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 has been hugely influential in the software industry for its emphasis on rapid iteration. It was originally designed for small, colocated teams [3], yet is used today even in large organizations of 20,000+ people, 78% of the time with multiple distributed teams [1]. This tension has led to a multitude of frameworks for employing agile practices in the large, the most popular [1] of which is the Scaled Agile Framework (SAFe) [2].

These frameworks aim to streamline coordination between teams and stakeholders; for example, SAFe organizes feature teams into *release trains* which focus on delivering units of business value, and time into *program increments* (PIs) which are extensively planned for. However, SAFe is challenging to actually implement [8]. Large projects require appropriate coordination and communication between teams [6] and cross-team dependencies cause huge communication overhead [7]; this is the fundamental problem SAFe tackles. Continuous Planning [4] is one solution; similar to how CI/CD reduces costs by automating software releases, automating the busywork involved in executing a planning session would enable rapid iteration, freeing time for collaboration. To this end, we implement a tool, Sapling, which facilitates collaboration across remote teams when carrying out SAFe PI Planning.

2 APPROACH & SAPLING TOOL

Sapling is a web application (Figure 1) which implements the workflow and processes that feature 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 *stories* and assigns them a *story point weight* representing how much effort the story will take. 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; that is the reason SAFe recommends that all teams should be colocated while planning. It is common to walk over to another team, hand them a sticky note representing a story, and have them fit it into their plan somehow. Sapling codifies this, allowing teams to send each other requests to add stories to each other's plans and see how their plans change before accepting.

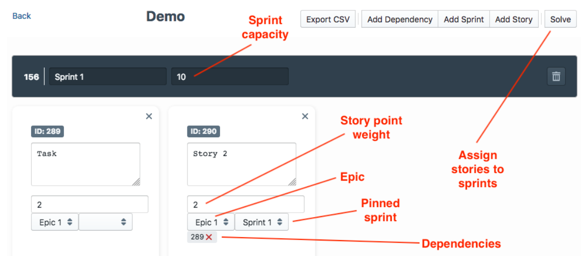

Figure 1: Sapling's main view: a story assigned to a sprint

To enable this, Sapling automates the assignment of stories to sprints entirely 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. We also support constraints involving story dependencies, stories which *must* be allocated to a given sprint, and epic priority.

Employing 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 stories.

3 DISCUSSION

Initial feedback has been positive: stakeholders appreciate the increased visibility and planning sessions proceed more seamlessly. Future work will focus on adding other soft constraints and improving the user experience of understanding 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.