Gottfried Wilhelm Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Konzept und Realisierung eines Ratgebers für die Softwareentwicklung basierend auf der Analyse von Informationsflüssen

Masterarbeit

im Studiengang Informatik

von

Lars de Vries

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr.-Ing. Helena Szczerbicka
Betreuer: M. Sc. Kai Stapel

Hannover, 4. Dezember 2008

Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst habe und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe.

Hannover, 4.12.2008 Lars de Vries

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung

Die Softwareentwicklung hat einen immer größeren Anteil an Produkten in inzwischen fast jedem Wirtschaftszweig, zum Beispiel in der Automobilindustrie. Anfangs als Zubehör oder Teil eines Produktes hat Software als eigenständiges Erzeugnis in den letzten Jahrzehnten immens an Bedeutung gewonnen, so dass von Software-Produkt-Management die Rede ist.

Bei neueren Betrachtungen aus wirtschaftlicher Perspektive findet sich in der Literatur ein anschaulicher Eindruck, zu welcher Wichtigkeit Software, ein immer als etwas abstrakt empfundenes immaterielles Wirtschaftsgut, inzwischen gelangt ist. Als „Manifestierung von menschlichem Know-How in Bits und Bytes“ wird Software hier neben den klassischen (und für Jahrhunderte in der Volkswirtschaftslehre einzigen) Wirtschaftsfaktoren *Kapital*, *Boden* und *Arbeit* in eine neue, vierte Kategorie eingeordnet: *Wissen* (vgl. [KRS2004]). Bei der Entwicklung von Software ist Wissen ein entscheidender Faktor.

Grundlage von Wissen sind Informationen. Die Mathematik beschäftigt sich seit Begründung der Informationstheorie nach Shannon um 1948 im Rahmen der Berechnung der Entropie mit dem Informationsgehalt von Nachrichten – die Informatik trägt „Information“ als Begriff sogar im Namen und befasst sich mit deren systematischer Verarbeitung.

Der richtige Umgang mit Informationen während des Softwareentwicklungsprozesses ist Gegenstand des Forschungsprojekts FLOW, das seit 2004 am Fachgebiet Software Engineering der Leibniz Universität Hannover die Wechselwirkungen von Informationsflüssen in Softwareentwicklungsprojekten untersucht. Es wurden bislang Methoden zur Notation, Simulation und zur allgemeinen Analyse entwickelt. Zuletzt wurde ein Katalog mit bisher bekannten Informationsflussmustern erstellt, auf den in dieser Arbeit auch wiederholt eingegangen wird.

Die Methoden, Werkzeuge und Hilfestellungen von FLOW sind auf die Analyse von Informationsflüssen in Softwareprojekten ausgerichtet. Konkrete Anleitungen, wie Informationsflüsse *verbessert* werden können, fehlen jedoch bisher. In dieser Arbeit sollen daher Ergebnisse der FLOW-Forschung so um- und eingesetzt werden, dass eine konkrete Nutzung der Erkenntnisse über Informationsflüsse möglich wird. Der Titel der Arbeit und die Aufgabenstellung benennen dabei als Nutzungsidee einen „Ratgeber“, der helfen soll „in einer Projektsituation auf Grundlage der FLOW-Prinzipien angemessen zu reagieren“, als Metapher für ein Werkzeug oder eine Methode. Ein entsprechendes Konzept zu erstellen um einen FLOW-Ratgeber realisieren zu können ist das Hauptziel dieser Arbeit.

Wenn so der Entwicklungsprozess optimiert oder zumindest verbessert werden kann, steigt auch die Qualität des Produktes: der Software selbst.

1.2 Aufgabenstellung und daraus abgeleitete Teilaufgaben

Das Ziel dieser Arbeit umfasst die Entwicklung eines Konzeptes, dessen praktische Umsetzung es ermöglichen soll, Projektsituationen auf Basis der Erkenntnisse der FLOW-Forschung zu bewerten. Sofern dies möglich ist, soll dann in Projektsituationen dazu angeleitet werden, angemessen zu handeln.

