Softwareentwicklung mit FlowProtocol

Wolfgang Maier

8. Oktober 2022

Inhaltsverzeichnis

Vorwort

FlowProtocol ist eine Webanwendung, mit der sich Protokolle, Analysen, Anleitungen und sonstige Dokumente anhand von beliebig verschachtelten Fragestellungen erstellen lassen. Der Name leitet sich ab von Flow wie Flussdiagramm und Protokoll und verdeutlicht die ursprüngliche Intention, dass Protokolle und Checklisten nicht mehr nur fest und unveränderlich gegeben sein sollen, sondern dynamisch durch gezielte Fragestellungen auf eine bestimmte Situation hin zugeschnitten werden, und somit deutlich mehr an Präzision und Nutzwert gewinnen. *FlowProtocol* selbst ist dabei nur die Anwendung, auf der Vorlagen ausgeführt werden können, die die Fragestellungen, Ausgaben und Protokollpunkte enthalten. Ein ganz wesentlicher Aspekt bei der Entwicklung des Systems bestand darin, dass die Erstellung von Vorlagen sehr einfach, und ohne besondere Werkzeuge oder entwicklungstechnische Kenntnisse möglich sein sollte, sodass damit innerhalb einer Einrichtung eine Plattform geschaffen werden kann, die von den Mitarbeitern nicht nur als Anwender genutzt, sondern auch gepflegt und angepasst und ständig durch neue Vorlagen erweitert wird. Dies wurde dadurch erreicht, dass Vorlagen mit einem beliebigen Texteditor in einer sehr einfachen und intuitiven Syntax erstellt werden können, und die reine Ablage in einem Verzeichnis ausreicht, um diese unmittelbar in die Anwendung einzubinden. Zusätzlich wurde ein sehr umfassendes Anwenderhandbuch mit vielen Demo-Beispielen erstellt.

Das Projekt wurde von mir im Winter 2021/22 begonnen und immer wieder um kleinere Funktionen erweitert, und wird wohl auch in Zukunft noch die eine oder andere Erweiterung erfahren. Es ist unter https://github.com/maier-san/FlowProtocol frei verfügbar. Die anfängliche Intention der Checklisten-Erstellung hat sich inzwischen in verschiedene Richtungen ausgedehnt, sodass die Anwendungsfälle nun um einiges breiter gefächert sind. Ausgangspunkt der verschiedenen Erweiterungen war dabei immer ein Bedarf, den ich bei meiner Arbeit als Teamleiter eines Softwareentwicklungsteams festgestellt hatte, und dementsprechend sind speziell für dieses Arbeitsumfeld nun so viele schöne und praktische Lösungen auf Basis von *FlowProtocol* entstanden, dass es sich lohnt, diese hier ausführlich zu beschreiben. Noch ein paar Worte zu meiner Person und dem Entstehungsumfeld von *FlowProtocol* . Nach meinem Studium der Mathematik und Informatik habe ich 2002 als Softwareentwickler bei einer damals noch kleinen Firma angefangen und konnte dort seit nunmehr über 20 Jahren meine Leidenschaft für das Programmieren, analysieren und ganz besonders den Aufbau von Prozess- und Unterstützungswerkzeuge tatkräftig einbringen und mich dabei auch permanent weiterentwickeln. Stetiges Wachstum und permanente Professionalisierung erfordern es, auch die Arbeitsmethoden immer wieder so auszurichten, dass das bestmögliche Zusammenspiel aller Beteiligten zugunsten der Anwender und Kunden möglich ist, und das sieht bei mehreren Scrum-Teams mit Product Owner, UX-Unterstützung und eigene Testabteilung anders aus, als bei drei individuellen Entwicklern, die sich Fragen und Antworten noch über den Schreibtisch hinwegzurufen können. Hinzu kommen definierte Abläufe und Prozesse, vielfach eingefordert durch die Normen ISO 9001 und ISO 27001 und nicht zuletzt jede Menge an Wissen, Erfahrung und Best-Practice, die es zu ordnen und gezielt zum Einsatz bringen gilt. Inzwischen angekommen in einer geteilten Rolle aus Teamleiter, Softwarearchitekt und technischer Product Owner habe ich den Vorteil, dass ich unmittelbar von den Teamkompetenzen profitieren kann, für deren Aufbau ich ebenfalls verantwortlich bin, und damit ein sehr direktes Feedback bekomme. Als wichtigste Erfahrungen kann ich hier schon vorwegnehmen, dass die Arbeitsatmosphäre wohl der wichtigste Faktor für ein gut

funktionierendes und effektives Team ist, aber gleich danach das Umfeld kommt, das eine möglichst reibungsfreie und direkte Umsetzung der eingebrachten Arbeitsenergie ermöglicht, und diese weder verschwendet oder an unnötigen Barrieren aufschreibt. Letzteres kann bei der oben genannten Menge an Vorgaben und Rahmenbedingungen auf jeden Fall zu einer Herausforderung werden.

