Joachim Pfeffer
Menschen. Entwickeln. Produkte.

V-Modelle – sie sind mitten unter uns

Immer wieder finde ich mich in Beratungsprojekten als Bestandteil einer Diskussion über das V-Modell wieder. Dabei geht es entweder um die Glaubensfrage „V-Modell oder Scrum?“ oder um Unschärfen im Verständnis darüber, was das V-Modell mit der eigenen Arbeit in der Produktentwicklung zu tun hat. Grund genug, hier eine Lanze für das V-Modell zu brechen – zumindest für das, was ich darunter verstehe. Aus meiner Sicht folgt die Entwicklung immer einem V-Modell. Die Frage, ob V-Modell und Scrum vereinbar sind, geht für mich daher am Ziel vorbei. Vielmehr können Gedanken über das eigene V-Modell dazu beitragen, die eigenen Engineering-Praktiken auf eine neue, reifere Stufe zu heben.

Was ist das V-Modell?

Klären wir zunächst einmal, was im üblichen Sprachgebrauch unter „V-Modell“ verstanden wird. Meistens steht dieses Verständnis nämlich im Widerspruch zu der Vielfalt von V-Modellen und Definitionen, die es dort draußen gibt. Falls Sie sich für die Historie des V-Modells interessieren, finden Sie dazu reichlich Informationen im Internet, ich möchte diesen Aspekt hier bewusst kurz halten.

Zum ersten Mal taucht ein V-Modell in den 1970er-Jahren in der Softwareentwicklung auf. Darin wurden verschiedene Tätigkeiten des Software Engineerings in Zusammenhang gesetzt, insbesondere die Zerlegung eines Systems in Unterkomponenten, die anschließende Integration der Komponenten zu einem System sowie die Beziehungen zwischen Definitions- und Testtätigkeiten.

Darauf aufbauend definierte die Bundesregierung in den 1980er-Jahren einen Entwicklungsstandard für Softwareprojekte, der für die Auftragnehmer staatlicher Aufträge verpflichtend wurde. Dies ist also eine andere Definition des V-Modells: das – zumindest in Deutschland – offizielle V-Modell (die Bundesrepublik Deutschland ist auch Inhaberin der Marke „V-Modell“). Das V-Modell der Bundesregierung wurde in mehreren Stufen den neuen Erkenntnissen und Strömungen im Engineering angepasst und existiert heute (2020) als V-Modell XT, das auch agile Ansätze berücksichtigt.

Abseits der staatlichen Vorschriften ist der Ansatz des V-Modells auch in verschiedene andere Normen und Standards eingeflossen. So nehmen zum Beispiel zwei in der Automobilindustrie wichtige Standards – Automotive und ISO 26262 – ebenfalls Anleihen am V-Modell.

In diesem Dokument möchte ich jedoch Prozessdefinitionen und Normen beiseitelassen und mich auf die Denkweise hinter dem V-Modell konzentrieren.

Ein atomares V-Modell

Das V-Modell heißt so, weil die Engineering-Tätigkeiten geometrisch in der Form eines V angeordnet werden. Dadurch lassen sich die Zusammenhänge zwischen den Tätigkeiten leichter beschreiben. Generell beschreibt der linke Schenkel des V die Definition, die Spitze des V den Bau und der rechte Schenkel den Test eines Systems. Für mich besteht ein generisches V-Modell also aus den drei Aspekten „denken“, „bauen“, und „testen“. Dieses Konzept ist universell und tritt überall in unserem Alltag auf, egal ob Sie sich etwas Leckeres zu essen kochen, ich diesen Artikel schreibe oder Sie Ihr Produkt entwickeln. Es geht immer darum, sich etwas zu überlegen, es umzusetzen und die Ergebnisse zu begutachten.

Dieses generische V-Modell trifft keine Aussage über Umfang und Dokumentation dieser drei Schritte. Ob beim Kochen, beim Schreiben, bei der Produktentwicklung oder bei einer anderen Tätigkeit: Die Schritte „denken“ und „bauen“ können nach außen hin zwar gleichzeitig wirken, genau genommen haben Sie aber eine Sekunde, bevor Sie das Ei in die Pfanne gehauen haben, unbewusst eine Erwartungshaltung an das Endergebnis im Kopf und Sie haben das Vorgehen definiert. Auch ein Softwareentwickler, der einfach seinen Editor öffnet und Code schreibt, folgt dem Muster denken-bauen-testen.

