# Wie managed man die Entwicklung von SCOPELAND®-Anwendungen?

Vom Wasserfall-Modell bis hin zu agiler Softwareentwicklung – Welches ist das optimale Vorgehensmodell für maßgeschneiderte Datenbankanwendungen?

These 1: Das Wasserfall-Modell ist nicht so schlecht, wie es aussieht.

These 2: SCRUM ist auch nicht die Antwort auf alle Probleme.

These 3: Phasenagiles Vorgehen kombiniert Agilität mit Planbarkeit.

Daher wählt Scopeland Technology als Vorgehensmodell das **phasenagile Vorgehen nach IPMA**.

## Inhalt

## Abbildungsverzeichnis:

# 1. Hintergrund

Maßgeschneiderte Softwarelösungen müssen in fast allen Fällen unter der unangenehmen Randbedingung entwickelt werden, dass die genauen Anforderungen zu Projektbeginn noch nicht exakt genug feststehen und sie somit noch nicht direkt umgesetzt werden können. Selbst wenn man glaubt, im Vorfeld schon alles zu wissen, wird man doch immer wieder eines Besseren belehrt: Wenn die Endanwender dann das fertige Programm zu sehen bekommen, stellen sie oft fest, dass das doch nicht so ganz das ist, was sie gemeint haben; oder dass sie dann erst die richtigen Ideen entwickeln können, wie das Programm viel besser aufgebaut sein sollte.

Ist also die „Agile Softwareentwicklung“ nach SCRUM die Lösung dieses Problems? Ja und nein. Agilität ist zwingend nötig, um all den Erkenntnisgewinn aufnehmen zu können, der sich erst im Laufe des Projekts abzeichnet. Aber heißt das wirklich, dass man erst im Projektverlauf entscheidet, was umgesetzt werden soll? Nein, das widerspricht zum einen dem unvermeidlichen Prinzip des Festpreisprojekts, und es wäre auch nicht wirklich effizient, wenn man ständig aufs Neue immer wieder von vorne anfangen würde nachzudenken, und immer wieder aufs Neue anfangen würde das Datenmodell umzubauen, Schnittstellen neu zu definieren oder gar das Bedienkonzept völlig über den Haufen zu werfen. Wenn die (SCRUM-) Schleifen, die man dann immer wieder dreht, zu groß sind, muss man damit rechnen, dass man letztlich alles mindestens zweimal macht; und dass das Projekt folglich auch mindestens das Doppelte kostet von dem, was nötig wäre. Außerdem sind turnusmäßige Bereitstellungen ständig neuer Versionen für Auftragsentwicklungen nur eingeschränkt sinnvoll, und wo genau der Product Owner angesiedelt sein soll – beim Auftraggeber oder Auftragnehmer oder auf beiden Seiten – ist auch nicht klar.

Als logische Konsequenz sollte man daher, ausgehend von den bereits zu Beginn bekannten Anforderungen, ein Vorgehensmodell verwenden, das sich an dem klassischen Wasserfall-Modell orientiert und auf dessen grundsolidem Kerngedanken „Erst denken, dann handeln“ aufbaut. Also ein Vorgehensmodell, bei dem man nicht immer wieder zurückkehrt zu den Grundfragen des „Was will ich eigentlich?“. Man muss nicht immer alles infrage stellen, denn der konkrete, unstrittige Bedarf hatte ja zuvor das Projekt generiert und den Projektrahmen definiert, und genau dafür hatte der Auftraggeber ein Budget bereitgestellt. Also sollte man auch genau das umsetzen, und nicht immer wieder erneut darüber nachdenken, was man entwickeln will und in welcher Reihenfolge.

Aber wie bekommt man die nötige Agilität in das Projekt, um angemessen mit der natürlichen Unschärfe der vorformulierten Anforderungen umgehen zu können? Ganz einfach: Man muss die Agilität auf die

Sachverhalte beschränken, für die sie erforderlich ist. Und man muss sie auf die Zeitfenster beschränken, zu denen sie möglich ist, ohne allzu große Mehraufwände zu erzeugen.

### ➔ Planvoll im Großen, Agil im Kleinen

Das Vorgehensmodell von Scopeland Technology ist deshalb ein geplantes, wohlstrukturiertes Vorgehen, mit wohldefinierten und weitgehend standardisierten Projektphasen. Innerhalb des Zeitfensters einer jeden Projektphase ermöglichen wir jedoch größtmögliche Agilität – und zwar immer für genau die in der jeweiligen Phase zu erarbeitenden Detailaspekte.

Durch präzise Definition der Phasen mit den in jeder Projektphase gemeinsam mit den Anwendern zu erarbeitenden Detailanforderungen sichern wir ab, dass die Anwender – ähnlich wie bei einer offenen unstrukturierten SCRUM-Methode – immer auch schon etwas auf dem Bildschirm sehen, wenn sie etwas entscheiden sollen. Und dennoch bleiben wir stets exakt im vorgesehenen Projektrahmen und Zeitplan.

# 2. Das Konzept der phasenagilen Vorgehensweise

Ausgangspunkt ist die Definition der Projektphasen in der Art, dass jede Phase in sich agil und visuell bearbeitet werden kann, sich aber nicht oder kaum auf die jeweils bereits abgeschlossenen Phasen auswirkt. Wie kommt man schrittweise, in kleinen agilen Projektphasen, vom Groben zum Feinen, von der Basis zu den Details?

