HANSER

Leseprobe

zu

„Produkt-Entwicklung“

von Joachim Pfeffer

Print-ISBN: 978-3-446-45908-3
E-Book-ISBN: 978-3-446-46127-7
E-Pub-ISBN: 978-3-446-46325-7

Weitere Informationen und Bestellungen unter
http://www.hanser-fachbuch.de/978-3-446-45908-3
sowie im Buchhandel

© Carl Hanser Verlag, München

Inhalt

Vorwort ... **XV**

Einleitung ... **XVII**

Teil I: Grundlagen ... **1**

1 Motivation und Lean ... **3**

1.1 Motivation ... 3
1.1.1 Warum agil? ... 3
1.1.2 Cynefin-Modell ... 5
1.1.3 Stacey-Matrix ... 8
1.2 Das Toyota-Produktionssystem ... 9
1.2.1 Historie ... 10
1.2.2 Veränderung ... 10
1.2.3 Kanban ... 12
1.3 Engpasstheorie ... 13
1.3.1 Wanderung der Pfadfinder ... 13
1.3.2 Drum-Buffer-Rope in der Produktion ... 14
1.4 Lean Development ... 15
1.4.1 Verzögerungskosten ... 15
1.4.2 Warteschlangen ... 19
1.4.3 Losgrößen ... 21
1.4.4 Abstimmungen ... 22

2 Agile Ansätze ... **25**

2.1 Grundlagen ... 25
2.1.1 Definition(en) ... 25
2.1.2 Das agile Manifest ... 26
2.2 Scrum ... 28
2.2.1 Überblick ... 28
2.2.2 Definition ... 32
2.2.3 Praxis ... 35
2.2.4 Skaliertes Scrum ... 41

2.3 Kanban ... 45
2.3.1 Kanban-Boards ... 47
2.3.2 Praxis ... 49

3 Menschen und Teams ... 53
3.1 Menschen ... 53
3.1.1 Motivation ... 53
3.1.2 Menschliches Leistungsvermögen ... 55
3.2 Teams ... 57
3.2.1 Teamaufbau ... 57
3.2.2 Co-Location ... 58
3.2.3 Teamzuschnitt ... 59
3.2.4 Kompetenzprofile ... 61

Teil II: Herausforderungen ... 63

4 Das Inkrement ... 65
4.1 Aufgabe des Inkrements ... 65
4.1.1 Feedbackschleifen in der Produktentwicklung ... 65
4.1.2 Früher Wert im Markt ... 68
4.1.3 Ausprägungen von Feedback ... 70
4.2 Exkurs: Inkrement in der Software ... 74
4.2.1 Automatisierung der Integration ... 75
4.2.2 Automatisierung von Tests ... 76
4.2.3 Kontinuierliche Auslieferung und Bereitstellung ... 78
4.3 Inkremente bei physischen Produkten ... 79
4.3.1 Herausforderungen in der Mechatronik ... 79
4.3.2 Definition des Inkrements ... 81
4.3.3 Praxisbeispiele ... 85

5 Verkürzung von Zykluszeiten ... 89
5.1 Technologisch bedingte Zeiten reduzieren ... 89
5.1.1 Modulare Architektur ... 90
5.1.2 Alternative Fertigungskonzepte ... 96
5.1.3 Automatisierter Test ... 99
5.1.4 Einsatz von Simulationen ... 103
5.2 Prozessbedingte Zeiten reduzieren ... 107
5.2.1 Losgrößen in der Entwicklung verkleinern ... 107
5.2.2 Entscheidungen beschleunigen ... 110
5.2.3 Kadenz und Synchronisation optimieren ... 114
5.3 Kulturell bedingte Zeiten reduzieren ... 116
5.3.1 WIP reduzieren ... 116
5.3.2 Warteschlangen managen ... 119
5.3.3 Expertenkultur verändern ... 122

5.4 Exkurs: Das WIKISPEED-Projekt ... 127
5.4.1 Historie ... 127
5.4.2 Arbeitsweise ... 128
5.4.3 Erfolgsfaktoren ... 130

6 Rahmenbedingungen für Agilität ... 133
6.1 Management und Leadership ... 133
6.1.1 Selbstorganisation ... 134
6.1.2 Leadership ... 140
6.1.3 Prioritäten und Impediments ... 144
6.1.4 Teamumgebung gestalten ... 148
6.1.5 Bedeutung der Linienorganisation ... 149
6.2 Prozesse ... 151
6.2.1 Projektmanagement ... 152
6.2.2 Produktentstehungsprozess (PEP) ... 154
6.2.3 Unterstützende Prozesse ... 161
6.2.4 Exkurs: Lean Development bei Harley-Davidson ... 164
6.3 Compliance ... 170
6.3.1 Allgemeines ... 170
6.3.2 Prozessreife, Automotive SPICE® ... 172
6.3.3 Funktionale Sicherheit, ISO 26262 ... 180

Teil III: Umsetzung ... 187

7 Startlöcher graben ... 189
7.1 Orientierung und Wissen ... 190
7.1.1 Umfang ... 190
7.1.2 Personen ... 194
7.1.3 Machbarkeit ... 196
7.1.4 Trainings ... 197
7.2 Grundlagen für das Konzept ... 200
7.2.1 Einordnung Lean und Agile ... 200
7.2.2 Scrum, Kanban oder ??? ... 201
7.2.3 Teamzuschnitte ... 204
7.2.4 Schnittstellen ... 207
7.3 Konzept auf Teamebene ... 209
7.3.1 Blackbox-Pull ... 210
7.3.2 Kanban ... 216
7.3.3 Scrum ... 224
7.4 Konzept auf Wertstromebene ... 229
7.4.1 Wertstrom mit Kanban ... 230
7.4.2 Skaliertes Scrum ... 233
7.4.3 Was tun? ... 239

8 Mit agilen Ansätzen unterwegs ... 241
8.1 Planen und verfolgen ... 241
8.1.1 Arbeiten mit den Backlogs ... 242
8.1.2 Planung ist alles ... 246
8.1.3 Messen und Korrigieren ... 250
8.1.4 Puffermanagement ... 255
8.2 Synchronisation ... 258
8.2.1 Management ... 258
8.2.2 Kunden und Lieferanten ... 261
8.2.3 Systemintegration und Test ... 263
8.3 Lernen und Korrigieren ... 265
8.3.1 Retrospektiven ... 265
8.3.2 Impediments ... 268
8.3.3 Communities ... 272

9 Performante Organisationen ... 277
9.1 Fokus für die Organisation: Portfolio-Management ... 277
9.1.1 Portfolio Kanban ... 277
9.1.2 Agile Budgetierung ... 280
9.1.3 Herausforderungen bei der Einführung ... 280
9.2 Mit Wertstrombetrachtungen Silos aufweichen ... 281
9.2.1 Motivation ... 281
9.2.2 Value Stream Mapping ... 284
9.2.3 Veränderung ... 290
9.3 Zur Veränderung einladen ... 292
9.3.1 Herausforderungen bei Kulturveränderungen ... 292
9.3.2 Open Space Technology ... 297
9.3.3 OpenSpace Agility ... 299

10 Literatur ... 305

Index ... 309

Vorwort

Schon wieder ein Buch über agile Entwicklung? Ich hoffe, dieses ist anders. Es soll das Buch sein, das ich bei meinem täglichen Bestreben, agile Ansätze bei der Entwicklung von physischen Produkten zu etablieren, all die Jahre vermisst habe. Die Grundkonzepte war mir aus meiner Zeit in der Softwareentwicklung vertraut, dennoch stellten sich mir unzählige Fragen, als ich Ende der 2000er Jahre wieder in meiner eigentlichen Tätigkeit, der Elektronikentwicklung tätig war und mir bekannte und neu aufkommende Ansätze bei der Entwicklung mechatronischer Produkte einsetzen wollte. Alle Bücher auf dem Markt bezogen sich damals auf den Einsatz agiler Ansätze in der Software-Entwicklung bzw. der IT. Diese Situation hat sich bis heute kaum geändert. Die wenigen Bücher, die zu agiler Produktentwicklung auf dem Markt sind, habe ich als zu philosophisch empfunden bzw. zu wenig konkret, um für mich oder meine Kunden daraus etwas für die konkrete Umsetzung von agiler Produktentwicklung und der täglichen Herausforderung damit etwas mitzunehmen. So habe ich 2016 damit begonnen, aus den damaligen Projekten inspiriert, ein Buch über den Einsatz von Kanban in der Mechanik- und Elektronikentwicklung zu schreiben. Parallel dazu hatte ich im Rahmen meiner Beratungstätigkeit die Möglichkeit, bei immer mehr Projekten meine Kunden dabei zu unterstützen, physische Produkte agil zu entwickeln. Darunter waren interessante Anwendungen wie Nockenwellen, Brennstoffzellen, Leuchtdioden und natürlich vielerlei Steuergeräte in der Automobilindustrie. Nachdem ich anfänglich sehr auf Kanban fokussiert war, habe ich mich in dieser Zeit mit dem mir aus der Software-Entwicklung vertrauten Scrum erneut immer mehr angefreundet. Kanban erscheint in diesen Branchen einfacher umzusetzen, Scrum verfolgt jedoch aus meiner Sicht einen sehr viel konsequenteren Teamgedanken und entfacht eine andere Energie, um Menschen gemeinsam ins Ungewisse aufbrechen zu lassen. So wurde über die Zeit aus einem Kanban-Buch ein Buch für die agile Produktentwicklung, welches Kanban und Scrum gleichermaßen enthält. Dabei war es mir wichtig, dem Leser so viel wie möglich konkrete Anleitung mit an die Hand zu geben – wie gesagt, es soll das Buch sein, welches ich immer vermisst habe.

Dieses Buch enthält also meine Sicht auf die Dinge und wie ich Veränderungsvorhaben in der agilen Produktentwicklung angehe. Das mag vielleicht den einen zu wenig konsequent agil sein, andere würden manche Aspekte der Agilität anders interpretieren, und vielleicht enthält das Buch auch Dinge, die ich einfach falsch verstanden habe. Auch wenn es aus meinem Marktverständnis heraus das erste Buch seiner Art ist, geht es mir nicht darum, ein Grundlagenwerk mit Absolutheitsanspruch zu schaffen, sondern einen Beitrag zur Diskussion über den Einsatz agiler Ansätze in der Produktentwicklung zu leisten.

Um in diesem Buch den Fokus auf die praktische Umsetzung hochzuhalten, habe ich den ausführlichen Grundlagenteil in ein eigenes Buch herausgelöst, das ich auch als Handout für meine Grundlagentrainings verwende [Pfeffer 2019], und habe hier im ersten Buchteil eine kurze Übersicht über die zugrunde liegenden Konzepte zusammengestellt.

