Peter Maurer

(E-Mail: p.maurer@maurer-treutner.de)
ist Diplom-Physiker und beschäftigt sich seit über 15 Jahren mit Software- und Systementwicklung sowie Architektur für Embedded und Echtzeitsysteme. Als Geschäftsführer seiner Firma Maurer & Treutner steht er seinen Kunden bei der Einführung und Betreuung agiler Prozesse und als Systemarchitekt zur Seite. Daneben arbeitet er auch als Trainer in diesem Bereich.

Best Practice für die Anforderungsmodellierung in agilen Projekten

Beim Einsatz agiler Entwicklungsmethoden gibt es keine festen Vorgaben für den Umgang mit Requirements. Bei der Entwicklung von Echtzeit- und Embedded-Systemen mit Sicherheitsfunktionen ist es wichtig, den Überblick über funktionale und nichtfunktionale Anforderungen zu behalten. In unseren Projekten hat es sich deshalb bewährt, eine begleitende SysML-Modellierung [omg] der Requirements einzusetzen, die mit dem Product Backlog synchron gehalten wird. Neben den eigentlichen Anforderungen setzen wir Test Cases und teilweise auch Use Cases ein. Das Requirementsmodell wächst dabei zusammen mit dem Produkt, wichtige Anforderungen – insbesondere sicherheitsrelevante – werden dabei kontinuierlich nachverfolgt, sich widersprechende Anforderungen fallen rasch auf.

Das Ziel des klassischen Anforderungsmanagements ist es, ein vollständiges Modell aller relevanten Anforderungen an das jeweilige System zu erhalten. Im Gegensatz dazu geht man bei einem agilen Vorgehen in der Entwicklung von einem wachsenden und über Anforderungsänderungen reifenden Produkt aus, d. h. das Wissen über die Anforderungen wächst und verändert sich wie das Produkt selbst im Laufe der Entwicklung.

Die gängigen agilen Entwicklungsmethoden treffen nur wenige oder gar keine expliziten Aussagen zum Thema Anforderungsmanagement. Im agilen Umfeld wird dieses Thema deshalb häufig ausgespart – teilweise wird sogar die Auffassung vertreten, dass Anforderungsmanagement und Agilität im Widerspruch zueinander stünden.

In der Entwicklung von Echtzeit- und Embedded-Systemen gibt es jedoch gravierende Gründe, die dafür sprechen, ein methodisches Anforderungsmanagement einzusetzen:

- Systematisches Anforderungsmanagement macht Zusammenhänge zwischen Anforderungen explizit, die sonst nicht erkannt werden. Bei Anforderungsänderungen können dadurch Widersprüche erkannt werden – häufig treten diese zum Beispiel zwischen funktionalen und nichtfunktionalen Requirements auf.
- Durch Anforderungsmanagement werden Zusammenhänge zwischen Hardwareelementen und Anforderungen deutlich sichtbar. Der Einfluss von Anforderungsänderungen auf die Hardware oder von Hardwareänderungen auf bestimmte Anforderungen wird dadurch frühzeitig erkannt. Dies ist insbesondere auch im Hinblick auf die im Vergleich zur Software langen Entwicklungszyklen bei der Hardware wichtig. So können Mehrkosten im Griff gehalten oder Verzögerungen verhindert werden.
- Der Zusammenhang zwischen Integrationstests und den zu testenden Anforderungen wird erst durch ein Anforderungsmanagement explizit. Dadurch kann die Testabdeckung verbessert und somit die Qualität der Produkte erhöht werden.
- Für Systeme mit sicherheitsgerichteten Funktionen ist ein ständig nachvollziehbares Anforderungsmanagement durch einschlägige Normen sogar zwingend vorgeschrieben.

Jedes dieser Argumente allein spricht schon dafür, die Themen Anforderungsmanagement und -modellierung als einen wertvollen Teil der Produktentwicklung anzusehen, dem auch beim Einsatz agiler Entwicklungsmethoden Platz eingeräumt werden sollte.

