3 Das Wesen von Softwareentwicklung

Softwareentwicklung hat wenig bis nichts mit der Herstellung von Massengütern gemein. Vielmehr befasst sich Softwareentwicklung mit der Lösung von Problemen. Welche Konsequenzen sich daraus ergeben, ist Thema des nun folgenden Kapitels. Wir stellen Ihnen drei Thesen vor, die aufgrund unserer Erfahrung den Kern der Softwareentwicklung ausmachen. Im weiteren Verlaufe des Buchs werden wir diese detaillieren und auf die Konsequenzen eingehen.

3.1 Was ist Softwareentwicklung eigentlich?

Softwareprozesse, die auf tayloristischen Prinzipien basieren, nehmen sich Fertigungsprozesse aus der Industrie zum Vorbild. Kennzeichnend für diese Prozesse ist, dass das jeweilige Endprodukt exakt bekannt und definiert ist. Im Rahmen einer Massenfertigung soll dieses tausend- oder gar millionenfach identisch und mit konstanten qualitativen Eigenschaften hergestellt werden.

Ausgehend von einem fertigen Prototyp leiten die Ingenieure der Forschungs- und Entwicklungsabteilung die dafür benötigten Fertigungsschritte ab und optimieren diese. Sie erstellen so zu einem standardisierten Produkt ein ebenso standardisiertes Rezept zu dessen Bau. Die Produktionsabteilung arbeitet anschließend das Rezept in der gewünschten Menge ab. Dank der definierten und kontrollierten Herstellungsweise erzeugt eine Firma so Produkte mit denkbar geringen Qualitätsschwankungen. Der Erfolg dieses Prinzips in Bezug auf Kosten und Qualität in der Industrie ist unbestritten.

Softwareentwicklung ist keine industrielle Fertigung

Wie verhält sich das nun in der Softwareentwicklung? Rolf, GUI-Entwickler, schildert eine typische Ausgangslage eines Softwareprojekts:

»Die Ziele waren gegeben, der Kunde hatte eine recht genaue Vorstellung, was er erreichen wollte, und nannte ebenfalls einige konkrete Features. Doch gab es auch Benutzer an einem anderen Standort, und

deren Bedürfnisse kannte niemand so genau. Selbstverständlich gab es unterschiedliche Lösungsalternativen mit Vor- und Nachteilen, doch so etwas wie ein Lösungskonzept war noch nicht da. Dies mussten wir zuerst entwickeln und die Alternativen entsprechend abwägen. Das ist uns – glaube ich – ganz gut gelungen. Leider änderten sich auch die Vorstellungen der Anwendervertreter im Projekt, gerade nach einer Demonstration des aktuellen Zwischenstands und insbesondere als zum ersten Mal Vertreter des anderen Standorts die Software begutachteten.«

Rolfs Schilderung ist exemplarisch für viele Projekte und deutet auf die wesentlichen Merkmale von Softwareentwicklung hin.

Das Ziel eines Softwareprojekts ist eine für die Benutzer zufriedenstellende Lösung, soviel ist klar. Allerdings gibt es verschiedene Lösungsalternativen – gute und weniger gute, effiziente und weniger effiziente. Welche es gibt, und welche unter Abwägung aller Umstände tatsächlich die beste ist, ist zu Beginn des Projekts unbekannt. Ein Projektteam tastet sich im Verlauf des Projekts an eine Lösung heran. Da es für gewöhnlich keine perfekte Lösung gibt, sondern Lösungen mit unterschiedlichen Stärken und Schwächen, wägt ein Projektteam also immer wieder Vor- und Nachteile einzelner Lösungen und Teillösungen ab.

Zudem ist zu Beginn für gewöhnlich auch die Ausgangssituation nur ungenügend bekannt. In Rolfs Beispiel ist es die Arbeitsweise am entfernten Standort, in anderen Situationen sind es unzureichende Kenntnisse der Fachdomäne, neue Geschäftsprozesse, Ergebnisse paralleler Entwicklungen und vieles mehr. Ein Projekt ist somit immer auch ein Lernprozess für die Beteiligten. Diese verstehen mit der Zeit die Bedürfnisse der Anspruchsgruppen besser und können so besser abschätzen, welche Lösungen eher gut und welche eher schlecht funktionieren könnten.