Bedanken möchte ich mich bei Dolores Omann, die einige meiner bisherigen Bücher lektoriert und mir auch bei diesem Buchkonzept mit Rat und Tat zur Seite gestanden hat, auch wenn sie nicht offiziell involviert war. Weiterer Dank geht an Michael Klein, Miriam Sasse und Sebastian Schneider für die kritische Diskussion meiner Gedanken in diesem Buch. Zuletzt auch noch ein großes Dankeschön an Brigitte Bauer-Schiewek vom Hanser Verlag, die dieses Projekt stets mit professioneller Coolness und viel Humor begleitet hat („ich habe Ihnen ein paar Cover-Vorschläge zugemailt. Rufen Sie einfach an, falls mehr Katzen drauf sollen“).

Westerland, Sylt, im Herbst 2019

Joachim Pfeffer

Einleitung

Dieses Buch wendet sich an agile Coaches und Führungskräfte, die in ihrer Organisation vor der Herausforderung stehen, agile Ansätze bei der Entwicklung physischer Produkte einzusetzen. Wenn ich Sie in diesem Buch als Leser direkt anspreche, gehe ich davon aus, dass Sie sich in einer solchen Rolle und Situation befinden. Die Abgrenzung zwischen Führungskraft und Coach habe ich dabei nicht aufgegriffen, weil dies aus meiner Sicht dem Lesefluss abträglich gewesen wäre. Aus demselben Grund verwende ich bei Rollen und Berufen immer die männliche Form, auch wenn damit alle Geschlechter gemeint sind.

Produktentwicklung ist ein technologisch breites Spektrum, was Technologien und Kompetenzen angeht. Dementsprechend kommen auch breit gestreute Beispiele vor, die dann vielleicht nicht immer genau Ihre Branche oder Ihr Unternehmen treffen. Vielleicht spielt Automotive SPICE für Sie ebenso wenig eine Rolle wie Moldflow-Simulationen – dann bitte ich Sie, über die entsprechenden Stellen hinwegzulesen. Bestimmt kommt an einer anderen Stelle etwas vor, in dem Sie sich wiederfinden können. Hinsichtlich der Engineering-Aspekte kann ich in einem solchen Buch natürlich nur einzelne Dinge herausgreifen. Wie Sie dieses jeweils bei Ihnen umsetzen können, müssen Sie selbst beurteilen. In Ihrer Branche haben Sie, was das Engineering angeht, auf jeden Fall eine deutlich höhere Expertise als ich.

Hinsichtlich der agilen Ansätze – allen voran Scrum und Kanban – gehe ich davon aus, dass Sie bestimmte Grundkenntnisse haben bzw. mehrere gute Fachbücher zu den Grundlagen bereits in Ihrem Regal stehen. In diesem Fall können Sie den ersten Teil des Buchs einfach auslassen oder lediglich kurz überfliegen und direkt mit Teil II in die Entwicklung physischer Produkte eintauchen. Falls Sie hingegen einen Überblick über die Konzepte gewinnen möchten oder einzelnes nachlesen möchten (vielleicht haben Sie ja noch kein Buch über Lean Development), werden Sie im ersten Teil fündig, in dem ich die wichtigsten Grundlagen kompakt – wie ich hoffe – zusammengetragen habe. Durch dieses Konzept kann es sein, dass einzelne Themen aus dem Grundlagenteil in den Teilen zwei und drei noch einmal kurz angeschnitten werden.

Der zweite Buchteil liefert wichtige Gedanken und Diskussionen zu den Reibungspunkten zwischen Agilität und physischen Produkten: Inkrement, Zykluszeiten, Expertenkulturen usw. Im dritten Teil zeige ich auf, wie Sie von den ersten Versuchen mit Ihren Produkten zu einer agilen Organisation kommen können. Gerade das letzte Kapitel ist eine kleine Gedankensammlung, in die Sie auch unabhängig von den vorhergehenden Kapiteln einsteigen können.

Zur besseren Strukturierung habe ich drei verschiedene Arten von Kästen eingesetzt. Es gibt Kästen für allgemeine Hinweise, für Praxistipps und für Praxisbeispiele. Auch diese können Sie beim Durchblättern des Buchs einzeln herauspicken und für sich verarbeiten.

Nun hoffe ich, dass Sie meine Ausführungen gewinnbringend in Ihren Zusammenhängen einsetzen können, und freue mich über Feedback an die Adresse *joachim.pfeffer@peppair.com.*

5 Verkürzung von Zykluszeiten

Eine Verkürzung der Zykluszeit für einen Entwicklungszyklus, also von der Spezifikation bis zur Auslieferung, verringert direkt die entstehenden Verzögerungskosten, in die alle bisher genannten Vorteile wie früheres Feedback einfließen. So naheliegend eine Neudefinition des Inkrements wie im vorigen Kapitel beschrieben ist, so schwer ist es auch für viele, sich eine nennenswerte Verkürzung der Zykluszeit vorzustellen. Oft sind hier die Grenzen fest verankerte Glaubenssätze und nicht die Technologie selbst. Die weit verbreitete „First time right"-Kultur trübt die Sicht auf das, was möglich ist. Meine Erfahrung: Das Wichtigste zur Verkürzung der Zykluszeit ist ein fester Wille, dies auch zu tun. Dies wird Geld kosten und viele etablierte Abläufe und Prozesse verändern.

Die vorherrschenden Zykluszeiten setzen sich aus folgenden beiden Aspekten zusammen:

1. Prozesszeiten: Wie viel Arbeitszeit müssen Sie für das nächste Inkrement investieren?
2. Liegezeiten: An welchen Stellen auf Ihrem kritischen Pfad wird nicht an Ihrem Inkrement gearbeitet?

Um die Prozesszeiten verringern zu können, werden Sie Architekturen und Technologien verändern müssen. Um die Liegezeiten zu verringern, werden Sie Ihre Arbeitsweise und Organisation verändern müssen. Beides kann nicht über Nacht geschehen.

Welcher Aspekt den größten Hebel hat, lässt sich nicht pauschal sagen. Dies hängt von Ihrem Produkt und Ihrer Organisation ab. In diesem Kapitel gehe ich auf die „üblichen Verdächtigen" ein und beziehe dabei bewusst eine extreme Position für eine wirklich agile Systementwicklung. Ob die damit verbundenen Konsequenzen hinsichtlich Architektur und verwendeter Komponenten für Ihr Produkt und für Ihren Markt erstrebenswert sind, können nur Sie selbst beurteilen.

5.1 Technologisch bedingte Zeiten reduzieren

Ein Teil der Liege- und Wartezeiten entfallen auf das Produkt selbst, also auf dessen Entwicklung, Bau und Test. Neben der Reduzierung der Liegezeiten und der Erzeugung eines Flusses geht es auch darum, die eigentlichen Prozesszeiten durch alternative Fertigungsverfahrung, flexible Architekturen und Automatisierung bei Bau und Test zu reduzieren.

5.1.1 Modulare Architektur

Bei Serienprodukten werden Systemarchitekturen in der Regel auf minimale Fertigungskosten oder andere Parameter wie zum Beispiel Gewicht optimiert. Diesem müssen sich andere Kriterien wie Wartbarkeit oder Modifizierbarkeit oft unterordnen. Letzteres, die Modifizierbarkeit, ist jedoch essentiell für die agile Produktentwicklung.

Sie können den Begriff der Modifizierbarkeit auf die Systemarchitektur beziehen oder auf das Produkt selbst. Bei unklaren Anforderungen und/oder Technologien, also im komplexen Umfeld, kommen Sie vielleicht nicht umhin, die Architektur hin und wieder grundlegend überarbeiten zu müssen, sie sollte also leicht modifizierbar sein. Zur Reduzierung der Zykluszeiten liegt der Fokus auf einem leicht zu modifizierenden Produkt, welches durchaus auf einer recht stabilen Architektur aufbauen kann.

Konsequente Modularisierung ist einer der Hauptfaktoren bei der Agilisierung der Produktentwicklung. Im Zeitraum von wenigen Wochen können Sie nur kleine Änderungen an Ihrem Produkt gestalten, bauen und testen. Dazu müssen die Komponenten in Ihrer Systemarchitektur voneinander möglichst unabhängig sein. So können Sie eine kleine Funktionalität an einer einzelnen Komponente verbessern oder hinzufügen, ohne dass dies einen Einfluss auf andere Komponenten hat.

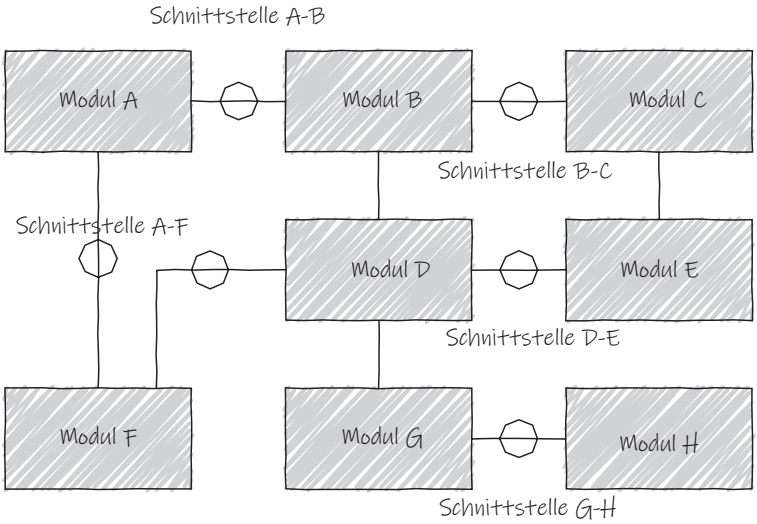

Bild 5.1 Modulare Systemarchitektur

Modularisierung erfordert eine Architektur mit sehr starken Schnittstellendefinitionen. Dadurch hängt eine Komponente nur von ihren Schnittstellen ab und nicht direkt von anderen Komponenten. Dies können mechanische oder elektrische Schnittstellen sein, Kommunikationsprotokolle (zum Beispiel von Bussystemen) oder auch Schnittstellen zwischen Software-Modulen (Bild 5.1).

Die Schnittstelle muss dabei vom Innenleben der Komponente abstrahieren. Sie definiert also nur, „was“ die Komponente leisten soll und nicht „wie“. Dies erfordert eine Rückbesinnung auf althergebrachte Tugenden aus dem Requirements-Engineering, nämlich die Lösung aus den Anforderungen an eine Komponente herauszuhalten.