Anforderungsmanagement im agilen Umfeld kann und muss keinen Anspruch auf Vollständigkeit haben. Vielmehr geht es darum, zu jedem Zeitpunkt den gerade erforderlichen Überblick über die Anforderungen an das System und ihre Zusammenhänge zu haben. Dem wachsenden Produkt aus dem agilen Entwicklungsprozess wird ein im gleichen Maß wachsendes Requirementsmodell gegenübergestellt, das jederzeit für Validierung und Verifikation herangezogen werden kann. Deshalb hängt es auch sehr stark vom jeweiligen Einzelfall ab, mit welcher Technik und in welcher Tiefe

Requirements Engineering betrieben wird. Die folgenden Ausführungen sind dementsprechend als Beispiel für ein mögliches Vorgehen zu verstehen.

Der Einsatz SysML-basierter Modellierung mit entsprechenden Tools [SMT] verspricht gerade bei der Entwicklung von Embedded Systemen gute Ergebnisse. SysML basiert auf Unified Modeling Language (UML) und erweitert die UML um Elemente für die Modellierung komplexer Systeme. Damit wird eine durchgängige Modellierung aller für das System relevanter Belange – also nicht nur die der Software – ermöglicht.

Für die Anforderungsmodellierung ist SysML dadurch besonders interessant, weil Anforderungen mit ihren Beziehungen untereinander und zu anderen Elementen des Modells dargestellt werden können. Wenn man diese Technik konsequent einsetzt, kann man den Einfluss von Anforderungsänderungen durch das gesamte Modell verfolgen. Neben dem Anforderungsdiagramm führt SysML auch das Zusicherungsdiagramm ein, das es erlaubt, Systemeinschränkungen und Randbedingungen zu modellieren.

Als Ausgangspunkt bei der Entwicklung von Embedded Systemen bietet sich eine Untersuchung der essenziellen Use Cases für das System an. Diese werden direkt aus den grundlegenden funktionalen Anforderungen an das System hergeleitet. Bei Scrum entsprechen diese Requirements den Items des Product Backlogs. Neben den funktionalen Requirements spielen bei Embedded Systemen häufig nichtfunktionale Requirements z. B. in Form von zeitlichen Bedingungen oder Sicherheitsanforderungen eine Rolle. Auch diese werden mit SysML-Requirements modelliert.

Um die Zusammenhänge zwischen verschiedenen Elementen des Modells und nichtfunktionalen Anforderungen zu verdeutlichen, ist es oft sinnvoll, Zusicherungen zu modellieren und in einen entsprechenden Zusammenhang zu setzen.

Die Use Cases werden dabei in einer für das jeweilige Projekt geeigneten Weise – zum Beispiel durch Szenarien – beschrieben. Bei der Analyse der Abläufe und der Diskussion zwischen Entwicklungsteam und Fachabteilung werden in aller Regel wichtige zusätzliche Anforderungen gefunden, die aus den ursprünglichen Anforderungen abgeleitet wurden oder diesen untergeordnet sind oder z. B. aus Normen und Bestimmungen stammen können.

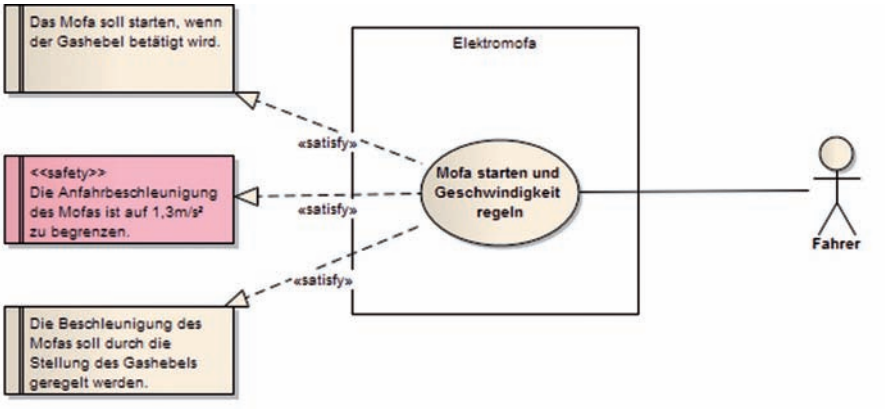

Abb. 1: Das Diagramm zeigt den Zusammenhang zwischen Requirements und essenziellem Use Case.