Unscharfes Ziel

Ein Softwareprojekt hat in aller Regel ein unscharfes Gesamtziel. Dies ist bedingt durch Unwissen über den Ausgangszustand, unklare Lösungsmöglichkeiten sowie Kombinationsmöglichkeiten aus verschiedenen Alternativen.

Nun steht eine Software nicht alleine da, sondern in einer engen Wechselwirkung zum Kontext. Wozu und wie werden Benutzer die Software verwenden? Wie sieht es mit konkurrierenden Lösungen und Bestrebungen aus? Wie ist die Wertschöpfungskette? Wie arbeitet das Unter-

nehmen heute, wie vielleicht in zwei Jahren? Eine Software soll für einen bestimmten Kontext funktionieren und dieser ist dynamisch. Die Dynamik kann dafür sorgen, dass selbst an den Grundfesten einer Softwareentwicklung gerüttelt wird, wie folgender Bericht von Evelyn, Softwarearchitektin, unterstreicht:

»Wir steckten mitten in der Programmierung, als der Konzern entschied, eine größere Restrukturierung vorzunehmen. Da unser System die Kommunikation zwischen den Fachleuten elektronisch unterstützte, die Reorganisation diese Kommunikation aber völlig auf den Kopf stellte, war die Software quasi obsolet. Wir mussten konzeptionell wieder an den Anfang zurück und überlegen, was unter den gegebenen Umständen noch Sinn macht und was nicht.«

Evelyns Beispiel ist ein Extrem: Die Änderung im Kontext stellt das ganze Projekt infrage. Weit häufiger sind in der Praxis graduelle Änderungen. Das Projektteam, das eine Lösung liefern soll, wird aufgefordert, die Lösung entgegen ursprünglichen Plänen auf die eine oder andere Art zu modifizieren, die Prioritäten zu ändern oder etwas zusätzlich zu bauen. Kommt Ihnen das bekannt vor?

Eigenschaften der Softwareentwicklung

Softwareentwicklung lässt sich demzufolge mit folgenden Punkten charakterisieren:

- Unwissen über Ausgangszustand
- Unscharfes Ziel
- Verschiedene, gegeneinander abzuwägende Lösungsalternativen
- Kompromissbildung
- Dynamik im Kontext

Damit besteht in der Softwareentwicklung eine komplett andere Situation als in der industriellen Fertigung. Sämtliche von dort übernommenen Ideen basieren darauf, dass das Endprodukt sowie der Prozess zum fertigen Produkt zweifelsfrei vorab bekannt sind. Beides ist in der Softwareentwicklung nicht gegeben.

Softwareentwicklung ist kein Herstellungsprozess

Der Weg zur entwickelten Software kann nicht im Voraus detailliert beschrieben werden.

Softwareentwicklung ist Problemlösen

Softwareentwicklung funktioniert grundlegend anders als die Fertigung eines Produkts. Denken Sie an ein Schachspiel: Es gibt eine Unmenge an Varianten, wie am Ende die Figuren stehen können und auch wie die Spieler diesen Stand erreichen. Ein Spieler weiß zudem nicht, wie der Gegenspieler auf die eigenen Züge reagieren wird. So

lässt sich eine Partie nicht im Voraus im Detail planen. Ein Spieler muss im Kopf Stellungen bewerten, Zugfolgen durchspielen, Gelegenheiten packen, also den Plan kontinuierlich anpassen. Softwareentwicklung hat viel mehr mit Schachspielen gemeinsam als mit der Produktion eines Autos.

Softwareentwicklung ist Problemlösen

Die Herausforderung ist das Finden einer guten und akzeptierten Lösung.