Hierbei muss man sich die Frage stellen, was beim Bau einer Software das Fundament und die „tragenden Wände“ sind. Niemals kann alles flexibel sein. Ein gänzlich flexibles Bauwerk wäre bestenfalls ein Zelt. Daher muss im Vorfeld festgelegt werden: Was ist das Fundament? Was sind die tragenden Wände einer Datenbankanwendung, die sich auch beim späteren agilen Feintuning mit Sicherheit nicht mehr ändern werden?

Uns kommt in diesem Zusammenhang entgegen, dass es bei Datenbankprojekten (und eigentlich sind alle Verwaltungslösungen und zudem fast alle Individuallösungen Datenbankprojekte) tatsächlich immer ein übergreifendes einheitliches Fundament gibt, nämlich die Datenbank. Die Entscheidung zu einer Implementierung auf Basis einer relationalen Datenbank fällt fast immer bereits im Vorfeld des Projekts und muss auch nicht infrage gestellt werden. Das Datenmodell einer relationalen Datenbank hat von Natur aus einen globalen, software- und modulübergreifenden Charakter. Egal, wie vielfältig das Paket an zu implementierenden Fachmodulen auch immer sein mag, sie alle greifen immer auf dieselbe Datenbank zu. Ein Aufsplitten des Datenmodells in kleine Modul-Datenbanken ist in den meisten Fällen nicht möglich und auch nicht sinnvoll. Das Datenmodell ist global und zumindest immer teilweise für alle Fachmodule da.

Aus diesem Grunde schlägt bei klassischen SCRUM-Projekten die Mehraufwandsfalle zu, wenn ständig Änderungen, Erweiterungen oder gar Umstrukturierungen an der Datenbank erforderlich werden. Entwickelt man das Datenmodell schrittweise, Feature für Feature oder Fachmodul für Fachmodul, dann erhält man zwar nicht unbedingt unpflegbaren Spaghetti-Code, aber auf jeden Fall unpflegbarere Spaghetti-Datenmodelle. Oder man muss immer wieder alles umbauen und wieder von vorn anfangen. Beides macht keinen Sinn. Also: zunächst das Datenmodell erstellen.

Will man möglichst effizient und kostenbewusst entwickeln, dann kommt man auch beim besten Willen nicht drum herum, zu Beginn des Projekts das gesamte Datenmodell komplett zu durchdenken und mehr oder weniger final auszugestalten, zumindest soweit, dass man später keine strukturellen Umbauten am relationalen Modell mehr vornehmen muss. Egal, ob man das nun so mag oder nicht – es geht nicht anders.

# 3. Wie man gemeinsam mit Fachanwendern ein Datenmodell entwickelt – agil und noch ohne Programmdetails

Hierzu muss man stets die Frage „Lieber Anwender, sind das hier wirklich Deine Daten?“ in den Mittelpunkt stellen. Dieser Ansatz scheint auf den ersten Blick verwirrend, gilt es doch als ausgemacht, dass Endanwender sich überhaupt nicht für Datenmodelle interessieren und das Programm nur „von außen“ sehen und verstehen können. Deshalb, so heißt es immer wieder, solle man nicht mit Daten-, sondern mit Prozessmodellen beginnen, wenn man mit den Anwendern reden möchte.

Diesem Vorurteil möchten wir energisch widersprechen. Ganz im Gegenteil: Die Anwender haben oftmals extreme Schwierigkeiten, sich in Bedien- und Prozessabläufe hineinzudenken, solange sie nichts auf dem Bildschirm sehen, und solange sie die Alternativen nicht kennen, zwischen denen sie sich entscheiden müssen. Ein geübter Prozessberater mag zwar spielend die Fachanwender mit seinen kleinen, bunten Tools begeistern und zum fleißigen Modelle malen motivieren, aber ob das, was dabei entsteht, wirklich widerspiegelt, wie ihr Programm am Ende aussehen und funktionieren soll, ist höchst fragwürdig. Für sehr grobe Sachzusammenhänge und auch in manchen anderen Fällen mag das hin und wieder funktionieren, aber im Normalfall definitiv nicht, denn Anwender denken von Natur aus nicht in Prozessen, sondern in Programmfunktionen. Das ist etwas ganz Anderes.

Dahingegen fällt es vielen Fachanwendern verblüffend leicht, ihre Daten zu benennen. Wer ohnehin dafür zuständig ist, bestimmte Daten zu erheben oder zu prüfen oder zu bearbeiten, der weiß auch sehr genau, um welche Daten es sich dabei handelt, und das in oftmals verblüffender Detailtiefe und mit allen Randaspekten dieser Daten. Man muss sie nur danach fragen.

Richtig ist allerdings, dass man Fachanwendern nicht zumuten kann, die komplette Datenbanktheorie mit Relationsarten, Primär- und Fremdschlüsseln oder gar Normalformen zu erlernen. Das ist auch nicht erforderlich, wenn man Tools im Einsatz hat, welche die relationalen Bezüge der Daten zueinander in einer für Endanwender verständlichen Form visualisieren können. Ein solches Tool ist der Ad-hoc-Modus von SCOPELAND®. Wenn der das Brainstorming begleitende SCOPELAND®-Berater die entstehenden Modelle immer direkt mittels SCOPELAND® aufnimmt und in Metadaten beschreibt – möglichst immer gleich mit ein paar automatisch generierten oder manuell erfassten Testdaten dazu –, dann kann man sich das Datenmodell im Ad-hoc-Modul interaktiv durchblättern. Hierbei handelt es sich nicht um Abbildungen abstrakter Modelle, sondern um realitätsnahe Ansichten der eigenen Daten – mitsamt aller Verknüpfungen in Form von Master-Detail-Ansichten, Katalogverweisen, Auswahlfunktionen und vielem mehr.

