AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
IT Design and Application Development
Selma Lagerlöfs Vej 300
DK-9220 Aalborg East
http://www.cs.aau.dk/

PROJECT INFORMATION

Title
Developing a Software Metrics Dashboard Supporting Multi-Team Agile Software Development

Project Period
Spring Semester 2017

Project Group
is1021f17

Participant(s)
Ivo Hendriks, Dan Meakin & Frederik Frode Nygart

Supervisor(s)
Peter Axel Nielsen

Page Count
25

Date of Completion
June 7, 2017

ABSTRACT

Increasing numbers of software development organisations are using agile methodologies, due to their perceived benefits when compared with more traditional software development methods. Managing the use of such methods within multiple team environments can be challenging. In this paper, we present the results of a study into the usefulness of software metrics within a multiple team agile software development, and their usefulness in aiding such a development when presented within a dashboard. Based on Van de Ven's principles of Engaged Scholarship and Hevner's framework for Design Science in IS, we undertook an iterative prototype development and evaluation study. The evaluation was undertaken with agile practitioners at one Danish software development organisation. The analysis of our evaluation sessions yielded fourteen findings, each of which related to the effectiveness of the dashboards and metrics contained within the prototype. We discussed these findings in relation to existing literature on metrics, dashboards and agile software development, identifying key metrics and design considerations for dashboards in multi-team agile development. We also contribute the artifact instantiation resulting from our design process, namely our final prototype. Finally, suggestions for possible future work in further development and evaluation of the prototype are presented.

Summary

In this paper, we explored the usefulness of software metrics and dashboards incorporating them within agile software developments involving multiple agile teams ('multi-team agile development'). Increasing numbers of software development organisations are using agile methodologies due to their perceived benefits compared with more traditional software development methods. In a previous case study on this topic, we found that managing the use of such methods within multi-team agile development can be challenging. In this paper, we present the results of a study into which software metrics are useful within a multi-team agile development, and how these can be combined in a dashboard to aid such a development.

Following Van de Ven's principles of Engaged Scholarship, which encourages researchers to cooperate with practitioners and conduct research that is relevant to practice, we conducted this study with the involvement of a number of agile practitioners. All of them were engaged in agile software development at a software development organisation in Denmark. To guide our research process, we used the Information Systems Research Framework of Hevner et al. Starting from a business need or problem, this this framework supports finding answers to that problem through interactively designing, building and evaluation of a design artifact.

Three build-and-evaluate loops were completed in the duration of this study. Each of those had a specific goal: understanding which tools may support multi-team agile development, and the ability of a metrics dashboard tool to provide this support; understanding which metrics and visualisations are of use in supporting multi-team agile development; and understanding which combinations and juxtapositions of metrics and visualisations are useful. In order to gain the required understanding, during each iteration one or more prototypes were created. These were evaluated with practitioners.

We used two methods to evaluate our prototypes: semi-structured interviews in the first two iterations; and usefulness testing for the final iteration. This method is similar to usability testing insofar that users are asked to perform certain preconceived tasks with the prototype system, but instead of uncovering interface problems, this aimed at gaining insight into the usefulness of the system: to what extent can the prototype's functionality assist the interviewee in accomplishing a particular task.

The prototype created during our third and final iteration contained the metrics that had been found most useful in previous sessions. These were combined on four thematic dashboards: Product Tracking; Product Quality; Agile Maturity; and Development Health. Each dashboard contained information on two levels. On the project level a single key visualised metric conveying information about the product or project was displayed, together with a number of traffic light indicators illustrating the performance or status of individual teams. These indicators provided access to the team level, on which a number of metrics relating to individual teams were visualised. A top level dashboard containing a high-level indicator for each of the themes was created to unify

the thematic dashboards.

Analysis of our evaluation sessions uncovered fourteen findings relating to the prototype dashboards and the elements contained in them. We discuss these findings in relation to existing literature on metrics, dashboards and agile software development. Our study confirmed the usefulness of a number of existing standard agile software metrics, and the applicability of a standard classification scheme to some agile software metrics, and it identified a number of new agile metrics. We confirmed a number of reasons to use agile metrics, and identified one new reason. We identified a number of desirable attributes possessed by a useful dashboard aiding multi-team agile development, and discovered that the information needs of individuals within agile software development do not vary between roles represented in our study. A final contribution is the artifact instantiation resulting from our design process, namely our final prototype, encapsulating the knowledge gained during the course of this study.

Future research may involve the evaluation of our prototype by researchers or practitioners within a different organisations undertaking multi-team agile development. It may include the development of the prototype to work with real data from such a development. Additionally, further research may be undertaken into the 'new' metrics identified within this study.

Developing a Software Metrics Dashboard Supporting Multi-Team Agile Software Development

Ivo Hendriks, Dan Meakin & Frederik Frode Nygart

7th June 2017

Abstract

Increasing numbers of software development organisations are using agile methodologies, due to their perceived benefits when compared with more traditional software development methods. Managing the use of such methods within multiple team environments can be challenging. In this paper, we present the results of a study into the usefulness of software metrics within a multiple team agile software development, and their usefulness in aiding such a development when presented within a dashboard. Based on Van de Ven's principles of Engaged Scholarship and Hevner's framework for Design Science in IS, we undertook an iterative prototype development and evaluation study. The evaluation was undertaken with agile practitioners at one Danish software development organisation. The analysis of our evaluation sessions yielded fourteen findings, each of which related to the effectiveness of the dashboards and metrics contained within the prototype. We discussed these findings in relation to existing literature on metrics, dashboards and agile software development, identifying key metrics and design considerations for dashboards in multi-team agile development. We also contribute the artifact instantiation resulting from our design process, namely our final prototype. Finally, suggestions for possible future work in further development and evaluation of the prototype are presented.

1 Introduction

In recent years, the number of companies using agile software development methodologies has been steadily increasing. In 2016, a majority of software development companies indicated that they used such methodologies (Sheehan, 2016, p. 4). Implementation of agile methodologies can be challenging in large organisations where there are a greater number of dependencies between teams, requiring formal coordination between them (Dikert et al., 2016; Waardenburg and Vliet, 2013).

We previously conducted a case study into the challenges that arise within the process of transformation from using traditional software development methods to using agile software development methods within a large organisation (Hendriks et al., 2017). Amongst other things, we found that the transformation process was highly challenging and required substantial method tailoring (Hendriks et al., 2017, pp. 13–14); that there was often tension between existing practices and newly introduced agile practices (Hendriks et al., 2017, p. 14); and the purpose and value of the agile approach was not always well communicated (Hendriks et al., 2017, pp. 14–16).

The challenges in our previous study can be related to two themes: managing and working within a software development spanning multiple teams and the availability of information to support this. Where multiple teams work together on a single project, coordination of these teams and information on their software development activities is required. It is crucial that different stakeholders in the development process have such information, and an effective means of providing this to stakeholders is through the use of software metrics.

Much has been written on software metrics over several decades (see e.g. Fenton, 1991; Grady, 1992; Kerzner, 2013). Software metrics are an essential part of a good software development process (Fenton and Bieman, 2014, p. 3). Agile practitioners regularly use a small number of metrics such as burndown, velocity and build status (Kupiainen et al., 2015, p. 144). Despite the importance of software metrics, comparatively little has been written about their use within agile software development, beyond those previously mentioned.

We wished to examine which software metrics might be valuable within the management of an agile software development involving several agile teams (hereafter, a 'multi-team agile development'), and how metrics could be used to address some of the challenges faced within such a development. Our aim was to develop a software prototype to allow us to examine the usefulness of particular metrics, and to display these within a dashboard to provide more comprehensive information on the software development, and particular aspects of it.

Based on the foregoing, we sought to answer the following research question:

> Which software metrics are useful within a multi-team agile development and how can they be usefully combined in a dashboard to aid such a development?

This paper is structured as follows. In Section 2, we provide an overview of relevant research in agile software development, software metrics, and their visualisation. In Section 3, we describe our research method for this study. In Section 4, we provide a full description of the prototype we develop in this exercise. In Section 5, we analyse our findings from our prototype evaluation sessions with interviewees. In Section 6, we discuss the implications of our findings. In Section 7, we conclude this paper with a brief summary of our results and proposals for future research.

2 Related research

In this section, we provide an overview of relevant research relating to agile software development, software metrics, and dashboards. Firstly, we describe some of the key concepts within agile software development and its use in developments involving multiple teams. Then, we explain software metrics and their use within agile software development. Finally, we describe dashboards and key concepts relating to them.

2.1 Agile software development

Agile software development is a model of software development based upon the notion of 'agility'. Agility has been defined as:

> the continual readiness of a [...] [software development] method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment (Conboy, 2009, p. 340).

For a practice to be agile it must reflect the approach to change within the definition of agility; must not reduce perceived customer value; and must be usable quickly and inexpensively (Conboy, 2009, p. 341).

Several agile software development methodologies have been developed, notably Extreme Programming (XP) (Beck and Andres, 2004) and Scrum (Schwaber and Beedle, 2002). Both place an emphasis on iterative development in short cycles to permit feedback and change to the software (Beck and Andres, 2004; Schwaber and Beedle, 2002, p. 50). They also envisage development taking place within small, self-organising teams. Notably, there is a focus on oral, informal communication and working software code rather than extensive written documentation (Beck and Andres, 2004; Sommerville, 2016).

Agile software development has generally been considered a model best suited to single team software development projects (Boehm and Turner, 2005, p. 28). Despite this, in recent years companies have been increasingly adopting agile development methodologies, which will involve their use in larger developments with multiple teams (Hendriks et al., 2017, pp. 2–3; Cockburn and Highsmith, 2001, p. 133). Changing from traditional to agile software development methods is often challenging for these organisations (Sommerville, 2016). Challenges related to method tailoring, and learning and understanding of agile values, and the existence of new roles in agile software development (Dikert et al., 2016, p. 95; Hendriks et al., 2017, p. 8).

The role of the Project Manager requires particular consideration within agile software development (Hendriks et al., 2017). In Scrum, for example, self-organising teams are responsible for managing the effort required to develop software as described by backlog tasks, with prioritisation of these tasks exclusively determined by Product Owners (Schwaber and Beedle, 2002, pp. 8,19). The role of the Project Manager is not addressed in Scrum or XP, the two dominant agile methodologies. Project Managers within agile development have a facilitative rather than directive role, focusing on removing impediments to effective development (Waardenburg and Vliet, 2013, p. 1261; Hendriks et al., 2017, p. 15).

The extent to which an agile team is experienced in practice use can be described using the framework of Wang et al. (2012). The framework provides a basis for understanding experience in practice use. This framework includes six stages, ranging from initiation—where a match between agile practice and its application environment is identified, and infusion—representing deeply customised use (Wang et al., 2012).

2.2 Software metrics

Software metrics is a term which refers to the activities involved in measurement of some aspect of software, and its development process (Fenton and Neil, 1999, p. 149; Fenton and Bieman, 2014, p. 17). Measurement involves the use of numbers or symbols to describe attributes of entities in the real world according to clearly defined rules (Fenton, 1994, p. 199).

Fenton and Bieman (2014) provide a framework for understanding and developing metrics to inform on the development process. The authors describe a number of 'entities' together with 'attributes' which may be measured (Fenton and Bieman, 2014, pp. 87–99). Effective measurement requires an empirical relation between entities based on particular attributes, and this empirical relation must be capable of being mapped to a numerical relation (Fenton, 1994, p. 200). An appropriate scale must selected for a particular metric, with implications for the type of analysis which can be undertaken based on this selection (Fenton, 1994, p. 201).

Different roles in the software development process use metrics for different purposes. Metrics are used by management to gain information on process, cost, productivity, quality and customer satisfaction, while developers use metrics to track requirements, defects and delivery goals (Fenton

and Bieman, 2014, p. 16).

Reflecting the distinctive approach of agile software development (when compared to traditional, plan-driven approaches), metrics use in agile software development differs from that in traditional software development. Many of the entities proposed by Fenton and Bieman (2014)—such as requirements, designs and associated entities—are clearly a poor fit to the interleaved, iterative approach of agile development (Sommerville, 2016).

Software metrics in agile development should align with the goals of agile software development (Lew, 2016, p. 53). Good agile metrics will affirm and reinforce agile principles (Hartmann and Dymond, 2006, pp. 1–2). People are almost never measured; metrics instead target the product under development and the development process. Measurement effort is directed mostly towards implementation, testing, and the whole development cycle, with requirements engineering, specification and design seldom measured (Kupiainen et al., 2015, p. 157).