Die allgemeine Frage nach Produktivität in der Softwareentwicklung lässt sich nun konkretisieren: Wie lösen Menschen produktiv Probleme? Dies ist Gegenstand von Untersuchungen von Kognitionspsychologen. Was die Forschung herausgefunden hat und was das für eine neue Sicht auf die Softwareentwicklung bedeutet, lesen Sie in Kapitel 4.

3.2 Team

Die von der Industrie übernommenen Prinzipien beeinflussen nicht nur unsere Vorgehensweisen im Projekt, sondern auch das Denken in der Zusammenstellung und der Funktionsweise von Teams. Tayloristisch geprägte Softwareteams bestehen aus Spezialisten, die im Rahmen der Arbeitsteilung eine jeweils eng umfasste Aufgabe erfüllen, beispielsweise Anforderungen erheben. Das Arbeitsergebnis einer Person wird dann zum nächsten Spezialisten weitergereicht, also zum Beispiel dem Softwarearchitekten, der es weiterverarbeitet, bevor auch dieser sein Ergebnis jemandem übergibt.

Damit wird, analog zur industriellen Massenfertigung, aus den Menschen ein letztendlich austauschbarer Produktionsfaktor. Sie erfüllen eine eng definierte Rolle mit klarem Input und Output, im Grunde ähnlich wie eine Maschine.

Der nächste Schritt für Unternehmen ist dann nur logisch. Zuerst standardisiert ein Unternehmen die Prozesse organisationsweit mit dem Ziel, diese quantitativ zu optimieren. Ist dies erreicht, können anschließend ganze Teile des Prozesses, beispielsweise die Entwicklung oder das Testing, einfach in ein Billiglohnland ausgelagert werden. So versucht die Firma, dieselbe Leistung für weniger Geld zu erhalten – in dem Gedankenmodell ist es ja nur eine Funktion, die irgendwie irgendwo erfüllt werden muss.

Nun, funktioniert das in der Softwareentwicklung wirklich genauso wie in der Industrie? Sind die Menschen in einem Software-

team austauschbare Produktionsfaktoren in einem standardisierbaren Prozess? Was sind die Konsequenzen dieser Haltung?

Andere Menschen, andere Problemlösung

Evelyn, Softwarearchitektin, erzählt:
»Wir wählten hier nicht eine Standardarchitektur, sondern bauten uns eine eigene Lösung zusammen, die insbesondere Multicasts auf dem Netzwerk zuließ. Einer unserer Entwickler hatte den Vorschlag gemacht, da er in einem früheren Projekt die Vorteile dieses Ansatzes schätzen gelernt hat. Das erwies sich im Nachhinein als Glücksfall. Wir stellten im Betrieb nämlich fest, dass deutlich mehr Anfragen auf unser System einprasselten als erwartet. Die Standardarchitektur hätte dies nicht geschafft. Ein anderes Team hätte vielleicht anders entschieden und sich dann an der schlechten Reaktionszeit die Zähne ausgebissen.«

Ob die getroffene Entscheidung nun richtig oder falsch war, sei dahingestellt, Evelyns Beispiel deutet auf etwas ganz anderes hin: Die an der Problemlösung mitarbeitenden Menschen sind ein höchst individueller Teil der Problemlösung. Mit dem Ersetzen durch andere Menschen würde nicht nur die Problemlösung in der Vorgehensweise, sondern sogar die spätere Lösung selbst ganz anders aussehen.

Wie ungewöhnlich dies auf den ersten Blick auch klingen mag, bei genauer Betrachtung macht dies Sinn. Jede Person bringt ein eigenes technisches und methodisches Erfahrungsspektrum, unterschiedliche Problemlösefähigkeiten und vieles mehr mit. Genau solche Faktoren entscheiden über das Ergebnis und den Weg einer Problemlösung. In Kapitel 4 werden wir noch näher darauf eingehen.

Ökosystem Team

Seit der Aufklärung prägt die Menschen der westlichen Welt ein sehr deterministisches Weltbild. Alles basiert auf dem Prinzip von Ursache und Wirkung. Die Physik der Welt um uns herum – so die Denkweise – muss nur verstanden werden, dann können wir sie uns untertan machen.