Wann immer möglich, dann sollen auch bereits in dieser Phase Altdaten übernommen und importiert werden, weil sich so in bestmöglicher Weise die Stimmigkeit des Datenmodells gegenprüfen lässt. Je nach Kritikalität und sonstigen Randbedingungen müssen diese hierfür jedoch noch anonymisiert oder pseudonymisiert und mit synthetischen generierten Testdaten angereichert werden. Wir entwickeln gemeinsam mit den Fachanwendern kein abstraktes, sondern ein „Interaktives" Datenmodell.

Die genaue Vorgehensweise in dieser Phase sieht typischerweise so aus, dass man sich in regelmäßigen Meetings in mehreren Runden durch dieses Thema durcharbeitet. Zeitaufwändige Erfassung bereits bekannter Strukturen und Testdaten macht der SCOPELAND®-Berater am Berliner Arbeitsplatz, und stellt dann in mehreren Runden die Stände ausführlich vor, in immer detaillierter und präziser werdenden Fragerunden. Die Kernfrage hierbei ist immer dieselbe: Ob das denn nun wirklich genau das ist, was Sie an Daten haben und benötigen, und ob Ihr Verweis- und Master-Detail-Zusammenspiel denn nun wirklich genau den tatsächlichen Sachzusammenhängen zwischen diesen Daten entspricht?

Diese Workshops sollten, sofern möglich, tatsächlich mit den Datenherren, also mit den realen Fachanwendern und Fachverantwortlichen durchgeführt werden, und nicht nur mit einem sogenannten vermeintlich allwissenden „Produkt Owner", der die Anwender zuvor befragt hatte, ohne ahnen zu können, welche Fallstricke da noch auftauchen werden. Der direkte und unmittelbare Kontakt zu den wichtigsten Vertretern der Fachbereiche ist ein ganz entscheidendes Erfolgskriterium für diese Herangehensweise – wenn nicht sogar jeglicher Softwareentwicklung.

Keinesfalls sollte sich der SCOPELAND®-Berater in dieser Phase dazu verleiten lassen, schon über Programmoberflächen und -abläufe zu diskutieren. Das geht schon deshalb nicht, weil für dieses andere Thema wahrscheinlich auch andere Teilnehmer am Workshop teilnehmen werden. Vor allem aber wäre das deshalb falsch, weil man sich zu schnell in die falsche Richtung lenken lässt. Solange man den Anwendern noch nicht zeigen kann, wie es möglicherweise aussehen könnte – eingebunden in objektiv gegebene Eckprinzipien und Rahmenfunktionen, modern in der Ausgestaltung, barrierefrei und sicher –, kennen diese auch noch nicht die möglichen Alternativen. Wenn Sie Pech haben, dann werden wir? Ihnen in dieser Phase aufmalen, wie ähnliche, aber inzwischen völlig veraltete Programme mal ausgehen hatten, die diese zufällig kennen? Es wäre definitiv der falsche Zeitpunkt.

Etwas anderes aber sollte man durchaus bereits in dieser Phase ausarbeiten und gleichfalls schrittweise ausreifen lassen: globale Eigenschaften der Daten, globale Regeln und sonstige globale Logik. Das sind Regelwerke, die z.B. besagen, dass ein Datum immer vor einem anderen liegen muss, dass eine Zeichenkette keine Ziffern enthalten darf, oder dass bestimmte Informationen nur als berechnete Spalten zur

Verfügung stehen. Oder in Ausnahmefällen auch schon mal ein Stück Workflow. Warum? Ganz einfach: All das sind Eigenschaften der Daten, und nicht Eigenschaften der darauf aufbauenden Programme. Pflegt man diese Art von Business Logik erst am Ende in die Programmoberfläche ein, dann verkompliziert es diese enorm, man erzeugt unnötige Fehlerquellen und provoziert auch noch ein übles Redundanzproblem.

Alles was zu den Eigenschaften der Daten gehört, und was ebenso global gilt, wie das Datenmodell selbst, sollte in dieser Phase erfasst werden. Also nicht nur das visuell verständliche Interaktive Datenmodell, sondern, bereits einen Schritt weitergehend, ein „um globale Business Logik angereichertes Interaktives Datenmodell“ – und dies so final und endgültig wie möglich.

# 4. Die nächste Phase: Der Programmrahmen

Die nächste große Etappe eines Datenbankprojekts ist die Erarbeitung und Entwicklung des globalen Programmrahmens. Auch hier mag man sich die Frage stellen, wie es denn sein kann, dass wir bereits jetzt mit den Anwendern über den äußeren Rahmen des Programms diskutieren wollen, obwohl wir immer noch nicht über die eigentlichen Programminhalte gesprochen haben.

Auch das ist sehr logisch und wohl begründet: Der äußere Programmrahmen, bestehend aus dem Programmdesign gemäß Corporate Design, aus Menüstruktur und -gestaltung, Benutzerverwaltung nebst Anbindung an das AD/LDAP-System, aus Kommunikation mit der IT-Außenwelt (Schnittstellen, WebServices etc.) – all dies ist fast nie das Resultat der Erarbeitung der Programminhalte, sondern fast immer objektiv gegeben oder mit der Auftragserteilung bereits vorgegeben. Es wäre auch viel zu ineffizient, über den grundlegenden Aufbau einer Datenbankanwendung immer wieder aufs Neue nachzudenken: die Rahmenarchitektur einer Anwendung steht i.d.R. bereits fest, ehe man sich mit den Einzelheiten überhaupt befassen muss.