Agile teams commonly use a variety of metrics, including those such as defect counts and customer satisfaction which may be equally applicable to plan-driven software development (Kupiainen et al., 2015, p. 158). Velocity, for example, presents a way to measure productivity and predict the quantity of work which can be completed in the current iteration. It is used for the prioritisation of content for a given iteration, and in planning releases involving development occurring over multiple iterations (Bumbary, 2016). Other metrics used in agile development include remaining effort, project completion rate, effort spent on rework versus new tasks, earned business value and total effort estimation (Javdani et al., 2012, pp. 128-130). There are several reasons why agile teams use metrics: sprint and project planning and progress tracking; understanding and improving quality; fixing software process problems; and motivating people (Kupiainen et al., 2015, p. 157).

2.3 Dashboards

A dashboard is a data driven tool designed to support the making of decisions (Yigitbasioglu and Velcu, 2012, p. 42). Information is presented using symbolic representations of the physical reality. Concretely, the most important information is displayed on a single-screen display (Yigitbasioglu and Velcu, 2012, p. 44). Decisions are made based on the functionality of the system, the environment in which the decision is being made, and the problem solving skills of the decision maker (Yigitbasioglu and Velcu, 2012, p. 43). The user should be able to explore areas requiring corrective action, with decisions made in order to achieve some individual or organisational objective (Yigitbasioglu and Velcu, 2012, p. 44).

Interactivity is a key part of the dashboard. Features such as 'drill-down' and 'drill-up' functionality and a flexibility in presentation format are useful in allowing the user to obtain further detail on particular information (Yigitbasioglu and Velcu, 2012, p. 48). However, excessive functionality is undesirable as this can distract the user (Tokola et al., 2016, p. 620). It is important to consider the appropriate level of functionality and detail such that there is no risk of 'information overload' (Yigitbasioglu and Velcu, 2012, p. 48).

Besides the amount of information represented on the dashboard, other characteristics influence whether a dashboard fulfils its purpose. It is recommended to use repetitive and uniform patterns to reduce visual complexity; bright colours should be used sparingly, except where they are required to direct attention to an element that needs immediate attention (Yigitbasioglu and Velcu, 2012, pp. 45–46,52). Complex data on dashboards is usually presented through tables and graphs. The former are preferable when the goal is to extract specific values from a dataset, whereas graphs are a better fit for tasks that require identifying and understanding of relationships (Yigitbasioglu and Velcu, 2012, p. 49). For displaying limited amounts of data, metaphorical indicators can be used. Single numbers can be appropriately visualised through gauges and meters, whilst traffic lights are an immediately recognisable option when the information can be condensed to three states (Staron et al., 2014, pp. 215–216).

The condensed nature of information presented on a dashboard makes it a useful instrument for managers for whom the information presented is crucial, as well as more casual users, who find dashboards to conform to the way they want to want to consume information (Eckerson, 2010, pp. 76–77). This does not mean, however, that different organisational roles benefit from using the same dashboard. Eckerson (2010, pp. 100-105) distinguishes three types of dashboards: operational, used to control operations; tactical, for optimising processes; and strategic, aimed at managing strategy. These dashboards are primarily used by operations staff, management and analysts and executives respectively. The interest of these roles is not statically tied to one dashboard though: middle management for example also use operational and strategic dashboards when managing 'up' or 'down' in the organisational hierarchy (Eckerson, 2010, p. 103).

3 Research Method

We adopted prototype design and evaluation as our research method for this study. Our approach is rooted in literature concerning the relationship between academia and practice—'engaged scholarship'—representing a collaboration *with* practitioners, rather than trying to produce knowledge that's useful *for* them (Van de Ven, 2007, pp. 7–10). Four forms of engaged scholarship are identified, with prototype design and evaluation falling within the two 'to design/control' quadrants (Van de Ven, 2007, pp. 26–27). The aims of this study lie on the border between between design and evaluation research (also referred to as design science research) and action/intervention research. Though our goals and interests matched those of action research insofar as we are working with one organisation, the lack of

opportunities for intervention—a concept crucial to action research—pointed towards design science research.

Hevner et al. (2004) provide a framework for design science research within Information Systems (IS). Design science in the IS discipline takes a business need or problem as a starting point (Hevner et al., 2004). Solving this problem requires design, a concept encapsulating both the design process and the artifact resulting from that process (Hevner et al., 2004, p. 78). A successful artifact instantiation demonstrates feasibility of both the design process and the designed product itself (Hevner et al., 2004, p. 84).

The process of designing a successful artifact involves a two-phase approach in which two research paradigms are used side by side: behavioural science is used to ensure *rigour* and *truth*, supporting the development and justification of theories that help explain phenomena related to the identified business need (Hevner et al., 2004, p. 79). Design science, on the other hand, provides *relevance* and *utility*, through the creation and evaluation of artifacts designed to meet the identified business need (Hevner et al., 2004, pp. 79–80). These two phases are executed iteratively in a build-and-evaluate loop—a sequence in which an informed artifact is produced, evaluated and improved upon (Hevner et al., 2004, p. 76).

3.1 Research design

Our study seeks to understand the usefulness of software metrics within multi-team agile software development, and to understand how they can be combined within a dashboard. We wished to investigate this through iterative prototype design and evaluation.

The build-and-evaluate loop required in the creation of a successful artifact is manifest in the iterations of this study. Our research process involved three iterations. For each iteration, we set a particular goal aimed at improving our understanding of a potential solution to the problem underlying our research. We developed one or more prototypes during each iteration, each of which was evaluated with a number of practitioners. The three iterations and their characteristics are outlined in Table 1. Each successive iteration built on the knowledge obtained in the previous, and was grounded in the evaluation of a prototype to obtain the required knowledge. Three iterations were undertaken as this allowed us to examine one specific aspect of our subject-matter: general concepts; specific metrics; and agile metrics dashboards.

We employed two different evaluation methods: interview, and usefulness testing. For the sake of consistency, throughout this paper we refer to those individuals participating in evaluation sessions as 'interviewees'. In the case of interview, we undertook semi-structured interviews with a number of potential metrics users to understand either something about the usefulness of metrics. During interviews, we displayed prototypes to the interviewee and asked for feedback on what was being displayed. The interviewee did not interact with these prototypes during interview.

Usefulness testing is how we term the evaluations we conducted where an interviewee interacts with a prototype to determine if it is 'capable of being used advantageously' (F. D. Davis, 1989, p. 320). It is similar to usability testing, which:

> involves representative users attempting representative tasks in representative environments, on early prototypes of computer interfaces (Lazar et al., 2010, p. 252).

However, we were interested in the usefulness of the prototype—to what extent could the prototype's functionality assists the interviewee in accomplishing a particular task—rather than questions of interaction with the interface. To address possible issues concerning interaction with the prototype, we provided advance guidance as to navigation and usage of the prototype and asked interviewees to say if they had difficulties in this regard. Each interviewee was given an identical small number of tasks to complete. As a dashboard requires the user's interpretation to be informative and useful, there was no right or wrong way to solve these tasks. This meant that rather than gathering quantitative measurements such as task completion and number of errors, our experiment focused on acquiring qualitative data—most importantly the reasoning and thoughts behind the considerations of the user in solving the task.

Qualitative methods are often criticised for a perceived lack of rigour. Three factors determine the rigour of qualitative usefulness testing: credibility, that is whether the test measures usefulness as perceived by the user; transferability, that is if the test environment mirrors its intended production environment; and dependability, how reliable test results are (Hughes, 1999, p. 493). For the testing in this study, credibility was provided through the use of a think-aloud protocol, which 'may be the single most valuable usability engineering method' (Hughes, 1999, p. 493). Transferability was enhanced by testing the prototypes with actual prospective users, in a familiar (work) environment, using hardware comparable to the hardware used within the organisation. We increased dependability by alternating between us who led the session, as suggested by Hughes (1999).

3.2 Prototype designs

A prototype is a simplified version of a particular system created to help evaluate that system (Mathiassen et al., 2000, p. 33). The requirements of each prototype were based on what we wished to evaluate in a particular iteration. Specific characteristics were then further specified using the 'fundamental prototyping principle' (Lim et al., 2008). The goal of prototyping according to this principle is

> finding the manifestation that in its simplest form, filters the qualities in which designers are interested, without distorting the understanding of the whole (Lim et al., 2008, p. 2).

Lim et al. (2008) describe prototypes in terms of his 'anatomy of prototypes'. The anatomy of prototypes contains

Table 1: Research iterations.

	Iteration 1	**Iteration 2**	**Iteration 3**
Description	Exploration of general metrics/-dashboards concepts	Development & testing of metrics & visualisations	Development & testing of a dashboard
Goal	Understand which tools may support multi-team agile development; the ability of a metrics dashboard tool to provide this support.	Understand which metrics are of use in supporting an agile transformation; understand which visualisations are useful in illustrating these metrics.	Understand how a dashboard combining useful metrics can aid in multi-team agile development.
Evaluation method	Interview	Interview	Usefulness testing
Result	Identification of a metrics dashboard as a concept to develop in the next iteration.	Identification of a subset of useful metrics to take forward into development of a dashboard.	A dashboard useful within multi-team agile development.

Table 2: Prototype dimensions. After Lim et al. (2008), A. M. Davis (1992) and A. M. Davis (1995).

Dimension	**Prototypes 1a, 1b & 1c (Iteration 1)**	**Prototype 2 (Iteration 2)**	**Prototype 3 (Iteration 3)**
		Filtering dimensions	
Appearance	Diverging, uninformed and ad-hoc representation of possible product	Detailed representation of metrics, minimal representation of interface elements	Detailed representation of dashboards minimal representation of interface elements
Data	Absent or hard-coded	Limited data store to allow for population of visualisations	Realistic data model populated with test data
Functionality	Viewing and limited modification of data to illustrate intended functionality	Viewing, modification and switching between individual metrics	Viewing, modification and switching between related groups of metrics and indicators
Interactivity	Linear walkthrough	Functional user interface—as far as data and functionality constraints permitted	Fully functional user interface—as far as data and functionality constraints permitted
Spatial structure	Diverging, but complete representation of possible product	Non-hierarchic, non-relational presentation of metrics	Combined representation of related metrics and indicators
		Manifestation dimensions	
Material	Computerised drawings & HTML web pages	Javascript web application using React & Redux libraries	Javascript web application using React & Redux libraries
Resolution	Overall limited representation of possible system	Performance and representation of individual metrics resembling intended product, limited functionality and interface	Performance, architecture and functionality resembling intended product, limited interface
Scope	Shallow representation of whole system functionality	Full representation of limited functionality	Full representation of full system functionality

two dimensions: filtering, corresponding to which aspects of design idea will be represented in a prototype; and manifestation, aspects of a design idea the designer must consider in the exploration and refinement of the design (Lim et al., 2008, p. 11). The authors distinguish five filtering dimensions: appearance, data, functionality, interactivity and spatial structure—the arrangement of and relation between interface elements (Lim et al., 2008, p. 11). Three manifestation dimensions are described: material, or medium; resolution, or fidelity, the level of detail of what is manifest; and scope, the range of what is manifest (Lim et al., 2008, p. 11). Table 2 fully describes each of our prototypes in terms of these dimensions. The reader should refer to both this table, and Table 1 to understand the nature of each prototype, the purpose for developing each and the outcome of so doing.

Prototypes 1a, 1b & 1c The first prototypes were developed during the first iteration. Three prototypes were developed for this iteration, each representing a different type of tool which may support a multi-team agile development. We evaluated all of these prototypes and then focused our evaluation on the metrics dashboard prototype. This permitted a focus on the concept of a metrics tool and acquire a general understanding of our interviewees' interest in software metrics.

Prototype 2 The second prototype focused on examining software metrics as they may be used at DanFin. A new prototype was developed containing a number of different metrics and visualisations of these. This allowed us to examine the usefulness of these metrics and visualisations with interviewees. It also provided a specific understanding both of the metrics in use at DanFin, and those which may be useful there, together with an understanding of how they may be effectively visualised.

Prototype 3 The final prototype focused on placing the most useful metrics into context within a dashboard. The previous prototype was developed further, reusing the metrics where they were desired for further evaluation, creating dashboards containing these metrics. The result of our prototype development is described at Section 4. This prototype provided us with knowledge on useful dashboards and the metrics contained therein.