Bei der Beschreibung der Use Cases lohnt sich auch ein Blick auf den Systemkontext – jedem Aktor müssen entsprechende Schnittstellen für die Kommunikation mit dem System zur Verfügung stehen. Wenn gleichzeitig mit der Software auch die Hardware entwickelt wird, kann es sich lohnen, bei der Betrachtung des Systemkontextes gefundene Beziehungen zwischen Elementen der Hardware-Architektur und den Anforderungen mitzumodellieren. Bei Änderungsanforderungen, von denen die Hardware betroffen ist, helfen diese Beziehungen bei der Validierung des Systems. **Fehler! Textmarke nicht definiert.**

Für den Integrationstest ist es sinnvoll, Test Cases mitzumodellieren, die wiederum mit den durch sie zu verifizierenden Anforderungen in Beziehung stehen. Auch bei Test Cases ist es sehr stark von den

Abb. 2: Das Zusicherungsdiagramm zeigt den Zusammenhang zwischen Anfahrstrombegrenzung und Anfahrbeschleunigung.

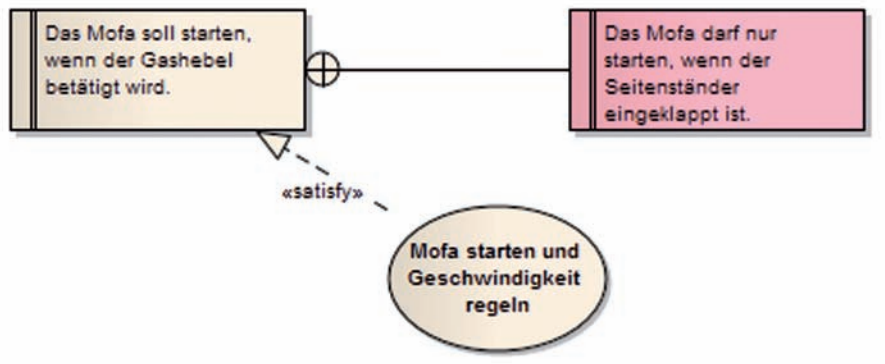

Abb. 3: Eine sicherheitsrelevante Anforderung muss bei der Implementierung des Use Case beachtet werden.