Genau so deterministisch soll auch die Softwareentwicklung sein. Die Autoren der diversen Vorgehensmodelle haben die Physik der Softwareentwicklung bereits für uns erforscht und eine Wegbeschreibung angefertigt. Die Projektteams können es sich daher einfach machen und nur noch mechanisch die Anweisungen dieses Rezepts befolgen. Voilà, die fertige Software steht glänzend und zitronenfrisch duftend im Regal. Alles ist vorher bestimmbar, die Arbeit ist simpel, überprüfbar, Ressourcen sind austauschbar und der Erfolg reproduzierbar.

Zugegeben, vielleicht ist das etwas überspitzt formuliert. Allerdings unterhalten gerade Großunternehmen häufig eigene Abteilungen, die nichts anderes tun, als die unternehmenseigenen Prozesse zu definieren und deren Einhaltung durch Audits in den Projekten zu

überwachen. Einen solchen Aufwand kann nur rechtfertigen, wer dieses Weltbild im Kopf hat.

Auch wenn es vielleicht einem inneren Bedürfnis nach Kontrollierbarkeit der Welt um uns herum widerspricht: Das deterministische Weltbild passt nicht zum Lösen von Problemen, und damit nicht zur Softwareentwicklung. Hier gibt es leider keine Garantie für Erfolg, und Ressourcen sind nicht ohne Konsequenzen auf das Ergebnis austauschbar. Sowohl das Produkt als auch der Weg dahin wird von den beteiligten Menschen bestimmt. So hängt dann auch eine produktive Zusammenarbeit zweier Personen individuell von diesen Personen ab, und nicht von einer extern gemachten Vorgabe. Arbeitsweisen, die in einem Team gut funktionieren, können daher in einem anderen ineffizient sein.

Ein Team ist ein Ökosystem, keine Maschine

Die Herausforderung besteht darin, ein Team so produktiv wie möglich zu machen.

Problemlösende Teams lassen sich also folgendermaßen charakterisieren:

- Jeder Mensch im Team ist ein individueller Produktionsfaktor.
- Jedes Teams ist einzigartig.
- Lösung und Lösungsweg sind für jedes Team und Problem anders.

Kapitel 5 befasst sich deshalb eingehend mit dem Ökosystem Team.

3.3 Projektführung

Auch wenn das Management akzeptiert, dass es nicht alles vorherbestimmen kann, so möchten die Führungskräfte logischerweise das Projekt und sein Ergebnis nicht dem Zufall überlassen. Das Thema Projektleitung bildet daher die Klammer um die in den beiden vorangegangenen Abschnitten thematisierten Gesichtspunkte Problemlösung und Team.

Wie lässt sich ein Softwareprojekt führen? Wie stellt die Projektleitung eine gute Lösung sicher und wie sorgt sie für einen möglichst effizienten Einsatz der Ressourcen?

Klassisches Projektmanagement

Toni, Projektleiter, erzählt:
»Ich erstelle typischerweise die zu erarbeitenden Arbeitspakete und teile diese auf die Leute im Team auf. Ich sorge auch dafür, dass alle die Prozessvorgaben einhalten und die Vorlagen mit den passenden Informationen gefüllt sind. Wie meine Leute diese Vorlagen ausfüllen, ist natürlich deren Spezialgebiet, und ich halte mich da zurück. Die Fort-

schrittskontrolle ist dann relativ einfach: Ich schaue mir regelmäßig an, woran die Leute gerade arbeiten und wann sie damit fertig sind. Natürlich muss ich auch für motivierte Mitarbeiter sorgen. Dazu sind eine Kaffeemaschine mit Gratiskaffee und gelegentliche Teamaktivitäten probate Mittel.«

Die klassische Projektmanagementliteratur sieht im Projektleiter den Noten- und Taktgeber sowie den Kontrolleur. Die Philosophie, die sich so auch z.B. bei PMI [PMI 2008] wieder findet, ist klar: Am Anfang wird das Ziel definiert, dann werden die Schritte dahin geplant und anschließend deren Ausführung überwacht. Toni, Projektleiter, erzählt weiter:

