Eagle: A Team Practices Audit Framework for Agile Software Development

Alejandro Guerrero
Rafael Fresno
aleguedia,raffrearaus@gmail.com
Universidad de Sevilla
Spain

Pablo Fernandez
Carlos Muller
Antonio Ruiz-Cortes
pablofm,cmuller,aruiz@us.es
Universidad de Sevilla
Spain

An Ju
Armando Fox
an_ju,fox@berkeley.edu
University of California
Berkeley, USA

ABSTRACT

Agile/XP (Extreme Programming) software teams are expected to follow a number of specific practices in each iteration, such as estimating the effort ("points") required to complete user stories, properly using branches and pull requests to coordinate merging multiple contributors' code, having frequent "standups" to keep all team members in sync, and conducting retrospectives to identify areas of improvement for future iterations.

We combine two observations in developing a methodology and tools to help teams monitor their performance on these practices. On the one hand, many Agile practices are increasingly supported by web-based tools whose "data exhaust" can provide insight into how closely the teams are following the practices. On the other hand, some of the practices can be expressed in terms similar to those developed for expressing service level objectives (SLO) in software as a service; as an example, a typical SLO for an interactive Web site might be *"over any 5-minute window, 99% of requests to the main page must be delivered within 200ms"* and, analogously, a potential Team Practice (TP) for an Agile/XP team might be *"over any 2-week iteration, 75% of stories should be '1-point' stories"*.

Following this similarity, we adapt a system originally developed for monitoring and visualizing service level agreement (SLA) compliance to monitor selected TPs for Agile/XP software teams. Specifically, the system consumes and analyzes the data exhaust from widely-used tools such as GitHub and Pivotal Tracker and provides team(s) and coach(es) a "dashboard" summarizing the teams' adherence to various practices. As a qualitative initial investigation of its usefulness, we deployed it to twenty student teams in a four-sprint software engineering project course. We find an improvement of the adherence to team practice and a positive students' self-evaluations of their team practices when using the tool, compared to previous experiences using an Agile/XP methodology.

The demo video is located at https://youtu.be/A4xwJMEQh9c and a landing page with a live demo at https://isa-group.github.io/2019-05-eagle-demo/.

ESEC/FSE '19, August 26–30, 2019, Tallinn, Estonia

ACM ISBN 978-1-4503-5572-8/19/08…$15.00
https://doi.org/10.1145/3338906.3341181

CCS CONCEPTS

• **Software and its engineering → Agile software development**; • **Social and professional topics → Project management techniques**.

KEYWORDS

team practice, agile, team dashboard, team practice agreement

ACM Reference Format:
Alejandro Guerrero, Rafael Fresno, Pablo Fernandez, Carlos Muller, Antonio Ruiz-Cortes, An Ju, and Armando Fox. 2019. Eagle: A Team Practices Audit Framework for Agile Software Development. In *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19), August 26–30, 2019, Tallinn, Estonia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3338906.3341181

1 INTRODUCTION

As Agile principles are becoming a prominent philosophy amongst the software industry, a number of project-planning management tools (such as GitHub [1] and Pivotal Tracker [2]) are becoming the common rule to coordinate teams in the software development process. However, in spite there are well established metrics and graph in the typical agile processes such as the completed story count or the sprint burndown in SCRUM, there are a wide number of potential team practices (TP) that could improve the team productivity and quality that are not measured or visualized automatically and rely in the skills or efforts of the project manager.

In order to boost the automation and systematization of such an improvement process, we propose a formalization of TP that could be operationalized. Specifically, we find an important similarity between a team practice structure and service level objectives (SLO) in software as a service; as an example, a typical SLO for an interactive Web site might be *"over any 5-minute window, 99% of requests to the main page must be delivered within 200ms"* and, analogously, a typical TP for an Agile/XP team might be *"over any 2-week iteration, 75% of stories should be '1-point' stories"*. In such a context, a set of TP could define a global agreement, that we coin as Team Practice Agreement (TPA), in a similar way that a set of SLOs (usually with defined compensations) conform a service level agreement (SLA).

[1] https://github.com
[2] https://pivotaltracker.com

Following this similarity, in this paper we present Eagle, a framework that supports a systematic way to define, measure and visualize team practices in software development teams following the agile principles. Specifically, the framework provides a microservice architecture based on the governify ecosystem[8] for SLA management extended in two different directions: (i) a new Domain Specific Language (DSL) and monitor for metrics related to agile project management tools and (ii) a dynamic dashboard customized for teams. As a result, the framework, provides a tooling ecosystem for organizations to define their best practices to follow and track the adherence of their teams and members in order to learn of their pitfalls and improve over time.