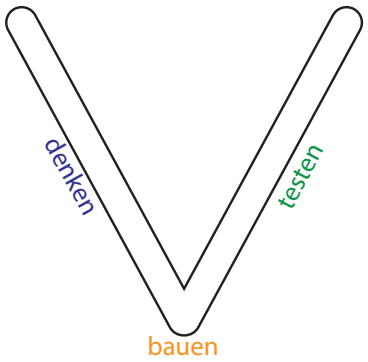

Bild 1: atomares V-Modell

Sie werden jetzt sagen: „In dieser Darstellung macht es sich Kollege Pfeffer zu einfach, denn die Aspekte der Dekomposition und Integration werden hier nicht berücksichtigt.“ Auch wenn die uns bekannten Interpretationen des V-Modells in der Software-und Systementwicklung eine solche Komposition vorsehen, bin ich dennoch der Meinung, dass auch ein wie von mir beschriebenes minimales V ein

V-Modell ist. Es zeigt bereits die horizontale Verbindung zwischen bauen und testen auf.

Dieser horizontale Zusammenhang zwischen linker und rechter Seite des V ist nach meiner Erfahrung nicht allen Anwendern des Modells bewusst: Wer sich ein technisches System, ein Spiegelei, oder einen Blog-Artikel ausdenkt, trägt damit automatisch die Verantwortung, auch den Test und die Testbarkeit des Vorhabens sicherzustellen. Dies mag im Rahmen meines Beispiels mit dem Ei noch am einfachsten sein. Doch schon beim Blogartikel muss ich mir im Vorfeld Gedanken machen, wie ich sicherstelle, dass er so aussieht, wie ich mir das vorgestellt habe und auf allen Browsern läuft. Bei Ihrem technischen System müssen Sie noch viel mehr Gedanken in Testkonzepte und die Testbarkeit der Anforderungen investieren. Das sollten Sie nicht erst beim Test machen, sondern schon während Sie die Anforderung an das System festlegen.

Handelsübliche V-Modelle

Auch an dieser Zwischenüberschrift können Sie erkennen: Sofern Sie nicht gerade ein Softwareprojekt für die Bundesrepublik Deutschland basteln, gibt es nicht das V-Modell. Unter einem üblichen V-Modell verstehe ich, dass sich die oben erwähnte Dekomposition eines Systems auch in der Beschreibung und dem Zusammenhang der Tätigkeiten widerspiegelt.

Dadurch besteht ein solches Modell aus mindestens zwei Ebenen. Ganz oben steht – wie Sie wahrscheinlich schon vermutet haben – das Gesamtsystem. Dieses zerfällt auf der zweiten Ebene in die Hauptkomponenten der Systemarchitektur. Auch eine Hauptkomponente könnte wieder in weitere Unterkomponenten zerfallen, dadurch würde sich ein Modell mit drei Ebenen ergeben. Sie könnten dies mit beliebig vielen Ebenen fortsetzen, um die Zusammenhänge zu beschreiben, erkläre ich hier jedoch ein einfaches Modell, in dem die Systemebene in lediglich eine weitere Komponentenebene zerfällt. Das Interessante hierbei ist, dass in einer solchen Denkweise jede Ebene nach demselben Muster aufgebaut ist – das Modell ist also fraktal.

Auf der linken Seite des V, also auf der Seite der Spezifikation und der Dekomposition, werden für jede Ebene Anforderungen erstellt. Auf der Systemebene sind das die Systemanforderungen, auf Komponentenebene sind dies Anforderungen an die jeweilige Komponente.

