Dr. Ernst Sikora

(ernst@sikora-online.de)
ist in leitender Position in der Entwicklung vorausschauender Fahrzeug-Sicherheitsfunktionen tätig. Er gestaltet den Produkt- und Systementwicklungsprozess unter Verwendung vielfältiger Methoden. Sein Werkzeugkasten beinhaltet neben einem Hammer sowohl klassische wie auch agile Techniken, textuelle wie auch modellbasierte Spezifikation, Reviews ebenso wie Prototyping und Simulation.

objektspektrum themenspecial: Requirements Engineering

Kein Requirements Engineering ist auch keine Lösung

Auch wenn es immer wieder versucht wird – ohne ein leistungsfähiges, zielorientiertes Requirements Engineering (RE) ist es kaum möglich, wirksame Lösungen und damit erfolgreiche Produkte zu entwickeln. Doch was gehört zu einem leistungsfähigen Requirements Engineering: Soll der Requirements Engineer nur die Stakeholder-Wünsche dokumentieren oder gestaltet er die Anforderungen selbst mit? Was gehört in die Anforderungsspezifikation mit rein und was nicht?

Es gibt zahlreiche Bemühungen, diese Fragen zufriedenstellend zu beantworten. Doch eine Klärung, die auf der Unterscheidung zwischen „Problem“ und „Lösung“ oder dem „Was?“ und dem „Wie?“ beruht, greift zu kurz. Um von der Idee zum Produkt zu kommen, müssen einerseits Entscheidungen über die Produkteigenschaften getroffen werden und andererseits Lösungen für jedes kleinere und größere Problem erarbeitet werden. Dies kann nur gelingen, wenn die Projektbeteiligten die Bereitschaft und die Zeit haben, tief genug in die Materie einzutauchen und sich dabei nicht von Rückschlägen beirren lassen.

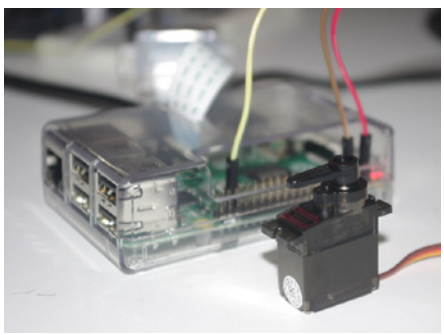
Ein Gedankenspiel: Das Ein-Mann-Projekt

Um einer Klärung, was RE leisten sollte, näher zu kommen, schauen wir uns zunächst exemplarisch den Verlauf eines Ein-Mann-Projekts an, in dem es naturgemäß noch wenig Spielraum für „Verantwortlichkeitsgerangel“ zwischen unterschiedlichen Projektrollen gibt.

Projekttagebuch Eintrag Nr. 1
Ich blicke auf einen blinkenden Cursor. Ich möchte etwas programmieren, aber mir fällt einfach nicht ein, was das Programm tun soll.

Projekttagebuch Eintrag Nr. 2
Heute Morgen beim Duschen kam mir eine Idee. Zu Weihnachten habe ich einen Raspberry Pi geschenkt bekommen. Mit dem werde ich einen Servo ansteuern, der später in einem Modellauto als Stellglied für die Lenkung dienen wird. Morgen überlege ich mir die Details.

Projekttagebuch Eintrag Nr. 3
Die Ansteuerung des Servos sollte kein Problem sein. Dem Datenblatt habe ich entnommen, dass der Servo ein PWM-Signal mit 50 Hz und einer Pulsdauer zwischen 1 und 2 Millisekunden erwartet. Bei einer Pulsdauer von 1 ms fährt der Servo in die linke Position, bei 2 ms in die rechte. Noch ein bis zwei Stunden, dann sollte alles laufen.



Projekttagebuch Eintrag Nr. 4
Ich habe ein Python-Programm zur Ausgabe der Soll-Position an den Servo geschrieben. Doch die Ansteuerung gestaltet sich schwieriger als gedacht. Der Servo-Arm bewegt sich sehr unruhig und zittrig. Daraufhin habe im Internet recherchiert und herausgefunden, dass das von der Standard-Input/Output-Bibliothek erzeugte PWM-Signal für eine Servo-Ansteuerung zu ungenau ist. Jetzt muss ich nach alternativen Lösungen suchen.

Dasselbe Projekt mit mehreren Stakeholdern

Der obige Projekttagebuch-Auszug verdeutlicht, wie wichtig die richtigen Vorgaben und Detailinformationen für die Softwareentwicklung sind. Zunächst gilt es, den Projektumfang grob abzustecken. Im weiteren Verlauf müssen Systemschnittstellen präzise definiert und ein tiefgehendes Verständnis über die Systemumgebung, in unserem Beispiel über einen Servo, erlangt werden. Und neben den funktionalen Eigenschaften, müssen auch Qualitätseigenschaften, wie z. B. die notwendige Genauigkeit eines Ansteuersignals, durchdacht und festgelegt werden.

Natürlich wird die Angelegenheit komplizierter, wenn nicht nur eine Person im Spiel ist, sondern mehrere Stakeholder wie Kunden, Programmierer, Tester und Qualitätsmanager im Projekt mitreden. Dann könnte zum Beispiel das folgende Gespräch stattfinden:
Programmierer: „Ich habe alles eins-zu-eins so umgesetzt, wie es in der Spezifikation steht."
Requirements Engineer: „Es war schwierig genug, vom Kunden in Erfahrung zu bringen, was er überhaupt will. Ich kann mich nicht um jedes kleine Detail kümmern. Und das Servo-Datenblatt habe ich mit beigefügt."
Programmierer: „Da steht aber nicht drin, dass der Servo so empfindlich auf kleinste Signalungenauigkeiten reagiert."
Requirements Engineer: „Gut, dann schreibe ich das in die Spezifikation rein. So oder so muss die Software geändert werden, und den Test müssen wir auch wiederholen. Und wer soll jetzt dem Kunden mitteilen, dass wir erst eine Woche später ausliefern können?"