Über diese Grundaussage hinaus bleibt die Aufgabenstellung bewusst allgemein, so dass zunächst konkrete Teilziele definiert werden müssen. In der Aufgabenstellung werden allerdings verschiedene Aspekte genannt, die zum Erreichen des Ziels von Bedeutung sein könnten. Während einige Teilaufgaben somit offensichtlich sind, war es Teil der Aufgabe, weitere zu betrachtende Aspekte überhaupt erst zu identifizieren.

1.2.1 Technische Teilaufgaben

Das Entwickeln eines „Ratgeber“-Konzepts ist Hauptteil der Arbeit. Die Ausgestaltung dieses Konzepts stellt die wichtigsten anstehenden Aufgaben dar. Hierbei geht es zunächst vor allem um die Eigenschaften des Ratgebers selbst:

- Einsatzbedingungen des Ratgebers
  Die Umstände der Verwendung, also die gegebenen Einsatz-Hintergründe sind zu untersuchen und der genaue Einsatzzweck des Ratgebers zu bestimmen. In diesem Zusammenhang ist eine Anforderungsanalyse für einen entsprechenden Ratgeber nötig. Zum Beispiel ist zu klären, wer den Ratgeber konsultiert, in welchen Projektsituationen und mit welchem Ziel.

- Implementierungsweise des Ratgebers
  Zu untersuchen ist weiterhin, in welcher Form ein Ratgeber-Konzept am besten umzusetzen ist – ein Handbuch ist genauso denkbar wie eine Software.

- Umsetzung des Ratgebers
  Zudem ist es ein Teilziel, den Ratgeber zumindest ausschnittsweise prototypisch zu realisieren. Es kann hierbei ausreichen, einzelne Aspekte exemplarisch zu zeigen, um jeweils Beispiele für die Umsetzung anzugeben, die Machbarkeit zu überprüfen und ggf. auf technische Schwierigkeiten und Details eingehen zu können. Es werden Erweiterungsmöglichkeiten betrachtet, wenn aus Zeitgründen im Rahmen dieser Arbeit nur Teilbereiche des Ratgebers konkret ausgestaltet werden können.

Die erhobenen Anforderungen, deren Ausarbeitung und spätere Umsetzung im Ratgeber stellen die Bearbeitung dieser „technischen Teilaufgaben“ dar. Die Ergebnisse werden in Abschnitt 2.3.1 *„Anforderungen an den Ratgeber“* zusammengefasst.

1.2.2 Theoretische Teilaufgaben

Die Arbeit an der Ausgestaltung des eigentlichen Ratgeber-Konzepts erfordert die systematische Suche nach neuen Erkenntnissen, da für diese Form der praktischen Anwendung von FLOW noch keine Erfahrungen bestehen. Um weiterhin unter Berücksichtigung relevanter Projektparameter für problematische Projektsituationen Lösungen vorschlagen zu können, gehen aus der Aufgabenstellung folgende zu untersuchende Bereiche hervor:

- Extraktion ergebnisorientierter FLOW-Prinzipien
  Ermöglicht erst eine verfeinerte Formulierung der FLOW-Prinzipien oder eine Definition und Auswahl einiger spezieller Regeln die praktische Verwendung des Informationsfluss-Gedankens?

- Anpassung der FLOW-Notation
  Da eine weitere Formalisierung oder Spezialisierung der FLOW-Notation denkbar ist, um diese Regeln darstellen zu können oder Metriken zur Bewertung zu erstellen, sind die Möglichkeiten zu überprüfen und gegebenenfalls zu ergänzen, die die bisherigen Erkenntnisse bieten.

- Erstellung von speziellen FLOW-Metriken
  Um Aussagen über eine Projektsituation machen zu können, die das Erteilen von Ratschlägen erlauben, muss eine Bewertung ermöglicht werden. Metriken für FLOW würden Projektsituationen bezüglich ihrer Informationsflüsse bewertbar machen. Es ist zu untersuchen, welche Situationen mit auf FLOW basierenden Metriken „gemessen“ werden können, und wie hierzu vorzugehen ist.