3.3 Data collection

This study took place during March, April and May 2017, with practitioners involved in the agile software development process at a large financial institution in Denmark, DanFin (not its real name). Data were collected at a series of evaluation sessions. Information on participants is in Table 4. Each evaluation session was conducted face-to-face, with the exception of one session which took place over Skype, and took between 45 and 75 minutes. Audio was recorded, and contemporaneous notes were taken at each session. Following each session, the audio was transcribed.

The evaluation sessions involved either interview or usefulness testing, as described above. During usefulness testing, interviewees were asked to complete the following tasks:

1. Evaluate the quality of the product under development.
2. How is development of product progressing? And how are different teams contributing?
3. Provide an agile maturity assessment for this project, and for team Beta.
4. Make an assessment of morale for both the whole project, and for team Lambda.
5. Assess the product and its development process. Where do you see challenges or success stories? Which metrics would you action?

3.4 Data analysis

For the evaluation sessions in the first and third iterations, each transcript was coded. This involved identifying researcher-denoted concepts or themes arising from the prototype evaluation (Lazar et al., 2010, p. 291). Comments relating to a particular topic were then grouped. Analysis was undertaken within each iteration. The results of each analysis session were used to inform decisions on what should be examined in the iteration to follow.

In the first iteration, we coded comments relating to the general usefulness of metrics or dashboards, and other general comments relating to the subject-matter of the prototypes. In the second iteration, the transcript was broken into sections, each relating to one metric. The text of each section was summarised and tabulated. In the third iteration, we coded comments concerning the usefulness of metrics and dashboards, and those relating to the ability of the interviewee to complete the tasks set for them. The latter was central to our usefulness evaluation.

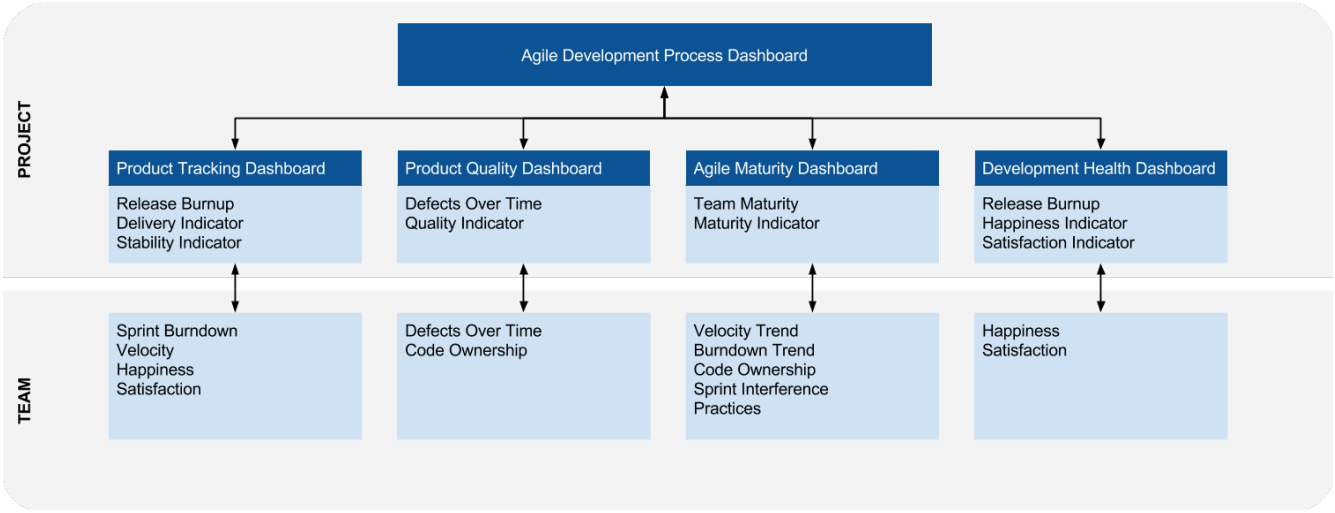
4 Prototype Description

Within this section, we describe the final prototype resulting from development over a series of iterations as explained in Section 3. The prototype can be accessed at https://goo.gl/931E3D; its source code at https://goo.gl/PGv1N5. Screenshots from the prototype can be found in Appendix A.

4.1 Prototype structure

The prototype takes the form of a top level dashboard containing four 'thematic' dashboards, that is dashboards providing information relating to a particular aspect of agile software development. Fig. 1 illustrates the prototype's structure. The top level and thematic dashboards are individually described in Section 4.2. The themes were derived

Figure 1: Flowchart.

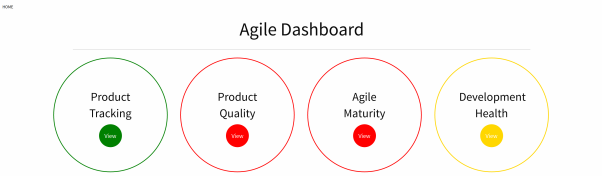


from an understanding of DanFin's approach to agile software development, from the metrics confirmed to be of value to interviewees in earlier iterations, and from knowledge of agile software development generally. Each thematic dashboard provides a project level overview, and a team level overview. These are described in more detail, below.

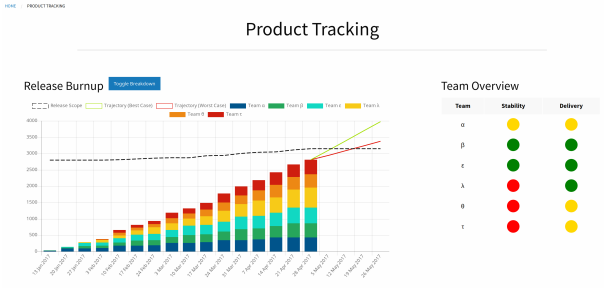
The metrics used within this prototype are listed in Table 3.

4.2 Dashboards

Top level dashboard This dashboard, illustrated below, informs users about how well the agile development process is doing by the use of four indicator elements. It also serves as navigation layer where each sub-dashboard can be drilled-down into to get further details about the different areas. Indicators summarise the information available within each thematic dashboard.



Product Tracking This dashboard, illustrated below, provides information about development progress focused upon scheduled and anticipated delivery of a product release.



This dashboard contains the following metrics:

- Release Burnup;
- Sprint Burndown;
- Velocity;
- Happiness; and
- Satisfaction.

Product Quality This dashboard provides information about the quality of the software being developed. It is focused upon measuring defects, both at present and over time. This dashboard contains the following metrics:

- Defects Over Time; and
- Code Ownership.

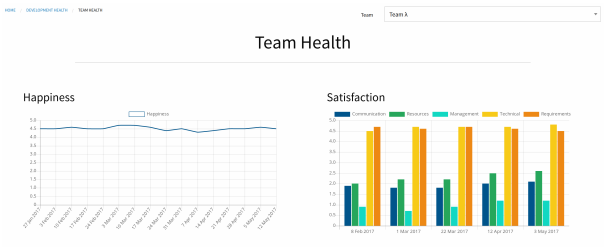
Agile Maturity This dashboard provides information about the maturity in agile methodology and practice use of a project and the teams working within it. It contains the following metrics:

- Team Maturity;
- Velocity Trend;
- Burndown Trend;
- Code Ownership;
- Sprint Interference; and
- Practices.

Table 3: List of metrics included within prototype.

Metric	Description
Burndown Trend	Overlays multiple sprint burndown trend lines to provide an overview of a series of sprints for one Scrum team.
Code Ownership	Illustrates the number of different team members making a contribution to different modules within the codebase.
Defects Over Time	Displays cumulative unresolved defects over a particular time period, categorised into criticality levels 1–5.
Happiness	Illustrates a team's self-assessed happiness level on a scale from 1–5.
Practices	Provides an overview of the maturity of a team's adoption of individuals agile practices using a scale adopted from Wang et al. (2012).
Release Burnup	Illustrates progress towards a scheduled release by displaying cumulative completed story points & total release scope, with an option to display per-team contribution.
Satisfaction	Illustrates a team's self-assessed satisfaction with a number of attributes of the development process (e.g. technical, management). Measured using a scale from 1-5.
Sprint Burndown	Displays a Scrum team's per-day completion of story points during one sprint.
Sprint Interference	Illustrates time in half-days spent by a Scrum team on tasks not contained within the sprint backlog, broken down into interference categories (e.g. bug fixing, unscheduled meetings).
Team Maturity	Displays a summarised per-team maturity assessment for all teams within a development.
Velocity	Illustrates the story points a Scrum team committed to complete during a sprint alongside the actual number of completed points.
Velocity Trend	Summarises the velocity of a team over the period of several sprints, including average velocity and completion rate.

Development Health This dashboard is concerned with the 'health' of a project and the morale of the teams involved in it. The team level dashboard is illustrated below.



This dashboard contains the following metrics:

- Happiness; and
- Satisfaction.

4.3 Design choices

The prototype is based on a number of specific design choices made during development. These choices were made based upon knowledge obtained from literature and developed throughout evaluation sessions with interviewees.

4.3.1 Architecture

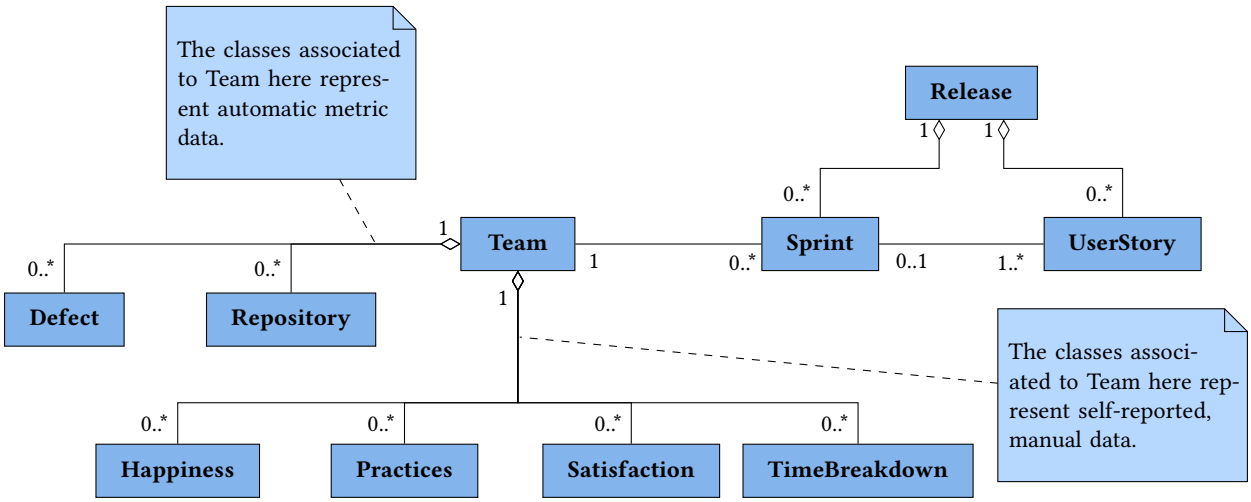
The following architectural features are present within the prototype:

Metric data model The prototype reflects as closely as possible the form and content of software metric data which would be obtained from a real multi-team agile development. It is based upon a data model which realistically represents the entities found in the software development process. This model is illustrated at Fig. 2. Each metric uses a subset of the data contained within the data model described above. The Sprint Burndown and Velocity metrics, for example, both use the same sets of sprints and user stories to illustrate the completion of tasks by teams involved in the development; each metric simply transforms the raw data in the required manner to display that metric.

Test data is located within a state container All test data is contained within a state container. This means that the prototype has one source for all data used in creating dashboards and metrics, which allows the prototype to mimic access to external data: instead of making a call to an external service holding software metrics data, the prototype makes a call to the state container. The use of alternative data within the prototype is straightforward. Substituting new data requires a change to be made in this one location alone.

Modular, dynamic visualisations The prototype is highly modular. Each dashboard and metric is an individual component which takes a dataset in a standardised format, and the component renders an indicator or chart based solely on this data. The metrics are totally decoupled from the data

Figure 2: Prototype data model.



model, allowing them to be reused without alteration. They are designed to easily work with data from the state container.