1 Die Situation im Entwicklungsumfeld

In diesem Abschnitt möchte ich genauer beschreiben, warum *FlowProtocol* gerade im Umfeld der Softwareentwicklung gut eingesetzt werden kann. Die Tatsache, dass dort vermutlich die meisten Arbeitsplätze EDV-Arbeitsplätze sind, die von einer fähigen IT-Abteilung betreut werden, bietet schon mal gute Voraussetzungen. Damit das System jedoch allgemein als Plattform innerhalb der Firma genutzt und auch von allen Mitarbeitern aufgebaut und weiterentwickelt wird, braucht es Menschen, die Abläufe und Informationen zu einem gewissen Grad abstrahieren und in eine maschinell verwendbare Form bringen können, was ja sozusagen die Kernkompetenz der Softwareentwicklung ist. Umgekehrt hilft einem diese Form der Abstraktion aber auch wieder dabei, einen Gegenstand besser zu durchdringen, und dementsprechend habe ich mir folgendes Motto als Zitat für ein Profilbild ausgesucht: „Erst wenn man einen Sachverhalt von Grund auf verstanden hat, ist man in der Lage, ihn einer Maschine beizubringen.“

1.1 Arbeiten mit Informationen

Bei Softwareentwicklung denkt man zwangsläufig zunächst einmal an Softwareentwickler, die Programmcode schreiben, der dann irgendwie vertrieben und ausgeliefert wird. So essenziell dieser Teil der Arbeit ist, ist er trotzdem nur ein Teil in der langen Kette der Produktentwicklung, an der viele Leute aus unterschiedlichen Fachrichtungen mit jeweils spezialisierter Arbeitsweise und eigenem Expertenwissen beteiligt sind, die Hand in Hand zusammenarbeiten müssen. Angefangen von der Bedarfserfassung und der Priorisierung der Projekte, weiter über die Ausarbeitung neuer Lösungen durch die Product Owner mit Unterstützung von Kollegen aus den Bereichen Design und Usability, die Zusammenarbeit mit den Entwicklungsteams, die Kontrollinstanzen des Qualitätsmanagements, Dokumentation, Marketing und Vertrieb, Dienstleistungen und Projektmanagement, bis hin zu Support und Hotline für die Betreuung bei Problemen, arbeiten alle primär mit Informationen, die erfasst, verstanden, bewertet, ergänzt und in eine neue Form gebracht werden müssen. In vielen Bereichen wird zudem kreativ gearbeitet, d.h. Dinge werden auf dem sprichwörtlichen weißen Blatt Papier neu erstellt. Grundsätzlich gibt es in jedem Bereich zahlreiche Aufgaben, die sich gut mit *FlowProtocol* umsetzen lassen, so dass man dieses Werkzeug wie das in jenem Firmen-Netzwerk zu findende interne Wiki gleich von Anfang an für alle Mitarbeiter verfügbar machen kann.

1.2 Kommunikation und Teamzusammenarbeit

Die oben beschriebene Prozesskette zeigt schon sehr deutlich, dass die Zusammenarbeit zwischen Personen und insbesondere Teams ein sehr wichtiger Aspekt ist, der einen enormen Einfluss auf die Effizienz und Qualität der Arbeit hat. Aus eigener Erfahrung weiß

ich, dass die Zusammenarbeit innerhalb der Teams meist ohne viel Mühe sehr gut funktioniert, und durch die hierarchische Personalstruktur dort auch ausreichend Unterstützung erfährt. Bei der Zusammenarbeit zwischen den verschiedenen Bereichen gestaltet sich das oft schwieriger, was schon damit anfängt, dass die Kommunikationswege abstrakter und formaler sind. Wenn jeder Mitarbeiter aus Bereich A seine Anfrage willkürlich an einen selbst gewählten Mitarbeiter aus Bereich B adressieren würde, wäre das Chaos vorprogrammiert. Für eine geregelte und verlustfreie Abarbeitung von Anfragen und Aufgaben werden dementsprechend Systemen und Posteingänge eingerichtet, die unabhängig von Kenntnissen über die Personalsituation des Zielbereiches genutzt werden können, da Zuweisungen und Vertretungssituationen bereichsintern geregelt werden. Standardmäßig verwendet man hierfür ein Ticketsystem, das mit statusbehafteten Vorgängen arbeitet, und den beteiligten Personen zu jedem Zeitpunkt Zugriff auf den aktuellen Informationsstand bietet, was man mit dem Weiterleiten von persönlichen E-Mails nicht mal ansatzweise abdecken kann.

Ganz ohne Wissen des Zielbereiches kommt man jedoch trotzdem nicht aus. Die Übermittlung von Informationen zwischen Bereichen wirkt immer die Gefahr von Verfälschungen und Verlusten. Schon das Weglassen einer kleinen Information kann die Verarbeitung eines Vorgangs mein großes Maß erschweren. Wird bei einem Bug-Report, der vom Support an die Entwicklungsteilung übergeben wird beispielsweise die Versionsnummer weggelassen, kann das dazu führen, dass unnötigerweise mehrere Versionen überprüft werden müssen, um den Fehler zu finden, was mit viel zeitlichem Aufwand verbunden ist. Die Abfrage und Bereitstellung der Versionsnummer wäre dagegen mit wenigen Sekunden machbar, und damit auch dann sinnvoll, wenn noch nicht klar ist, ob man sie wirklich benötigt. Entsprechend ist es auch nicht die Zeiteinsparung die zu einem solchen Fehler führt, sondern meist einfach das fehlende Wissen um die Bedeutung, die bestimmte Informationen innerhalb der Arbeitsvorgänge in den anderen Bereichen haben. Manche Firmen verwenden viel Geld und Mühe, um mit Hilfe von externen Beratern den Punkt herauszufinden, an dem die Mitarbeiter den größten Engpass sehen, und landen so ebenfalls bei dem Punkt Kommunikation, oder wenn der Berater sein Geld wert ist, bei teamübergreifende Kommunikation und laterale Führung.

