is a core agile value, we claim that many teams will then be happy to share the information with external stakeholders. When a platform is introduced to track software and process metrics, it should make it possible for the team to define goals and to track progress towards these goals. This is a key ingredient to get a sense of *mastery*. We have explained how ambient interfaces, seamlessly embedded in the environment, have proven to be very effective to maintain awareness about this progress. A certain level of fun and playfulness should also be seen positively, because of its impact on engagement. Finally, measurement initiatives should be ambitious in their goals and not be limited to the presentation of low-level metrics. The impact of these metrics on broader issues should be communicated with the target audiences, so that a *sense of purpose* can be conveyed. This cannot be done with basic dashboards, which need to be complemented with narratives and rich visualizations. We thus believe that a continuous data-driven process requires a combination of feedback channels. Figure 6 shows that the feedback can vary from raw metrics to sophisticated interpretations, as well as in terms of latency.

Acknowledgment

The authors would like to thank the teams mentioned in the case studies. They have contributed to our understanding of the problems faced by agile teams and have helped us frame and evaluate the solutions presented in the paper.

References

[1] D. H. Pink, *Drive: The surprising truth about what motivates us*. Penguin, 2011.
[2] O. Liechti, "Awareness and the www: an overview," *ACM SIGGROUP Bulletin*, vol. 21, no. 3, pp. 3–12, 2000.
[3] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. ACM, 1992, pp. 107–114.
[4] C. Wisneski, H. Ishii, A. Dahley, M. Gorbet, S. Brave, B. Ullmer, and P. Yarin, "Ambient displays: Turning architectural space into an interface between people and digital information," in *International Workshop on Cooperative Buildings*. Springer, 1998, pp. 22–32.
[5] A. Cockburn, *Agile Software Development Agile Software Development*. Addison-Wesley Professional, 2001.
[6] O. Liechti, J. Pasquier, L. Prévost, and P. Gremaud, "The wot as an awareness booster in agile development workspaces," in *International Conference on Web Engineering*. Springer, 2016, pp. 598–602.
[7] O. Liechti, J. Pasquier, and R. Reis, "Supporting agile teams with a test analytics platform: a case study," in *12th IEEE/ACM Workshop on Automation of Software Test (submitted)*, 2017.
[8] E. Segel and J. Heer, "Narrative visualization: Telling stories with data," *IEEE transactions on visualization and computer graphics*, vol. 16, no. 6, pp. 1139–1148, 2010.
[9] J. Heer and B. Shneiderman, "Interactive dynamics for visual analysis," *Queue*, vol. 10, no. 2, p. 30, 2012.
[10] J. Hullman and N. Diakopoulos, "Visualization rhetoric: Framing effects in narrative visualization," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2231–2240, 2011.
[11] T. Menzies and T. Zimmermann, "Software analytics: so what?" *IEEE Software*, vol. 30, no. 4, pp. 31–37, 2013.
[12] V. R. Basili, "Software modeling and measurement: the goal/question/metric paradigm," 1992.
[13] F. B. Viegas, M. Wattenberg, F. Van Ham, J. Kriss, and M. McKeon, "Manyeyes: a site for visualization at internet scale," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 6, 2007.