Praxisbeispiel: Unglückliche Schnittstellendefinition

Ein Negativbeispiel zu Schnittstellen in der Systemarchitektur, das ich einmal erlebt habe: Der Systemarchitekt hat die Vorgabe herausgeben, dass Sensoren möglichst „dumm" zu halten sind. Dadurch wurde die Anbindung der Sensoren über den CAN-Bus so definiert, dass die rohen ADC-Digits des AD-Wandlers im Sensor über den Bus übertragen wurden. Die ungewöhnliche Begründung: Es sollten keine Fließkommawerte über den Bus gehen. Jede Änderung am Sensor oder dessen Mikrocontroller verursachte dadurch Änderungen bei mehreren anderen Modulen. Nachdem die Sensoren ohnehin über Mikrocontroller verfügten, wäre es problemlos möglich gewesen, die Schnittstelle in physikalischen Einheiten zu definieren, den Temperatursensor zum Beispiel in Millikelvin, um wie gewünscht Fließkommaoperationen zu vermeiden.

Falls Sie in der Rolle eines Zulieferers sind, zum Beispiel für die Automobilindustrie, ist die Trennung von „was" und „wie" nicht immer einfach: Oft enthalten die von Ihrem Kunden übermittelten Lastenhefte bereits Vorgaben zur technischen Umsetzung. Beziehen Sie in diesem Fall die hier von mir vorgebrachten Betrachtungen auf die Bereiche in Ihrem Produkt, auf die Sie Einfluss nehmen können.

Modularisierung in der Elektronikentwicklung

Viele Elektronikentwickler gestalten die Schaltpläne bereits modular. Netzteil, Mikrocontroller, Busschnittstellen, ADCs usw. werden separat gestaltet und über definierte Signale miteinander verbunden. Traditionell werden jedoch bei Layout, Aufbau und Test alle Module zusammengefasst. Die Leiterplatte hat schon die endgültige Größe, und die Bauteile werden nicht mehr nach ihrer Modulzugehörigkeit angeordnet, sondern nach Aspekten wie verfügbarem Platz, Anzahl der Durchkontaktierungen oder EMV. Die Integration und die Inbetriebnahme erfolgen daher nach dem „Big Bang"-Ansatz. Können Fehler nicht eingegrenzt werden, muss die Integration manchmal rückgängig gemacht werden – ich denke, alle Elektroniker meiner Generation haben schon einmal Leiterbahnen durchgekratzt, um Fehler einzugrenzen.

Für eine agile Elektronikentwicklung können Sie das Modulkonzept durch alle weiteren Schritte, also Layout, Aufbau und Test weitertragen. Dies bedeutet konkret, für jedes Modul eine eigene Leiterplatte oder bei einfachen Modulen vielleicht nur einen Steckbrettaufbau durchzuführen. Diese Modulleiterplatten können Sie unter Verzicht auf Platzoptimierung schnell mit dem Autorouter layouten und bei einem Expressanbieter in Auftrag geben, falls Sie nicht gerade mit 16-lagigen Leiterplatten unterwegs sind. Gegenüber dem endgültigen Layout müssen Sie dann natürlich zusätzliche Bauteile für Steckverbinder vorsehen. Sie können nun Modul für Modul aufbauen und für sich testen. Sie können die Integration Schritt für Schritt durchführen: Am Anfang können Sie Module durch bestehende Module oder Alternativen ersetzen (z. B. ADC, Netzteil).

Der so entstehende „fliegende Aufbau" ist natürlich keineswegs ein potenziell auslieferbares Produkt im Sinne des Scrum Guide, er reduziert jedoch die Komplexität von Integration und Test und erlaubt Ihnen, im Takt von wenigen Wochen vorzugehen. In diesen Intervallen können Sie es durchaus schaffen, ein einzelnes Modul im Schaltplan zu verbessern, aufzu-

bauen und zu testen. Hochfrequenz- und EMV-Aspekte werden Ihre Möglichkeiten hier einschränken, zumindest was die Testbarkeit angeht. Hierfür müssen Sie dann in der Tat auf die später entstehende „endgültige“ Leiterplatte mit allen Modulen warten.

Modularisierung in der Mechanikentwicklung

Mechanisches Design ist aus physikalischer Sicht schon modular, wenn Sie zwischen einzelnen Komponenten lösbare Verbindungen einsetzen, also auf Kleben, Schweißen, Pressen o. ä. verzichten. Dies macht Verbindungen teurer, aber eben auch flexibler. So können Sie von Iteration zu Iteration einzelne Module/Komponenten gegen überarbeitete oder neue Module austauschen. Ob Sie im nächsten Sprint einen Gehäusedeckel verbessern, ein Stück Software oder eine Antriebsachse einer Werkzeugmaschine – die Modularisierung ist der Schlüssel zu agiler Produktentwicklung.

Mechanische Schnittstellen wie zum Beispiel Lager oder Flansche werden in der Regel so entworfen, dass sie zu der Komponente passen, die „angedockt“ werden soll. Im Rahmen der agilen Produktentwicklung kommen Sie jedoch vielleicht in die Verlegenheit, Komponenten gegen solche zu tauschen, die über andere mechanische Schnittstellen verfügen. Dann müssen Sie mit Adaptern arbeiten. Versteifen Sie sich also nicht zu sehr auf die eine mechanische Verbindung, die Sie im Moment im Kopf haben, vielleicht benötigen Sie später ohnehin einen Adapter.

Produkte mit Adaptern zusammenbauen? Das klingt für Sie wahrscheinlich nach einer Bastellösung. Aus agiler Sicht ist ein solches Produkt zum einen ein funktionierendes Inkrement, zum anderen aber nicht der Endzustand. Die Grenze zwischen Bastellösung und High-End-Produkt ist in der agilen Entwicklung fließend. Das Produkt entwickelt sich weiter, und irgendwann ist der Zeitpunkt gekommen, an dem sich das agile Team dazu entscheidet, die Architektur anzupassen, weil sie nun nicht mehr zum aktuellen Produkt passt. In der Software nennt man dies „Refactoring“: ein interner Umbau des Produkts, ohne die äußeren Eigenschaften oder die Funktionalität zu verändern. Refactorings dienen dem im Grundlagenteil beschriebenen Abbau von technischen Schulden. Im Fall der Mechanikentwicklung werden bei einem Refactoring die meisten der über die Zeit eingeführten Adapter wieder verschwinden, egal ob es wirklich eine agile Produktentwicklung ist, also das Produkt immer weiter entwickelt wird, oder ob es einen Übergang zur Serienproduktion gibt.

Modularisierung in der Software-Entwicklung

Die Domäne der Software-Entwicklung hat es einfacher mit der Modularisierung: Die Konzepte sind erprobt, und Adapter können auch in ausgelieferten Produkten noch enthalten sein, sie erzeugen kein Gewicht, keine Seiteneffekte und benötigen keinen Bauraum. Der Vorteil der Software – alles ist möglich – ist auch ihr entscheidender Nachteil: In der Software ist es viel leichter als bei Elektronik- und Mechanikentwicklung, an der Architektur vorbeizuarbeiten, um schnell zu funktionierenden Lösungen zu kommen. Solche Architekturverletzungen schaffen also neue, nicht geplante Verbindungen zwischen existierenden Modulen. Ohne regelmäßiges Anpassen der Architektur im Rahmen von Refactorings bauen sich schnell technische Schulden auf.

In der Software-Entwicklung können Sie ebenso wie in der Mechanik- und Elektronikentwicklung bestehende Module gegen andere, selbst gefertigte oder zugekaufte Module austauschen und so schon früh im Entwicklungsablauf ein funktionierendes Produkt enthalten.

Frühe funktionierende Systeme

Verfügt Ihre Architektur über klar definierte Schnittstellen, können Sie eventuell bestehende Komponenten aus Vorgängerprojekten oder vom Wettbewerber einsetzen, um früh im Entwicklungsfortschritt ein funktionierendes System zu bekommen. Dazu müssen Sie sich unter Umständen Adapter bauen, welche die Schnittstelle einer alten/anderen Komponente verbergen und als Ihre neu definierte Schnittstelle anbietet. Bei mechanischen und elektrischen Schnittstellen kosten solche Adapter Bauraum und sind nicht immer einfach umzusetzen. Damit entkommen Sie aber dem Druck, alle Komponenten gleichzeitig reifen lassen zu müssen.

Beispiele für Adapter:

- Mechanisch
  - Adapterflansche
  - Reduzierhülsen
  - Gehäuse
  - Adapterstecker
- Elektrisch/Kommunikation
  - Transformatoren
  - Signal-/Frequenzumsetzer
- Software
  - Adapterklassen/-funktionen

Praxisbeispiel: Fahrassistenz

Beim ersten Treffen stellte der Kunde sein Projekt für ein Fahrassistenzsystem vor. Der Projektplan sah vor, dass das aktuelle Arbeitspaket „Systemarchitektur" noch für vier Monate läuft, danach sollte die Software implementiert werden. Die notwendige Hardware (Steuergeräte) stand schon zur Verfügung. Als Vorbereitung für die Scrum-Einführung hat der Kunde schon ein Backlog erstellt, dieses enthielt aber keine Anforderungen, sondern Aufgaben aus dem Projektplan (Architektur erstellen, entwickeln …).

Der erste Schritt war nun, die Backlog-Items von Tätigkeiten auf die geforderten Funktionen umzustellen, die inkrementell wachsen konnten:

- Fahren bei Tag, Bundesstraße, 30 km/h
- Fahren bei Tag, Bundesstraße, 80 km/h
- Fahren bei Tag, Landstraße, 30 km/h
- Fahren bei Tag, Landstraße, 80 km/h
- Fahren bei Nacht, Bundesstraße, 30 km/h
- usw.

Da ein lauffähiges System in ferner Zukunft zu liegen schien, priorisierten wir zunächst nicht die Anforderung mit dem größten Kundennutzen nach oben, sondern die technisch einfachste Funktion, um möglichst früh ein lauffähiges

System zu bekommen. In der weiteren Diskussion stellte sich heraus, dass die Architektur auf der obersten Ebene bereits stabil ist und die Schnittstellen klar und ausreichend abstrakt sind. Die weitere Architekturzeit sollte in das Design einzelner Komponenten gehen, so der ursprüngliche Plan. Meine Frage: „Wäre es möglich, in zwei Wochen mit der ersten Funktionalität zu fahren?“ führt zu der Erkenntnis, dass aus den Vorgängerprojekten so viel funktionierender Quellcode verfügbar ist, damit die neue Architektur mit altem Code hinter den neuen Schnittstellen in zwei Wochen für eine Testfahrt ausreichende Funktionalität liefern konnte. Das alte System hatte zwar eine gänzlich andere Architektur, aber die Kernfunktionalität konnte dort entnommen werden. Damit wurde die weitere Architekturarbeit herunterpriorisiert und der Fokus auf ein – wie auch immer – lauffähiges System gesetzt.