Da diese „Rahmenanwendung“ weitgehend unabhängig von den konkreten Inhalten ist, kann man sogar noch einen Schritt weitergehen in Richtung bestmöglicher Effizienz: sie ist funktional sogar weitgehend unabhängig vom konkreten Projekt und kann für einen Kunden einmal zentral entwickelt und für fast alle Anwendungen übergreifend eingesetzt werden. Und schließlich bringen wir mit SCOPELAND® sogar schon eine noch nicht fertig gestaltete, aber in allen wichtigen Rahmenfunktionskomponenten bereist vorgefertigte Rahmenanwendung als Musterlösung mit, die wir für Sie entsprechend anpassen und auf das zuvor final entwickelte Datenmodell aufsetzen.

Damit allein ist es aber noch nicht getan, denn der Teufel liegt bekanntlich immer im Detail, und es gibt Dutzende unterschiedliche kundenspezifische Randaspekte und Sonderanforderungen, die hier eingearbeitet und ausoptimiert werden müssen. Zudem ist diese Rahmenanwendung auch genau die Stelle, an der sich das Aussehen und die Akzeptanz der späteren Anwendung herauskristallisiert. Es handelt sich um einen sehr wichtigen und wesentlichen Teil des gesamten Projekts.

Der technische Grundaufbau und die grundlegende Art der Menüführung muss nicht mit den Vertretern der Fachbereiche diskutiert werden, aber die Ausarbeitung des exakten Erscheinungsbildes des Programms sollte durchaus einer gemeinsamen Ausarbeitung und agilen Umsetzung in mehreren schrittweisen Optimierungsrunden unterliegen. Folglich haben wir auch in dieser Phase einen agilen Prozess, allerdings mit einem etwas anderen Teilnehmerkreis.

Auch hierbei ist es von großer Wichtigkeit, dass diese Phase zu Ende gebracht wird, eher die darauffolgende fachlich-inhaltliche Phase startet. Warum? Ganz einfach: damit die Prototypen, über die wir dann gemeinsam diskutieren, bereits mit richtiger und endgültiger Einbettung in den Programmrahmen präsentiert werden können. Nur dann kann man erwarten, dass nicht-IT-kundige Vertreter der Fachbereiche eine ausreichende Vorstellung vom Gesamterscheinungsbild der Anwendung entwickeln, um in den Workshops wirklich verbindliche Entscheidungen treffen zu können.

# 5. Die nächste Phase: Die Ausarbeitung der eigentlichen Fachmodule

Der nächste Schritt umfasst die Ausarbeitung der eigentlichen fachlichen Inhalte und des Erscheinungsbildes und der konkreten Bedienerführung innerhalb der Masken der jeweiligen Fachmodule. Bei großen Anwendungen kann dies für die einzelnen Fachmodule getrennt und wahlweise parallel oder nacheinander oder überlappend erfolgen, zumindest sofern diese nicht allzu sehr miteinander vernetzt sind.

Da wir alles Globale, nämlich sowohl das Datenmodell nebst globaler Logik als auch die gesamte Rahmenanwendung und auch das Design und das Gesamtkonzept der Bedienerführung aus den Fachmodulen herausgelöst und bereits im Vorfeld final ausentwickelt haben, können wir uns nun voll und ganz auf die eigentlichen fachlichen Inhalte konzentrieren, und dabei die einzelnen Fachmodule entspannt getrennt voneinander betrachten und bearbeiten.

Und auch dies erfolgt nun im Rahmen eines hochgradig agilen Prozesses, wobei sich die Agilität wiederum auf das beschränkt, was in dieser Phase zu entwickeln ist. Auch hierfür schlagen wir eine Folge regelmäßiger Workshops mit den jeweils relevanten Vertretern des Auftraggebers vor, in denen die einzelnen Fachmodule Schritt für Schritt immer weiter ausreifen.

Auch dies funktioniert nur, wenn man die richtigen Tools dafür im Einsatz hat, und SCOPELAND® ist eine optimale Plattform, um Programmabläufe und -oberflächen in mehreren Änderungsrunden agil ausreifen zu lassen. Hierbei ist der Ansatz „Konfigurieren statt Programmieren“ das entscheidende Erfolgsrezept.

# 6. Die allererste und die allerletzte Phase

Das gesamte Projekt wird nun noch durch eine initiale Start- bzw. Konzeptions- und eine abschließende Finalisierungsphase eingerahmt und abgerundet.

Immer getreu dem Grundsatz „Erst denken, dann handeln“ sehen wir für jedes Projekt eine vorangehende Konzeptionsphase vor, unabhängig davon, ob diese vom Auftraggeber gefordert wird, oder ob vielleicht sogar ein von Dritten ausgearbeitetes Feinkonzept vorliegt. Die Tiefe der Ausarbeitung des Konzepts wird von unterschiedlichen Randbedingungen beeinflusst, und der Projektleiter des Auftraggebers wird ggf. im Zusammenwirken mit dem Projektleiter des Auftragnehmers per Checkliste ein entsprechendes Tailoring vornehmen. Dies kann im Bedarfsfall auch ein Adaptieren dieses Vorgehensmodells als Ganzes beinhalten, z.B. wenn nicht nur nach logischen Fachmodulen, sondern auch nach technisch getrennten Modulen auf Metadatenbank- oder Generierebene getrennt werden soll.