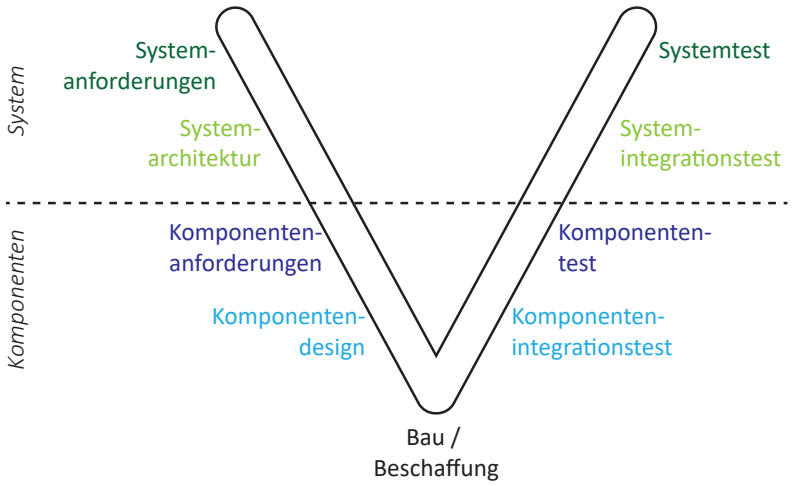

Bild 2: V-Modell mit zwei Ebenen

Anforderungen beschreiben immer das „Was“. Sie beschreiben also eine Blackbox-Sicht ohne Rücksicht auf das Innenleben der jeweiligen Ebene. Aus den Anforderungen wird für jede Ebene ein technisches Konzept abgeleitet. Das ist die Whitebox-Sicht, die das „Wie“ beschreibt. Aus den Systemanforderungen wird also eine Systemarchitektur abgeleitet, mit der die Komponenten des Systems sowie deren Interaktion definiert werden. Die Systemarchitektur liefert damit einen wesentlichen Anteil für die Anforderungen an die Komponenten (wiederum Blackbox-Sicht). Ebenso gibt es für jede Komponente wieder eine technische Beschreibung des Innenlebens, meist als Komponentenarchitektur oder Komponentendesign bezeichnet.

Damit bin ich in diesem Zwei-Ebenen-Beispiel an der Spitze des V angelangt, an der die Spezifikationen und Konzepte in ein Produkt umgesetzt werden. Die Umsetzung geschieht auf der niedrigsten spezifizierten Ebene, in diesem Beispiel werden also die einzelnen Architekturelemente der Komponenten gebaut.

Sobald dies geschehen ist, gewinnt die rechte Seite des V-Modells an Bedeutung. Sie steht für Integration und Test. Auf Komponentenebene werden nun die gefertigten Einzelteile zu einer Komponente zusammengesetzt (integriert) und danach wird jede Komponente für sich getestet. Integration und Test entsprechen dabei den links gegenüberliegenden Ebenen „Architektur“ und „Anforderungen“. Beim Zusammenbau, also bei der Integration, kann zum ersten Mal getestet werden, ob die Interaktion der Einzelteile den Vorgaben der Architektur entspricht (Whitebox). Danach wird im Test der Komponente sichergestellt, dass die Komponente die an sie gestellten Anforderungen erfüllt (Blackbox). Gemäß dem fraktalen Ansatz wiederholt sich dieses Spiel auf der darüber liegenden Systemebene: In der Systemintegration wird die Interaktion der Systemkomponenten getestet, das integrierte System wird dann gegen die Systemanforderungen getestet. Damit steht, egal auf welcher Ebene, den Anforderungen auf der linken Seite des V immer ein Blackbox-Text auf der rechten Seite gegenüber. Einem technischen Konzept bzw. einer Architektur auf der linken Seite stehen eine Integration und ein Integrationstest auf der rechten Seite gegenüber.

Zusammenspiel der Aktivitäten

Ein V-Modell beschreibt also zunächst Aktivitäten und Zusammenhänge. Es beschreibt nicht, ob diese in definierten Prozessen abgearbeitet und schriftlich dokumentiert werden, oder ob dies alles im Kopf talentierter Entwickler geschieht. Das Zusammenspiel der Aktivitäten ist für mich ein wesentlicher Punkt: Gerade hier erlebe ich in der Praxis oft eine unzureichende Interpretation des V-Gedankens. Zum einen möchte ich kurz auf das Prinzip der Traceability, also der Nachverfolgbarkeit von Anforderungen, eingehen – zum anderen geht es mir um das, was ich „horizontale Verantwortung" nenne.

Mit Ihrem Kunden reden Sie auf der Ebene der Systemanforderungen. Vielleicht war es auch Ihr Kunde, der diese aufgestellt hat. In Ihrem Entwicklungsablauf müssen Sie für sich nachweisen, dass alle Anforderungen umgesetzt und auch getestet wurden. Diese Nachverfolgbarkeit wird als „Traceability" bezeichnet. Dabei gibt es zwei Achsen: Die vertikale Traceability weist nach, dass alle Anforderungen in der Produktentwicklung berücksichtigt wurden, also welche Anforderungen durch welche Architekturelemente, Komponenten und Bauteile umgesetzt werden. Bei Software zieht sich das von der Systemebene bis auf die entsprechende Code-Zeile. Die horizontale Traceability weist nach, ob es zu jeder Anforderung auf der entsprechenden Ebene auch Testfälle gibt, um die Umsetzung der Anforderung verifizieren zu können.