4.3.2 Functionality

The following functionality is contained within the prototype:

Dashboards display project level & team level information Each thematic dashboard provides a project level view and a team level view. This allows users to view information pertaining to the software development project from two consistent vantage points. At team level, the dashboard provides information on one agile team, with the user able to select different teams at this level. At project level, the dashboard provides information about the development as a whole, or aggregated information about all teams participating in the development.

Partially interactive metrics Metrics within the prototype are partially interactive. Where multiple groups of data are contained within a metric, such as multiple satisfaction criteria with the Satisfaction metric, groups can be hidden and made visible by the user. This permits a limited level of interaction, allowing the user to focus on relevant information.

5 Analysis

In the following section, we describe and analyse the results of our prototype evaluation. Our findings are summarised in Table 6. Table 4 lists the individuals with whom we conducted evaluation sessions, together with the iterations in which these sessions took place. Table 5 describes the different roles listed in Table 4. Where we quote from interviewees, we use a code such as 'SM1:it3' to indicate which interviewee is quoted ('SM1', in this example) and the iteration during which that evaluation session took place (iteration 3, in this example).

Table 4: Interviewees & participation.

Interviewee	Role	Iteration		
		1	2	3
AC	Agile Coach	✓	✓	✓
AL	Agile Leader		✓	✓
PM1	Project Manager		✓	✓
PM2	Project Manager			✓
SM1	Scrum Master	✓		✓
SM2	Scrum Master		✓	
SM3	Scrum Master			✓

5.1 Overview

Evaluation sessions yielded a number of findings in relation to the overall structure of the prototype, to common elements in the prototype, and in relation to the main dashboard.

5.1.1 Dashboards are useful, but the main dashboard required more detail (1)

Dashboards supported the use of metrics at DanFin, providing an easily accessible overview of these:

> I think it's great to have that overview and [...] [to have] that in a central way (AC:it1).

The breakdown between the thematic dashboards was found to be useful by all interviewees. Interviewees commented on the lack of detail on the main dashboard. The

Table 5: Interviewee roles.

Role	Description
Agile Coach	Responsible for training teams on agile and Scrum and making maturity assessments for those teams. Involved in identifying impediments to effective agile software development and discussing these with management.
Agile Leader	Responsible for the composition and general performance of agile teams allocated to them. Involved in hiring, firing & managing team members, but don't engage in day-to-day team operations.
Project Manager	Responsible for delivery and quality of one or more development projects. Also assigned one or more teams for which they served as an Agile Leader.
Scrum Master	Responsible for the Scrum process at team level. Ensure that team members follow Scrum practices, and offer explanation on them where needed. Responsible, with their team, for tailoring the Scrum process to the teams wishes and organisational culture.

red-yellow-green indicators did not convey enough information to provide a useful overview of the information contained within the thematic dashboards. One interviewee commented on this and suggested an improvement:

> My first impression was why don't we have something below to show what is behind[?] […] [W]hen you call it a dashboard I expect information on the dashboard, not just options. […] [Take] some of the central charts and do like a miniature version of that, where I can quickly see if there is something interesting behind. (AC:it3)

Furthermore, it was not clear to the interviewees that the indicators on the main dashboard were, in fact, indicators. None of our interviewees understood until prompted that the main dashboard contained four indicators, each relating to one thematic dashboard.

Thus, the dashboards within the prototype were found to be useful, but the main dashboard lacked sufficient detail.

5.1.2 Red-yellow-green indicators provide an immediate overview, but lack clarity (2)

Red-yellow-green indicators (hereafter, 'indicators') were used within the thematic dashboards to provide an immediate summarised indication to the user of a particular piece of information about the development. These indicators were recognised by one interviewee as being useful to indicate which teams were currently requiring specific attention:

> I really would like if this tool could give me some kind of indication of which teams is it interesting to drill into and have a look at (AC:it1).

Interviewees generally found these indicators useful and effective in providing an immediate overview of an important piece of information. They allowed interviewees to understand quickly where attention may be required due to possible problems in the development process. One interviewee commented that:

> I would assume that the yellow ones and the red ones are the teams to look out for […] So I would go look into [detail on why there is a problem] […] (SM3:it3).

On one thematic dashboard, we embedded arrows within the indicators to explicitly illustrate change trends. These indicators were also found to be useful by the interviewees.

Interviewees commented that it was unclear to them how a judgement was made on whether the indicator was red, yellow or green. Several interviewees noted that they wished to know the time period for the indicator where it was based upon a trend. Specifically, these interviewees wished to know whether the period was when the team was still newly formed, something not clear from the indicator:

> [Is it] looking […] [at] the trend for the last 3 sprints or is just since last sprint[?] […] I would need some information and some knowledge about that (AC:it3).

The time period can influence whether an interviewee would consider a team's performance satisfactory:

> So whenever you, in the first [few sprints] […] you're trying out […] new stuff. And after that then usually you'll be stable after several sprints. […] I wouldn't be disappointed in a team if they are just trying to be agile doing this, because all of the methods are new […] (SM3:it3).

One interviewee expressed the view that the indicator within the product quality dashboard was simply incorrect:

> I would say that, as long as you have criticality 5 issues, you can't be green. Because that's a critical issue and it's— you're not going anywhere with that code. (PM1:it3)

Thus, the indicators provided a useful, quick summary of important information but did not reflect the assessment interviewees would make of the status of the development in every case.

5.2 Development Health

Evaluation sessions yielded a number of findings in relation to the usefulness of the development health dashboard, and the metric visualisations contained within this.

Table 6: Findings.

Category	No.	Description	Section
Overview	1	Dashboards are useful, but the main dashboard required more detail.	5.1.1
	2	Red-yellow-green indicators provide an immediate overview, but lack clarity.	5.1.2
Development Health	3	Development Health dashboard did not provide users with useful information at project level.	5.2.1
	4	Happiness is a key metric in assessing agile development.	5.2.2
	5	Satisfaction is a useful metric, though the Satisfaction criteria were questioned.	5.2.3
	6	The relationship between Happiness & Satisfaction is unclear	5.2.4
Agile Maturity	7	Assessment of Agile Maturity is a team level, not a project level activity.	5.3.1
	8	Practice assimilation levels are an important indicator of maturity.	5.3.2
	9	Sprint Interference is a useful metric, but what constitutes interference is unclear.	5.3.3
	10	Code Ownership's value in assessing agile maturity is low.	5.3.4
Product Tracking	11	Product Tracking dashboard gives a useful overview on the project level, but not on the team level.	5.4.1
	12	Velocity and Sprint Burndown metrics useful together.	5.4.2
Product Quality	13	Product Quality dashboard is useful for assessing quality on project and team levels, but might not give a complete insight into it.	5.5.1
	14	Code Ownership's relation with quality is unclear.	5.5.2

5.2.1 Development health dashboard did not provide users with useful information at project level (3)

Interviewees were unimpressed by the information provided by the development health dashboard at the project level. At this level, information concerning health of individual teams was aggregated in a combined average (mean). Interviewees commented that Average Happiness did not convey useful information about the status of the development itself, or of teams within it.

One interviewee commented:

> If I could filter out different teams that would be good. Because [...] [within the Average Happiness visualisation] I have a nice 2.5 to 3, but I don't know if I have 2 teams which are 4.5, 2 teams which are 1. That makes a nice average but, big trouble (PM1:it3).

Another interviewee was confused by the Average Happiness chart being positioned next to per-team indicators. This interviewee thought that the dashboard would be more useful if the Average Happiness chart was removed entirely: 'I would just remove the Happiness' (SM1:it3).

It was suggested that the notion of development health, rather than team health, was not meaningful as this information is only capable of being considered at the level of the team:

> This Average Happiness is really difficult because [...] I'd be a lot more interested in what's going on in the teams. Because that's where the dynamics are. That's where you can [...] make a difference. (PM1:it3)

Thus, the development health dashboard—and it's project level focus—was not useful to users of the prototype in making an assessment of the health of development. Development health within the prototype was only meaningful in relation to the team, not the project.

5.2.2 Happiness is a key metric in assessing agile development (4)

Interviewees viewed the Happiness metric as very important in assessing the status of an agile development. DanFin used a Happiness metric prior to our study. Happiness, according to one interviewee, is concerned with measuring the opportunity the team gets to work on tasks that they wish to work on. It is useful for discussing which things are going well, and which are going badly at a particular point in time. Its value is in stimulating those discussions:

> I focus on the last sprint: how happy were you with regard to stories, people, [etc.] [...] And then we discuss what it takes to get us higher. And that, I think that makes good sense to people [...] because it's a way of getting to say what's bugging everyone. (SM1:it1)

When making assessments of team health, all interviewees considered the team's happiness. Interviewees were able to easily obtain an understanding of a team's current happiness, as well as trends over a period. One interviewee commented that it was important to be aware of teams' happiness, even if they were successfully delivering software:

> [Y]ou look at the team and team members' well-being [happiness], you have to do that, even though you're reaching deadlines. That has nothing to do with that. You would still have to see how are the team doing[...] (PM2:it3)

Happiness was linked to other aspects of the development process by different interviewees, suggesting that the metric provided an insight into more than whether or not a team and its members were happy. One interviewee connected Happiness to the ability of the team to deliver, noting

that a failure to complete the committed story points for a sprint and Happiness may be linked:

> [S]omething happened in March, I can see: they were not very happy. But actually that was when they [...] didn't complete all the story points they committed. Maybe there's a relation to that[...] (SM3:it3).

Another interviewee suggested that Happiness (together with Satisfaction) had a broad impact on all aspects of development:

> I mean that, if we go in and fix and give some attention to some of the things they are not satisfied with—the technical and the requirements and we target also their work Happiness so that they actually get to a state where they enjoy working—they will most likely increase every parameter on the board actually (AC:it3).

In addition to the thematic dashboards in which it appears, Happiness may be useful in assessing the agile maturity of a team. One interviewee commented that Happiness was important in understanding a team's agile maturity:

> [Within agile maturity] I would like to see the Happiness as well, because here's only about velocity and burndown and sprints and code ownership, and that's okay but if we're very high on that one then we're very unhappy in the team, I would be worried. (SM3:it3)

Happiness was found to be a widely used and useful metric. Interviewees considered it was connected to many different aspects of the development process and did more than inform solely on Happiness.

5.2.3 Satisfaction is a useful metric, though the Satisfaction criteria were questioned (5)

Interviewees expressed positive views about the Satisfaction metric. They liked the idea of teams being able to track specific attributes of the development process and assign a subjective rating to them. Some interviewees reflected that the categories represented topics often discussed in the sprint retrospective; thus, this related to information discussed but not currently captured by DanFin. The Satisfaction metric provided a good overview of the team. One interviewee commented on the effectiveness of Satisfaction in portraying her team:

> I think this is a good picture of our team. [...] This is kind of like, it could be interpreted as noise from the outside, things you can't control yourself (SM1:it3).

The meaning of the Satisfaction metric was not understood in the same manner by all interviewees. Some viewed it as representing something akin to the happiness of the team broken down into specific topics. Others viewed it as representing impediments facing the team in relation to specific aspects of the development. One specific interviewee would use this metric as the basis for discussion between the interviewee and the team, aimed at removing any of these barriers, commenting:

> When you see a Satisfaction graph that [...] [drops] from 3.5 to somewhere between 1.5 and 2.5, then you have an issue. There's something here that's not going in the right direction. [...] I would use this as a basis for dialogue with the team (PM1:it3).

The differing interpretations between interviewees indicate a possible inconsistency in use across teams were this metric used in development. It could be difficult to compare teams' Satisfaction where the teams understand Satisfaction in different ways.

A number of interviewees commented on the specific Satisfaction criteria measured and displayed within the metric. One interviewee suggested that they should be open to change:

> Maybe if you could choose some other kinds of [...] [criteria] I don't think it should be this every time. And they could change over time. (AL:it2)

Several other interviewees expressed similar views. One interviewee expressed the view that he did not wish for there to be a greater number of criteria displayed in the metric, only for the specific criteria to be changeable; the number chosen was appropriate.

Satisfaction was evidently useful to the interviewees, though there was a difference of opinion in whether the chosen criteria were correct, and whether these should be subject to change by users.

5.2.4 The relationship between Happiness & Satisfaction is unclear (6)

Interviewees recognised that Happiness and Satisfaction were related metrics, given their proximity within the Development Health dashboard and the similar meanings of 'Happiness' and 'Satisfaction'. However, interviewees had differing views on the nature and strength of that relationship. One interviewee thought it would be desirable to understand a team's Happiness through the use of the Satisfaction metric. This was not, however, something he considered would be possible:

> If I am trying to find out why they are being more happy [...] I can't find the answer [...] with this Satisfaction chart. [...] I would like to look at this Satisfaction chart to explain that. But maybe you can't use it to do that (AL:it3).