Zwingend in allen Projekten umfasst die Konzeptionsphase aber zumindest eine Präzisierung der gesamten logischen und technischen Architektur der Anwendung, eine systematische Analyse des gesamten Zusammenwirkens mit der Außenwelt (insb. Schnittstellen, Services etc.), eine Analyse von Risikofaktoren bis hin zur möglichst führzeitigen Veranlassung von eventuell erforderlichen Programmierjobs für Anforderungen, die sich mit dem Funktionsvorrat des Produkts nicht oder nur eingeschränkt „out of the box“ umsetzten lassen. Je nach Ergebnis des Tailorings werden sodann die erforderlichen Unterlagen bzw. Konzepte erarbeitet, wie z.B. das Rollen- und Rechtekonzept, das Datensicherheitskonzept, Installations- und Betriebsanleitungen u.v.m.

In der abschließenden Finalisierungsphase wird alles zusammengeführt, und die Software unterliegt nochmal als Ganzes einer Prüfung und Abrundung. In dieser Schlussphase sind keine schwerwiegenden kundenseitigen Änderungswünsche mehr zugelassen (bzw. nur als Change Request).

# 7. Gestufte Qualitätssicherung in allen Phasen

In allen Phasen sehen wir jeweils darauf zugeschnittene Qualitätssicherungsmaßnahmen vor, um Defizite auf allen Ebenen möglichst zeitig zu erkennen. So erfolgen beispielsweise bereits am Ende der Konzeptionsphase Prüfungen hinsichtlich der Qualität der eventuell zu liefernden Dokumente, Prüfungen hinsichtlich der Qualität der Metadaten im Verlauf und am Ende der Datenmodellierungsphase, sowie im Zuge der Entwicklung der Rahmenanwendung Qualitätsprüfungen hinsichtlich der Barrierefreiheit und Datensicherheit.

**Abbildung 1: Phasenagiles Vorgehensmodell**

Selbstverständlich erfolgen ebenso akribische Tests der in der Entwicklung befindlichen Fachmodule: Dies geschieht nach unterschiedlichen, vordefinierten Qualitätskriterien und jeweils zum frühestmöglichen sinnvollen Zeitpunkt. Jede Phase für sich bedarf einer internen Abnahme durch die Qualitätssicherung von Scopeland Technology. Die Qualitätssicherung ist unabhängig und unterliegt nicht den Weisungen des jeweiligen Projektleiters. Nicht behobene Mängel sollen soweit möglich zeitnah behoben werden, sodass es zu keinem Zeitverzug für die weiteren Projektphasen kommt.

# 8. Wie die agilen Prozesse gemanaged werden

Sämtliche Workshop-Runden in den einzelnen Phasen werden „agil“ durchgeführt, allerdings nicht nach dem formalen SCRUM-Modell, sondern nach den Kriterien und Grundprinzipen des Design Thinking, wie es von etlichen innovativen und marktführenden Unternehmen zunehmend favorisiert wird.

Anders als der Name dieser Methode assoziiert, geht es dabei nicht vorrangig um das Design, sondern darum, in interdisziplinären Teams von Auftraggeber und Auftragnehmer optimale Lösungen gemeinsam zu erarbeiten. Und es geht um Usability & User Experience (UUX), hier allerdings weniger in Bezug auf Spaß am Benutzen einer Software, sondern mehr darauf bezogen, wie eine Software aufgebaut sein muss, damit man damit effizient seine Arbeit erledigen kann.

Typischerweise geht man dabei in kurzen Zyklen voran (z.B. je Fachmodul ein Workshop wöchentlich oder zweiwöchentlich), und in jeweils kurzen Workshops werden die jeweiligen Arbeitsstände mit allen jeweils relevanten Projektbeteiligten besprochen. Wichtig ist hierbei, dass sich die Zusammensetzung der Workshop-Teams über die gesamte Bandbreite der Rollen im Projekt erstreckt – vom typischen, echten Anwender bis hin zum Softwareentwickler, der das betreffende Modul umsetzen soll. Je nach Thema und Projektphase werden aber unterschiedliche Personen in die Diskussionen mit einbezogen, z.B. auch mal Webdesigner, Datenbankadministratoren, Datenschutzbeauftragte oder sonstige Fachspezialisten. Auf jeden Fall sollten sowohl die Anwendervertreter des Auftraggebers als auch die eigentlichen Entwickler des Auftragnehmers involviert sein.

Das jeweilige Design-Thinking -Team ist dabei auch das eigentliche Entscheidergremium, und eine etwa gleichstarke Präsenz von Vertretern der Auftraggeber- und Auftragnehmerseite garantiert ausgewogene Entscheidungen und Kompromisslösungen. Ein Übergewicht an Nice-to-Have-Wünschen von Anwendern wird folglich sofort ausnivelliert – vor dem Gegengewicht der Aufwände und Konsequenzen all dieser vielen Ideen. Und andersherum werden etwaige zu sparsame Interpretationen von Kundenanforderungen sofort auf die Ebene ihrer tatsächlichen Erfordernisse zurückgeholt. Dieses sofortige Aushandeln aller Detailaspekte erspart beiden Seiten ganze Berge unnötig beschriebenen Papiers, und damit in erheblichem Umfang reduzierte Projektmanagementaufwände. Außerdem bietet dies eine optimale Plattform, um nachträgliche Änderungen im Rahmen eines Festpreisbudgets einvernehmlich auszuhandeln.