2 EAGLE TOOL

2.1 Modelling TPAs

In order to define a Team Practice (TP), we require a language with enough expressivity and easy to understand by teams so it can be transformed from and to natural language. In previous works [8], authors proposed the *iAgree* language to define SLAs joint with a tooling ecosystems that can be extended. From a language perspective, a wide range of SLA aspects can be included from basic information to advanced elements such as limitations (e.g. rates or quotas) or compensations (e.g. penalties or rewards). In this work we have extended this language to describe Team Practice Agreements (TPA) that are composed of different TPs. In order to show its expressiveness we use an specific example of TP to encourage team members to deliver 1-point stories in less than 3 days. The rationale of this TP is the assumption that 1-point stories should be concrete enough to be quickly finished. In this case, the natural language description could be *"over any 3-days period, 75% of assigned 1-point stories should be finished"*. Using the extended iAgree to formalize this TP we can define unambiguosly the metric and objective of the TP as long with the measuring resolution (Figure 1 shows the excerpt [3] of a TPA including such a TP). Specifically, using this extension a team practice is defined by a combination of:

- One or more **metrics**, which specify how the system should compute the different measures that are then used to check the adherence to the practice (Lines 20 to 35 in Figure 1).
- One **objective**, based on one or several metrics and defining operations to be calculated between them, as well as the values that are considered appropriate for the practice (Line 45 in Figure 1).
- A computing **period**, which indicates how frequently the practice adherence will be calculated (Line 47 in Figure 1).
- A computing **scope**, which defines the target of the metrics, that is, if they should be computed for a whole team or each member separately (Line 8 in Figure 1).

While the objective, the period and the scope are already included in the *iAgree* specification and thus they are available out of the box, it was required to extend the language with a DSL to define metrics in the context of Agile/XP software development teams.

[3] A full sample of formal TPA can be found in https://isa-group.github.io/2019-05-eagle-demo/

```
1  id: TPA_Sample:
2  context:
3    validity:
4      initial: 2019-03-07
5      timeZone: America/Los-Angeles
6    ...
7    definitions:
8      scopes:
9          development:
10             member:
11                 name: Member; description: Member of a project;
12             project:
13                 name: Project; description: Project;
14             ...
15     computers:
16         pivotaltracker:
17             url: http://pt.computer.eagle.governify.io
18 terms:
19   metrics:
20     PERCENT_SUCCESSFULLY_DELIVERED_1P_STORIES:
21       computer:
22         name: /indicators/PERCENT_SUCCESSFULLY_DELIVERED_1P_STORIES
23         url: http://pt.computer.eagle.governify.io
24       measure:
25         computing: actual
26         element:
27           percentage:
28             transition:
29               fromState: started; toState: delivered;
30               duration: < 4320 //3 days in minutes
31         filters:
32           type: feature
33           state: delivered, accepted, rejected
34           estimate: 1
35         scope:...       //as definitions scopes
36   ...
37   guarantees:
38     id: 1P_STORIES_SUCCESSFULLY_DELIVERED_ON_TIME
39     notes: 75% of 1-point stories should take less than 3 days...
40     scope:
41       project:
42         name: Project; description: Project;
43       of:
44         scope: project: 2317518
45         objective: PERCENT_SUCCESSFULLY_DELIVERED_1P_STORIES ≥ 75
46         window:
47           type: static; period: daily; initial: 2019-03-07
```

Figure 1: Excerpt of TPA in YAML syntax of *iAgree* language.

The DSL implemented define a metric definitions composed of four different parameters:

- **computing**: this parameter indicates whether the metric must be aggregated or not, to compute an actual value or the maximum, minimum, etc. of a given period.
- **element**: this parameter specifies what property is relevant to the metric calculation, such as the *number* of stories or their estimation *points*. In case we require a computation of a percentage we can refine the metric with the usage of the **percentage** keyword that compute the number of elements that pass a refined set of **filters**.
- **filters**: they define a series of restrictions that must be fulfilled by a story in order to be taken into account for the metric calculation, such as the *type*, the *state* or the *duration* of the *transition* from one state to another.
- **offset**: an optional parameter that indicates that the metric should be computed for a different period which is obtained by adding or subtracting days to the given period.