Im oben genannten Beispiel ist es auch naheliegend, den Aufbau von Wissen über Abläufen und Informationen in anderen Bereichen einzufordern, da dies verständnisfördernd ist und es immer besser ist, wenn man weiß, aus welchem Grund man Dinge macht. In einem so komplexen Gebiet wie der Softentwicklung kommt man hierbei allerdings schnell an die Grenzen, denn zumeist gibt es viele potentielle Informationen, von denen je nach Situation nur wenig relevant sein können. Mit welchen Tests kann ein bestimmter Fehler schon auf dem Ausgangssystem gut eingegrenzt werden? Wann benötige ich sogar die Bereitstellung einer Datenbanksicherung? Mit der Zeit werden Mitarbeiter dafür ein Gespür entwickeln können, was aber einen aufwendigen Lernprozess und entsprechend viele Fehler voraussetzt. Wir werden später noch deutlich komplexere Beispiele zu diesem Thema sehen.

Die Alternative dazu sieht so aus, dass man die benötigten Informationen einfach generell abfragt, zum Beispiel in Form von Pflichtfeldern oder Checklisten oder der Zurückweisung unvollständig ausgefüllter Vorgänge. Wenn Informationen nur noch über den Zaun geworfen werden, verliert man zwangsläufig einen wichtigen Teil der Zusammenarbeitskultur und läuft auch Gefahr, dass die Sinnhaftigkeit der Arbeit infrage gestellt wird, was nicht zu unterschätzen ist. Bei immer gleich ablaufenden Tätigkeiten besteht zumindest die Chance, dass auch die dafür benötigten Informationen immer die gleichen sind, und

man am Ende mit einen Formular am besten bedient ist. Sobald die Aufgabenstellung jedoch variieren kann und man auf Fallunterscheidungen reagieren sollte, bekommt man hier das Problem, dass man entweder nicht benötigte Informationen abfragt, oder insgesamt auf die Abfrage der für die Spezialfälle benötigten Informationen aus em gleichen Grund verzichtet.

Unabhängig davon ist das roboterartige abfragen aller potentiell benötigten Informationen im Allgemeinen auch nicht unbedingt wirtschaftlicher, als bei Bedarf über eine Rückfrage in eine zweite Iteration zu gehen, insbesondere, wenn man den zeitlichen Aufwand und die damit einhergehende Zufriedenheit des Anwenders entsprechend hoch bewertet.

1.3 Kategorisierungshilfen und Unterstützungssysteme

Wie kann man also allgemein kommunizieren, welche Informationen man in welcher Situation am besten benötigt oder welche nächsten Schritte sich abhängig von bestimmten Voraussetzungen unmittelbar anschließend?

FlowProtocol bietet die Möglichkeit die auf internen Abläufen eines Bereiches begründete Entscheidungsketten in einer formalen Form nach außen bereitzustellen, sodass sie auch von Mitarbeitern ohne dieses spezielle Fachwissen verwendet werden können. Die vorliegende Situation wird durch eine Reihe von Multiple-Choice-Fragen erfasst, wobei es die Möglichkeit zur Verschachtelung erlaubt, je nach Teilantwort beliebig in die Tiefe zu gehen, und bekommt am Ende der Ausführung dafür genau die Informationen, die auf diese Situation passen. Das entspricht auch einer der wesentlichen Grundideen der Anwendung, nämlich Wissen für andere nutzbar zu machen, ohne dass es dafür vermittelt und erworben werden muss. Der wesentliche Vorteil ist hierbei, dass der Verwender die Vorlage als Hilfestellung und nicht als fehlerbedingte Iteration wahrnimmt, und sich sicher sein kann, dass die abgefragten Informationen auch wirklich benötigt werden.

Erklärt man die Abfrage der Informationen zusätzlich durch die Ergänzung von Erläuterungstexten, die man direkt in der Vorlage anzeigen lassen kann, fördert man zusätzlich den Transfer des zugrundeliegenden Wissens und befähigt die Mitarbeiter damit zusehmend, die Situationen auf Basis von Verständnis zu bewältigen.

Ein klassisches Beispiel für diese Anwendung bilden sogenannte Unterstützungssystem für die Problemanalyse, die sich sehr gut mit *FlowProtocol* aufbauen lassen. Man spricht hier auch von einem einfachen Expertensystem und meint eine interaktive Anwendung, die eine vorliegende Situation sukzessive mit Hilfe von Fragen erfasst und dazu passende Handungsoptionen anbietet. Bei der Problemanalyse in Verbindung mit einem Softwareprodukt besteht die Zielsetzung darin, die Einschränkung nach Möglichkeit durch Anwendung direkter Maßnahmen zu beheben, und falls dies nicht möglich ist, alles Notwendige an Informationen zu erfassen, um das Problem im Rahmen eines Vorgangs zielgerichtet und möglichst ohne weitere Iteration beheben zu können.

