Agiles Vorgehen, LabVIEW und Fixpreis schliessen sich nicht aus

1 Februar 2016 | **Insight Zühlke** | **Amadeo Vergés**
Lesezeit: 4 Minutes

Während heute der agile Software-Entwicklungsprozess in klassischen Softwareprojekten standardmässig zum Einsatz kommt, ist dies im industriellen Bereich oder bei LabVIEW-Anwendungen noch nicht überall der Fall. Im nachfolgend beschriebenen Projekt hat Zühlke die Software mit dem agilen Entwicklungsprozess erfolgreich realisiert, wobei die Software terminlich den kritischen Pfad darstellte und ein Fixpreis vorgegeben war.

Das Projekt

Für ein internationales Industrie-Unternehmen haben wir ein grosses und komplexes Testlabor schlüsselfertig realisiert. Dieses dient dazu, sicherheitsrelevante Sensoren unter verschiedensten Umgebungseinflüssen in Grenzbereichen prüfen zu können. Neben den komplexen und vernetzten Hardwarekomponenten, mit über 300 in Echtzeitsignalen, ist auch die entsprechende kundenspezifische Software Teil des Lieferumfanges. Zur Realisierung wählten wir LabVIEW unter Anwendung des objektorientierten Ansatzes (LVOOP). Die Testsoftware ist für die Steuerung der kompletten Anlage, die Datenerfassung, -verarbeitung und -auswertung verantwortlich.

Das Projekt wurde zum Fixpreis realisiert und war in eine Konzept- und Realisierungsphase unterteilt.

Wie wurde die LabVIEW-Software agil realisiert?

Für die Erfassung der Anforderungen und der Testfälle verwendeten wir das Softwaretool „Polarion“. Jedes Team-Mitglied und auch die Mitarbeiter des Kunden (*Product-Owner* und S*ystem-Tester*) hatten jederzeit einen webbasierten Zugriff auf die Inhalte.

Beim Projektstart wurden die Anforderungen in Form von *User-Stories* in Polarion beschrieben und nur soweit verfeinert, um eine Aufwandsschätzung mit den sogenannten *Story-Point*s und eine Priorisierung vornehmen zu können. Die Summe aller *Story-Points* stellt den gesamten, aber nicht skalierten Arbeitsumfang des Softwareprojektes dar. Während des Projektes kann die Summe der abgearbeiteten *Story-Points* mit dem effektiven Aufwand verglichen werden. Als Ergebnis resultiert daraus die „Geschwindigkeit“ (*Velocity*), mit welcher man vorwärts kommt. Der Restaufwand kann dadurch genauer geschätzt werden, weil laufend eine aktuelle Skalierung der *Story-Points* vorliegt.

Die Implementierung erfolgte in Form von zweiwöchigen *Sprints*. Zu Beginn des *Sprints* bzw. schon während des vorherigen *Sprints* wurden die *User-Stories* für den Sprint in der Reihenfolge ihrer Priorität und der technischen Erfordernisse (Verfügbarkeit der Mitarbeiter, Umsetzung von Funktionsgruppen) ausgewählt und für ihre Implementierung verfeinert. Die Umsetzung erfolgte, indem entsprechende Testfälle erstellt wurden, und nicht indem die *User-Stories* selbst detailliert wurden.

Vorteile dieses Vorgehens:

- Nach jedem *Sprint* liegt ein konkretes Arbeitsergebnis vor, welches sofort vom Kunden überprüft werden kann.
- Als Grundlagen für die Implementierung werden die Testfälle selbst verwendet. Die Anforderungen sind damit sehr konkret und es muss nur der Funktionsumfang implementiert werden, bis die Testfälle erfüllt sind.
- Die zusätzliche Detaillierung der *User-Stories* selbst kann gespart werden.
- Am Ende jedes Sprints kann die implementierten *User-Stories* sofort mit Hilfe der bereits fertiggestellten Testfälle getestet werden.
  Durch die kontinuierliche Ermittlung der Implementierungsgeschwindigkeit des Teams (*Velocity*), konnte der Aufwand bzw. der Fertigstellungstermin schon sehr früh verifiziert werden. Das iterative Vorgehen ermöglichte es auch einfach und kostengünstig *User-Stories* wieder zu streichen und nötigenfalls neue User-Stories aufzunehmen, weil noch kein Aufwand für die Detaillierung und Umsetzung der User-Stories angefallen war.

Dank dem Einsatz von *Polarion* war der Zustand der verschieden Work-Items (*User-Stories*, Testfälle, Fehler) für alle Projektmitarbeiter und den Kunden transparent und der Kunde (Product-Owner) war in der Lage, die Testfälle zeitnah zu reviewen. Die Ergebnisse der Testläufe (insbesondere die aufgetretenen Fehler) wurden auch in Polarion eingetragen, so dass die Fehlerbeseitigung zeitnah erfolgen konnte.

Durch die Erfassung der Anforderungen (*User-Storie*s), der Testfälle und ihrer Abhängigkeiten in Polarion war die Nachvollziehbarkeit (*Tracability*) jederzeit gegeben.

An den Sprint-Meetings waren wenn möglich immer alle Teammitglieder anwesend. Damit war auch der direkte Kundenkontakt und das unmittelbare Feedback gewährleistet.

Die Moral von der Geschichte

Agiles Vorgehen setzt die entsprechenden Kenntnisse und Fähigkeiten voraus, um den Prozess konkret anzuwenden. Selbstverständlich auch fundiertes LabVIEW-Know-How, denn der Prozess alleine garantiert nicht automatisch auch guten Code!

Das Software-Projektteam sah, gemittelt über die gesamte Projektdauer (ca. 11 Monate), vereinfacht in etwa wie folgt aus:

- 1 Agiler-Prozess Coach, der sich auch mit dem Requirements Engineering und den Test-Spezifikationen beschäftigte, 50%
- 1 LabVIEW-Architekt (CLA), 100 %
- 2 bis 3 LabVIEW-Developer (CLD), je 100 %

Und bei Ihnen?

Wie sehen Ihre Softwareprojekte im vergleichbaren industriellen Umfeld aus? Welche Erfahrungen mit dem agilen Vorgehen haben Sie gemacht?

Ich bin interessiert an Ihrem Feedback!