Das Design-Thinking-Team wird von den Projektleitern beider Seiten gesteuert – bei großen Projekten ggf. jeweils noch unterteilt in fachliche und technische Projektleiter –, welche gemeinsam die Meetings leiten und Entscheidungsfindungen durchsetzen. Dieses ebenso bilaterale Projektleiter-Team sichert also auch gleichzeitig das Änderungsmanagement ab, unterstützt von einem ebenso pari-pari besetzten Lenkungsausschuss als Eskalationsinstanz.

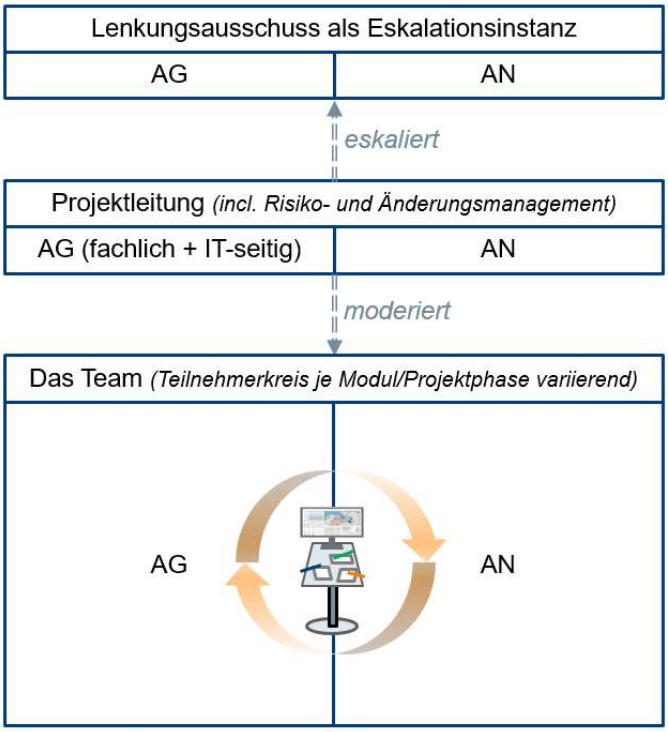



**Abbildung 2: Vereinfachte Projektstruktur**

Die vielleicht wichtigste Aufgabe der Projektleiter besteht darin, die Teammitglieder zu verbindlichen Entscheidungen in allen offenen Punkten zu drängen, und zwar soweit möglich zu sofortigen Entscheidungen bzw. bis spätestens im Folgemeeting nach dem Aufkommen der jeweiligen Frage. Primär geht es darum, der Bequemlichkeit des Vertagens entgegenzuwirken. Die Projektleiter beider Seiten sollten sich dabei stets als ein Team sehen, welches gemeinsame Entscheidungen des gemischten Teams herbeiführt. Sie sollen nicht lediglich die Meinungen „ihrer“ Leute konsolidieren, um dann auf der Ebene der Projektleiter dafür zu kämpfen, sondern sich auf eine moderierende Rolle beschränken und gemeinsam die Einigung zwischen Fach- und Realisiererebene erwirken.

Auf diese Weise werden Reibungsverluste und vor allem das altbekannte „Stille-Post-Problem“ vermieden. Zudem wird die große Problematik umgangen, dass es diesen einen allwissenden Product Owner – den SCRUM zwingend erfordert – in der Realität nur sehr selten gibt. So vermeiden wir den Product-Owner-Flaschenhals, durch den sonst der gesamte Wissensfluss unverfälscht und vollständig durchgeleitet werden muss.

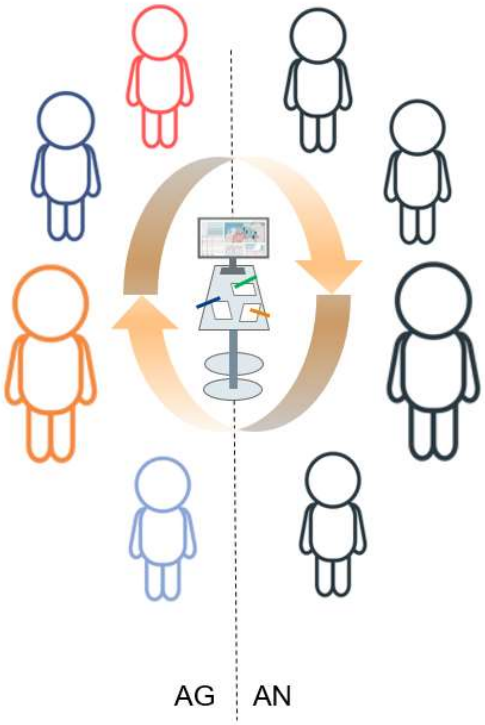



**Abbildung 3: Design Thinking in agilen Projektphasen**

Damit solche Design-Thinking-Runden effizient ablaufen, haben sich dafür einige typische Spielregeln herausgebildet, wie beispielsweise die, dass alle Teilnehmer des Brainstormings im Meeting gleichberechtigt sind, dass Laptops und Handys ausgeschaltet bleiben, und dass Ideen, Konzepte, Anregungen vorrangig aufgemalt und nicht aufgeschrieben werden. Dieses „Zurück zum Papier"-Prinzip hat bei der Anforderungsaufnahme und der Erarbeitung von Lösungsansätzen einige bedeutende Vorteile, sodass hier dem Papier klar der Vorzug zu geben ist – unabhängig davon, mit welchen Tools und Formalien das Projekt als Ganzes gemanaged wird.