Dies gelingt, indem man zunächst die Art des Fehlers genau kategorisiert und dazu die Fehlerstelle so präzise wie möglich beschreibt. Ersteres kann z.B. so aussehen:

> *Um welche Art von Fehler handelt es sich?*
>
> ◯ *Ausnahmefehler mit Ausnahmecode. Es wird eine Fehlermeldung mit einem achstelligen Fehlercode angezeigt.*
> ◯ *Inhaltlicher Fehler. Ein angezeigter oder ausgegebener Wert entspricht nicht den Erwartungen.*

◯ *Fehler in der Zugriffssteuerung. Die Sichtbarkeit oder der Zugriff auf bestimmte Daten entspricht nicht den Erwartungen.*
◯ *Technischer Fehler. Die Anwendung startet nicht, stürzt ab oder zeigt ungewollte Auffälligkeiten an der Programmoberfläche.*
◯ *Sonstiger Fehler.*

Für die Erfassung der Fehlerstelle kann etwa die Menüstruktur der Anwendung als verschachtelte Auswahl nachgebildet werden. Der Vorteil dabei liegt weniger in der Zeitersparnis gegenüber einer Tastatureingabe, sondern vielmehr in der Einheitlichkeit der resultierenden Ausgabe. Bei der Softwareentwicklung strebt man die Verwendung einer allgegenwärtigen Sprache an und meint damit, dass der Sprachgebrauch rund um die Kerndomäne der Anwendung über alle Bereiche der Firma und Entwicklung hinweg durchgängig ist, und einer einheitlichen Terminologie folgt, was sich jedoch nur mit sehr viel Mühe bis zur letzten Konsequenz durchsetzen lässt. Gerade im Zusammenhang mit Supportfällen wird allzugern der Sprachgebebrauch des Kunden übernommen und selbst an der Programmoberfläche klar benannte Elemente und Menüpunkte werden teilweise sehr phantasievoll umbenannt. Für einen erfahrenen Mitarbeiter mag die Zuordnung zwar auch dann noch gut möglich sein, doch eine Volltextsuche im System kommt hier sehr schnell an ihre Grenzen. Die Wiederauffindbarkeit von Informationen in Vorgängen ist eine wichtige Qualität, die viele Stunden Zeit einsparen kann, und die leider oft vernachlässigt wird.

Das Beispiel oben zeigt schon recht gut, wie tief und systematisch das Wissen über potentielle Probleme in der Software sein muss, um eine zielführende und vollständige Kategorisierung hinzubekommen, und es lässt auch erahnen, mit welchen Fragen man für jeden der Punkte in die Tiefe gehen könnte.

Im nächsten Schritt kann man versuchen, Eigenschaften des Systems abzufragen und Abhängigkeiten zu finden, die das Auftreten des Problems beeinflussen oder auslösen, wie Programmbenutzer oder Arbeitsplatz. In Abhängigkeit davon kann man vielleicht auch schon bestimmte Sofortmaßnahmen ausmachen, die potentiell hilfreich sein könnten. Diese lassen sich ebefalls als Frage formulieren und führen so auch bei schon versuchter erfolgloser Anwendung zumindest zum Festhalten dieser Information, was ansonsten nicht immer geschieht.

Wurde schon versucht, die Benutzereinstellungen zurückzusetzen?

◯ *Ja, jedoch ohne Erfolg*
◯ *Nein, diese Maßnahme steht noch aus.*
◯ *Nein, diee Maßnahme ist aktuell nicht möglich.*

Je nach Antwort bekommt man damit am Ende der Ausführung entweder die Protokollierung, dass man den entsprechenden Schritt ohne Erfolg durchgeführt hat oder die Auflistung dieser Maßnahme in einer Liste unter der Überschrift „Mögliche nächste Schritte“.

Der Aufbau eines tief verschachtelten Expertensystems wird meist nicht in einem Anlauf gelingen und gerade am Anfang werden sich zusätzliche Iterationen dadurch nicht vermeiden lassen, da man immer wieder Fälle haben wird, die durch die Vorlage nicht abgedeckt sind, und auch noch später werden immer wieder Sonderfälle auftreten, die nicht vollständig durch das allgemeine Schema der Vorlage abgebildet werden, was bei der Anwendung der eigenen Erfahrung allerdings auch nicht anders ist. Wichtig ist es, jetzt die Fälle zu erkennen, die man noch in die Vorlage einarbeiten sollte, und das unmittelbar auch zu tun. Auf diese Weise kann schon der nächste Mitarbeiter in der nächsten

Minute von der Verbesserung profitieren, ohne dass er überhaupt am zugrundeliegenden Austausch beteiligt war.

1.4 Der Umgang mit Fachwissen

Erfahrung und Fachwissen sind zwei weitere wichtige Säulen, die den Prozess der Softwareentwicklung im Wesentlichen bestimmen. Natürlich sind auch viele operative Dinge entscheidend, aber um innovative Lösungen zu finden und diese mit modernen Mitteln umsetzen zu können, braucht es Fachwissen auf Höhe der Zeit und möglichst viel Erfahrung im Umgang damit, und in einer Branche, in der sich die eingesetzten Technologien und Methoden so schnell weiterentwickeln, wie bei der Entwicklung von Anwendungen, ist das eine besondere Herausforderung. Entsprechend ist es wichtig, dem Wissenserwerb und der Wissensvermittlung ausreichend Raum und auch Freiraum zu geben. Für Ersteres gibt es Fortbildungen und sehr gute Online-Portale mit tausenden von gut gemachten Tutorials. Die Vermittlung von Wissen und Erfahrung muss dagegen meist in Eigenregie organisiert werden, zum Beispiel in Form von kleinen hausinternen Fachkonferenzen, was auf jeden Fall einen besonderen Charme, aber auch seine Tücken hat. Der Umfang an benötigtem Wissen ist groß, und was man nicht regelmäßig benötigt, wird man wieder vergessen. Insofern wird man nicht umher kommen, dass Wissen in irgendeiner Form festzuhalten und bei Bedarf abrufbar zu machen.