Wie dieser kurze Dialog zeigt, sind es oft die Feinheiten, auf die es ankommt. Doch im Projektalltag sind es häufig genau diese Feinheiten, für die niemand genügend Zeit hat. Sollte sich also der Requirements Engineer für derartige Themen verantwortlich fühlen, oder sind dies Lösungsdetails, die in den Softwareanforderungen tunlichst nichts zu suchen haben?

Verantwortung des Requirements Engineers

Meiner Ansicht nach sollten die für die Softwareentwicklung notwendigen Recherche- und Experimentieraktivitäten nicht an den Entwurf, die Implementierung oder den Test delegiert werden. Die Softwarearchitekten, -designer, -programmierer und -tester haben die bereits genügend herausfordernde Aufgabe, anhand der Anforderungen ein robustes, performantes, flexibles, wartbares, sicheres – die Aufzählung lässt sich noch fortsetzen – Softwaresystem zu konstruieren.

Bei einer arbeitsteiligen Entwicklung trägt der Software Requirements Engineer die Verantwortung dafür, dass die Softwarekonstrukteure auch in Detailaspekten eine solide Arbeitsgrundlage bekommen und so wenig wie möglich „für den Papierkorb" arbeiten müssen.

Gleichwohl ist dies nicht immer in Gänze erfüllbar, denn in einer innovativen und komplexen Entwicklung lässt sich nicht alles vorhersehen. Damit müssen alle Beteiligten leben, denn sonst wäre es keine Softwareentwicklung, sondern eine Softwareproduktion, in der mit hoher Perfektion immer wieder das gleiche Werkstück erzeugt wird. Innovation hingegen beinhaltet immer auch Rückschläge, aus jedem Rückschlag ergeben sich neue Erkenntnisse und es erfolgt ein neuer Versuch, eine bessere Lösung zu finden.

Facetten des Requirements Engineering

Die Autoren dieses Themenspecials stellen verschiedene Facetten eines leistungsfähigen RE dar und geben Ihnen zahlreiche Anregungen dafür, das RE zum Motor Ihrer unternehmerischen Wertschöpfungskette auszugestalten. Dies beginnt schon damit, wie RE im Grundsatz verstanden und gelebt wird.

Dr. Kim Lauenroth tritt der Auslegung des RE als „Schreiben von Dokumenten, die niemand lesen möchte" entgegen. In seinem Artikel arbeitet er die Anteile aus Wissenschaft, Ingenieurwesen und Gestaltung heraus, die in einem richtig verstanden RE vorzufinden sein sollten.

Larissa Endriss und **Markus Eberhardt** befassen sich in ihrem Beitrag mit den organisatorischen Voraussetzungen für die erfolgreiche Entwicklung komplexer und innovativer Systeme. Ihr Beitrag beruht auf der Beobachtung, dass Systemarchitekturen häufig die Organisationsstruktur des Unternehmens widerspiegeln, von dem das System entwickelt wird. Die Autoren geben dabei auch Anregungen zur Weiterentwicklung der Organisationsstruktur.

Jörg Holtmann, **Markus Fockel**, **Thorsten Koch** und **David Schmelter** charakterisieren unterschiedliche Ausgestaltungen des RE in Unternehmen anhand eines Reifegradmodells. Die Reifegrade decken ein Spektrum von unvollständig (RG 0) über systematisch (RG 3) bis hin zu optimiert (RG 5) ab. In dem Beitrag wird für jeden Reifegrad die Ausprägung des RE hinsichtlich der Dokumentation der Anforderungen, der Anforderungs-Traceability und des RE-Prozesses dargestellt.

Dr. Sebastian Adam, **Dr. Norman Riegel** und **Özgür Ünalan** setzen sich mit typischen Vorbehalten gegenüber der Einführung eines Werkzeugs für RE auseinander. Die Autoren plädieren für eine frühzeitige Unterstützung des Entwicklungsteams durch Werkzeuge. Gerade Teams, die noch über wenig Erfahrung im Bereich RE verfügen, können von einer werkzeuggestützten Prozessanleitung sowie von den Überprüfungs- und Verwaltungsfunktionen eines fortgeschrittenen RE-Werkzeugs profitieren.

Hendrik Richter gibt einen Überblick über NLP (Neurolinguistisches Programmieren), ein Modell, das seine Wurzeln in der Psychologie hat und im RE zum erfolgreicheren Identifizieren, Dokumentieren und Abstimmen von Anforderungen beiträgt. Beispielsweise postuliert NLP, dass jeder Mensch ein bis zwei bevorzugte Wahrnehmungskanäle hat, visuell, auditiv oder kinästhetisch.

Stan Bühne berichtet in seinem Beitrag über die Einführung von Compliance-Bewertungen in Softwareprojekten. Compliance-Bewertungen werden von dem Softwarelieferanten zu wichtigen Projektmeilensteinen erstellt und verschaffen dem Auftraggeber Transparenz, welche Anforderungen zu welchem Grad bei dem jeweiligen Projektmeilenstein erfüllt sind. ■

Viel Vergnügen beim Lesen dieses Themenspecials!

Dr. Ernst Sikora