Design-Thinking-Workshops werden übrigens bevorzugt an Stehtischen abgehalten. Warum ist das so? Man ist im Stehen lockerer, und die Gedanken finden leichter neue Wege. Die Hemmschwelle, mal schnell ans Whiteboard zu gehen ist geringer, und wenn man in der Mitte des Stehtisches etwas auf Papier aufmalt, was man sich gerade überlegt hat, dann können das alle anderen besser verfolgen als an einem Konferenztisch. Und falls mal eine kleine Videokonferenz eingeschoben werden muss, dann geht auch das unkomplizierter und einfacher, weil man näher dran ist. Und „last but not least": Der nette Nebeneffekt besteht darin, dass die Meetings auch schneller ein Ende finden. Design Thinking steht also nicht nur für eine neue Vorgehensweise, es bringt auch eine ganz andere Kultur des Arbeitens und Zusammenarbeitens zwischen den Projektpartnern mit sich.

# 9. Die Standard-Projektphasen eines typischen SCOPELAND®-Projektes nach dem Modell der phasenagilen Entwicklung

Typische SCOPELAND®-Projekte lassen sich nach dem Modell der phasenagilen Entwicklung in fünf Phasen mit den folgenden Charakteristika unterteilen:

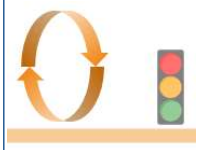
**Phase 1**

**Projektstart – Erst Denken, dann handeln**

- Grobkonzept mit Tailoring je nach Projektanforderungen
- Mindestanforderungen: Gesamtarchitektur, Schnittstellenkonzept, Risikoanalyse, Initiierung eventueller Programmier- und sonstiger langlaufender Tasks
- Workshop-Beteiligte: Projektleiter und technische Experten von AG und AN

**Phasenbezogene Qualitätssicherung**

**Phase 2**



**Interaktives Datenmodell**

- Um globale Business Logik angereichertes interaktives Datenmodell, befüllt mit Test- oder anonymisierten Altdaten
- Workshop-Beteiligte: Projektleiter und Datenbankexperten von AG und AN, Fachanwender und Entwickler

**Phasenbezogene Qualitätssicherung**

**Phase 3**

**Der Programmrahmen**

- Aufsetzen des Programmrahmens, inklusive aller modulübergreifenden Aspekte
- Workshop-Beteiligte: Projektleiter, Sicherheits- und Barrierefreiheitsexperten von AG und AN, Web-Designer, Entwickler

**Phasenbezogene Qualitätssicherung**

**Phase 4**

**Die Fachmodule – Der Content**

- Entwicklung der Fachmodule, als Content der Rahmenanwendung (ggf. getrennt für die einzelnen Fachmodule, wahlweise parallel, nacheinander oder überlappend)
- Workshop-Beteiligte: Projektleiter von AG und AN, Fachanwender und Entwickler

**Phasenbezogene Qualitätssicherung**

**Phase 5**

**Finalisierung**

- Finalisierung und Projektabschluss
- Workshop-Beteiligte: Projektleiter von AG und AN

**Phasenbezogene Qualitätssicherung**

**Abbildung 4: Die Standard-Projektphasen der phasenagilen Entwicklung**

# 10. Eine andere Vorgehensweise in der Softwarepflegephase

Wie geht man weiter vor, nachdem das Projekt komplett umgesetzt wurde? Jede Software muss weiterentwickelt werden, und für kundenspezifische Anwendungssoftware gilt dies im besonderen Maße. Alles ändert sich, nichts bleibt wie es ist.

Anders als bei der Neuentwicklung einer Individuallösung ist in der Pflegephase der regelmäßige Output weiterentwickelter Versionen durchaus wünschenswert, und das zumeist limitierte Budget legt nahe, dass man Änderungswünsche sammelt (Backlog), priorisiert und turnusmäßig in die Entwicklung einstellt – passend zu den jeweils verfügbaren finanziellen und personellen Ressourcen, mit regelmäßigen qualitätsgesicherten Versionen als Output dieses Prozesses. Dieses Szenario deckt sich mit der Theorie von SCRUM-Projekten, nebst einer Task-Organisation innerhalb der Sprints nach KANBAN, so dass es als sinnvoll erscheint, eine entsprechende Projektmanagement-Organisation nach SCRUM aufzubauen.

Dies gilt allerdings mit einer Einschränkung: Auch in der Pflegephase haben wir den fehlenden allwissenden Product Owner immer noch nicht, und wir werden ihn nie haben; jedenfalls nicht bei Auftragsentwicklungen. Das Problem, dass man nicht so recht sagen kann, wo er idealerweise anzusiedeln wäre, beim Auftraggeber oder beim Auftragnehmer, ist ebenfalls noch ungelöst.