Einige Wochen später: Das Management ist begeistert, dass es bereits Testfahrten statt Präsentationsfolien gibt. Bei den Testfahrten stellte sich heraus, dass ein Sensor Probleme bereitet, mit denen keiner gerechnet hat. Diese Erkenntnis wäre nach dem klassischen Projektplan erst Monate später gekommen. Selbstverständlich wurde die Entwicklung in den weiteren Sprints hin zu der beabsichtigten mächtigen und eleganten technischen Lösung getrieben.

Eine modulare Architektur und die konsequente Nutzung dieser mit alten Komponenten erlaubt es, im Scrum-Modus zu arbeiten, also in kleinen Schritten vorzugehen und dabei ständig ein funktionierendes System zu haben – auch wenn ein solches frühes System eine „Bastellösung“ ist und nicht ausgeliefert werden kann. Die Risiken werden minimiert und die Motivation maximiert.

Neben existierenden Komponenten, die Sie in Ihrer Schublade oder auf dem freien Markt finden, haben Sie auch die Möglichkeit, zu Beginn an manchen Stellen Komponenten einzusetzen, die nur minimale oder keine Funktionalität liefern und einfach herzustellen sind. Die Aufgabe solcher „Platzhalter“-Komponenten ist, die entsprechenden Schnittstellen rudimentär zu bedienen, damit Sie das komplette System mit wenig Aufwand zum Leben erwecken können. Bei elektronischen Komponenten können diese eventuell auch durch eine PC-basierte Simulationen ersetzt werden, die mit Ihrem System über die definierten Schnittstellen interagieren.

Emergentes Systemdesign

Wasserfall-artige Entwicklungsansätze setzen darauf, die Systemarchitektur vorab detailliert zu durchdenken, dann einzufrieren und auf Basis dieser – möglichst nicht zu verändernden – Architektur weiterzuentwickeln. Basis für die umfangreichen Architekturüberlegungen sind im Vorfeld eingefrorene Anforderungen. In einem agilen Entwicklungsumfeld werden sich jedoch diese Anforderungen laufend ändern und damit auch die Architektur. Eine Änderung der Architektur, besonders spät im Entwicklungsablauf, ist sehr teuer. Wenn sich jedoch die Anforderungen gegenüber den ersten Annahmen verändert haben, bleibt Ihnen keine andere Wahl, als die Architektur anzupassen. Dies trifft zu, egal ob Sie klassisch oder agil entwickeln. Scrum oder Kanban verändern ja nicht Ihr Marktumfeld. Es

erscheint nur deshalb ungewohnt, weil bei klassischen Ansätzen schlicht versucht wird, Änderungen zu unterbinden. Änderungen als Chance für den Erfolg des Kunden zu begrüßen, wie es das Agile Manifest vorsieht, scheitert oft am Projektdenken mit festem Inhalt, fester Zeit, festem Budget.

Vorab erdachte, feste Systemarchitekturen basieren oft auch auf Annahmen über das, was sich in der Zukunft noch ereignen wird, sowohl hinsichtlich der Technologie wie auch hinsichtlich des Markts. Viele klassische Architekturen versuchen daher, der Unsicherheit zu begegnen, indem sie möglichst viele potenzielle Optionen einplanen. In der Software wird eine solche, auf Annahmen basierende, für den aktuellen Produktstand überdimensionierte Architektur als „Frameworkeritis“ bezeichnet. Die Praxis zeigt, dass im weiteren Verlauf der Entwicklung nur ein Teil der antizipierten Funktionalitäten umgesetzt und die Architektur dennoch für unvorhergesehene Funktionalitäten erweitert wird.

Mit emergenten Architekturen zu arbeiten bedeutet, dass die Architektur für die aktuelle Funktionalität des Produktinkrements angemessen sein sollte und keine Annahmen über die ungewisse Zukunft treffen sollte. Damit verbunden ist, wie erwähnt, eine Bereitschaft, die Architektur wenn nötig anzupassen oder im Extremfall ganz neu zu erstellen. Um Missverständnisse zu vermeiden: Dies bedeutet nicht, dass die Architektur im Zweiwochen-Takt neu überdacht und ständig angepasst wird. Die Produktaspekte, von denen Sie wissen, dass sie in naher Zukunft sicher in das Produkt einfließen werden, sollten Sie auch in Ihre Architektur einfließen lassen. Bildhaft können Sie sich diese Unterschiede in der Vorgehensweise wie in Bild 5.2 gezeichnet vorstellen: Beim klassischen Vorgehen wird die gesamte Architektur „auf Halde“ erstellt.

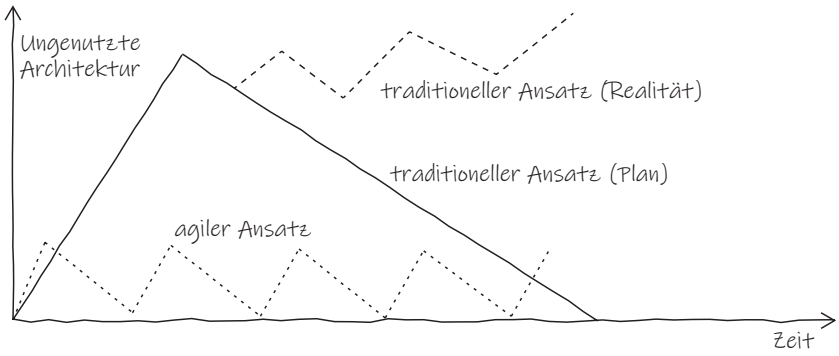

Bild 5.2 Inkrementelle Architekturentwicklung (angelehnt an *scaledagileframework.com*)

Die Annahme dazu: Die gesamte Architektur wird auch genutzt. Die Praxis hingegen ist: Sobald die Architektur genutzt wird, ergeben sich neue Erkenntnisse – technologisch und funktional. Eine Anpassung der Architektur wird notwendig. Dennoch bleibt immer mehr ungenutzte Architektur zurück und macht die weitere Entwicklung langsam und unflexibel. Emergente Architekturen hingegen werden nur soweit durchdacht, wie Anforderungen und Technologie stabil erscheinen, vermutlich also durchaus über mehrere Sprints hinweg. Dieses Denken können Sie auch auf Plattformkonzepte übertragen. Eine gute Plattform basiert auf vielen Kundenprojekten oder wird zusammen mit Kundenprojekten iterativ entwickelt. Eine Plattform im stillen Kämmerlein zu entwickeln, birgt ein hohes Risiko zur „Frameworkeritis“.

Wie weit Sie bei emergenten Architekturen vorausplanen können bzw. sollten, hängt von der Änderbarkeit der Technologie und von der Stabilität der Anforderungen ab. In jeder Technologie müssen Sie irgendwann technische Entscheidungen treffen, die später nur zu enormen Kosten wieder zu ändern sind. Bei der Software können dies verwendete Plattformen, Betriebssysteme usw. sein, bei physischen Produkten konstruktive Entscheidungen: Wenn Sie zum Beispiel ein Flugzeug agil entwickeln, kommt der Punkt, an dem das Grundkonzept nur noch sehr aufwendig zu ändern ist. Auch normative Forderungen wie zum Beispiel bei der funktionalen Sicherheit (z. B. nach IEC 61508 oder ISO 26262) können ebenfalls einen Einfluss auf die Sichtweite bei den Architekturbemühungen haben. Dazu später mehr in einem eigenen Kapitel über den Umgang mit Compliance-Anforderungen.

5.1.2 Alternative Fertigungskonzepte

Wenn Sie Ihr Produkt mit einer modularen Architektur und vielleicht mit teilweise vorhandenen Modulen aufgebaut und funktionsfähig haben, werden Sie damit beginnen, in kleinen Iterationen einzelne Module zu verbessern. Während in der Software der Bau in der Regel automatisiert ist und in überschaubaren Zeiten durchgeführt werden kann, sind die Zeiten für Beschaffung und Aufbau bei physischen Produkten ein signifikanter Bestandteil eines Entwicklungszyklus. Um dies zu durchbrechen, müssen Sie sich mit dem Gedanken anfreunden, über andere Fertigungsverfahren und andere Lieferanten nachzudenken.

Beschaffung

Die Zeit für die Beschaffung von Materialien teilt sich prinzipiell auf in die Prozesszeit bei Ihrem Einkauf und die Lieferzeit Ihrer Lieferanten. Um die Liegezeiten im Einkaufsprozess zu reduzieren, müssen Sie Ziele und Prozesse Ihres Einkaufs für die Produktentwicklung anpassen. Oftmals sind die Einkaufsziele primär auf Kosten und nicht auf Zeiten optimiert. Wenn Sie das Rechenbeispiel im Kapitel über Cost of Delay betrachten, wird ersichtlich, dass die Preise eines in sehr geringen Stückzahlen für die Entwicklung zugekauften Produkts vernachlässigbar sind, gegen die durch Lieferzeiten entstehenden Verzögerungskosten. Die Faustregel für den Einkauf bei der agilen Entwicklung lautet: „Egal was es kostet, es muss morgen bei uns im Haus sein.“ Das ist natürlich etwas provokant und in der Praxis mit Augenmaß einzusetzen: Nur durch Kenntnis von COD können Sie hier das Optimum erkennen.

Für Preisverhandlungen oder umfangreiche Preisvergleiche ist also kein Raum mehr, wenn Sie schnell und flexibel sein wollen. Teilweise erfolgen Lieferantenauswahl und Bestellung dann direkt durch die Entwickler oder der Einkauf bereitet mit den Lieferanten Rahmenverträge vor, aus denen die Entwicklung schnell und unkompliziert Aufträge generieren kann. Eine solche Konstellation macht Sie schnell in der Entwicklung, passt jedoch nicht zu einer eventuell später vorgesehenen Serienentwicklung Ihres Produkts. Sie müssen daher den für Ihr Produkt optimalen Punkt finden, ab dem der Beschaffungsmodus von „schnellen Prototypen“ zu „Serienfertigung“ schwenkt. Für die zweite Phase sind Ihre bestehenden Beschaffungsprozesse schon optimiert. Hier geht es weiterhin darum, auf den Preis zu achten und strategische Partnerschaften mit Lieferanten zu bilden.