Another interviewee considered that the two were related, and that addressing unsatisfactory areas within the development would improve an already-high Happiness:

> If they are truthful about their Happiness then they are in good spirit. There is no question that there something [relating to Satisfaction] that is annoying them, and that should be look[ed] [...] into [to see] if there is something that can be done there. That would actually help the team (AC:it3).

A further interviewee noted that, in practice, her team was generally quite happy in spite of the existence of impediments. These impediments might manifest themselves within the Satisfaction metric, were it in practical use:

> [The team is] actually pretty satisfied even though we have a lot of impediments because we do very good teamwork.

(SM3:it3)

This interviewee thus does not recognise a strong relationship between the two metrics; this reflects the general lack of clarity amongst interviewees as to the relationship between these two metrics.

The lack of clarity in the relationship between these two metrics could make it challenging for users to draw conclusions from a dashboard involving both of them. In spite of this, interviewees were able to make judgements using these metrics. Thus, clarification of this relationship is desirable but not immediately necessary.

5.3 Agile Maturity

We have made a number of findings in relation to the Agile Maturity dashboard, and the metrics contained within it.

5.3.1 Assessment of Agile Maturity is a team level, not a project level activity (7)

The Agile Maturity dashboard contained a number of metrics aimed at assessing the agile maturity at both project and team level. A majority of interviewees utilised the dashboard to obtain a detailed understanding of the performance of the team. When asked to assess the challenges one team faced, an interviewee explained:

> I would say that actually it looks like they are doing a fairly good job, there [is] room for improvement, and they're actually working with improvement, and that is the [...] most important thing [...]. [M]aybe there is something that indicates that it might be time for them to look at the technical practices in the foreseeable future. Otherwise that might hit them hard very soon (AC:it3).

When asked directly whether the prototype permitted a team level agile maturity assessment to be undertaken, one interviewee replied:

> I have a reasonably good understanding of what's happening in Team Beta. And why they're in trouble (PM1:it3).

Another interviewee however emphasised the limitations of a maturity assessment based on the information presented on the dashboard. In a real-life environment this information would be combined with other measures and observations:

> [I]t would be a small overview of the assessment, it can be detailed, but that's not needed here, and if this was a team [with which] I had interaction [...], then it could be a lot more detailed and this could be put into a context to support different observations (AC:it3).

The project level, indicating a condensed overview of agile maturity for individual teams involved in a project's development, proved less informative. Interviewees found the task of assessing agile maturity on this level to be challenging. Though most interviewees were able to identify teams that needed attention based on the information on the dashboard, interviewees clicked through to specific teams almost immediately. One interviewee suggested that a combined maturity assessment for all teams working on a project might simply not be meaningful:

> I think that the team level is the right level, and not [...] the whole project, because the next time you have another team in the project and you should look at this team. Not together with all the other teams, you should look at it separately. (AC:it3)

The team level was therefore useful, allowing all interviewees to successfully complete an agile maturity assessment for individual teams. However, the project level was found less useful.

5.3.2 Practice assimilation levels are an important indicator of maturity (8)

All interviewees agreed on the importance of having information on the maturity of teams' use of agile practices. The Practices metric—and its measurement scale—was liked by a majority of interviewees. One interviewee explained why she valued this metric over more traditional agile metrics such as Velocity:

> [W]hen you're talking to teams they say 'oh, yeah, we have a stable velocity and...', yeah yeah yeah, but you're not doing stories, work, or base your work on business value (SM1:it3).

Another interviewee stated how he would use the Practices metric:

> Well this is [...] an important way for me to have a look at how we're doing on our agile principles. [...] [W]hat do we need to improve? Then I would prepare activities based upon this (PM1:it3).

This metric appears to aid in the learning and improvement of teams.

Some interviewees expressed views on how the metric could be improved. One interviewee gave one example of an additional practice which could be added:

> [When assessing agile maturity] I would look at [...] the definition of ready and definition of done. You don't have definition of ready [in your prototype] (SM3:it3).

Another concerned the designation of a number of practices as 'key' practices, which could be emphasised in assessing the progress of a team towards agile maturity:

> [I]t would be great to be able to see, right, we have decided in the coming sprints or five sprints or something like that, we will look at this area and see if we can improve that (PM1:it3).

The same interviewee suggested that each team's scores on these key practices could be a more meaningful alternative to the aggregated maturity scores at the project level in the dashboard.

All interviewees thus agreed on the usefulness of measuring a team's maturity in their use of agile practices. The Practices metric appears to permit a focus on learning and facilitates the improvement of teams.

5.3.3 Sprint Interference is a useful metric, but what constitutes interference is unclear (9)

All interviewees agreed on the value of tracking time spent on non-sprint tasks during sprints and being able to have an overview of this. Interviewees felt it important to understand the time spent on these tasks, as it allowed them to understand where time was being spent during a sprint, and thus whether time was being spent in the desired way.

Interviewees disagreed, however, on which tasks should be classified as interference. Time spent on unscheduled meetings and miscellaneous non-sprint tasks were generally accepted to be interference. Two interviewees suggested the latter could be used for capturing the effects of scope change during a sprint, with one stating:

> You can either say that our sprint goal is locked so we won't do it, but usually when you go to a Product Owner saying 'this is reality so we take it in' [...] [Through tracking non-sprint tasks] you could of course explain why [...] you didn't make your sprint goal (SM3:it3).

Two interviewees considered that effort spent on reducing technical debt or refactoring did not constitute interference, as those activities are tied into development of stories from the sprint backlog. Another interviewee considered bug fixing not to be interference for the same reason. Team members working on backlog tasks outside the sprint's scope was considered normal practice by one interviewee, and should not be measured by this metric.

Thus, the usefulness of tracking effort spent on non-sprint tasks was generally accepted. Some of the specific activities measured by this metric were not, however, generally viewed as constituting interference.

5.3.4 Code Ownership's value in assessing agile maturity is low (10)

The Code Ownership metric was not found to be useful in assessing agile maturity. Half of the interviewees did not use the metric in making an agile maturity assessment, and only two discussed its potential use. One of the interviewees who did elaborate on the use of this metric considered it an indicator for how team members are working together:

> So it is about working close together, and pair programming or whatever is not really working, it could be improved, [...] and I would talk to the team about how are you actually working together on this, could we share more knowledge[?] (PM2:it3)

Another interviewee also saw it as an indicator, but stressed the fact that its value would be highly dependant on contextual factors, such as code structure and applied practices. The same interviewee referred to this metric in relation to the overall value of the prototype dashboard, hinting at its usefulness in a broader context:

> [W]hat I have here is a great support [...] Code Ownership is something that I couldn't get directly from the teams by talking to them, and they're not really available anywhere else (AC:it3).

However, this interviewee did not explain how the metric could be used in making an agile maturity assessment.

Thus, whilst there might be a relation between agile maturity and Code Ownership, this relationship—and the ability to use Code Ownership as part of an agile maturity assessment—was not widely recognised or understood.

5.4 Product Tracking

In this section we describe findings made in relation to the Product Tracking dashboard. The Happiness and Satisfaction metrics contained within the team level Product Tracking dashboard are discussed in Section 5.2.

5.4.1 Product Tracking dashboard gives a useful overview on the project level, but not on the team level (11)

Interviewees considered the Product Tracking dashboard to give useful insights into development progress on the project level. All interviewees were able to use the information on the dashboards to gain an insight into how development was progressing and elaborate on how different teams were performing. One interviewee explained his interpretation of the information displayed:

> Best case, if we progress as we are at the moment, then on 5th May we will have our delivery in place. Worst case [...] on the 15th. [...] I can also see how much different teams [...] have delivered. [...] And if I look at Lambda [...] [,] it's a team that's improving over time, doing more [...] than what they expected initially (PM1:it3).

One Scrum Master valued the dashboard's straightforward illustration of the relation between scope change and delivery date. The two Project Managers appreciated the clear best and worst case delivery window it provided. The indicators on the dashboard, providing information on team stability and delivery, were generally liked, as they provided a quick insight into which of the teams needed immediate attention.

At the team level, interviewees were able to understand how teams were performing, but struggled to relate the teams' development effort to the whole project. This was attributed to the relative nature of stories points as a unit of measurement:

> [W]hat I can see here is that [...] they are almost doing their commitments[.] [...] [B]ut what I can't see is [...] what they are doing, how much is that. Because we are doing relative estimates [.] (AL:it3)

A possible solution for this problem was proposed by three interviewees: this consisted of adding an indicator for each sprint indicating whether the sprint goal had been achieved.

The Product Tracking dashboard at the project level was thus found to provide useful information on the development status of the whole product. Interviewees found it more difficult, however, to obtain a useful overview of a team's status and contribution the development project.

5.4.2 Velocity and Sprint Burndown metrics useful together (12)

The Velocity and Sprint Burndown metrics were considered useful measurements of team performance by the majority interviewees. These were seen as closely related metrics, and were frequently used together. One interviewee observed:

> [V]elocity [indicates] they've not been meeting their targets, [...] which causes issues with regards to the Sprint Burndown. (PM1:it3)

Both metrics were used to make an assessment of a team's ability to estimate their work capacity for one sprint. The Velocity metric could additionally provide insight in stability of a team's delivery, with one interviewee explaining:

> [S]o [...] I can see that actually are they able to work toward a sprint commitment and get there. [...] [I] can see on the burndown if they're actually working with [it] and getting progress or if they are just flat lining in the beginning and then just hurrying up to get something done on the last day. And can see on the Velocity, is that stable or is it fluctuating? Are they consistently not meeting their commitment? (AC:it3)

Another interviewee monitored the Sprint Burndown metric on a daily basis at a previous workplace, and wished it was currently available at DanFin:

> I miss it here in DanFin because I'm used to having a dashboard, and I'm used to having all of these measurements. I was used to going through the burndown on every morning meeting [at] the IT company I came from (SM3:it3).

There was some criticism of the usefulness of these metrics. One interviewee considered information about individual sprints to be less valuable than the end goal. Another considered that the Sprint Burndown metric was not as valuable as having a sprint goal and ensuring that agile practices were followed:

> I don't think [...] it was really useful, [not] [...] when you have a sprint goal, [...] and you do good dailies the team looks at (SM2:it2).

While one Project Manager stressed the importance of having access to these detailed metrics, an- other Project Manager considered information about individual sprints to be of less value to her:

> [A]s a Project Manager I don't really care about the different sprints, what's important to me is [...] this end goal (PM2:it3).

Thus, most interviewees agreed on the usefulness of one or both of the Velocity and Sprint Burndown metrics, most thought these metrics valuable for assessing teams' performance in the development process.

5.5 Product Quality

We made a number of findings in relation to the product quality and the metrics contained within it. These are described below.

5.5.1 Product Quality dashboard is useful for assessing quality on project and team levels, but might not give a complete insight into it (13)

The Product Quality dashboard contained a number of metrics relating to product quality, both on the product level as on the team level. Interviewees used the information displayed in the dashboard to gain an understanding of the quality of the product under development. One interviewee explained his interpretation of this:

> [I] would have said there were a lot of defects if it was in production. [...] [F]or me it seems that they are still in the start of the project, because they still have a pretty constant level of defects in each criticality I think. [...] [I]t seems like they are not trying to solve the problem. They are just developing new stuff (AL:it3).

On the project level, the main metric displayed Defects Over Time, broken down in a one to five criticality scale. This metric was generally viewed as important and useful:

> I care about the defects as a Project Manager, then I need to get the defects out of the way, get through all of the test cases, [and] then I'm good. (PM1:it3)

The ability to break down defects per team on the team dashboard was equally appreciated by interviewees:

> [It is] very useful to look into the different teams, because I can see we need to do something in order to raise quality in this first team [...] (PM2:it3).

While the nature of the Defects Over Time metric was generally not considered problematic, half of the interviewees felt it was difficult to interpret artificial data without any context:

> I don't have anything to compare this... I have no experience from other projects, how many critical defects do we have in other projects at this time (AL:it3).