2.2 Dashboard

In order to track a team's adherence to the different practices, we developed an interactive dashboard that is automatically generated

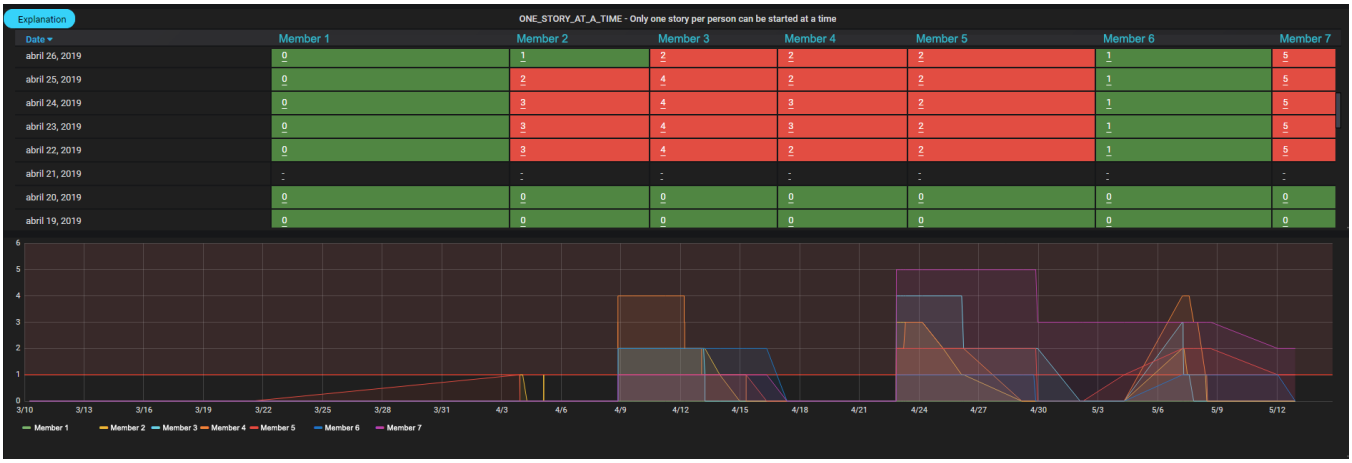

Figure 2: Dashboard screenshot representing a TP adherence for a team.

from the TPA[4] which both, team members and project managers can use. Specifically, Figure 2 shows a screenshot of a fragment [5] of the dashboard including a graph representing a specific TP adherence in the team identified as *"ONE_STORY_AT_A_TIME"* that can be described in natural language as *each member can only be working on one task at the same time*[6].

The dashboard allows to select a period of time to analyze, for instance, an iteration that was unsatisfactory in order to detect the practices that where violated. In addition, an annotation over the graph is possible to highlight persistent comments over the time line for further analysis.

An explanation of each TP can be found in the right upper corner (blue button) of every graph, to allow the users to understand what the graph is representing. The limit to fulfill or not the TP is delimited by a red area. In addition, each graph is linked to a companion table above it with the fulfillment degree of TP's for each day; in case the objective is not fulfilled it is highlighted with a red background and a green background otherwise. In case more information is required, the users can obtain a list of evidences that support that fulfillment degree. As an example in the TP shown in the Figure 2, by clicking in any point of the graph, we will see the list of active stories assigned to each member in that specific point in time.

3 CASE STUDY

We run a case study in a software engineering course at a University of California, Berkeley. With this case study, we would like to understand

- Can TPAs change student backlog delivery behaviors?
- Can Eagle change student backlog delivery behaviors?

We also collect feedback from students about TPAs and the system.

3.1 Settings

In this software engineering course, 120 students are divided into 20 groups. Students work with a real customer on a web service project over four 2-week iterations. Students are supposed to use Agile development methods.

In this case study, we focus on backlog delivery practices. Backlog and user stories are fundamental components in major Agile methods, such as XP [3] and Scrum [9] to address the principles of fast delivery of stories and reasonable velocity [7]. The TPAs in natural language are given to students at the beginning of the second iteration. The Eagle system is open to students at the end of the third iteration.

Each student is required to finish an optional self-assessment survey at the end of each iteration, in which they are asked to answer likert-scale questions on their Agile practices. Furthermore, a survey focused on TPAs and Eagle is sent to students in the middle of the forth iteration.

