CASE STUDY

Using Sapience data for Sprint Retrospectives

sapience

Background

Agile software development – the use of a set of methods and practices where solutions evolve through collaboration between self-organizing, cross-functional teams – has become very popular in recent years. According to the VersionOne 2019 State of Agile Survey, 97% of respondents were practicing Agile. Though the survey participants were mostly from North America and Europe, Agile development is popular all over the world, including in India.

The Scrum management framework and its variants are by far the most popular agile methodology. A key aspect of Scrum is the Sprint Retrospective, the final event of each Sprint during which teams reflect on their contributions and work approach then make an actionable plan for future improvements. It is a key component of Agile which focusses on inspection and adaptation.

The challenges

Sprint Retrospectives can prove to be invaluable. However, often the Retrospective is the most dreaded of all meetings. There are very few committed participants. Often, it becomes a gripe session where managers and senior managers vent on how things didn't go as planned and how fewer stories than planned were completed. The list of "what didn't go well" is long compared with "what went well". And sometimes it's a mere formality to check off this item in the Scrum guidelines, with little or no actionable goals and improvements.

Even when the team has put in significant effort, the Sprint Retrospective meeting may indicate inadequate performance, and fewer completed stories than planned. Developers may blame this on bug backlog, customer issues, and other unplanned effort, reducing the time available for new stories. At times, the stories are not well-defined resulting in re-work and more meetings to clarify the specs. There is a cascading effect on the quality assurance team, which gets multiple low-quality builds to test. These complaints fall on deaf years as senior management and the customer are largely focused on the end results. The blame game can have an adverse effect on a team's morale. After all, it's only natural to feel bitter and resentful when three to four weeks of hard work are undervalued and/or considered to be inadequate.

The findings

Sapience data analysis of the activities in each Sprint can identify the root cause and provide continuous inspection and improvement to help achieve more productivity and completed stories in your teams' Sprints.

- Correlating 'work' time with Sprint achievements
- Root cause analysis of a poor Sprint based on time on various activities
- Reviewing estimates and fine tuning them
- Setting actionable commitments for future Sprints
- Continuous inspection and improvement

The solution

With factual data, you obtain valuable insights in assessing the true reality and underlying factors behind why user stories are not being completed.

Sapience provides a feature that enables managers to define 'phases' for a project. A phase can be anything, such as 'development' or 'design' in the case of a traditional waterfall model, or the successive Sprints in the case of Scrum. Each phase has a start and an end date, which, in this the example, will correspond to the Sprint start/end dates:

Phase	Start	End
Sprint 1	July 1	July 22
Sprint 2	July 28	Aug 17
Sprint 3	Aug 25	Sept 7
Sprint 4	Sept 15	Oct 5

Sapience trends related to work effort break-down across purposes and activities can then be viewed in terms of phases. The Purpose tab will show the team's daily average work time (on-PC and off-PC in meetings, etc.) on the project and any other company work for each of the Sprints. The Activity tab will show the break-down across key activities in each of the Sprints. During the Sprint Retrospective meeting, these insights can be valuable in assessing the true reality and underlying factors behind why user stories are not being completed.

The phase-wise analysis in Sapience is retrospective, though you can also compare the trend for a partial Sprint.