A more fundamental issue was raised by another interviewee. He stated that the number of defects would primarily inform him on a possible lack of quality. To gain an insight into the presence of quality, he would require information from a statistical code analysis system in use at DanFin, that provides measurements on code complexity, test coverage and average class size.

Thus, the Product Quality dashboard was useful to interviewees at both project and team level. It did, however, require more context for effective use, and may be unable to provide an insight into the presence of quality, only its absence.

5.5.2 Code Ownership's relation with quality is unclear (14)

A majority of interviewees agreed that having multiple developers contribute to a certain part of the product code

usually leads to an increase of quality. Most interviewees however found it difficult to see a direct and clear relation between the Code Ownership metric and Product Quality dashboard.

One interviewee explained that under specific circumstances, Code Ownership could be informative on quality:

> [There] could be [such a relationship]. Because, if the team is doing something that is shared or split across multiple repositories [...] [,] if they then have really low shared code ownership [...] [,] that might be something that has a substantial impact on the quality. But it doesn't say anything in itself. You have to combine it with all the background knowledge of the team and what they're doing (AC:it3).

Another, however, rejected the idea of such a relationship altogether:

> In my mind [quality] is not related to Code Ownership. I could be related to so many other things (SM1:it3).

Thus, whilst some interviewees were able to recognise the value of information about Code Ownership, its ability to inform on product quality is unclear. Interviewees recognised that Code Ownership had a bearing on quality, but in this context it was ultimately uninformative.

6 Discussion

To answer our research question, in this section we discuss our findings as presented in Section 5 as they relate to existing literature. Our research question, as stated above, is:

> Which software metrics are useful within a multi-team agile development and how can they be usefully combined in a dashboard to aid such a development?

Several topics emerge from the analysis which contribute to answering our research question. Specifically, we discuss the metrics which were particularly valuable; the implications of our findings in relation to dashboards supporting multi-team agile development; and use of the prototype by individuals with different roles in such a development.

6.1 Key metrics

The metrics literature identifies a number of metrics which are regularly used in agile software development, and considered important. These include velocity, effort estimate, defect count and defect trend (Kupiainen et al., 2015, p. 155). We found in this study that the Velocity and Sprint Burndown metrics—the latter being a means of estimating effort—were useful (Finding 12). We also found that our Defects metric was of value (Finding 13), this incorporating both a defect count and a defect trend over time. Our findings thus confirm the usefulness of a number of metrics identified in literature as being regularly used within agile software development.

Software metrics are often categorised and described using the classification scheme of Fenton (1994) (see, e.g. Kupiainen et al., 2015; Gómez et al., 2006). Team and personnel are both mentioned as entities for measurement within this scheme, possessing attributes such as productivity and experience (Fenton, 1994; Fenton and Bieman, 2014). Experience in agile software development can be equated with the level of assimilation for a given set of agile practices; this can be measured in a rigorous manner (Wang et al., 2012). We found that the Practices metric (Finding 8) was very useful in permitting an assessment of the agile maturity of a team. It is suggested in our finding that this metric would allow users of it to make a meaningful assessment of how agile a team actually is, rather than simply whether a team is using nominally agile practices. This metric is not explicitly mentioned in the metrics literature, though it can be seen that it is a means of measuring the experience of an agile team; it measures an attribute of a named entity of Fenton's. Our findings in this regard thus confirm that the classification of metrics in Fenton (1994) holds in relation to this new metric. As a metric not currently identified in the literature, however, our findings add to current knowledge by identifying a novel metric specifically applicable to agile software development.

A number of different reasons for using metrics within agile software development are identified in the metrics literature (Kupiainen et al., 2015, p. 150). Generally, metrics used within agile software development should support agile principles (Hartmann and Dymond, 2006). Agility (in software development) means focusing on embracing change and learning from it; this should be supported by the use of agile practices (Conboy, 2009, p. 340). We found two metrics to be the most useful within this study: those measuring the Happiness and Satisfaction of the team (Finding 4 & 5). These metrics were universally viewed as providing a substantial amount of useful information on an agile team. These metrics are, however, entirely unrepresented in the literature. They do not fit easily into the classification scheme of Fenton (1994), as described above; nor do they fit the reasons for using metrics from Kupiainen et al. (2015). Happiness and Satisfaction relate not to the measurement of teams or personnel *per se*, but to the views of teams in relation to their working environment. Users identified a change in happiness or satisfaction as being indicative of changes in the development environment that might not be otherwise identified, and took decisions based on this knowledge. The centrality of change within agile software development requires the knowledge of the existence of such a change, so that it can be learned from. These metrics convey information which facilitates the learning from change so central to agility. Thus, the existence and usefulness of these metrics adds to existing knowledge by identifying two previously unidentified metrics, and by identifying a new reason for metrics use within agile software development, namely to understand, and thus learn from, change within the development process.

6.2 Dashboard design

A dashboard should display a single page containing limited information (Yigitbasioglu and Velcu, 2012, p. 44). It should contain a degree of interactive functionality to accommodate the need for more detailed information. The appropriate level of functionality should be considered in light of the purpose of the dashboard and the quantity of information required for that purpose (Yigitbasioglu and Velcu, 2012, p. 48). One of the more common functional features is 'drill-down' and 'drill-up' functionality, which allows dashboard users to acquire more detailed information about a particular element within the dashboard (Yigitbasioglu and Velcu, 2012, pp. 47–48). Our findings show that users found that detailed information is necessary for making decisions. Indicators where used within a dashboard must be accompanied by the ability to obtain detailed information (Finding 1 & 2). Even though there was a difference of opinion as to the most valuable level of detail for different dashboards, users found it useful to access different levels of detail to obtain required information (Findings 3, 7, 11 & 13). This is suggestive of a need for different levels of detail of information within a dashboard aiding multi-team agile development. We have been unable to identify any literature on this specific point. Thus, our findings in this regard represents a new contribution.

It is important that information presented by a dashboard is relevant to the user and focuses on key measurements (Yigitbasioglu and Velcu, 2012, p. 48). A flexible representation can be beneficial in improving relevance of a dashboard to the user (Yigitbasioglu and Velcu, 2012, p. 52). During our research we found that users have different information needs. There were differences of opinion between users as to which metrics and visualisations were useful within a dashboard (Findings 10, 12 & 14), and as to what specific data were useful within metrics and visualisations (Findings 5, 8 & 9). This suggests a need to provide options to users to omit or alter dashboard contents to best suit their information needs. Flexible representation is therefore a desirable functional feature in the design of a dashboard for agile development. Thus, we confirm the usefulness of a flexible representation of dashboards as presented in literature. Further, the desirability of flexible representation in a dashboard aiding multi-team agile development represents a new contribution to knowledge on agile dashboard design.

Five major reasons for using metrics in agile software development are distinguished by Kupiainen et al. (2015): sprint and project planning, sprint and project progress tracking, understanding and improving quality, fixing software process problems, and motivating people. These reasons were however derived by the authors from measurement goals of individual metrics, and therefore not validated in practice. Our prototype contained four different themes around each of which a dashboard was created: Product Tracking, Product Quality, Agile Maturity, and Development Health. These themes arose from our iterative prototype development process, emerging from the purposes for which metrics were considered useful during evaluation. Thus, each theme may be considered to relate to one reason or purpose for using the dashboard based upon it. Practitioners found the themes to represent useful groupings of dashboard functionality (Finding 1). There is a relationship between our themes and the reasons in Kupiainen et al. (2015). Product Quality relates to understanding and improving quality; Product Tracking relates to sprint and project progress tracking; and Agile Maturity relates to fixing software process problems. Development Health does not relate to any of the reasons found in literature. However, as this theme contains only the Happiness and Satisfaction metrics, the reason for this is explained in Section 6.1 in the discussion on these metrics. Thus, three of our four themes correspond to three of the reasons for using metrics in agile software development. Therefore, this partially confirms and validates in practice the literature relating to the reasons one would use software metrics in agile software development.

6.3 Roles & dashboard usage

Metrics literature commonly makes a sharp distinction between the information requirements of management, developers and other operational staff (Fenton and Bieman, 2014, pp. 14–16; Eckerson, 2010, p. 103). In agile software development, however, the distinction between different roles in the development process is much less apparent, as agile teams themselves, and not management, are responsible for managing the teams' processes and deliveries (Schwaber and Beedle, 2002, pp. 8,19). Project Managers are suggested to take on a facilitative, rather than directive role in agile environments (Waardenburg and Vliet, 2013, p. 1261). Our findings indicate that interviewees, irrespective of their role in the development processes, made similar assessments of the value of elements of the prototype. This was apparent for example in the high value placed on the Practices and Happiness metrics, which was shared by Scrum Masters, Project Managers and the Agile Coach (Finding 4 & 8). Some metrics were found useful by different roles for different reasons, as was the case with the Release Burnup metric (Finding 11). A Scrum Master valued this metric for the insight it provided into the consequences of scope change, whereas a Project Manager found the expected release date useful. Where diverging opinions on usefulness existed, such as for the Sprint Burndown and Velocity metrics, different opinions could not be related to the roles of the interviewees (Finding 12). The common values placed on different metrics indicates that, despite undertaking different roles, individuals within agile software development have largely the same information requirements. These can be related to their responsibilities and interests as described by agile literature. This finding modifies the current knowledge concerning software metrics insofar as it relates to agile software development, as different roles within this environment have the same informational needs. This is in accordance with the agile software development literature.

6.4 Limitations

Our study took place over a relatively short period (March to May 2017). It built on a previous case study at one software development organisation, and involved the evaluation of prototypes at that same organisation. As with any study undertaken solely within one organisation, our findings may be inapplicable outside of the context of this organisation. However, DanFin appears to have adopted many standard agile practices, and thus we expect our results to have more general applicability to other, similar environments.

The short duration of the study had an impact on the number of iterations we were able to complete. Metrics were evaluated in each iteration, however dashboards were only evaluated in the final iteration. Ideally we would have undertaken a further iteration to iterate on its design and evaluate dashboards in more detail. This was mitigated, however, by having a comparatively large number of interviewees in the final iteration which we intended to result in a more thorough evaluation of the final prototype.

Selection of participants was dependent on the willingness of individuals to participate in our study. It is possible that our participants were not representative of practitioners at DanFin, or of agile practitioners generally. Furthermore, we only had one participant who was an Agile Coach, one who was a Agile Leader, and no Product Owner or Developer. Our findings were consistent across participants; as they were involved in different teams and projects, it is unlikely that the participants were consistently unrepresentative of agile practitioners. The Scrum Master is, to some extent, able to represent the views of Developers as they work closely with them to resolve impediments and promote agile processes.

Our prototype used test data rather than real data throughout our study. As such, our findings may not be reflective of a real-world evaluation of our prototypes. The test data was instantiated within a complete data model, however, and intended to reflect what might be encountered in the real world. It is unlikely that the test data were so different from real data that they substantially affected our findings.

The Happiness and Satisfaction metrics, as described in Section 6.1, may not use an appropriate scale. The numerical values used in the Happiness and Satisfaction metrics are not absolute measurements, but are closer to ordinal values (Fenton, 1994). The scale might be improved by using very dissatisfied through to very satisfied, with intermediate values between. As such, it may be that the Happiness and Satisfaction metrics do not meet the requirements for software metric Fenton describes. However, interviewees were accustomed to using these scales as Happiness was previously in use at DanFin. Thus, it appears that the scales were understood and the metrics found useful in any case.

7 Conclusion

This paper presents the findings of our prototype development and evaluation study. Our study provides an insight for both practitioners and researchers into the software metrics useful within a multi-team agile development. It also supports combining these in a dashboard to assist with understanding the status of such a development, and making decisions in relation to it. We identify several contributions arising from this study.

In relation to software metrics our study confirms the usefulness of standard agile software metrics already in use. It confirms the applicability of the classification scheme of Fenton (1994) to some agile software metrics. We have identified several novel metrics not present within the literature: Practices, Happiness and Satisfaction. Further, the latter two of these practices do not fit within the aforementioned classification scheme, representing we believe a new type of software metric. We confirm a number of reasons for using metrics within agile software development. We also identify a new reason for so doing: to understand and learn from change within the software development environment.

In relation to dashboards, this study confirms the usefulness of having a flexible representation of a dashboard to enable it to be customised based on the needs of the individual user. This desirability for a flexible representation is new knowledge as it relates to agile software development dashboards. We also find that users of an agile software development dashboard require to have access to information at varying levels of detail for such a dashboard to be useful.