Es ist nicht immer einfach, die Beschaffung für die Entwicklung von der Serienbeschaffung zu trennen. Wenn Sie zusammen mit Ihrem Lieferanten entwickeln, ist dies ohnehin ein gemeinsames Entwicklungsvorhaben und kein Einkauf im eigentlichen Sinne. Wenn es für die benötigte Technologie nur einen Lieferanten gibt und dieser volle Auftragsbücher hat, wird es Ihnen schwerfallen, diesen mit höheren Preisen zu kürzeren Lieferzeiten zu motivieren. Für den Großteil aller zugekauften Teile gibt es jedoch Expressoptionen. Additiv und subtraktiv gefertigte Prototypen bekommen Sie bei entsprechenden Anbietern innerhalb weniger Werktage, ebenso wie Leiterplatten mit üblichen Layer-Anzahlen.

Rapid Prototyping

Um die eigentlichen Produktionszeiten zu verkürzen, lohnt es sich, einen Blick auf alternative Materialien und Fertigungsverfahren zu werfen – auch wenn diese in späteren Inkrementen eventuell wieder gegen andere ersetzt werden. Oft haben Entwicklungsorganisationen keinen Überblick über das, was an Technologie und potenziellen Lieferanten am Markt verfügbar ist, denn bisher galt es, das Produkt in einem Zug in den serienreifen Zustand zu bringen, mit Serienlieferanten und -technologien. Es lohnt sich also, im Rahmen der inkrementellen Entwicklung den Rapid-Prototyping-Markt zu untersuchen. Zerspanende Bearbeitung und additive Fertigungsverfahren bringen in wenigen Stunden mechanische Bauteile hervor. Da diese Verfahren oft teure Anlagen benötigen, müssen Sie hier vielleicht auch mit Lieferanten zusammenarbeiten.

Insbesondere in der additiven Fertigung gibt es immer wieder neue Technologien und Konzepte. Mit Sicherheit ist es nützlich, für Ihr agiles Team einen 3D-Drucker zur Verfügung zu stellen, um Konzepte überprüfen zu können. Die damit erstellten Kunststoffteile sind jedoch nur begrenzt im eigentlichen Produkt einsetzbar. Jedoch sind auch für mechanisch und thermisch belastbare Metallteile additive Fertigungsverfahren verfügbar, mit denen Sie aus der 3D-Konstruktion direkt ein komplexes Teil fertigen können. Teilweise finden additive Fertigungsverfahren auch Eingang in die Serienfertigung, denn mit ihnen können zum Beispiel Strukturen in umschlossenen Räumen gefertigt werden, was mit zerspanenden Verfahren oder Gussverfahren so nicht möglich wäre. Tabelle 5.1 zeigt eine kleine Übersicht über etablierte additive Verfahren.

Tabelle 5.1 Gängige additive Fertigungsverfahren

Verfahren	Mögliche Materialien	Bemerkungen
Selektives Lasersintern (SLS)/Selektives Laserschmelzen	Breite Auswahl an Kunststoffen und Metallen, auch z. B. Aluminium oder Edelstahl	Oft ist es möglich, hier schon mit den später in anderen Fertigungsverfahren verwendeten Materialien zu arbeiten. Oberfläche muss unter Umständen mit subtraktiven Verfahren nachbehandelt werden.
3D-Druck (Pulver)/ Schmelzschichtung (FDM)	Kunststoffe	Preiswerte Anlage, Druck im Büro möglich. Teile nur begrenzt mechanisch belastbar.
Stereolithographie	Vordefinierte fotosensitive Materialien	Höchste Genauigkeit aller additiven Verfahren, sehr gute Oberflächen. Einschränkung auf bestimmte Materialien, dadurch nur begrenzt mechanisch belastbar.

Neben diesen sehr ausgereiften Verfahren sollten Sie auch einfache Lösungen zur Überprüfung weniger komplexer Dinge berücksichtigen. Damit meine ich zum Beispiel Bastellösungen aus Karton, Klebeband oder Lego-Steinen. Dies mag zunächst unprofessionell und wenig ingenieursmäßig klingen, gerade aber für Diskussionen zu Bauraum und Montierbarkeit von Produkten ist ein preiswertes echtes 3D-Modell manchmal dem detaillierten 3D-Modell im CAD vorzuziehen: Das ganze Team kann das Problem bzw. das Konzept im wahrsten Sinne des Wortes „begreifen“, und Änderungen sind schnell und preiswert umzusetzen.

In der Elektronikentwicklung sind die Möglichkeiten nicht ganz so vielfältig. Dennoch gibt es für einfache Leiterplatten eine preiswerte Möglichkeit, diese Dienstleister liefern mehrlagige Leiterplatten innerhalb weniger Arbeitstage. Falls Sie mit einer zweilagigen Leiterplatte auskommen, könnte sich die Investition in eine CNC-Fräsmaschine lohnen, mit der Sie zweilagige Leiterplatten in sehr kurzer Zeit selbst herstellen können. Während die klassische Herstellung über fotochemische Verfahren geschieht, werden beim Isolier- oder Trennkanalfräsen genannten Prozess die Kupferschichten mechanisch abgetragen. Durchkontaktierungen (Vias) werden hier in der Regel über eingepresste und verlötete Nieten realisiert.

Praxisbeispiel: Auto im 3D-Druck

Der italienische Hersteller XEV hat ein kleines, zweisitziges Elektroauto entwickelt. Strukturelemente, Antrieb und Fenster werden klassisch produziert, der Rest besteht aus 57 Kunststoffteilen, die auch in der Serie im 3D-Druck erstellt werden sollen. Die im FDM-Verfahren entstehende raue Oberfläche wird überlaminiert, damit entfällt auch der Arbeitsschritt der Lackierung. Laut Internetmedien soll die italienische Post bereits Fahrzeuge vorbestellt haben [Zieler 2018].

Automatisierung des Aufbaus

Der Traum in der agilen Produktentwicklung ist ein sogenannter „Hardware-Compiler“, der – analog zur Software – nach erfolgtem Engineering das Produkt ohne weiteres menschliches Zutun aufbaut. Auch automatisierte Herstellungsverfahren wie additive Verfahren in der Mechanik oder das Trennkanalfräsen bei Leiterplatten liefern nur einzelne Bauteile oder Komponenten. Der 3D-Druck muss eventuell nachbearbeitet und die (einzelne) Leiterplatte von Hand bestückt und verlötet werden. Spätestens beim Zusammenbau der Komponenten ist jedoch in der Regel Handarbeit angesagt. Auch wenn es hierzu spannende Forschungen gibt wie zum Beispiel, ein Fahrzeug um einen bestehenden Antriebsstrang „herumzudrucken“.

Wenn Sie das automatisierte Fertigen einzelner Komponenten durch Dreh-/Fräszentren, 3D-Drucker und Trennkanalfräsen konsequent ausnutzen, haben Sie noch keinen „Hardware-Compiler“, der Zusammenbau durch das Team liefert, aber wichtiges Feedback, und es ist eine Tätigkeit, die im Querbalken des „T-shaped Skill Set“ verortet ist, also flexibel – je nach Auslastung der einzelnen Teammitglieder – vom Entwicklungsteam geleistet werden kann.

5.1.3 Automatisierter Test

In der agilen Software-Entwicklung ist es die Regel, dass das Software-Produkt innerhalb einer agilen Iteration mehrfach automatisch gebaut und getestet wird. Die Infrastruktur dazu ist verfügbar und kann auch relativ einfach eingerichtet werden. Bei der Software wie auch bei physischen Produkten dauert in der Regel ein Testzyklus länger als der Zusammenbau. Daneben lassen sich Tests bei physischen Produkten leichter automatisieren als der Zusammenbau. Je nach Art und Technologie des Produkts ergeben sich verschiedene Optionen zur Automatisierung der Tests.

Testkonzepte

Egal, ob Sie agil oder klassisch arbeiten, müssen Sie sich Gedanken über Ihr Testkonzept machen, also was Sie auf welcher Architekturebene testen können und müssen. Da Sie im agilen Umfeld Tests deutlich öfters durchlaufen und deshalb auch nach weiteren Möglichkeiten zur Automatisierung suchen werden, wird dies unter Umständen auch Ihre Testkonzepte verändern.

Falls Sie bereits in Architekturebenen denken und Modultests gegenüber Integrationstests unterscheiden, können Sie diesen Abschnitt gerne auslassen. Ansonsten ist in den folgenden Ausführungen vielleicht etwas Inspirierendes für Sie dabei.

Generell wird zwischen Komponenten- bzw. Modultests und den Integrationstests auf der jeweiligen Architekturebene unterschieden. Bei Komponententests werden Komponenten isoliert gemäß ihrer Spezifikation getestet, also an ihrer definierten Schnittstelle (Blackbox-Tests) und unter Umständen auch im Innenleben der Komponenten (Whitebox-Tests). Integrationstests testen das Zusammenspiel einzelner Komponenten, oft indem im integrierten Zustand auch die Schnittstellen beobachtet und gemessen werden. Das mag für Sie jetzt alles recht konservativ klingen, doch die Engineering-Grundlagen ändern sich nicht durch die Agilität. Lediglich die Bürokratie dahinter, also wie viel formal spezifiziert und dokumentiert wird, sollten Sie deutlich reduzieren – soweit es Branche und Produkt zulassen.

Für die Automatisierung Ihres Produkttests müssen Sie überlegen, auf welchen Ebenen Sie automatisieren können. Falls Sie bisher nicht in den Kategorien von Komponenten- und Integrationstests denken, möchte ich dies kurz beispielhaft anhand zweier Grafiken beschreiben. In diesem Beispiel gehe ich davon aus, dass sich das Produkt in verschiedene Hauptmodule unterteilen lässt. Deren Schnittstellen und Interaktionen sind in der Systemarchitektur beschrieben. Jedes Modul hat für sich wieder Unterkomponenten, deren Schnittstellen und Interaktionen durch die Modularchitektur beschrieben werden können (Bild 5.3). Architektur ist also fraktal. Wenn Ihr Produkt ein Steuergerät ist, ist es für Ihre Kunden ein Modul in deren Architektur.