Im zweiten Schritt müssen auch die Ergebnisse durchgeführter Tests (Testreports) mit den entsprechenden Testfällen in Beziehung gesetzt werden. Damit weisen Sie nach, dass jede Anforderung auch wirklich getestet wurde. Die Verwaltung von Traceability ist in der Regel nur mit entsprechenden Software-Werkzeugen möglich. Da sich alle Dokumente im Entwicklungsprozess regelmäßig ändern, ist es nur so möglich, die Traceability nachweisen zu können.

Der Grundgedanke in einem V-Modell ist, dass auf der Achse der horizontalen Traceability links die Grundlagen für die jeweils gegenüberliegende Aktivität auf der rechten Seite geschaffen werden. Wer Anforderungen aufschreibt, hat auch die Verantwortung, dass diese getestet werden können und dass grundlegende Testkonzepte vorliegen.

Ebenso haben die Architekten links im V die Verantwortung, die Reihenfolge für die Integration vorzuschlagen und die Grundlagen für den Integrationstest zu legen. Dies kann auch beinhalten, dass die Architektur Elemente oder Schnittstellen enthält, die nicht der Erfüllung der Anforderungen, sondern der Testbarkeit dienen (z. B. zusätzliche Messpunkte oder Schnittstellen). Jede Konzeptstufe im V-Modell trägt also auch die Verantwortung für die zugehörige Teststufe rechts im V. Diese, wie ich sie nenne, „horizontale Verantwortung" im V-Modell ist in der Praxis nicht immer anzutreffen. Oft werden die Aktivitäten auf den beiden Seiten des V strikt getrennt.

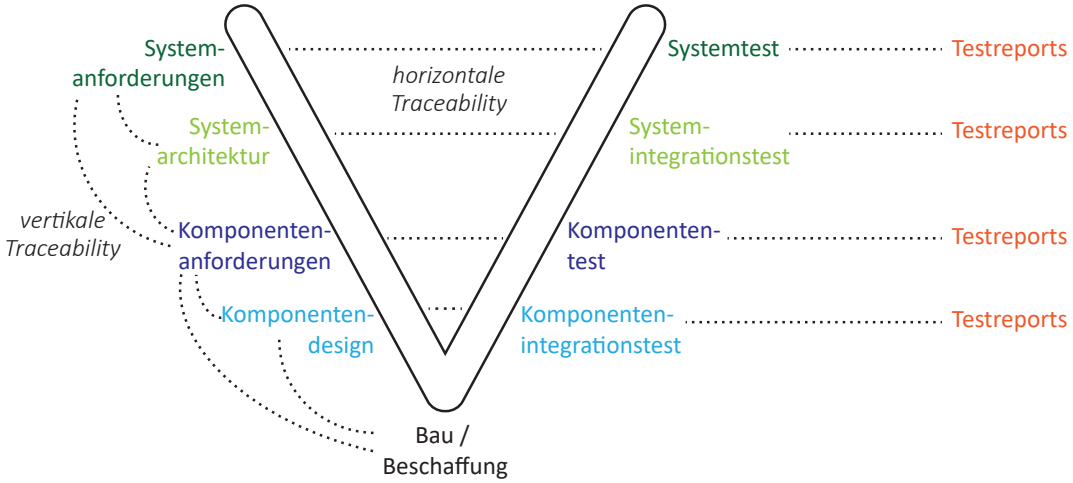

Bild 3: V-Modell mit Traceability

Scrum im V-Modell

Jeder Sprint in Scrum entspricht einem klassischen Projektdurchlauf mit Planung, Review und Lessons Learned. Am Ende jedes Sprints steht ein fertig getestetes und dokumentiertes Produktinkrement. In jedem Sprint wird also einmal das entsprechende V der Produktentwicklung durchlaufen. Die Tätigkeiten, die in einem V-Modell definiert sind, können dabei beliebig oft iteriert werden, interagieren oder parallelisiert werden. Jedes Scrum Team arbeitet also implizit mit einem V-Modell.