This study also modifies current understanding of the need to provide different dashboard information to different stakeholders. We find that the information needs of individuals within multi-team agile development do not vary between roles represented in our study.

A final contribution is the artifact instantiation resulting from our design process, namely our final prototype, encapsulating the knowledge gained during the course of this study.

This study indicates a number of areas for future research. We conducted this study at an organisation that tailored agile development methodologies to their specific requirements. Further research may involve similar such studies in organisations using different agile development methodologies and practices, to ascertain whether our findings are applicable more broadly than our case.

Our prototype represents a purposeful starting point for further development by practitioners wishing to support their own agile development processes or researchers aiming to further investigate the information needs of a multi-team agile software development. A future study may evaluate our prototype within a different organisation with a wider range of interviewees.

Whilst we used test data during our study, our prototype is capable of operation with ‘real’ data. Valuable future research would include conducting a study using our prototype populated with real data from within a working

multi-team agile software development, to test whether our findings hold under these conditions.

On the level of individual metrics, the high value practitioners attributed to the Happiness and Satisfaction metrics is noteworthy. It is unclear to us the extent to which either of these metrics are measured in other organisations. Further research could also involve introducing these metrics to organisations that have no previous experience with this type of measurement to establish whether experience influences valuations of these metrics.

References

Beck, Kent and Cynthia Andres (2004). *Extreme Programming Explained: Embrace Change, 2nd Edition*. 2nd edition. Boston, MA: Addison-Wesley. 224 pp. ISBN: 978-0-321-27865-4.

Boehm, B. and R. Turner (2005). 'Management Challenges to Implementing Agile Processes in Traditional Development Organizations'. In: *IEEE Software* 22.5, pp. 30–39. ISSN: 0740-7459.

Bumbary, Karen M (2016). 'Using Velocity, Acceleration, and Jerk to Manage Agile Schedule Risk'. In: *Information Systems Engineering (ICISE), 2016 International Conference on*. IEEE, pp. 73–80.

Cockburn, Alistair and Jim Highsmith (2001). 'Agile Software Development: The People Factor'. In: *Computer* 34.11, pp. 131–133. ISSN: 0018-9162.

Conboy, Kieran (2009). 'Agility from first principles: Reconstructing the concept of agility in information systems development'. In: *Information Systems Research* 20.3, pp. 329–354.

Davis, Alan M (1992). 'Operational Prototyping: A New Development Approach'. In: *IEEE Software* 9.5, pp. 70–78.

– (1995). 'Software prototyping'. In: *Advances in computers* 40, pp. 39–63.

Davis, Fred D (1989). 'Perceived usefulness, perceived ease of use, and user acceptance of information technology'. In: *MIS quarterly*, pp. 319–340.

Dikert, Kim, Maria Paasivaara and Casper Lassenius (2016). 'Challenges and success factors for large-scale agile transformations: A systematic literature review'. In: *Journal of Systems and Software* 119, pp. 87–108. ISSN: 0164-1212.

Eckerson, Wayne W. (2010). *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*. 2 edition. New York: Wiley. 336 pp. ISBN: 978-0-470-58983-0.

Fenton, Norman (1991). *Software Metrics: A Rigorous Approach*. London, UK, UK: Chapman & Hall, Ltd. ISBN: 978-0-442-31355-5.

– (1994). 'Software measurement: a necessary scientific basis'. In: *IEEE Transactions on Software Engineering* 20.3, pp. 199–206. ISSN: 0098-5589.

Fenton, Norman and James Bieman (2014). *Software metrics: a rigorous and practical approach*. Third. CRC Press.

Fenton, Norman and Martin Neil (1999). 'Software metrics: successes, failures and new directions'. In: *Journal of Systems and Software* 47.2, pp. 149–157.

Gómez, Oswaldo et al. (2006). 'A systematic review measurement in software engineering: state-of-the-art in measures'. In: *International Conference on Software and Data Technologies*. Springer, pp. 165–176.

Grady, Robert B. (1992). *Practical Software Metrics for Project Management and Process Improvement*. Englewood Cliffs, NJ: Prentice Hall. 282 pp. ISBN: 978-0-13-720384-0.

Hartmann, D. and R. Dymond (2006). 'Appropriate agile measurement: using metrics and diagnostics to deliver business value'. In: *AGILE 2006 (AGILE'06)*. AGILE 2006 (AGILE'06), 6 pp.–134. DOI: 10.1109/AGILE.2006.17.

Hendriks, Ivo et al. (2017). 'Understanding the Challenges of a Large-Scale Agile Transformation'. Aalborg, Denmark.

Hevner, Alan et al. (2004). 'Design Science in Information Systems Research'. In: *MIS Quarterly* 28.1.

Hughes, Michael (1999). 'Rigor in Usability Testing'. In: *Technical Communication: Journal of the Society for Technical Communication* 46.4, pp. 488–94. ISSN: 0049-3155.

Javdani, Taghi et al. (2012). 'On the Current Measurement Practices in Agile Software Development'. In: *International Journal of Computer Science Issues (IJCSI); Mahebourg* 9.4, pp. 127–133. ISSN: 1694-0814.

Kerzner, Harold (2013). *Project Management Metrics, KPIs, and Dashboards: A Guide to Measuring and Monitoring Project Performance*. 2 edition. Hoboken, New Jersey: Wiley. 448 pp. ISBN: 978-1-118-52466-4.

Kupiainen, Eetu, Mika V. Mäntylä and Juha Itkonen (2015). 'Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies'. In: *Information and Software Technology* 62, pp. 143–163. ISSN: 0950-5849. URL: http://www.sciencedirect.com/science/article/pii/S095058491500035X.

Lazar, J., J. H. Feng and H. Hochheiser (2010). *Research Methods in Human-Computer Interaction*. John Wiley & Sons Ltd. ISBN: 978-0-470-72337-1.

Lew, Philip (2016). 'Agile Testing Metrics: Quality Before Velocity'. In: *Software Quality Professional Magazine* 18.3.

Lim, Youn-Kyung, Erik Stolterman and Josh Tenenberg (2008). 'The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas'. In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 15.2, p. 7.

Mathiassen, Lars et al. (2000). *Object-Oriented Analysis & Design*. Aalborg, Denmark: Marko. 443 pp. ISBN: 978-8-777-51150-9.

Schwaber, Ken and Mike Beedle (2002). *Agile Software Development with Scrum*. Upper Saddle River, NJ: Prentice Hall. ISBN: 978-0132074896.

Sheehan, Megan (2016). *State of lifecycle management: journey towards quality*. Hewlett Packard Enterprise Community.

Sommerville, Ian (2016). *Software Engineering*. Tenth Edition. Pearson. ISBN: 978-1-292-09613-1.

Staron, Miroslaw et al. (2014). 'Dashboards for continuous monitoring of quality for software product under development'. In: *Relating System Quality and Software Architecture*. Ed. by Ivan Mistrik et al. Morgan Kaufmann Publishers.

Tokola, Henri et al. (2016). 'Designing Manufacturing Dashboards on the Basis of a Key Performance Indicator Survey'. In: *Procedia CIRP* 57, pp. 619–624.

Van de Ven, Andrew H. (2007). *Engaged Scholarship: A Guide for Organizational and Social Research*. 1 edition. Oxford ; New York: Oxford University Press. 344 pp. ISBN: 978-0-19-922630-6.

Waardenburg, Guus van and Hans van Vliet (2013). 'When agile meets the enterprise'. In: *Information and Software Technology* 55.12, pp. 2154–2171. ISSN: 0950-5849.

Wang, Xiaofeng, Kieran Conboy and Minna Pikkarainen (2012). 'Assimilation of agile practices in use'. In: *Information Systems Journal* 22. ISSN: 1365-2575.

Yigitbasioglu, Ogan M and Oana Velcu (2012). 'A review of dashboards in performance management: Implications for design and research'. In: *International Journal of Accounting Information Systems* 13.1, pp. 41–59.

A Prototype Screenshots

Figure 3: Top level dashboard.

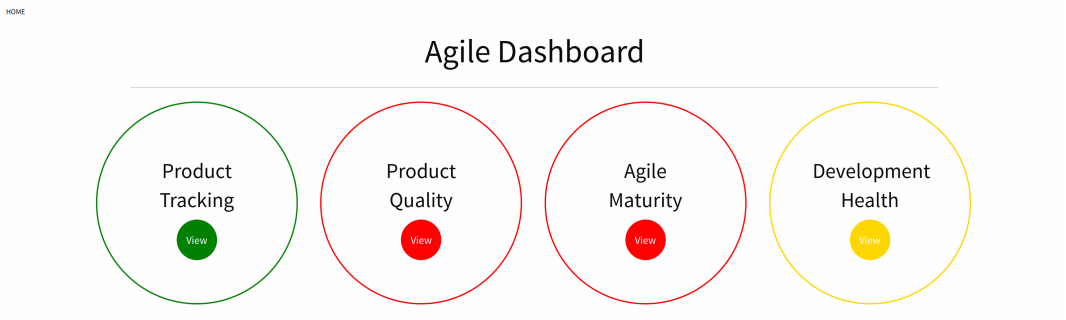


Figure 4: Product Tracking dashboard (project level).

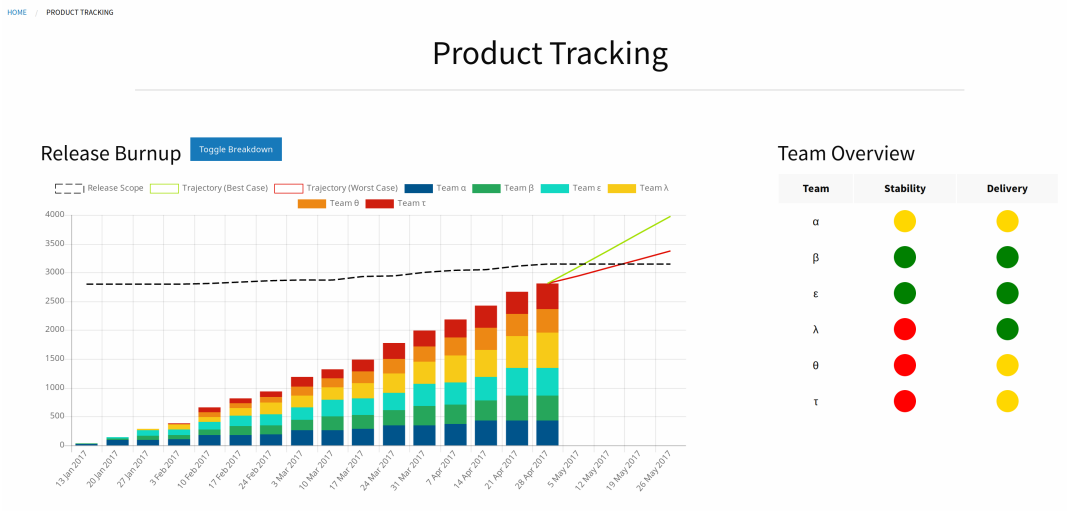


Figure 5: Product Tracking dashboard (team level).

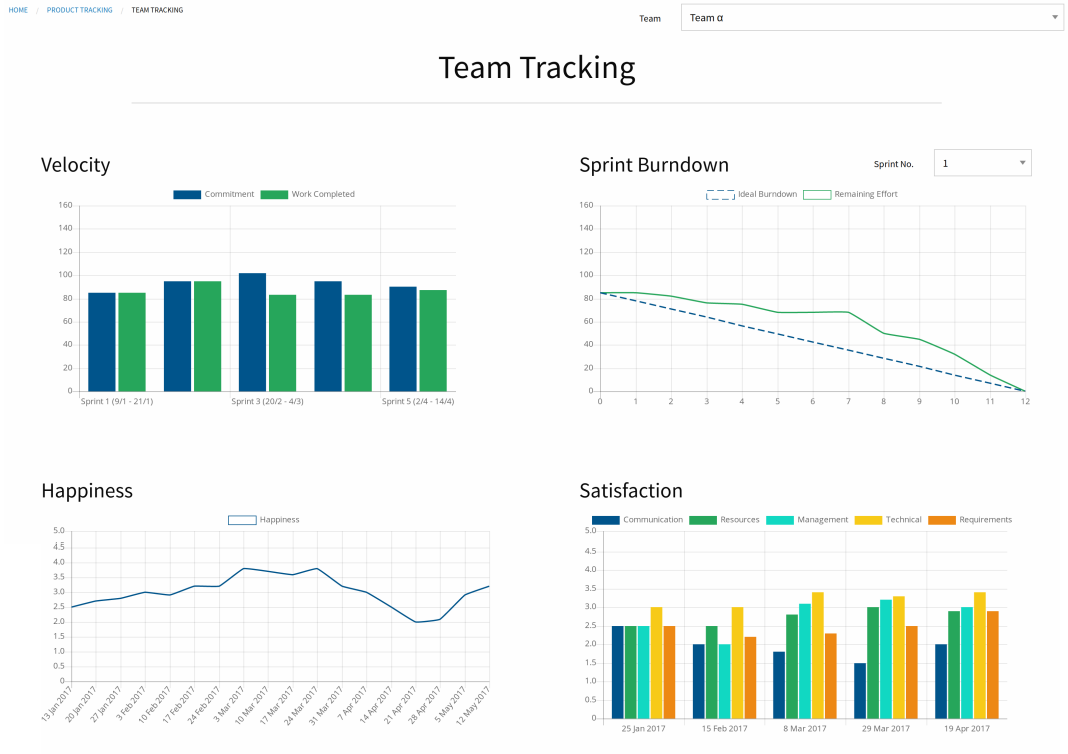


Figure 6: Product Quality dashboard (project level).

