Softwareentwicklung mit FlowProtocol

# 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.

# Die Vorteile 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."

## Vielseitige Einsatzmöglichkeiten

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, gibt es in jedem Bereich Aufgaben, die sich gut mit FlowProtocol umsetzen lassen. Ähnlich wie das vermutlich in jenem Firmen-Netzwerk zu findende interne Wiki kann man dem entsprechend auch FlowProtocol generell für alle Mitarbeiter verfügbar machen und als allgemeines Werkzeug etablieren.

## Förderung von Teamzusammenarbeit und Prozesskultur

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 der 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 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.

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

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. Wenn man möchte, kann man die Begründung auch in Form von Erläuterungstexten direkt in der Vorlage anzeigen lassen, um gleichzeitig auch den Aufbau von Wissen und Verständnis zu fördern.

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.

## Ausschöpfung von Fachwissen

%%% Zusammenspiel mit anderen Systemen

% Bewertungen

%%% Die Wichtigkeit von Objektivität

%%% Arbeiten mit Kriterien

%%% Rangfolgen und das gute alte Bauchgefühl

%%% Umfragen

% Kleine Expertensysteme

%%% Zielsetzung und Anwendungsbeispiele

%%% Wissen verfügbar machen ohne Wissensaustausch

%%% Der steinige Weg zum guten Expertensystem

% Das schwere Los des Product Owners

%%% Es allen Leuten Recht getan\dots

%%% Tellerfertige Stories für Entwickler

%%% Die Kunst nix zu vergessen

%%% Die SekretärIn im Browser

% Die Einbeziehung der Teams

%%% Durchgängigkeit beginnt bei der Story

%%% Bereitstellung von Codefragementen

%%% Synergieen mit BestPractice und CleanCode-Prinzipien

%%% Risikobewertung und Risikovermeidung

%%% Unterstützung bei Integrations- und Unittests

% Werkzeug zur Reflektion

%%% Selbstverbesserung durch reflektierende Fragen

%%% Diskretion, Verfügbarkeit und Protokollierung

%%% Beispiel der PO-Abnahme-Vorbereitung

% Schnell gebaute Werkzeuge

%%% Das große Feld der Textbausteine

%%% Einbeziehung von Fallunterscheidungen

%%% Navigationsbäume

%%% Anwendungsfälle für generierte Links

% Einrichtung, Pflege und Schulung

%%% Eine Aufgabe für CI und IT

%%% Speicherort, Berechtigungen, Versionsverwaltung

%%% Mit guten Beispielen vorangehen

%%% Einbindung in Prozesse und Infrastruktur

%%% Kultur aufbauen und am Leben halten

% Schlusswort