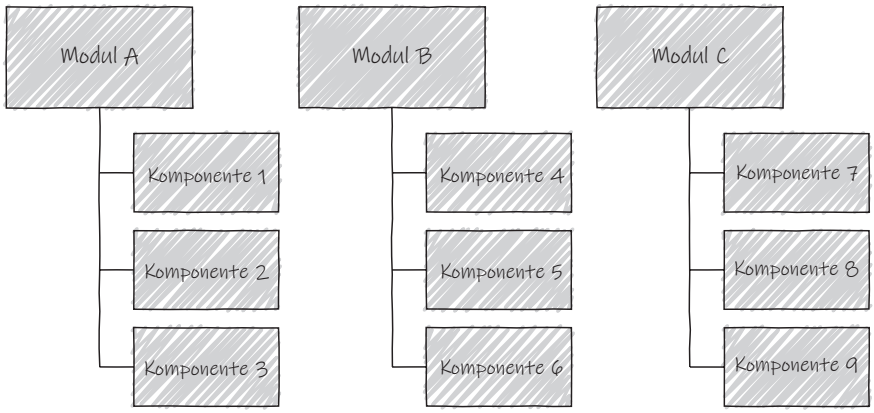

Bild 5.3 Fraktale Architektur

Formal gibt es für jede Ebene zwei Spezifikationen, also einmal für die Architekturelemente und einmal für die Architektur selbst. Architekturelemente werden mit Modul-/Komponententests getestet, die Architektur mit Integrationstests. Die Zusammenhänge lassen sich am besten in der klassischen V-Darstellung abbilden (Bild 5.4). Bitte denken Sie jetzt nicht, dass ich jetzt das V-Modell einführen will. Es geht mir lediglich um die Zusammenhänge zwischen Spezifikation und Test. Auch wenn Sie dies im Agilen weniger formal und vielleicht weniger strikt handhaben werden, verändern sich die Grundlagen des Entwickelns und Testens nicht.

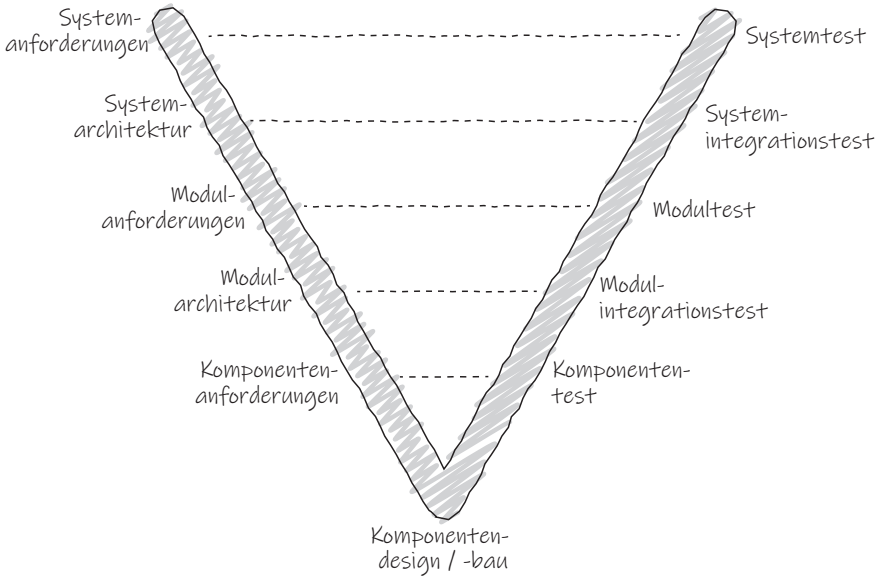

Bild 5.4 Beispiel: Design- und Teststufen in der V-Darstellung

Wenn Sie eine solche Struktur für Ihr Produkt definiert haben, können Sie besser einschätzen, auf welcher Architekturebene Sie während Ihrer Iterationen Elemente tauschen oder optimieren, wie Sie die Veränderungen, also das neue Inkrement testen möchten und wie Ihre Möglichkeiten zur Testautomatisierung sind, um die Testdurchläufe häufig, schnell und in hoher Qualität durchführen zu können. Auf Chancen und Herausforderungen der Testautomatisierung verschiedener Technologien gehe ich im Folgenden kurz ein.

Elektronische Komponenten

Whitebox-Tests von bestückten Leiterplatten können Sie mit automatisierten Tests durch Nadeladapter (Flying Probe Tester) automatisieren. Damit können Sie an bestückten Bauteilen oder an eventuell extra im Layout angelegten Testpunkten, Leiterplatten, Bestückung sowie Bauteiltoleranzen und -funktion vollautomatisch testen. Sie können über eine solche Testinfrastruktur auch Embedded Software aufspielen und die Hardware-Software-Integration testen. Dazu können Sie eine spezielle Testsoftware oder die eigentliche Produktsoftware verwenden.

Für die Blackbox-Sicht auf eine elektronische Komponente, zum Beispiel ein Steuergerät, können Sie die Testautomatisierung recht weit treiben und haben das vermutlich auch schon getan. Das Schlagwort dazu ist der Hardware-In-the-Loop-Test (HIL-Test), bei dem das Produkt von außen mit elektronischen Signalen stimuliert und sein Verhalten automatisch ausgewertet und bewertet wird. Dazu wird dem Gerät von der HIL-Testumgebung seine spätere Umgebung (z. B. ein Fahrzeug) über Simulationen vorgegaukelt. Diese Tests sind auf elektronischer Ebene voll automatisierbar. Umgebungsbedingungen wie Temperatur, Vibration usw. können teilweise auch automatisch simuliert werden, jedoch ist es dazu in der Regel nötig, dass Sie das Gerät manuell von der einen Umweltsimulation in die andere bringen.

Wenn Sie bereits mit einem HIL-Prüfstand unterwegs sind, gilt es hier, die Testabdeckung und den Automatisierungsgrad weiter in die Höhe zu schrauben, denn in einer agilen Entwicklung müssen Sie Ihre Tests in der Regel viel häufiger ausführen als in einer traditionellen Umgebung. Auf das Thema Dauerlauf gehe ich weiter unten ein.

Mechanische Komponenten

Im Gegensatz zu Steuergeräten oder Ähnlichem lassen sich mechanische Komponenten deutlich schwieriger testen. Die Maßhaltigkeit können Sie mit verschiedenen 3D-Messverfahren halbautomatisch überprüfen. Wenn es jedoch um Aspekte wie z. B. Oberflächengüte geht oder um mechanisches Verhalten, werden die Dinge komplizierter. Sicherlich können Sie mit einem Industrieroboter Ihre soeben entwickelten und produzierten Teile mechanisch beanspruchen oder anderweitig Dinge testen, die Anschaffung und der Unterhalt einer solchen Infrastruktur rein für die Entwicklung rechnet sich jedoch in der Regel nicht.

Oftmals können Sie die beabsichtigte Funktion einer mechanischen Komponente nur in „verbautem“ Zustand testen, also in der Interaktion mit anderen Komponenten Ihres Produkts. Dies bedeutet in der Regel einen Aufbau des ganzen Produkts oder zumindest von Teilen davon. Eine Simulation der Umgebung, wie oben bei den elektronischen Komponenten beschrieben, ist hier nicht so umfassend oder gar nicht möglich.

Integrierte mechatronische Systeme

Auch komplexe integrierte mechatronische Systeme, wie zum Beispiel ein Brennstoffzellensystem oder Antriebe, können Sie auf einem entsprechenden Prüfstand weitgehend automatisiert testen, wenn das System elektrisch bzw. elektronisch angesteuert werden kann. Die Herausforderung besteht dann neben einer eventuell notwendigen Simulation des Systemkontexts in der Erfassung von Messwerten. Oft wird es nötig sein, auch Messwerte/Daten aus dem Inneren des Systems zu erhalten. Sie müssen also bei den in der Systemarchitektur festgelegten Schnittstellen unter Umständen auch vorsehen, Daten für einen späteren Integrations- oder Systemtest zur Verfügung zu stellen. Die Architektur muss also auch die Testbarkeit des Systems berücksichtigen.

Kann das System nicht oder nur begrenzt elektronisch angesteuert werden, kann es sich lohnen, über den Einsatz von Robotik nachzudenken. Insbesondere wenn es um automatisierte Tests von Benutzerinteraktionen geht, wenn also irgendwelche Knöpfe gedrückt werden müssen, können preiswerte Spiel- oder Ausbildungsroboter schon ausreichend sein. Alles, was dienlich ist, können Sie in der Entwicklung einsetzen, auch wenn es in traditionellen Denkmustern unprofessionell oder unseriös wirken mag. Und wenn Sie einen USB-Raketenwerfer, der im Büro kleine Schaumstoffraketen verschießt, für die automatisierte Interaktion mit Ihrem Produkt sinnvoll nutzen können: Kaufen Sie ihn.

Praxisbeispiel: Testautomatisierung bei einer Netzwerkkomponente

Das Entwicklungsteam einer Netzwerkkomponente begann, mit Scrum zu arbeiten. Dadurch ergaben sich die bekannten Herausforderungen, die Durchlaufzeiten von Bau und Test deutlich zu verkürzen. Zwei wesentliche Veränderungen bei der Arbeit dieses Teams möchte ich hier kurz herausstellen.

Die Testautomatisierung aus elektronischer Sicht war zwar Aufwand für das Team, jedoch ohne große Hürden umsetzbar. Da das Gerät ein kleines Display und ein paar Drucktaster zur Bedienung besitzt, war der Test bis zu diesem Zeitpunkt nicht voll automatisiert. Das Team beschloss, auch die Benutzerinteraktion mit in die Testautomatisierung aufzunehmen, indem es einen kleinen Roboter beschaffte, wie er in Schulen und Hochschulen zur Ausbildung eingesetzt wird. Diese Investition von wenigen hundert Euro rechnete sich schnell. Der kleine Plastikroboter kann zuverlässig die Bedienelemente des Geräts bedienen. Auch wenn der Aufbau mit Kabelbindern und Klebeband für die Kritiker der Agilität im Unternehmen unseriös wirkte, das Team hat durch den unkonventionellen Ansatz mit einer kleinen Investition eine neue Dimension im Automatisierungsgrad erreicht. Da ein Auslesen des Displays mittels Bilderkennung zu komplex wäre, entschloss sich das Team, die Ansteuerleitung des Displays im Testaufbau nach außen zu führen und so auch den Rückkanal der Benutzerinteraktion mit in die Testautomatisierung aufzunehmen.

Grenzen der Automatisierung

In meinen bisherigen Ausführungen bin ich von Tests zu funktionalen Anforderungen ausgegangen. Ziel der Automatisierung ist in diesem Fall, die Anzahl der getesteten Anforderungen pro Zeiteinheit zu erhöhen. So kann bei einem entsprechenden Automatisierungsgrad die Testzeit verkürzt werden. Beim Test von nicht-funktionalen Anforderungen sind oft nur kleine oder gar keine Abkürzungen möglich. Laufzeittests von mehreren Tausend Stunden lassen sich unter Umständen durch das Testdesign ein wenig verkürzen, doch die Laufzeit auf der Zeitachse, die über viele agile Iterationen hinausgeht, bleibt. Das Entwicklungsteam kann den Testlauf in solchen Fällen nicht abwarten und wird sich einem parallelen Vorhaben widmen. Extreme Beschaffungs-, Bau- oder Testzeiten werden auch die fokussiertesten und agilsten Teams in eine Multiproduktumgebung zwingen.

