Funktionale Sicherheit in agilen Software-Projekten
Wie STPA und BDD helfen können

Prof. Dr. Stefan Wagner, Institut für Softwaretechnologie, Universität Stuttgart

Während agiles Software Engineering stark verändert hat, wie in der Praxis Software entwickelt wird, spielt es bei sicherheitskritischen Systemen noch eine untergeordnete Rolle. Die Integration von Sicherheitsanalysen in ein Vorgehen nach Scrum ist aber möglich ohne an Agilität zu verlieren. Im Folgenden betrachten wir ein Beispiel dafür und verknüpfen die Sicherheitsanalysen mit Behaviour-Driven Development zur verbesserten Qualitätssicherung.

Einleitung

Software wird zunehmend eine zentrale Komponente in sicherheitskritischen Systemen, da sie oft für die Steuerung der Systeme verantwortlich ist. In allen sicherheitskritischen Domänen, sei es Fahrzeugbau, Flugzeugbau oder auch Werkzeugmaschinen, ist die Digitalisierung und damit auch der massive Einsatz von Software heute angekommen.

Für die Entwicklung von Software außerhalb eingebetteter und sicherheitskritischer Systeme setzt sich zunehmend das agile Software Engineering mit selbstorganisierenden Teams, kurzen Zyklen und flexiblem Anforderungsmanagement durch. Dadurch entstehen am Anfang des Projekts aber keine umfassenden und detaillierten Anforderungsdokumente und Architekturentwürfe, was im Widerspruch zu vielen Sicherheitsanalysemethoden und Sicherheitsstandards steht. Dort erwartet man eine detaillierte Architektur des Systems, um die Analysen durchführen und die Ergebnisse für eine Zertifizierung dokumentieren zu können.

Wie kann man sich dieser Herausforderung nun stellen? Wie kann man die Vorteile der agilen Software-Entwicklung auch für sicherheitskritische Systeme nutzen?

Sicherheitsanalyse in Scrum mit STPA

Der am häufigsten verwendete agile Entwicklungsprozess ist Scrum [1]. Er schreibt die Rollen des Scrum Masters, des Product Owners und des Teams vor. Scrum betont, dass das Team selbstorganisierend ist, während der Scrum Master auf die Einhaltung der Scrum-Regeln achtet und der Product Owner die Schnittstelle zum Kunden ist. Anforderungen werden in einem Backlog gehalten und priorisiert. Anforderungen aus dem Backlog werden dann in zwei- bis vierwöchigen Sprints abgearbeitet. Die Sprints bestehen neben der eigentlichen Entwicklungsarbeit aus einem Planungstreffen, dem Sprint-Review und der Retrospektive. Im Planungstreffen wird festgelegt, was im Sprint getan werden soll. Im Sprint-Review wird am Ende des Sprints ein potentiell auslieferbares Produkt-Inkrement dem Kunden präsentiert. In der Retrospektive sammelt das Team Prozessverbesserungen.

Es gibt einige wenige Vorschläge, wie Scrum genutzt werden kann, um sicherheitskritische Systeme zu entwickeln. Safe Scrum [2] und R-Scrum [3] sind die zwei bekanntesten Beispiele, die es bereits schaffen viele Dokumentationsanforderungen in Scrum einzubauen, dass die Anforderungen aus Standards wie der IEC 61508 eingehalten werden können. Eine Schwierigkeit bleibt aber, dass die Anteile, die die funktionale Sicherheit betreffen außerhalb des Sprints bleiben. Die Agilität leidet hier etwas.

Deshalb haben wir uns dazu entschieden, als Sicherheitsanalysemethode STPA (System-Theoretic Process Analysis) [4] einzusetzen. Die von Nancy Leveson am MIT entwickelte Methode bringt für uns hier zwei entscheidende Vorteile: (1) Es geht von einer iterativen Weiterentwicklung von System und Sicherheitsanalyse im Wechsel aus und (2) fokussiert sich im Gegensatz zu anderen Methoden nicht auf einzelne Komponenten sondern auf deren Zusammenspiel, was die Einbindung von Software aber auch Nutzern einfach ermöglicht.

Wir haben als entsprechende Scrum-Erweiterung *S-Scrum* [5] vorgeschlagen, das in vielen Aspekten stark an Safe Scrum orientiert ist. Der grundlegende Prozess ist in Abbildung 1 dargestellt. Hier gibt es zwar auch vor den Sprints eine Software-Sicherheitsspezifikation (SSRS), aber die Durchführung von STPA ist innerhalb des Sprints und sogar im täglichen Ablauf. Die Kommunikation zwischen Sicherheitsexperten und Team ist durch ein explizites Regular Safety Meeting sichergestellt, das nicht unbedingt täglich durchgeführt werden muss. Die direkte Einbindung der Ergebnisse wird durch eine Abbildung auf Tests in der kontinuierlichen Integration (TDD/BDD/CI) erzielt. Nach den Sprints gibt es noch eine *Final STPA Validation*, die alle Sicherheitsanforderungen nochmal explizit validiert, damit dies in die Dokumentation einfließen kann.