Traditionell wird Wissen in Texten festgehalten, die dann später von denen gelesen werden, die es benötigen. Das ist schon bei der Erstellung mit viel Aufwand verbunden, da je nach Umfang und Komplexität des Gegenstandes viel Text mit zusätzlichen Beispielen und Materialien notwendig ist, um wirklich alles zu vermitteln. Entsprechend mutiert das Durchlesen dann auch eher zum Durcharbeiten und setzt so die Schwelle für die Auseinandersetzung mit dem Thema unnötig hoch. Es ist besser, hier andere Medien zu einzusetzen.

Erfahrungsgemäß orientierten sich Mitarbeiter bei der Anwendung von Wissen lieber an einem Beispiel, als an einer theoretischen Beschreibung. Was liegt also näher als die Theorie direkt an einem bereits umgesetzten Anwendungsfall zu erklären, und das nicht schriftlich, sondern in Form einer Screencast-Aufzeichnung, bei der man einfach auf die wichtigsten Stellen zeigt und diese kurz erläutert, was ohne große Vorbereitung und mit wenig Zeitaufwand möglich ist. Ergänzt man das Ganze noch durch eine kurze Wikiseite, auf der man die Grundidee und die wesentlichen Stichpunkte notiert, kann man damit im Laufe der Zeit eine beachtliche Bibliothek an Wissenseinheiten aufbauen, die sich auch noch gut überblicken und auffinden lassen. Auch das Anschauen einer solchen Aufzeichnung ist natürlich deutlich einfacher, als die Durcharbeitung einer langen schriftlichen Dokumentation und bietet darüber hinaus auch noch die Wahl, in welcher Tiefe man sich in das Thema einarbeiten möchte. Verwendet man die Aufzeichnung in Kombination mit den Artefakten des dort beschriebenen Anwendungsfalls, so hat man alles was man benötigt, um das dort vermittelte Wissen auf eine neue Situation anzuwenden.

Wichtig ist, die Dinge an dieser Stelle schlank zu halten. Die Länge der Aufnahmen sollte 10 Minuten nicht überschreiten, und die müssen nur inhaltlich richtig, aber in keinem Fall perfekt präsentiert sein. Entsprechend sollten weder Schnittprogramme oder Nachvertonungen zum Einsatz kommen. Eine solche Aufzeichnung sollte ohne großen Schmerz weggeworfenen und neu gemacht werden können, wenn sich etwas ändert. Auch die Zusatzdokumentation im Wiki sollte sich auf die wesentlichen Stichpunkte beschränken. Im

Idealfall reicht diese sogar aus, wenn man sich später erneut mit dem Gegenstand auseinandersetzen muss.

Dieses Beispiel zeigt gut, wie wichtig das Medium bei der internen Wissensvermittlung ist. Eine funktionierende und aktive Wissensvermittlung steht und fällt mit der Bereitschaft der Mitarbeiter, ihr Wissen zu dokumentieren, und sich mit dem dokumentierten Wissen anderer Mitarbeiter auseinanderzusetzen. Das geschieht nur dann, wenn die Hürden dafür niedrig, und der Nutzen klar erkennbar ist. Unter den richtigen Bedingungen kann hier sogar eine richtige Kultur der Wissensvermittlung entstehen, die gleichermaßen effektiv, als auch ressourcenschonend ist.

Auch *FlowProtocol* ist unnwillkürlich ein Medium zum Umgang mit Wissen und ähnlich wie oben ist auch hier die Niederschwelligkeit ein entscheidender Faktor. Dieser besteht beim Aufbau einer umfangreichen Vorlage im Wesentlichen darin, dass diese eben nicht in einem Rutsch erstellt werden muss, sondern ausgehend von einer ersten Version bei Bedarf Stück für Stück erweitert werden kann. Oft ist der Anfang das schwierigste, deshalb ist es wichtig, diesen möglichst einfach zu gestalten. Im Gegensatz zu einer Screencast-Aufzeichnung ist eine *FlowProtocol* -Vorlage jedoch eher ungeeignet, um vollständig neues Wissen zu vermitteln, sondern vielmehr dafür gedacht, um bei der Anwendung von vorhandenem Wissen anzuleiten und zu unterstützen. Allein das Verstehen und die richtige Beantwortung der in der Vorlage gestellten Fragen setzen ja schon eine gewisse Fachlichkeit voraus, die erst erworben werden muss. Ob nun in Form eines Unterstützungssystems, eines Konfigurators oder einer interaktiven Anleitung, jede *FlowProtocol* -Vorlage enthält essentielles Systemwissen, die von dort wieder relativ bequem abgerufen oder angewendet werden kann. Idealerweise kombiniert man also die Vermittlung des dafür notwendigen Grundwissens in der herkömmlichen Form mit dem Aufbau passender Vorlagen, die dabei helfen, dass Wissen auf einem höheren Level einfach und präzise anzuwenden.