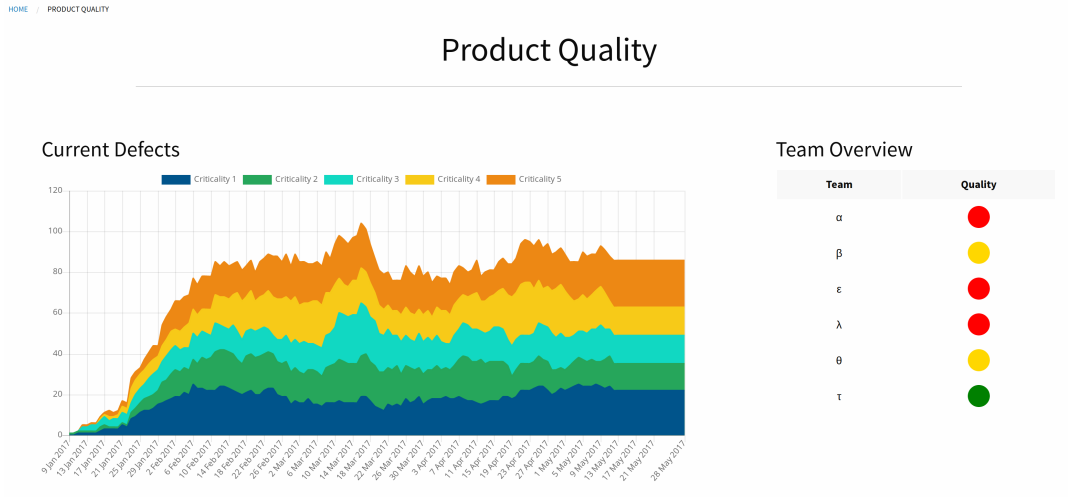


Figure 7: Product Quality dashboard (team level).

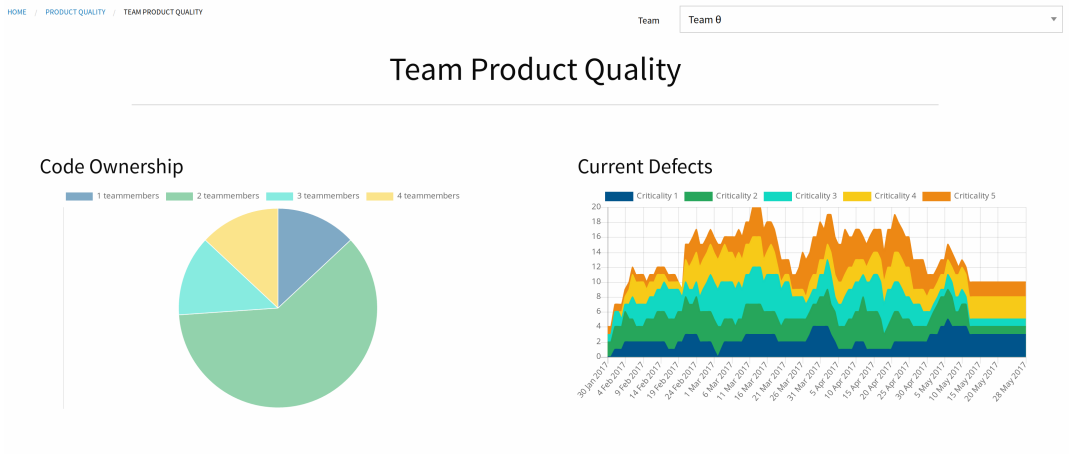


Figure 8: Agile Maturity dashboard (project level).

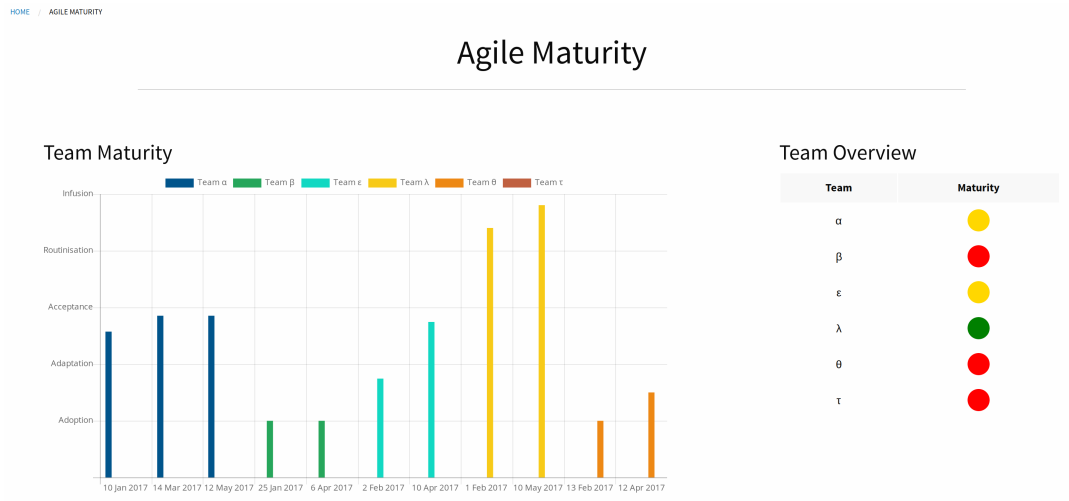


Figure 9: Agile Maturity dashboard (team level).

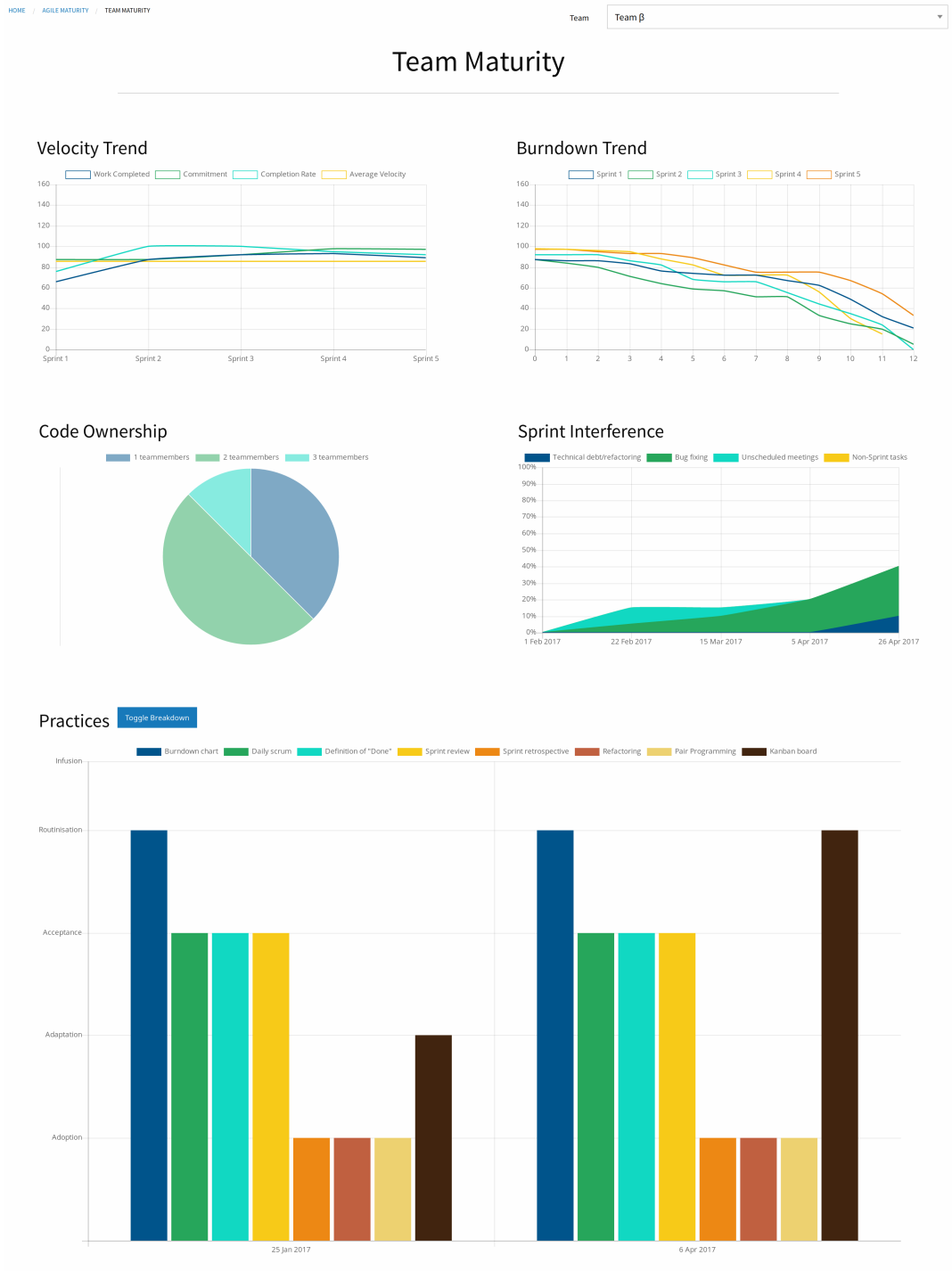


Figure 10: Development Health dashboard (project level).

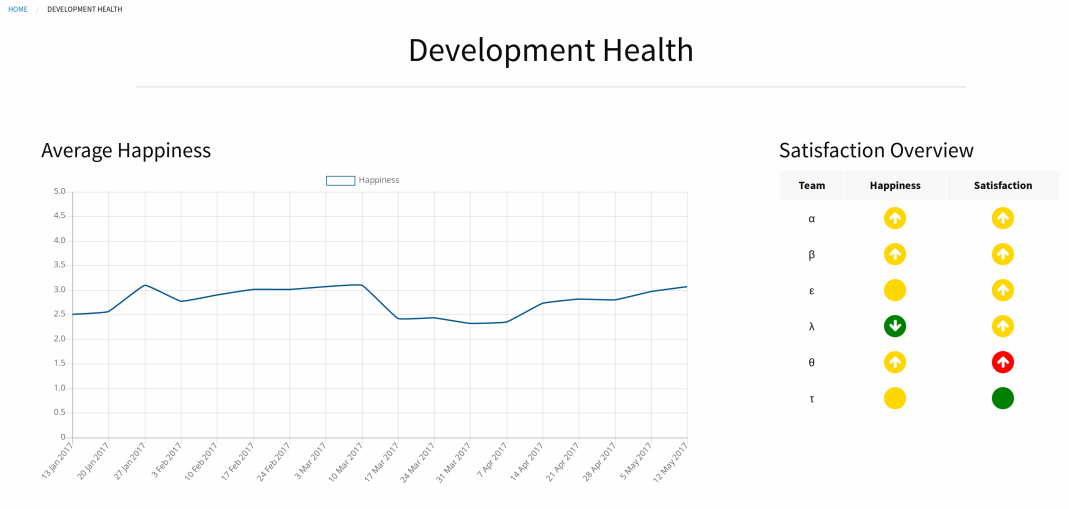


Figure 11: Development Health dashboard (team level).