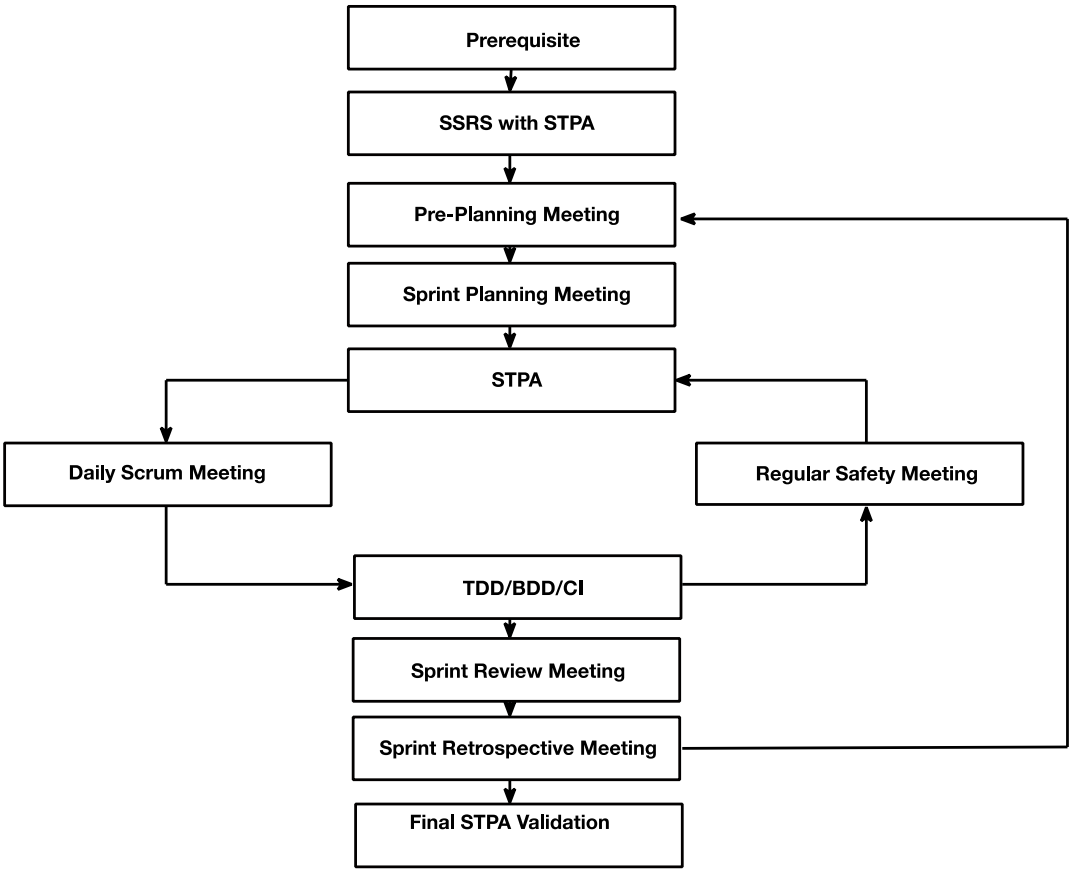

Abbildung 1: Vereinfachter Ablauf von S-Scrum

Trotz doch einiger zusätzlicher Aktivitäten und Dokumente, die in S-Scrum gegenüber Scrum existieren, konnten wir in einer Studie [5] zeigen, dass die Agilität

aus Sicht der Entwickler darunter kaum leidet. Wir sind also der agilen Entwicklung sicherheitskritischer Systeme einen Schritt näher gekommen.

Behaviour-Driven Development und Sicherheitsanforderungen
Neben der Einbindung einer geeigneten Sicherheitsanalysemethode ist auch eine einfache Verknüpfung zu agiler Qualitätssicherung notwendig, um in der Entwicklung sicherheitskritischer Software agil sein zu können. Aus der agilen Software-Entwicklung stammt ein Ansatz, der vielversprechend ist, um ihn mit Sicherheitsanforderungen zu verbinden: *Behaviour-Driven Development* (BDD) [6].

Die Grundidee ist hier, dass das Verhalten vorab mit der Hilfe von Beispielen und Szenarien in einem sogenannten *Three Amigos Meeting*, bestehend aus einem Entwickler, einem Tester und einem Kundenvertreter, spezifiziert wird. Die Spezifikation ist dabei meist in Form von einfachem natürlichsprachigen Text. Beispielsweise wird dazu die Sprache *Gherkin* verwendet. Diese Gherkin-Spezifikationen werden dann von Testern automatisiert eingelesen und als Testfall gegen das System geprüft. Diese Spezifikation und der Test werden geschrieben, bevor das spezifizierte Szenario implementiert ist, sodass die Entwickler dann ein klares Kriterium haben, wann sie eine Anforderung erfüllt haben: Dann, wenn alle Tests erfolgreich durchgeführt werden können.