Die Frage, inwiefern Scrum und V-Modell zusammenpassen, entsteht meistens durch die Existenz starr definierter Prozesslandschaften, die auf dem V-Gedanken basieren. Diese sind darauf ausgelegt, dass ein Durchlauf des V, also ein Entwicklungszyklus, viele Monate oder gar Jahre benötigt. Entsprechend umfangreich sind die im Prozess hinterlegten Anforderungen an die Dokumentation des Vorgehens. Nicht selten entspringen diese dem Teufelskreis der Produktentwicklung: „Management macht unrealis-

tische Vorgaben, Entwicklung versucht diese unter Qualitätseinbußen zu erfüllen, Management macht wegen der schlechten Qualität mehr Vorgaben zu Dokumentation und Kontrolle, Zeitvorgaben werden wegen zusätzlichem Overhead noch unrealistischer.“ Um solche gewachsenen Monster-V-Prozessmodelle mit Scrum vereinbar zu machen, muss der Prozess Stück für Stück entschlackt werden, Checklisten müssen durch Vertrauen und Wunschdenken durch Transparenz ersetzt werden. Dazu müssen große Anstrengungen unternommen werden, um Aktivitäten auf beiden Seiten des V-Modells maximal zu automatisieren. Nur so kann das V irgendwann in einen Sprint passen.

Sie werden nun einwenden, dass es bei der Entwicklung von physischen Produkten – dem natürlichen Lebensraum von V-Modellen – nicht immer möglich ist, einen Durchlauf des V in einen Sprint zu bekommen. Das ist richtig. In einem solchen Fall haben Sie zwei Möglichkeiten.

Die erste (und häufig angewendete) Möglichkeit ist, erst ab einer bestimmten Ebene mit agilen Sprints zu beginnen. In diesem Fall haben Sie einen nicht-agilen Vor- und Nachspann im V-Durchlauf (oft auch als Hybrider Ansatz bezeichnet). Die agilen Korrekturmöglichkeiten beziehen sich nur auf die Umsetzung, nicht auf Spezifikation und Test, und bleiben also in einem überschaubaren Rahmen. Inwiefern dies noch agil ist, müssen Sie selbst beurteilen.

Die zweite und zugleich agilere Möglichkeit ist, die agile Lernschleife von Wochen auf Monaten auszudehnen. Ein kompletter V-Durchlauf in drei Monaten ist bei vielen Systemen realistisch, wenn die einschlägigen Impediments wie Einkaufs-, QM- und Musterbauprozesse angegangen werden.

Vorschläge

Egal, was Sie entwickeln und wie Sie es entwickeln: Stellen Sie sich die Tätigkeiten in einem V vor. Schaffen Sie auf der linken Seite im V die Grundlagen für alle Schritte auf der rechten Seite. Dies erhöht die Geschwindigkeit und verringert das Risiko bei Ihrem Entwicklungsvorhaben. Falls Sie die Prozesse, Templates und Checklisten für Ihr V-Modell definiert haben, prüfen Sie diese auf Notwendigkeit und Sinnhaftigkeit. Dies hilft Ihnen schneller durch das V zu kommen, wenn Sie agile Ansätze verwenden. Erliegen Sie nicht der Versuchung der hybriden Ansätze, nur weil das im Moment einfacher erscheint. Wenn Ihnen ein hybrider Ansatz plausibel und nützlich erscheint, überlegen Sie, ob agiles Arbeiten für Sie überhaupt Vorteile bringt. Wenn Sie hingegen agil arbeiten möchten, arbeiten Sie darauf hin, das V in einem festen Takt von wenigen Monaten durchlaufen zu können. Regelmäßige Durchläufe des Entwicklungs-Vs und die ständige Verbesserung der Organisation machen Agilität aus, auch wenn die Durchläufe zu Beginn der Veränderung noch Monate dauern. Arbeitspakete hingegen inmitten eines starren Projektplans mit unveränderten Prozessen im Takt von zwei Wochen abzuarbeiten und mit Zetteln an der Wand zu visualisieren, ist vielleicht nützlich, aber nicht agil.

Autor

Joachim Pfeffer ist Experte in der agilen Produktentwicklung für physische Produkte. Seit vielen Jahren berät er Unternehmen in der Automobilindustrie und im Maschinenbau, um mit ihnen schlagkräftige Entwicklungsabläufe in einer zunehmend komplexen Welt zu entwerfen. Dabei arbeitet er mit Scrum und Kanban, angereichert mit Konzepten aus dem Lean Development und der Engpasstheorie.

Veröffentlicht am 27.01.2020
auf joachim-pfeffer.com