1.5 Wiederkehrende Muster und Konfigurierbarkeit

Konfiguratoren funktionieren ähnlich wie die oben beschriebenen Unterstützungssysteme, nur dass anstelle einer Situation Anforderungen abgefragt werden, in deren Abhängigkeit bestimmte Schritte durchzuführen sind oder nicht. Jeder kennt vermutlich die Konfiguratoren der Fahrzeughersteller, die die Zusammenstellung eines Neuwagens und die Auswahl der gewünschten Extras ermöglichen. Am Ende der Konfiguration erhält man ein Bild seines Traumautos samt Preiskalkulation, sowie eine Liste, mit der man direkt zum Händler gehen kann. Dies ist möglich, weil die Auswahl innerhalb klarer Vorgaben variiert und sich dort klare Abhängigkeiten abbilden lassen, wie z.B. die verfügbare Motorisierung für ein bestimmtes Fahrzeugmodell oder der Preisaufschlag für Nebenscheinwerfer.

Entwicklungen zur Erweiterung einer großen Software funktionieren meist analog ähnlich und orientieren sich idealerweise an bewährten Mustern, die schon an anderen Stellen eingesetzt wurden, und die relativ schnell mit dem vorhandenen Framework auf weitere Stellen übertragen werden können. Auch hier kann es Variationen, zwischen denen man auswählen kann, und auch hier lässt sich ein Preis in Form von Aufwand berechnen. Ansonsten geht es wie im Fall oben darum, die für eine spezielle fachliche Arbeit benötigten Informationen abzufragen und bereitzustellen.

In der am Scrum-Prozess ausgerichteten Softwareentwicklung übernimmt normalerweise

der Product Owner die Aufgabe, die Anforderungen mit den verfügbaren Umsetzungsmustern abzugleichen und gegebenenfalls in die passende Zielform zu bringen, nicht zuletzt, um die Einheitlichkeit der Anwendung aufrechtzuerhalten, und der Aufbau eines Konfigurators scheint sich dafür nicht unbedingt zu rechnen. Umgekehrt hat man die wenigen Abhängigkeiten eines Musters relativ schnell in einer Vorlage abgebildet und profitiert so von einem System, dass sowohl bei der präzisen Formulierung, als auch bei einer standardisierten Form der Umsetzung unterstützt. Je anch Aufbau kann der Konfigurator auch direkt durch die Kollegen genutzt werden, die dem Product Owner die Anforderungen zutragen, so dass die darüber ausformulierten Entwicklungen nur noch geprüft, priorisiert und eingeplant werden müssen.

Einmal festgelegte Standards für Form und Umsetzungen werden gleichermaßen fest in die Konfiguration eingearbeitet und erlauben dahingehend keine Abweichungen. Es entfallen mühsame Diskussionen an Stellen, wo die Dinge schon entschieden sind und die Entwicklungsbausteine können in der Form verwendet werden, wie es vorgesehen ist – ein gerader Implementierungsprozess ohne unnötige Ecken und Kanten.

Im Abschnitt über interaktive Anleitungen wird die Idee des Konfigurators nochmals aufgegriffen und mit der Möglichkeit verknüpft, die aus einer Konfiguration resultierenden Entwicklungsschritte innerhalb einer Vorlage zu erstellen.

2 Bewertungen

Wie wichtig ist eine Anforderung, wie gravierend ein Fehler und wie gut passt eine Entwicklung in eine vorgegebene Strategie? Im Entwicklungsalltag treten permanent Fragen auf, bei denen die Antwort die Grundlage für wichtige Entscheidungen ist, und die das Handeln ganzer Teams bestimmen. Meistens handelt es sich dabei um Einschätzungen oder Bewertungen von bestimmten Situationen und allzu oft erfolgt die Beantwortung dann größtenteils auf der Grundlage von Intuition und Bauchgefühl.

2.1 Entscheidungsprozesse

Der Umgang mit Bewertungen hat nicht einen großen Einfluss auf die tatsächliche Festlegung der Arbeit, sondern auch darauf, wie diese wahrgenommen wird. Entscheidungen, die transparent und im Detail nachvollziehbar sind, haben generell eine höhere Akzeptanz als solche, die einfach nur von oben festgelegt werden. Entsprechend wird auch die Arbeit an einem Projekt anders ablaufen, wenn die hohe Priorität des Projektes für den Mitarbeiter gut nachvollziehbar ist, und er nicht den Eindruck hat, eigentlich an etwas wichtigerem Arbeiten zu müssen. Unsicherheit hinter Entscheidungen bremst unwillkürlich das Handeln aus, unabhängig davon wie klar die Entscheidung getroffen wurde. Dabei ist das Vorhandensein einer bewusst getroffenen, und bestenfalls dokumentierten und kommunizierten Entscheidung schon mal eine gute Ausgangssituation, selbst wenn diese weder begründet oder durchdacht ist. Nicht selten entstehen entscheidungswürdige Zustände auch vollständig durch Eigendynamik, und im Nachhinein fragt man sich, wer eigentlich das Ruder in der Hand hat. Der Ausgangspunkt kann ein einfacher Vorschlag sein, der am Ende nur mangels Alternativen, tagesaktueller Begeisterung oder aus Gründen der Konfliktvermeidung umgesetzt wird.