Hierzu schlagen wir vor, im Groben den Prinzipien von SCRUM zu folgen, die Anforderungserhebung im Detail, und das gegebenenfalls erforderliche Abwägen von Nutzen versus Aufwand und Konsequenzen der einzelnen Änderungswünsche auch weiterhin den Design-Thinking-Teams aus Anwendern und Realisierern zu überlassen. Die Projektleiter beider Seiten sollten hierbei in gleicher Weise wie bei der Erstentwicklung der Software als PM-Team agieren, nur mit dem Unterschied, dass sie nun einen Backlog als kontinuierlichen Stapel an Wünschen gemeinsam verwalten, gewichten und eintakten. Abweichend von der SCRUM-Theorie müssen die Sprints auch nicht immer gleichlang sein, und es muss auch nicht unbedingt permanent an dem Projekt gearbeitet werden. Sprinttermine sind Plantermine für Folgereleases der Softwarelösung und natürlich bedarfsgerecht abzustimmen.

# 11. Qualitätssicherung und Abnahmen

Die Qualitätssicherung erfolgt, wie oben erläutert, phasengerecht und stets zeitnah. Es ist selbstredend, was in den jeweiligen Phasen zu testen ist, und wie mit Testergebnissen – je nach Gewicht eines Mangels – umzugehen ist. Eine ausführlichere Erläuterung der branchenüblichen Einstufung von Fehlern von „geringfügig“ über mittlere Einstufungen bis hin zu „betriebsbeeinträchtigenden“ und „betriebsverhindernden“ Mängeln muss hier nicht erneut ausgeführt werden, und inwieweit diese jeweils abnahmeverhindernd sind, wird i.d.R. durch die vertragliche Grundlage eines Projekts vorgegeben.

Zu betrachten ist aber die Frage, ob die vollständige Qualitätssicherung nach einer Projektphase abnahmerelevant ist oder nicht. Die Antwort: Ja, aber nur, wenn sich der Auftraggeber den Luxus leisten kann, die daraus resultierenden Mehraufwände und Zeitverzögerungen in Kauf zu nehmen, nur um etwas mehr Planungssicherheit und etwas mehr Sicherheit bezüglich der zu leistenden Teilzahlungen zu gewinnen. Der Zeitverzug, der sich allein daraus ergibt, dass bestimmte Arten von Mängeln den Projektfortschritt durch Fehlerbehebungsrunden und wiederholte Abnahmeprozeduren ausbremsen, sowie die entsprechenden PM-Mehraufwände lassen den Nutzen dieser Sicherungsmaßnahme zweifelhaft erscheinen.

Für die jeweils phasenbezogenen Qualitätssicherungen sollte lediglich die Durchführung der Qualitätstests meilensteinrelevant sein, nicht aber die vollständige Behebung der erkannten Mängel. Da diese größtenteils nicht sofort auf die Nachfolgeaktivitäten durchschlagen, und auch noch in den darauffolgenden Phasen behoben werden können, kann der Entwicklungsprozess weitgehend unbeeinträchtigt weiterlaufen. Unser Anliegen, für alles die Qualitätssicherung so zeitig wie möglich anzusetzen, dient der Optimierung des weiteren Vorgehens, und ist nicht dafür gedacht, das Erreichen der geplanten Meilensteine künstlich auszubremsen.

Was aber abnahmeähnlich hierbei zu sehen ist, ist die Fixierung der in der Phase erarbeiteten Inhalte. Es wird stets angestrebt, dass alle Anforderungen einer Phase bis zu deren Ende komplett aufgenommen wurden. Nachträgliche Änderungen, wie zum Beispiel schwerwiegende Änderungen am Datenmodell während der Erarbeitung der Benutzeroberflächen, sind nicht tolerierbar und eindeutig als Change Request zu betrachten. Es geht also in den Zwischen-QS-Phasen nicht darum, dass der Auftragnehmer nachweist, bereits alles richtig gemacht zu haben, sondern dass der Auftraggeber zum Phasenende anerkennt, dass das Erarbeitete dem vereinbarten, vertraglichen Umfang und Inhalt entspricht.

Und noch etwas halten wir für relevant in diesem Zusammenhang: Bekanntlich ist es bei IT-Projekten oftmals strittig, ob ein bestimmtes Programmverhalten fehlerhaft ist oder nicht, und als wie schwerwiegend dieses einzustufen ist. Um endlose Diskussionen an dieser Stelle zu vermeiden, ist es naheliegend,

auch das Bewerten der Fehlersituation durch entsprechende Design-Thinking-Teams einvernehmlich durchführen zu lassen: effizient und im (notfalls auszuhandelnden) Konsens.

# 12. Projekt- und Terminplanung

Wir empfehlen unbedingt, bereits zu Projektbeginn bzw. spätestens nach der Startphase nicht nur alle Projektphasen und Meilensteine zu planen, sondern mit AG und AN gemeinsam bereits sehr zeitig die Zusammensetzung der Design-Thinking-Teams festzulegen und sämtliche Meetings vorausschauend durchzuplanen, einschließlich Vertretungsregelungen und Reserveterminen. Dies ist deshalb so wichtig, weil nur so eine kontinuierliche Verfügbarkeit der jeweiligen Teammitglieder gesichert werden kann, und weil es eine Voraussetzung dafür ist, die nötige Verbindlichkeit für die gemeinsam erarbeiteten Entscheidungen und Ergebnisse zu erreichen.

**Scopeland Technology GmbH**
Düsterhauptstraße 39 - 40
D - 13469 Berlin
Tel. +49 30 209 670 - 0
Fax +49 30 209 670 - 111

**info@scopeland.de**
**www.scopeland.de**

Geschäftsführer: Karsten Noack
Amtsgericht Charlottenburg HRB 176787 B
USt.-Nr.: DE 183446349