3.2 Findings

Student backlog delivery behaviors. Figure 3 shows that students report better backlog delivery behaviors in iteration 3 and 4. As Eagle is deployed in iteration 3 and 4, the result suggests that Eagle may help students improve their self-perceived backlog delivery behaviors.

Adherence improvement. In spite the TPAs were known to students during Iteration 2 and the dashboard itself at the end of Iteration 3, the eagle system was monitoring data since the very beginning. Figure 4 shows the degree of adherence to the practices by aggregating the fulfillment of the TPs during the iteration by all teams (depicted as the line in the top of the graph) and the improvement from the previous iterations (depicted as the blue bars in the bottom of the graph); as we can see there was an important increase in iteration 2 when student had the explicit list of TP (though they were expressed in natural language) but there was a small reduction

[4]based on the Grafana framework available at https://grafana.com/
[5]A live demo of the whole dashboard for a number of public projects is available at https://isa-group.github.io/2019-05-eagle-demo/
[6]The formal TP description can be found in https://isa-group.github.io/2019-05-eagle-demo/
[7]The complete description of the TPA can be found in https://isa-group.github.io/2019-05-eagle-demo/

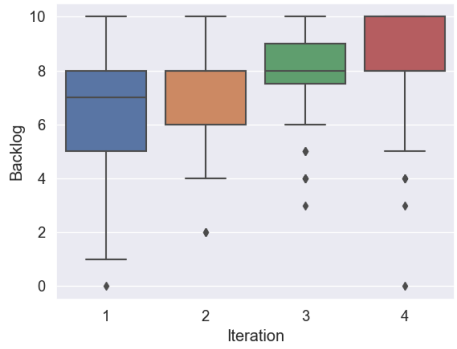

Figure 3: Self-assessment results on backlog delivery practices.

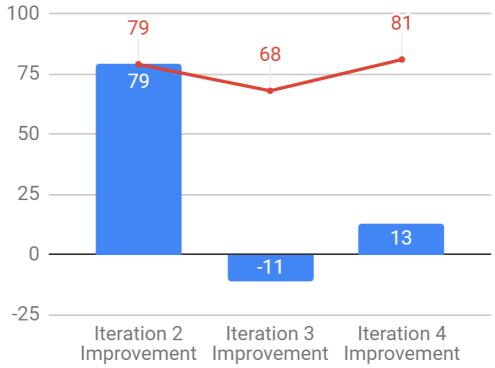

Figure 4: Improvement analysis amongst iterations.

during iteration 3 that could be caused by a fatigue phenomenon from lazy adherence to the rules and the lack of observance feeling; finally the highest adherence was observed in iteration 4 when the dashboard was available to students as they have an explicit measurement of their TP fulfillment.

Student attitude. 16 students respond to the voluntary survey on TPAs and Eagle; 11 (69%) respondents agree or strongly agree that *TPAs and the Eagle system drive you to deliver features in a way that is different from your previous course team projects.* This indicates that overall students agree that using TPAs and the Eagle system change their backlog delivery behaviors. Furthermore, 10 (63%) respondents would like to use TPAs in the future and 9(56%) would like to use the Eagle system. The results suggest that students generally appreciate the values that TPAs and the Eagle system bring to their team. In the qualitative feedback, students mention that while TPAs set concrete requirements on student behaviors, they are not strictly enforced (in terms of score implications). In contrast, in other courses, rules are either strict and specific, or general and lenient. In such a context, TPAs could have had a stronger impact when they have a direct impact on their score.

4 RELATED WORK

To the best of our knowledge, currently no related tooling has been proposed in the literature to provide insight about how closely the teams are following customized team practices in Agile/XP development. However, a number of related works can be found [2, 4, 5, 7, 10] providing a set of software development practices that could be defined as TPs in order to be measured and analyzed with our proposed tool. Thus, Baltes et al. propose in [2] a theory describing software developers properties and practises that are distinctive for experts in their field. For example, the concept of getting feedback from peers are remarked as an important factor and it could be used as a metric in our TPs. Huijgens et al. provide in [4] metrics with high predictive power towards a subset of lagging variables. Thus, as part of their results, metrics like *"planned stories completion ratio"*, or *"planned points completion ratio"* should be defined as TPs in our work because the authors provide them a high predictive power. Treude et al. in [10], and Meyer et al. in [7] conducted empirical studies with 156 GitHub users and 379 software developers, respectively. From their studies metrics emerged from both, (1) the analysis for objective measures of development activity, and (2) the improvement of the productivity through the development of new tools and the sharing of best practices. Complementary, Eagle could be extended in that direction to incorporate such a set of new metrics. Finally, Kupiainen et al. in [5] establishes a motivational ground for the Eagle system by providing a systematic literature review analyzing why and how metrics are used by industrial agile teams in order to infer enforcing process improvements.