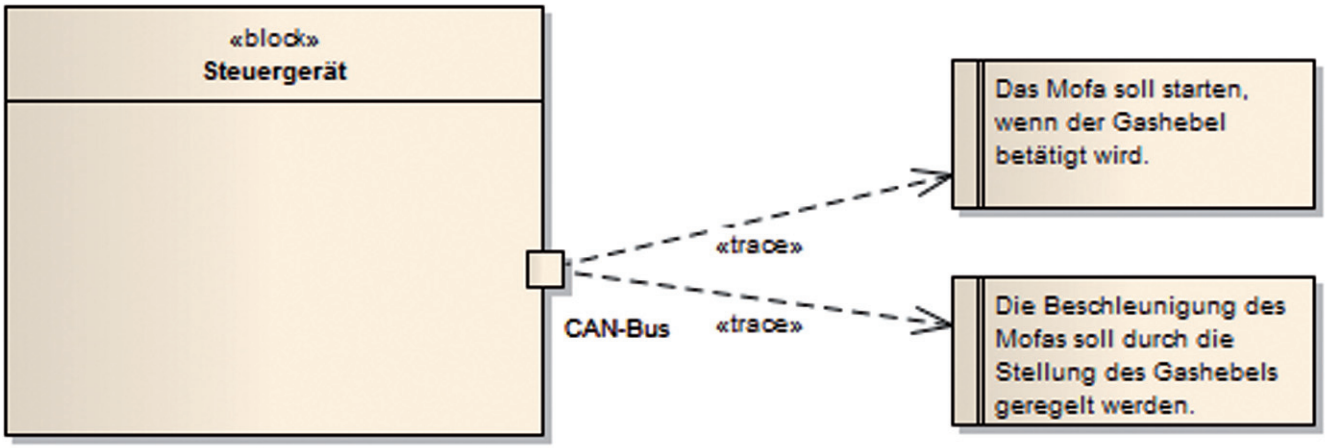

Abb. 4: Elemente der Hardware werden benötigt, um Anforderungen zu erfüllen.

Umständen abhängig, welche Art der Beschreibung gewählt wird und wie tief diese geht. Für Tests mit hohem manuellen Anteil haben sich Ablaufbeschreibungen in Textform oder auch als Aktivitätsdiagramm bewährt – bei automatisierten Testabläufen kann ein Link zum entsprechenden Test genügen.

Wir verstehen in unseren Projekten das Anforderungsmodell als notwendiges Artefakt für das System – die Erweiterung und Pflege des Anforderungsmodells ist entsprechend Teil jeder Iteration. In unseren Augen ist es sinnvoll, wenn die Verantwortung für das Modell bei den Entwicklerteams liegt, die sich den Input bei den Fachabteilungen holen. Je nach Anforderung des Projekts – bei Systemen mit sicherheitsgerichteten Funktionen sind durch entsprechende Normen weitere Vorgaben für das Anforderungsmanagement zu beachten – kann das Anforderungsmodell im Rahmen des regelmäßigen Reviews am Ende jeder Iteration oder in gesonderten Review- oder Auditterminen behandelt werden. Um die jeweils passende Tiefe der Anforderungsmodellierung zu finden, ist es unbedingt erforderlich, in Retrospektivmeetings bewusst auf das Thema Anforderungsmodellierung und die dabei angewandte Methodik einzugehen.

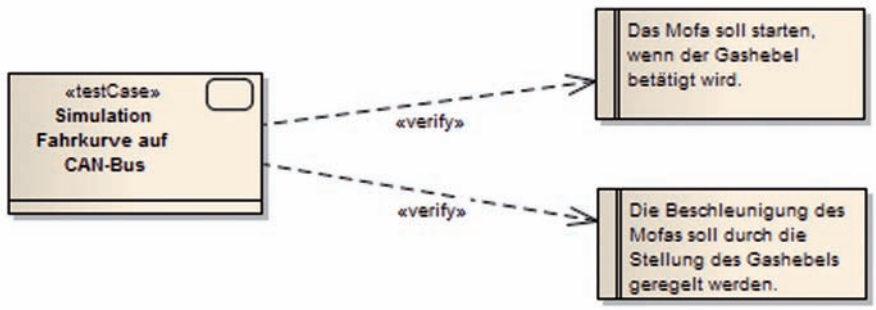

Abb. 5: Test Cases verifizieren ein oder mehrere Requirements.

Das SysML-Anforderungsmodell wird wie das entstehende Produkt ständig weiterentwickelt. Dabei darf es niemals zum Selbstzweck werden, vielmehr muss sich das Anforderungsmodell daran messen lassen, ob es zu einem tieferen Verständnis für das System bei Entwicklern und Fachabteilungen beiträgt. Wenn dies gelingt, hilft das Anforderungsmanagement dabei, Widersprüche und Inkonsistenzen aufzufinden und damit Fehlentwicklungen zu verhindern und trägt so dazu bei, Entwicklungszeit und Kosten zu sparen. ■

Referenzen

[omg] http://www.omg.org/spec/SysML/

[SMT] Einen Überblick über SysML-Tools, finden Sie auf http://www.omgsysml.org/#Vendors