Das spannende für Sicherheitsanalysen ist, dass wir uns bei der funktionalen Sicherheit auch mit dem Verhalten des Systems beschäftigen: Mit welchem Verhalten kann das System in einen unsicheren Zustand kommen und eine Gefährdung auslösen? Damit lassen sich die Sicherheitsanalysen, die wir in S-Scrum mit der STPA entwickeln leicht in Gherkin-Spezifikationen umbauen, die wiederum als Testspezifikation dienen können. Ein Beispiel einer solchen Spezifikation ist in Abbildung 2 gezeigt.

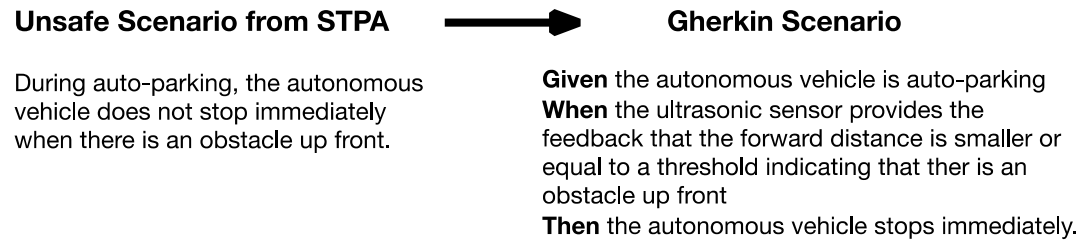

Abbildung 2: Ein Beispiel für ein Unsafe Scenario und zugehöriges Szenario in BDD

Wir haben diese Verknüpfung der STPA-Sicherheitsanalyse und BDD in Experimenten untersucht [7]. Wir konnten dabei zeigen, dass gegenüber dem herkömmlichen Erstellen von Akzeptanztests insbesondere die Kommunikation zwischen den Beteiligten durch BDD erhöht wurde. Das Three Amigos Meeting zusammen mit der natürlichsprachigen Szenario-Spezifikation scheint es deutlich leichter zu machen, um über die Anforderungen und Tests zu sprechen. Damit ist BDD eine sehr vielversprechende Methode für den Kontext sicherheitskritischer Systeme.

Zusammenfassung und Ausblick

Unsere Ergebnisse haben das Thema sicher noch nicht erschöpfend gelöst. Durch S-Scrum mit eingebettetem BDD steht aber nun ein empirisch erprobter Entwicklungsprozess zur Verfügung, der sowohl iterative Sicherheitsanalyse als auch eine enge Integration mit agiler Qualitätssicherung erlaubt. Wir gehen davon aus, dass dies aber nur der Start ist. Um die empirischen Ergebnisse zu erhärten suchen wir Partner für weitere Studien. Auch BDD und die dazugehörige Art Szenarios zu spezifizieren könnte noch viele weitere Bereiche in der Entwicklung eingebetteter Software-Systeme unterstützen. Der Fokus auf einfaches Verständnis und Kommunikation bei gleichzeitig vollautomatischer Testausführung stellt eine interessante Kombination für viele Arten von Anforderungen dar. Wir planen auch hier weiter die Möglichkeiten zu untersuchen.

Literaturverzeichnis

1. www.scrumguides.org
2. T. Stålhane, T. Myklebust, G. Hanssen. The application of Safe Scrum to IEC 61508 certifiable software. In: *Proceedings of the 11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference*. 2012.
3. B. Fitzgerald, K.-J. Stol, R. O'Sullivan, D. O'Brien. Scaling agile methods to regulated environments: An industry case study. In: *Proceedings of the 35th International Conference on Software Engineering*. IEEE, 2013.
4. N. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2011.
5. Y. Wang, J. Ramadani, S. Wagner. An exploratory study of applying a Scrum development process for safety-critical systems. In: *Proceedings of the 2017 International Conference on Product-Focused Software Improvement*. Springer, 2017.
6. M. Wynne, A. Hellesoy. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers.* Pragmatic Bookshelf, 2012.
7. Y. Wang, S. Wagner. Combining STPA and BDD for safety analysis and verification in agile development: A controlled experiment. In: *Proceedings of the 2018 International Conference on Agile Software Development. Springer*, 2018.

Autor

Stefan Wagner ist Professor für Software Engineering und Geschäftsführender Direktor des Instituts für Softwaretechnologie der Universität Stuttgart. Er studierte Informatik in Augsburg und Edinburgh und promovierte an der TU München. Seine Forschungsschwerpunkte sind Requirements Engineering, Software-Qualität, funktionale Sicherheit und agiles Software Engineering. Er bearbeitet diese Themen gerne mit empirischen und psychologischen Methoden in enger Kooperation mit der Industrie. Daneben ist er zu diesen Themen auch als Berater und Trainer freiberuflich tätig. Er ist Mitglied bei GI, ACM und IEEE.

Kontakt

Internet: www.iste.uni-stuttgart.de/se, www.stefan-wagner.biz
Email: stefan.wagner@iste.uni-stuttgart.de