»Es liegt in der Natur der Sache, dass nicht alles immer ganz rund verläuft. Vorlagen werden nur schlecht ausgefüllt, ich muss da immer wieder nachhaken. Oder Leute arbeiten aneinander vorbei und ich muss diese koordinieren und auch zur Zusammenarbeit motivieren. Des Öfteren haben Dokumente zwar die Reviews und Meilensteine geschafft, jedoch mit ungenügendem Inhalt, wie sich erst später herausstellt. Solcherlei Dinge führen immer wieder zu ungeplanten Verzögerungen und letztlich zu Kosten. Hier zu überwachen und genügend Reserven einzuplanen ist ebenfalls meine Aufgabe.«

Tonis Auffassung der Aufgabe eines Projektleiters zeigt eine Diskrepanz zu den Aussagen der beiden vorangegangenen Abschnitte. Sind Problemlösen und Team zwei bestimmende Faktoren von Softwareentwicklung, stellt sich sofort die Frage, warum Toni die Problemlösung derart ungeführt seinen Spezialisten überlässt? Kann Toni mit seinen Kontrollen sicherstellen, dass diese Spezialisten eine gute Lösung finden? Und reicht für eine effiziente Zusammenarbeit im Team tatsächlich, sich auf Vorlagen, Kaffeeküche und Teamevents zu verlassen?

Toni konzentriert sich auf Verwaltungstätigkeiten und unterstützt den Problemlösungsprozess nur ungenügend. Dieses Verhalten entspricht den Ansätzen des klassischen Projektmanagements. Dessen Ursprünge liegen wiederum außerhalb der Softwareentwicklung, und entsprechend findet sich auch dort kaum etwas zum Thema Führung des Problemlöseprozesses. Warum auch – beim Bau eines Hauses oder eines Tunnels steht ja das Ergebnis vorher bereits fest. Und findet sich doch etwas zum Thema Problemlösen, dann sind es bestenfalls Kreativitätstechniken. Das greift natürlich bei Weitem zu kurz. Auch wird kaum je diskutiert, was effektive Teams ausmacht.

Führung einer Entwicklung ist mehr als Verwaltung

Ein Projekt zu führen heißt, die beiden zentralen Aspekte einer Softwareentwicklung zu steuern und zu fördern: den Lösungsfindungsprozess und die Produktivität im Team.

Projektleitung in agiler Softwareentwicklung

Einer der wichtigsten Unterschiede zwischen traditioneller und agiler Softwareentwicklung ist gerade das Verständnis, wie Projektleitung zu geschehen habe. Agile Prozesse definieren keine explizite Rolle des Projektleiters – möglicherweise als Resultat aus Erfahrungen, wie sie von Toni geschildert wurden. In einer agilen Softwareentwicklung ist das Team insgesamt für eine gute Problemlösung zuständig. Kurze Iterationen mit ausführbarer, dem Kunden vorgeführter Software sorgen für eine regelmäßige Überprüfung des Fortschritts und des Ergebnisses. Vorgaben für die Zusammenarbeit innerhalb des Teams gibt es kaum, die Leute sollen sich selbst organisieren und dabei ein paar agile Prinzipien beherzigen.

Evelyn, Softwarearchitektin, erzählt:
»Ich habe an der Entwicklung einer Steuerungssoftware für eine Anlage mitgearbeitet. Das Team, es waren an die dreißig Personen, hat Scrum angewandt und wurde auch von einem angesehenen externen Experten gelegentlich beraten. Sie haben alle agilen Techniken gewissenhaft ausprobiert, sich dann aber doch in der Komplexität verzettelt und immer mehr die agilen Praktiken einschlafen lassen. Eine Hauptschwierigkeit für das Team war, dass sich die Hardware selbst stark änderte und auch die Vision der Anlage, hohe Qualität vs. hoher Durchsatz, schwankte. Hier fehlte meiner Meinung nach eine klare strategische Führung, wie das Gesamtproblem anzupacken und worauf zu optimieren sei.«

Gemeinsamer Nenner: eine starke Hand

Auch wenn agile Prozesse keinen Projektleiter vorsehen, so benötigen auch nach agilen Prinzipien durchgeführte Projekte Führung. Je größer und komplexer das zu lösende Problem, desto wichtiger ist eine solche Führung.