Obwohl die Entscheidungsprozesse innerhalb eines Bereiches zu den wichtigsten Prozessen gehören, werden sie selten formal als solche abgebildet. Dies gründet vermutlich

hauptsächlich darauf, dass das Treffen von Entscheidungen mehr als Privileg oder Verantwortung der höheren Führungsebenen gesehen wird, und man sich hier eher nicht einem Prozess unterordnen möchte. Diese Einstellung spiegelt sich dann auch im Umgang mit Entscheidungen, und lässt sich oft schon daran erkennen, dass Entscheidungen nur dann getroffen werden, wenn jemand bewusst dazu auffordert, und das obwohl die Situationen regelmäßig mit klaren Abhängigkeiten im Geschäftsalltag auftreten. Vermutlich ist so die Nachbestellung neuer Kaffeebohnen vielerorts deutlich systematischer geregelt, als die strategisch relevanten Entscheidungsprozesse.

Hat man jedoch erst einmal erkannt, an welchen Stellen Entscheidungen benötigt werden, ist die wichtigste Hürde überwunden, und es muss noch geklärt werden, wer die Entscheidung unter welchen Optionen auf Basis welcher Grundlage treffen soll. Gemeinschaftlichkeit und Objektivität

Der Anspruch, Entscheidungen objektiv und gemeinschaftlich zu treffen, ist weit verbreitet. Dies hat sich auch damit zu tun, dass der Ärger für die Verantwortung einer schlecht getroffenen Entscheidung schwerer wiegt, als die Anerkennung für eine gut getroffene. Unabhängig davon gelten mehrheitlich getroffene Entscheidungen jedoch als besser, was sicher auch an dem größeren Maß an Auseinandersetzung liegt, das dafür notwendig ist. Wichtig ist die Unterscheidung zwischen Gemeinschaftlichkeit und Objektivität. Eine gemeinschaftlich getroffene Entscheidung muss nicht objektiv sein und eine objektiv getroffene Entscheidung kann auch alleine getroffen werden. Beides sind Faktoren, die das Treffen der Entscheidung weg vom subjektiven Bauchgefühl einer einzelnen Person rücken und damit sowohl Legitimation, als auch Qualität erhöhen sollen.

Objektivität wird schon allein deshalb angestrebt, weil man ja im Grunde davon ausgeht, dass es überhaupt die richtige Entscheidung gibt, die sich dementsprechend auch absolut bestimmen lassen sollte. Im Detail geht man hierbei von Ursache- und Wirkungsprinzipien aus, die man verstehen und für Vorhersagen verwenden kann.

Im besten Fall trifft man Entscheidungen mehrheitlich und objektiv. Während Ersteres nur organisatorisch eine Herausforderung ist, die inzwischen jedoch mit vielen guten Abstimmungstools bewältigt werden kann, ist Objektivität schwieriger zu erreichen. Nachweisliche Objektivität gelingt meist nur mit Kriterien, die man zuerst aufstellen und teilweise begründen muss, was manchmal recht schwierig ist. So bleibt es oft dabei, dass die Entscheidung für oder gegen eine Sache zwar mit einer augenscheinlich objektiven Begründung getroffen wird, man jedoch nicht mehr auf die Details der dahinterliegenden Wirkungsweise eingeht.

Oft sind subjektive Entscheidungen emotionsgetrieben und man muss sich anstrengen, um sich hier auf die Sachlichkeit zu beschränken. In anderen Fällen ist es umgekehrt, und man muss eine Sache kategorisieren, ohne dass man emotional zu einen oder anderen Richtung tendiert. Hier muss man zwangsläufig auf Dinge wie Kriterien zurückgreifen oder sich bewusst auf eine Gefühlsentscheidung einlassen.

2.2 Arbeiten mit Kriterien

Wie schon gesehen sind Kriterien ein wichtiges Werkzeug, wenn es darum geht, Dinge und Situationen in Bezug auf festgelegte Aspekte möglichst objektiv zu bewerten und einzustufen. Deren Aufstellung ist sowohl fachlich, als auch methodisch anspruchsvoll, und kann viel Zeit in Anspruch nehmen.

Der große Vorteil bei der Arbeit mit Kriterien ist der, dass sich diese immer wieder ver-

wenden lassen, und jede darin investierte Zeit also vielfachen Nutzen bringt. Ein weiterer großer Vorteil ist der, dass die Arbeit mit Kriterien in großem Umfang dazu beiträgt, Sachverhalte und Zusammenhänge tiefgreifend zu verstehen, und bei der Anwendung in gleicher Weise zu vermitteln.

Kriterien sind die Einzelaspekte, die in ihrer Gesamtheit die Qualität oder Eignung einer Sache für eine bestimmte Zielverfolgung ausmachen, und die damit eine Begründung für Planungsentscheidungen liefern. In geeigneten Fällen lassen sich Kriterien sogar gewichten und am Ende sogar mit einer Metrik beaufschlagen, so dass man verschiedene Gegenstände direkt über einen Zahlenwert vergleichen kann.

Für die meisten Bereiche werden die Kriterien dauerhaft sein, und sich im Laufe der Zeit nur im Detail verfeinern. Die zu bewertenden Dinge (z.B. Anforderungen und Wünsche) haben ihren Ursprung an permanenten Eingangsströmen und Anwendung der Kriterien ist fest in einem Prozess verankert. Das schnelle Aufstellen und Verfügbarmachen von Kriterien ermöglicht es jedoch auch, Kriterien auch nur im Rahmen einer kurzfristigen Strategieverfolgungen oder eines Projektes aufstellen, und damit z.B. die Wunschdatenbank nach passenden Kandidaten zu durchsuchen.