Eine weitere Art von Tests, welche aus der Software-Entwicklung unbekannt sind, sind zerstörende Tests. Wenn Sie solche Tests benötigen, werden Sie entsprechend mehrere Inkremente pro Iteration bauen, damit zumindest eines Ihre ausgefuchsten Tests überlebt und dazu dienen kann, von Stakeholdern Feedback einzuholen.

5.1.4 Einsatz von Simulationen

Agile Ansätze versuchen, Feedback darüber, ob das Produkt dem Business dienlich ist, wann immer möglich über das Produktinkrement selbst zu erhalten. Konzepte, Zeichnungen und dergleichen markieren in dieser Sicht keinen Fortschritt, sie sind bloße Theorie. Um die Bau- und Testzeiten von komplexen Produkten umgehen zu können, sollten Sie auch den Einsatz von Simulationen in Erwägung ziehen. Diese können in gewissem Umfang Bau und Test ersetzen. Ähnlich wie beim Thema „Rapid Prototyping“ setzen viele Organisationen bisher Simulationen nicht in dem Maß ein, wie es technisch möglich wäre. In eher sequentiell vorwärts gerichteten Entwicklungsabläufen werden Simulationen nur für Teilaspekte genutzt, da die Anzahl der Entwicklungsiterationen und Testzyklen im Vergleich zum agilen Vorgehen sehr klein ist.

Wo Berechnungsverfahren an der Komplexität der Aufgabe scheitern, weil zum Beispiel bei Verformungen, Strömungen oder EMV-Effekten nichtlineares Verhalten auftritt, werden die Annahmen traditionell durch Versuche und physische Tests überprüft. Ein solches Vorgehen ist jedoch oft teuer, zum Beispiel bei Crash-Tests von Fahrzeugen, oder gar gefährlich: Für Herzschrittmacher oder Stents ist es nicht möglich, im menschlichen Körper einen „Crash-Test“ durchzuführen.

Mit der Entwicklung immer größerer Rechenleistung in den letzten Jahrzehnten wurde es möglich, die Physik in viele kleine Berechnungselemente herunterzubrechen und Tausende von diesen sowie deren Interaktion mit viel Rechenleistung abzubilden. Diese Simulationsansätze werden als „Finite Elemente Methode“ (FEM) bezeichnet. Dadurch ist es inzwischen möglich, auch ungefährliche und relativ preiswerte Versuche durch Simulationen zu ersetzen, um die Produktentwicklung zu beschleunigen. Ein Beispiel hierfür könnte die Optimierung einer Verbundverpackung von Getränken sein: Wird Ihr eben gekaufter Orangensaft explodieren, wenn er von Ihrem Küchentisch auf den Boden fällt?

Dieses Buch kann keinen umfassenden Überblick über Simulationsmöglichkeiten für jede Spezialanwendung bieten. Ich greife ein paar Möglichkeiten heraus, und vielleicht ist für

Ihr Produkt und Ihre Vorgehensweise eine Inspiration dabei, tiefer in das Thema einzusteigen.

Simulation von mechanischen Produkten/Komponenten

Der inzwischen weit verbreitete Einsatz von 3D-CAD-Programmen für die dreidimensionale Konstruktion von Produkten und Komponenten liefert einen Teil der Eingangsdaten für eine mechanische Simulation, nämlich die Geometrie des Produkts. Angereichert mit Daten zum Material und anderen Parametern können so über FEM die Verformungen eines Bauteils unter Krafteinwirkungen analysiert sowie Lebensdaueranalysen für mechanische Belastungen durchgeführt werden. Hinsichtlich des thermischen Verhaltens eines Produkts lassen sich mit solchen Simulationen auch Aussagen zur Wärmeverteilung und zur thermisch bedingten Ausdehnung erstellen. Auch sehr dynamische Aspekte wie zum Beispiel Vibrationen oder Strömungen von Gasen und Flüssigkeiten lassen sich inzwischen mit ausreichend Rechenleistung so gut simulieren, dass nur noch wenige physikalische Versuche notwendig sind, um das Modell der Simulation zu validieren.

Neben diesen allgemeinen mechanischen Aspekten gibt es unzählige Spezialanwendungen wie zum Beispiel die Simulation des Verhaltens von Laminaten. Weit verbreitet ist inzwischen die Simulation von Spritzgussverfahren, um das Werkzeug soweit wie möglich vor der zeitaufwendigen und teuren Herstellung zu optimieren. Diese sogenannten „Mold Flow"-Simulationen können sowohl das Fließverhalten beim Einspritzen des Werkstoffs in das Werkzeug simulieren wie auch die Wärmeverteilung beim Füllen und das Schwinden des Werkstoffs beim Abkühlen.

Vermutlich haben Sie schon einige der eben beschriebenen Simulationsmöglichkeiten im Rahmen Ihrer Mechanikentwicklung im Einsatz. Darüber hinaus gibt es inzwischen Ansätze, um komplexe Interaktionen von Komponenten zu untersuchen, zum Beispiel die Simulation eines kompletten Antriebsstrangs von Nutzfahrzeugen. Neben Motor, Getriebe und Steuergeräten wird der komplette LKW simuliert, und es können Testfahrten am Computer durchgeführt werden, um Verbrauch und Fahrkomfort zu optimieren. Dazu wird unter anderem das Vibrationsverhalten des Antriebsstrangs simuliert wie auch die Auswirkungen der Vibrationen auf die Fahrerkabine. Auch in der Luftfahrt gibt es Möglichkeiten, dynamisch die Flugsteuerung eines Flugzeugs zu simulieren und damit auch zu optimieren, bevor der erste Prototyp aufgebaut wird.

Praxisbeispiel: Simulation eines Antriebsstrangs

In einer von Siemens zusammen mit dem Lkw-Hersteller Scania veröffentlichten Fallstudie ist der Einsatz von Simulationen bei der Entwicklung und Optimierung von Lkw-Antriebsträngen beschrieben. Scania hat zuvor schon isolierte Simulationen eingesetzt, die voll integrierte Simulation reduzierte die Zeit für die Modellierung laut Scania um Faktoren zwischen zwei und zehn. Ziel dieser sehr umfangreichen komplexen Simulation ist, das Design des Antriebsstrangs hinsichtlich Getriebeverlusten, Fahrkomfort, Ölfluss im Getriebe und pneumatischer Betätigung zu optimieren. Die Simulationskette geht dabei von verschiedenen Fahrerprofilen über verschiedene Straßenbeschaffenheiten und -steigungen, Getriebe, Antrieb, Reifen bis zu den Chassis- und Karosserie-

teilen. Simulation und Design werden iterativ weiterentwickelt. Dieses Vorgehen erlaubt es, ohne Zeit und Kosten für physische Aufbauten und Tests den Antriebsstrang für die angesprochenen Parameter zu optimieren. Im Endeffekt können mit dieser Simulation komplette Testfahrten des LKWs am Computer durchgeführt werden, was letztendlich auch einen positiven Einfluss auf die Umwelt hat. Datenvergleiche mit wirklichen Testfahrten haben laut der Fallstudie nur minimale Abweichungen ergeben und somit die Wirksamkeit der Simulation bestätigt.

Simulation von elektronischen Produkten/Komponenten

Soll ein Regelkreis an einem physischen Produkt mit einer elektronischen Steuerung versehen werden, ist es inzwischen üblich, die Regelung mit Simulationsmodellen auszulegen und zu untersuchen und aus dem Modell heraus auch Teile des Quellcodes für die Embedded Software zu generieren. Im Folgenden beschreibe ich beispielhaft, an welchen Stellen im weiteren Elektronikentwicklungsprozess durch Unterstützung von Software-Werkzeugen Geschwindigkeit generiert werden kann. Auch für analoge Schaltungen stehen seit vielen Jahren bewährte Simulationswerkzeuge zur Verfügung.

Das Entwerfen einen Schaltplanes kann Ihnen die Software nicht abnehmen; jedoch ist es wichtig, schon bei diesem Arbeitsschritt Informationen zu Preisen und zur Verfügbarkeit der verwendeten Elektronikkomponenten verfügbar zu haben. Manche Software-Werkzeuge zum Schaltplanentwurf bieten diese Auskunft sogar integriert an. Damit haben können Sie Ihr Schaltungskonzept direkt auch hinsichtlich der Beschaffungszeiten optimieren.

Moderne Software zur Entflechtung der Platine (Layout) hat in ihren Bibliotheken nicht nur die Daten, die für die Bestückung relevant sind, sondern auch Informationen zur 3D-Geometrie und zu weiteren physikalischen Eigenschaften. Dadurch steht die Platine im Computer auch fertig bestückt als 3D-Modell zur Verfügung und kann virtuell in Gehäusen verprobt werden, zum Beispiel hinsichtlich der Kollisionsfreiheit bei gestapelten Platinen, Montierbarkeit oder Kabelführung. Sind alle geometrischen Fragen geklärt und das Layout ist – unterstützt durch leistungsstarke Autorouter – soweit stabil, kann am Computer die Versorgungsintegrität überprüft werden: Sind alle Leiterbahnquerschnitte für alle Betriebszustände ausreichend oder tritt an einem Bauteil ein „Voltage Drop“ auf? Die Suche nach solchen Fehlerbildern bei einer physischen Platine kostete früher viel Zeit und Nerven, jetzt können die Problemzonen schon identifiziert und beseitig werden, bevor das erste Zinn für die Bestückung schmilzt.

Durch immer höhere Prozessor- und Signaltaktungen wird neben der Versorgungsintegrität auch die Signalintegrität immer mehr zum Thema. Ist das Layout geeignet für schnelle Signale? Oder werden an irgendeiner Stelle Flanken zu sehr verschliffen (was dann bei einer physischen Platine ebenso wie bei den oben geschilderten Versorgungsproblemen zu unangenehmen latenten Fehlerbildern führen würde)? Neben der Signalintegrität durch hausgemachte Einflüsse wie dem Layout kann durch Simulationen auch das EMV-Verhalten sowohl hinsichtlich Immission wie auch Emission abgeschätzt, das Layout optimiert und gezielte Gegenmaßnahmen ergriffen werden, anstatt wie früher auf Verdacht „Angst-Kondensatoren“ zu verteilen.