Die Aufgabe der Führung ist es, ein schlagkräftiges Team zu schaffen und eine produktive Problemlösung zu bewirken. Ob nun ein Projektleiter, ein Zweigestirn aus Product Owner und Scrum Master oder sogar andere Personen die Führungsfunktionen übernehmen, sei dahingestellt. Problemlösen lässt sich nicht fremd steuern und Produktivität nicht verordnen. Auch passives Verwalten oder Laisser-faire sind nicht Erfolg versprechend.

Kapitel 8 wird sich intensiver mit diesem Themenkreis auseinandersetzen.

Führung ist aktive Unterstützung des Teams und des Problemlösens

Die Herausforderung einer Projektführung ist es, das Team bestmöglich im Problemlöseprozess zu unterstützen und so dessen Produktivität zu steigern.

3.4 Wissensarbeit als Herausforderung

Die an einer Softwareentwicklung beteiligten Menschen leisten nicht nur Routinearbeit, sondern setzen insbesondere ihr Wissen und ihr Denken ein, um Neues zu entwickeln. Softwareentwicklung ist damit Wissensarbeit. Die bestimmenden Faktoren in der Wissensarbeit sind die individuellen Menschen und wie diese im Team gemeinsam Probleme lösen. Diese Ansicht widerspricht dem aus dem tayloristischen Ansatz gewonnenen physikalischen Verständnis, bei dem Spezialisten einer vordefinierten Wegbeschreibung wie einem Rezept folgen.

Aus diesem Blickwinkel postulieren wir drei Thesen zur Softwareentwicklung:

- Softwareentwicklung ist Problemlösen, kein Herstellungsprozess.
- Ein Team ist ein Ökosystem, keine Maschine.
- Führung ist aktive Unterstützung des Teams und des Problemlösens.

Diese Thesen sollen dabei helfen, das Wesen der Softwareentwicklung besser zu verstehen und so wirksamere Vorgehensweisen zu entwickeln. Die damit einhergehenden Fragen sind:

- Wie funktioniert Problemlösen?
- Wie funktioniert ein Team?
- Wie können Teams produktiver werden?
- Welchen Mehrwert kann und muss Führung leisten?

Im weiteren Verlauf des Buchs vertiefen wir diese Thesen und Fragen und geben Ihnen viel Denkstoff, wie Sie die damit verbundenen Herausforderungen besser meistern können.

3.5 Zusammenfassung

Softwareentwicklung befasst sich mit Wissensarbeit. Problemlösende Tätigkeiten, wie sie auch die Kognitionspsychologie untersucht, sind dabei essenziell. In der Wissensarbeit werden Individuen und Teams zu einzelnen Produktionsfaktoren und können nicht ohne Konsequenz auf das Ergebnis ausgetauscht werden. Lösung und Lösungsweg sind für jedes Problem und jedes Team anders. Mit dieser Betrachtungs-

weise ändern sich auch die Anforderungen an die Führung: Diese soll Teams zu produktiven Problemlösern entwickeln. Diese Aspekte werden in den folgenden Kapiteln vertieft behandelt. Damit möchten wir Ihnen Denkanstöße für ein grundlegendes Modell hinter der Softwareentwicklung mitgeben und Ihnen einen darauf basierenden, idealen Softwareentwicklungsprozess näherbringen.

Im nächsten Kapitel legen wir dazu das Fundament mit Betrachtungen des Problemlösens aus der Sicht der Kognitionspsychologie. So machen wir Sie mit der Denkweise der Softwareentwickler und anderer im Projekt beteiligter Personen vertraut und entwickeln ein neues Verständnis für Projektarbeit.