2.3 Fehlerpriorisierung nach Schweregrad

Um auf die Vorgehensweise bei der Aufstellung von Kriterien genauer einzugehen, schauen wir uns das folgene Beispiel an: Die Priorisierung der Fehlerbehebung soll sich an einem Schweregrad-System orientieren, um die vorhandenen Ressourcen bestmöglich auf die wichtigsten Vorgänge zu konzentrieren. In dieser einfachen Formulierung stecken schon relativ viele Punkte, die schon das Ergebnis grundlegender Gedanken sind. Die Ausgangslage für eine solche Optimierung ist am Anfang erst mal nur die Feststellung, dass die bestehende Vorgehensweise in Bezug auf bestimmte Aspekte Schwächen hat, im vorliegenden Fall zum Beispiel: bei der chronologischen Abarbeitung von Fehlervorgängen kommt es zu größeren Verzögerungen bei der Behebung von Engpässen, die die Arbeitsfähigkeit des Kundensystems beeinträchtigen.

Da die chronologische Priorisierung als Ursache des Problems festgestellt wurde, wird man diese zuerst gegen eine Priorisierung nach Schwerengraden ersetzen und z.B. die vier Schweregrade „kritisch“(1), „hoch“(2), „moderat“(3) und „niedrig“(4) einführen, alle Fehlervorgänge zuordnen und entsprechend des Schweregrades abarbeiten.

Da sich die Zuweisung eines Schweregrades nun unmittelbar auf die Reihenfolge der Umsetzung auswirkt, wird jeder direkt oder indirekt von einem Fehler Betroffene ein Interesse daran haben, den größtmöglichen Schweregrad zuzuordnen, um eine zeitnahe Bearbeitung sicherzustellen. Überlässt man diese Zuordnung dann auch noch den Erstellern der Vorgänge, erlebt man schnell eine Schweregrad-Inflation, bei der am Ende selbst Tippfehler mit „hoch“eingestuft werden, insbesondere wenn die Abarbeitungsreihenfolge gar nicht mehr sichergestellt ist, dass Vorgänge mit dem Schweregrad „moderat“überhaupt umgesetzt werden.

Letzteres kann zumindest durch das Autobahnprinzip verhindert werden. Hierbei wird die Abarbeitung von Vorgängen der einzelnen Schweregrade so organisiert wie der Verkehrsfluss auf den Spuren einer Autobahn. Auf der linken Spur kommt man am schnellsten voran, auf der mittleren etwas langsamer und auf der rechten noch etwas langsamer, aber auf allen Spuren kommt man vorwärts. Übertragen auf die Schweregarde könnte das so aussehen, dass man kritische Aufgaben umgehend bearbeitet, und die Abarbeitung der

rechtlichen Schweregrade mengenmäßig in einem entsprechenden Verhältnis einplant, etwa so:

$$\text{hoch} : \text{moderat} : \text{niedrig} = 8 : 5 : 3$$

Innerhalb jedes Schweregrades kann man nun wieder chronologische Reihen bilden, bei denen sich neue Vorgänge unten einreihen, innerhalb der Reihe mit der dort vorgegebenen Geschwindigkeit nach oben wandern und dann dort abgeschöpft werden. Ungenutzte Kapazitäten kann man auf den nächstniedrigeren Schweregrad übertragen.

Im nächsten Schritt geht es darum, festzulegen, unter welchen Umständen ein Fehler kritisch ist oder einen entsprechend hohen Schweregrad hat. Am sinnvollsten ist es, die Bestimmung hier absteigend durch hinreichende Kriterien durchzuführen, d.h. man listet zuerst alle möglichen Aussagen auf, deren Zutreffen den Fehler allein schon als kritisch klassifizieren, und ergänzt diese durch die Option „Keine davon.“.

Welche der folgenden Aussagen trifft auf den Fehler zu?

- ◯ *Der Fehler blockiert die Anwendnung vollständig. Kein Benutzer kann aktuell arbeiten.*
- ◯ *Der Fehler wirkt sich schädigend auf den Datenbestand aus. Daten gehen verloren oder werden unwiederbringlich in einer nicht vorgesehenen Form verändert.*
- ◯ *Der Fehler führt zu Verfälschungen von Daten, die finanzielle oder juristische Folgen haben können.*
- ◯ *Der Fehler führt zu einer Einschränkung einer explizit zum Datenschutz eingesetzten Funktion.*
- ◯ *Keine davon.*

Wählt man dies letzte Option, so werden im nächsten Schritt explizit die Aussagen aufgelistet, die dem nicht kritschen Fehler zumindest noch einen hohen Schweregrad bestätigen, usw., bis man dem Fehler am Ende nach dem Ausschlussprinzip den Schweregrad „niedrig“ zuordnet, ohne dafür noch explizit Aussagen zu formulieren.

Das System könnte auch umgekehrt aufgebaut werden, allerdings dürfte es dann deutlich schwieriger sein, am Anfang alle Aussagen für einen niedrigen Schweregrad aufzulisten.

Auch hier gilt es natürlich wieder, die Kriterien bei Bedarf zu erweitern. Der Auslöser dafür wird dann sein, wenn der ausgegebene Schweregrad nicht mit der eigenen Einschätzung übereinstimmt, und man eine relevante Aussage in der Auflistung vergisst. So einfach eine entsprechende Erweiterung in der Vorgehensweise auch ist, sollte sie trotzdem nach festgelegten Regeln ablaufen, um zu verhindern, dass die Vorlage willkürlich von jedem angepasst werden kann, um für seine Vorgänge die gewünschte Einstufung zu erhalten.