Neben den rein elektromagnetischen Eigenschaften können Sie mit modernen Simulationswerkzeugen auch den Wärmehaushalt Ihrer Platine simulieren sowie deren Vibrationsverhalten untersuchen. Um die Elektronik virtuell in höhere Integrationsstufen führen zu können, stehen virtuelle Steuergeräte zur Verfügung sowie vorbereitete Simulationen von komplexen Umgebungen wie zum Beispiel Verbrennungs- oder Elektromotoren oder Batterie- oder Brennstoffzellensysteme. Mit simulierten Fahrzeugen können Sie simulierte Testfahrten mit realitätsnaher Fahrdynamik durchführen und Fahrassistenzsysteme reproduzierbar am Computer testen, unabhängig von Wetter und Gefährdung der Testfahrer.

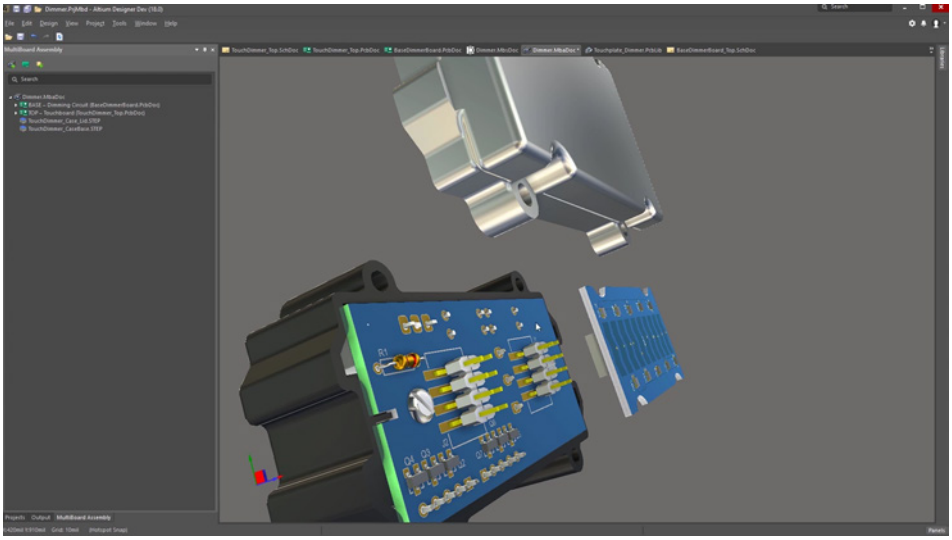

Bild 5.5 Test der Einbausituation direkt im Elektronik CAD (Altium Designer)

Die technischen Möglichkeiten kennen kaum Grenzen. Mechatronische Systeme können über weite Strecken innerhalb von Simulationen entwickelt und zur Reife gebracht werden. Den dazu erforderlichen Aufwänden und Kosten steht die Möglichkeit gegenüber, mit schnellen Zyklen ohne externe Abhängigkeiten die Entwicklung voranzutreiben.

Simulation der Mensch-Maschine-Schnittstelle

Die bisher von mir aufgeführten Simulationsmechanismen erlauben es, elektromechanische Produkte hinsichtlich ihres Verhaltens und ihrer Eigenschaften zu verifizieren. Ein Teilaspekt der agilen Feedbackschleife ist jedoch auch die Bedienbarkeit oder auch die Wartbarkeit von Produkten. Dies sind Punkte, an denen Menschen mit Ihrem System interagieren. Hierzu lohnt es sich, einen Blick auf Simulationen mit Virtual Reality (VR) oder Augmented Reality (AR) zu werfen. Derzeit liegt die Hauptanwendung dieser in hoher Reife verfügbaren Technologien auf Ausbildung und Unterstützung bei der Wartung. Im Rahmen der Produktentwicklung lassen sich solche Simulationen jedoch auch ideal einsetzen, um Bedienkonzepte zu validieren oder die Produzierbarkeit zu optimieren.

Index

Symbole

3D-Druck 97

A

Abhängigkeiten 79, 220, 251
Abstimmungen 22
Agiles Manifest 26
Agile Transformation 293, 300
Auslastung 19
Automotive 67
Automotive SPICE 172
– Level 1 174
– Level 2 177

B

Backlog 30, 49, 120, 227, 236
Backlog Replenishment Meeting 49
Blockade 49
Budgetierung 153, 280
Burndown Chart 38, 254, 266

C

CCPM *siehe* Critical Chain Project Management
CFD *siehe* Cumulative Flow Diagram
COD *siehe* Verzögerungskosten
Co-Location 58, 149, 207
Community of Practice 60, 272
Compliance 299
– Dokumentation 170
– Rollen 171
Continuous Integration 75
– Prinzipien 76
Cost of Delay 96 *siehe* Verzögerungskosten
Critical Chain Project Management 115, 255, 264
Cumulative Flow Diagram 50
Cynefin-Modell 5

D

Daily Scrum 31, 34
DBR *siehe* Drum-Buffer-Rope
Definition of Done 38, 51, 84
Definition of Ready 36, 51, 245
Delegation Poker 112
Design Pattern 92, 115, 128
De-Skalierung 41
Development Team 31 f., 227, 245
DoD *siehe* Definition of Done
Drum-Buffer-Rope 13 f., 46
Durchlaufzeit 108, 210, 212, 246, 252, 267, 288
Durchsatz 251

E

Einkaufsprozess 96, 129, 161
Einladung 142, 295, 301
Engpasstheorie 13, 255, 267, 270
Entscheidungen 112, 137
Entwicklung
– iterativ-inkrementelle 65, 71, 292, 294

Experiment 72, 294, 299
Experte 60, 122, 124

F

Feedback
– zum Produkt 66
– zum Prozess 67
FORDEC 113
Fortschrittsmessung 38, 73
Funktionale Sicherheit 180
– ISO 26262 180

G

Geschwindigkeit
– nachhaltige 56

I

Impediment 195, 147, 259, 269, 163
Impediment Backlog 33, 40, 269
Inkrement 31, 34, 65, 81, 85, 236, 250
– Elektronik 91
– Mechanik 92
Integration 75
Iteration 159, 167, 231 f.
iterativ-inkrementell 299

K

Kadenz 114, 155, 166, 208, 221, 235, 237
Kanban 12, 46, 199, 201, 246, 278
– Board 47, 216, 223
– Policies 51, 221
Kompetenzprofil 61, 98, 122
Kontextwechsel 55

L

Large Scale Scrum 43
Lastenheft 242
Leadership 133, 140, 195, 226, 259, 269, 273
Lean Development 15
Lean Production 10
LeSS *siehe* Large Scale Scrum
Liberating Structures 266
Lieferant 157, 192
Lieferzeit 212
Liegezeiten 21, 61
Linienorganisation 149
Losgröße 10, 21, 107, 110

M

Manifesto for Agile Software Development *siehe* Agiles Manifest
Matrix-Organisation 60, 272
Meilenstein 248, 250, 263
Metriken 251, 266
Motivation 53, 266, 295, 301
– extrinsische 54
– intrinsische 54

N

Nachhaltige Geschwindigkeit 56
Nexus 43

O

One Piece Flow 10
Open Space Agility 293
Open Space Technology 266, 297

P

Pairing 62, 123
PBI *siehe* Product Backlog Item
PEP *siehe* Produktentstehungsprozess
Pflichtenheft 242
Portfolio-Management 118, 146, 166, 211, 277
Prioritäten 145, 214, 243, 259
Product Backlog 30, 33, 36, 242
– Qualität 37
– Schätzungen 37
– User Story 36
Product Backlog Item 33, 36, 245
Product Backlog Refinement 35, 245

Product Owner 30, 32, 111, 150, 194, 225, 242, 245
Produktentstehungsprozess 154, 158, 231
Prototypen 82, 97
Puffer 115, 208, 255, 257, 264
Pull-System 12, 15, 46, 117, 211, 279
Push-System 116

Q

Qualität 37
Qualitätsmanagement 163

R

Reifegrade
– Inkrement 83
Reporting 142, 259
Retrospektive 49, 265
Rollen 57
– handlungsorientierte 57
– kommunikationsorientierte 57
– wissensorientierte 57

S

SAFe® 44
Schätzungen 37, 211, 246 f., 253
Scrum 28, 54, 198, 170, 200, 183
– Artefakte 30, 33
– Events 30, 34, 111
– Guide 30
– Leader-Scrum-Team 143
– Rollen 30
Scrum Master 32 f., 150, 226, 266, 269
Scrum@Scale 44
Scrum Team 135
Selbstorganisation 134, 164
Servant Leadership 140
Server
– Warteschlange 19
Serviceklassen 48
Set-Based Design 168
Shu Ha Ri
– Definition 204
Simulation 104
Skalierung 41, 233, 265
Skill Set *siehe* Kompetenzprofil
Spielemechanik 54
Spike 72
Sponsor 194
Sprint 31, 34, 241
Sprint Backlog 34
Sprint Planning 31, 34
Sprint Retrospective 32, 35
Sprint Review 31, 35
Stacey-Matrix 8
Swarming 124
Swimlanes 48
Synchronisation 115, 166, 208
Systemarchitektur 90, 94, 99, 128
Systemintegration 168, 208, 263

T

Taktung 166
Taskboard 219
Team 57
– co-located 58
– Feature-Team 59, 204
– Komponententeam 59, 204
– Teambildung 57
– Zuschnitt 59
Teambildung 57
Team Happiness 252
Technische Schulden 38
Testautomatisierung 76, 101 f., 109
Testkonzepte 99
Theory of Constraints *siehe* Engpasstheorie
Timebox 228, 35
TOC *siehe* Engpasstheorie
Toyota-Produktionssystem 9, 29, 138, 200, 260
TPS *siehe* Toyota-Produktionssystem
Trainings 197
Transaktionskosten 22, 109
Transparenz 4, 38, 55, 118, 138, 254, 261, 289, 292, 294
T-shaped skill set 62

U

Umgebung
– komplexe 30, 90, 284, 292
User Story 36

V

Value Stream Mapping 282
Variabilität 15, 37
Velocity 37, 40, 247, 249, 251
Verschwendung 10, 259, 289
Verzögerungskosten 17, 96, 243f., 264
V-Modell 100, 154
Vorhersage 253

W

Warteschlange 19, 119, 122, 214
Weighted Shortest Job First 243, 280
Wertstrom 153, 193, 203, 229, 189, 277
WIP 48, 50, 116, 145, 210, 214, 223, 252, 278
Work in Process *siehe* WIP
WSJF *siehe* Weighted Stortest Job First

Z

Zielvereinbarung 139
Zykluszeit 80, 89, 166