Automating Continuous Planning in SAFe

Darius Foo
dfoo@veracode.com
Veracode

Jonah Dela Cruz
jdelacruz@veracode.com
Veracode

Subashree Sekar
ssekar@veracode.com
Veracode

Asankhaya Sharma
asharma@veracode.com
Veracode

ABSTRACT

The Scaled Agile Framework (SAFe) is a popular realisation of the agile methodology for large organisations. It is widely adopted but challenging to implement. We describe a new tool which automates aspects of the SAFe PI Planning process to enable continuous planning and facilitate collaboration between remote teams.

1 INTRODUCTION & MOTIVATION

The Agile Manifesto, originally designed for small, colocated teams [3], is used today even in large organizations of 20,000+ people, 78% of the time with multiple distributed teams [1]. This has led to a multitude of frameworks around agile practices in the large, the most popular [1] being the Scaled Agile Framework (SAFe) [2].

SAFe is challenging to implement, especially in a large-scale, distributed setting [8]. Large projects require significant coordination between teams [6] and cross-team dependencies cause huge communication overhead [7]. SAFe's solution is to wrangle these dependencies during *Program Increment (PI) Planning* sessions: large, centralized meetings, which are fundamentally unable to scale or serve remote teams. Continuous Planning [4] is one way to bridge this gap: enabling global visibility into a continuously-updating plan and automating away the overhead would free time for actual collaboration. We implement a tool, Sapling, to realize this.

2 APPROACH

Sapling[1] is a web application (Figure 1) which implements the planning workflow that teams follow. It is intended to be used concurrently by teams during a PI Planning session.

A given feature team begins PI Planning with a prioritized list of *epics*: high-level business objectives to meet at the end of each PI. Depending on the length of the PI, their time is divided into *sprints*. The team then breaks each epic into atomic *user stories* and assigns them a *story point* value representing how much effort the story will require. Depending on the number of people in the feature team, each sprint is given a *capacity* representing how many story points' worth of work the team can take on. The stories are then assigned to sprints respecting the weight-capacity constraints, as well as unstated ones such as dependencies between stories.

Much of PI Planning is wrangling cross-team dependencies (which is why SAFe recommends that teams be colocated while planning): it is common to walk over to another team, hand them

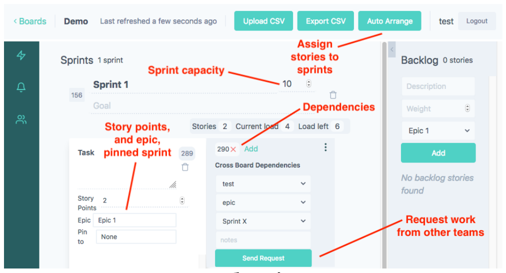

Figure 1: Sapling's main view

a sticky note representing a story, and have them fit it into their plan somehow. Sapling codifies this, allowing teams to add stories to each other's plans and preview the result before accepting.

To enable this, Sapling automates the assignment of stories to sprints using an answer set solver [5]. One might recognize story assignment with weights and capacities as the 0-1 Knapsack Problem; an answer set solver is well-suited to such NP-hard problems. Sapling supports domain-specific constraints such as story dependencies, fixed story-sprint assignments, epic priority, as well as optimization criteria such maximizing story parallelism.

Using a solver in this manner and making it easy to add constraints enables an interactive, *iterative* planning workflow: one simply lists the stories for a particular epic, refines the plan with constraints, and is always up to date on the feasibility of the plan. Conversations between teams are no longer punctuated by having to mitigate the ripple effect of having to pack in unforeseen work.

3 DISCUSSION

Feedback has been positive: stakeholders appreciate the increased visibility and planning sessions proceed more smoothly. Future work will focus on the user experience of solver decisions.

REFERENCES

[1] [n.d.]. State of Agile Survey. https://www.stateofagile.com. Accessed: 2019-12-26.
[2] Mashal Alqudah and Rozilawati Razali. 2016. A review of scaling agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology* 6, 6 (2016), 828–837.
[3] Barry Boehm and Richard Turner. 2005. Management challenges to implementing agile processes in traditional development organizations. *IEEE software* 22, 5 (2005), 30–39.
[4] Breno Bernard Nicolau De França, Rachel Vital Simões, Valéria Silva, and Guilherme Horta Travassos. 2017. Escaping from the time box towards continuous planning: an industrial experience. In *2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering (RCoSE)*. IEEE, 43–49.
[5] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. 2011. Potassco: The Potsdam answer set solving collection. *Ai Communications* 24, 2 (2011), 107–124.
[6] Deepti Mishra and Alok Mishra. 2011. Complex software project development: agile methods adoption. *Journal of Software Maintenance and Evolution: Research and Practice* 23, 8 (2011), 549–564.
[7] Maria Paasivaara and Casper Lassenius. 2016. Scaling scrum in a large globally distributed organization: A case study. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. IEEE, 74–83.
[8] Abheeshta Putta, Maria Paasivaara, and Casper Lassenius. 2018. Benefits and challenges of adopting the scaled agile framework (safe): Preliminary results from a multivocal literature review. In *International Conference on Product-Focused Software Process Improvement*. Springer, 334–351.

[1] https://sapling.netlify.com

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7963-2/20/05.
https://doi.org/10.1145/3387940.3391536