In software engineering courses, there are some recent studies on exploiting metrics and dashboards to provide fast feedback [1, 6]; Matthies et al. show in [6] how they use a metric dashboard, ScrumLint, to provide fast feedback to students. Bai et al. use metrics to deliver continuous feedback to students in a software engineering course [1]. Compared with the previous studies, our work focuses on backlog delivery behaviors. Furthermore, TPA (joint with the underlying SLA model) provide a generalizable approach for metric definitions, which makes Eagle also applicable to other learning scenarios and have an easier integration with other tools.

5 CONCLUSIONS AND FUTURE WORK

The Eagle tool represent a fist attempt to create a framework to audit the Agile software development teams by providing a mean to express, monitor and visualize their Team Practices (TPs). We leverage the pre-existing Service Level Agreement (SLA) management platform Governify[8], and extend it with a DSL for Agile software development metrics and an interactive dashboard. A case study over 20 student teams in a software engineering course has provided promising findings in terms of improved adherence to audited practices and positive student self-evaluation. In this learning context, as future work we plan to extend the infrastructure and incorporate other SLA elements such as the compensations by defining scoring penalties or rewards in case of under-fulfilling or over-fulfilling a given TP. From an industrial perspective, we plan to validate the system in other scenarios such as software companies and to mine interesting TPs from historic project databases.

ACKNOWLEDGMENTS

This work is partially supported by the European Commission (FEDER), the Spanish Government under projects BELI (TIN2015-70560-R) and HORATIO (RTI2018-101204-B-C21).

REFERENCES

[1] Xiaoying Bai, Mingjie Li, Dan Pei, Shanshan Li, and Deming Ye. 2018. Continuous delivery of personalized assessment and feedback in agile software engineering projects. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 58–67.

[2] Sebastian Baltes and Stephan Diehl. 2018. Towards a Theory of Software Development Expertise. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. ACM, New York, NY, USA, 187–200. https://doi.org/10.1145/3236024.3236061

[3] Kent Beck and Erich Gamma. 2000. *Extreme programming explained: embrace change*. addison-wesley professional.

[4] Hennie Huijgens, Robert Lamping, Dick Stevens, Hartger Rothengatter, Georgios Gousios, and Daniele Romano. 2017. Strong Agile Metrics: Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)*. ACM, New York, NY, USA, 866–871. https://doi.org/10.1145/3106237.3117779

[5] Eetu Kupiainen, Mika V. Mäntylä, and Juha Itkonen. 2014. Why Are Industrial Agile Teams Using Metrics and How Do They Use Them?. In *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics (WETSoM 2014)*. ACM, New York, NY, USA, 23–29. https://doi.org/10.1145/2593868.2593873

[6] Christoph Matthies, Thomas Kowark, Keven Richly, Matthias Uflacker, and Hasso Plattner. 2016. How Surveys, Tutors and Software Help to Assess Scrum Adoption in a Classroom Software Engineering Project. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 313–322.

[7] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. 2014. Software Developers' Perceptions of Productivity. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*. ACM, New York, NY, USA, 19–29. https://doi.org/10.1145/2635868.2635892

[8] Carlos Müller, Pablo Fernandez, Antonio M. Gutierrez, Octavio Martín-Díaz, Manuel Resinas, and Antonio Ruiz-Cortés. 2018. Automated Validation of Compensable SLAs. *IEEE Transactions on Services Computing* (2018). https://doi.org/10.1109/tsc.2018.2885766

[9] Ken Schwaber and Mike Beedle. 2002. *Agile software development with Scrum*. Vol. 1. Prentice Hall Upper Saddle River.

[10] Christoph Treude, Fernando Figueira Filho, and Uirá Kulesza. 2015. Summarizing and Measuring Development Activity. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. ACM, New York, NY, USA, 625–636. https://doi.org/10.1145/2786805.2786827