- Untersuchung neuer FLOW-Muster und Abstraktionsgrade
  Ausgangspunkt für eine Bewertung mittels Metriken wären abstrakte Darstellungen von typischen Projektsituationen. Dies könnte z.B. durch FLOW-Muster oder -Mustergruppen geschehen, wie sie im FLOW-Pattern-Katalog [GE2007PK] beschrieben werden. Dieser Katalog befasst sich mit FLOW-Mustern auf der untersten, detailliertesten Ebene. Hier sind andere Betrachtungshöhen und Abstraktionsgrade denkbar, die von Bedeutung sein könnten.

Im Rahmen der Erforschung dieser Bereiche wurden einige Thesen aufgestellt und Ideen entwickelt, die im Verlauf dieser Arbeit geprüft wurden. Diejenigen, die sich als relevant herausstellten, werden im Folgenden diskutiert und ergeben Kernpunkte der Arbeit. Sie bilden die theoretische Grundlage für das Konzept des Ratgebers.

Die Entwicklung der Ratgeber-Idee und die Untersuchung aller dafür notwendigen Hintergründe stellen die Bearbeitung dieser „theoretischen Teilaufgaben“ dar. Die Ergebnisse werden im Abschnitt 2.3.2 *„Ratgeber-Idee: vom Problem zur Ursache“* zusammengefasst.

1.3 Gliederung der Arbeit

Diese Arbeit ist wie folgt aufgebaut:

Kapitel 1 In der Einführung wird die Motivation hinter der Arbeit erläutert. Die Aufgabenstellung um die Erstellung eines Ratgebers und daraus abgeleitete Teilaufgaben werden beschrieben.

Kapitel 2 Die Teilaufgaben betreffen die Untersuchung verschiedener Aspekte, die die Ausarbeitung eines Konzepts für den angestrebten Ratgeber ermöglichen soll.

In diesem Kapitel werden die hierzu durchgeführten Untersuchungen beschrieben.

Der erste Abschnitt diskutiert die oben genannten Teilaufgaben. Die Forschung zu dieser Arbeit stützt sich zu großen Teilen auf Interviews; deren Durchführung wird im zweiten Abschnitt beschrieben. Die gezogenen Schlüsse werden im dritten Abschnitt zusammengefasst.

Kapitel 3 Der Ratgeber als Umsetzung der Ergebnisse wird in den zwei Teilen dieses Kapitels beschrieben.

Zunächst wird das theoretische Konzept des Ratgebers erläutert und auf die Zusammenhänge mit der betriebenen Forschung eingegangen, die Ergebnisse werden detailliert diskutiert.

In einer prototypischen Realisierung wird schließlich die Ausarbeitung des Ratgeber-Konzepts demonstriert und in einer beispielhaften Konsultierung die Anwendung des Ratgebers selbst.

Kapitel 4 Im vierten Kapitel finden sich Bewertungen der Ergebnisse und ein Ausblick auf die ausstehende Forschung.

2 Untersuchungen

Bevor die Entwicklung des Konzepts begonnen werden konnte, das als Ergebnis dieser Arbeit angestrebt wurde, erforderten die Vorarbeiten einen immensen Forschungsaufwand, der in diesem Kapitel dargestellt werden soll. In Abschnitt 2.1 *„Theoretische Grundlagen“* werden die Aspekte der FLOW-Theorie zusammengefasst, die untersucht wurden. Es wird zudem erläutert, warum diese für die Thematik von Bedeutung sind. Abschnitt 2.2 *„Interviews“* beschreibt die Gespräche, die zur Untersuchung, Absicherung und Validierung der Theorie geführt wurden. Abschließend werden in Abschnitt 2.3 *„Grundlegende Ergebnisse“* die wichtigsten Erkenntnisse diskutiert, auf denen später das Ratgeber-Konzept aufgebaut wird.

2.1 Theoretische Grundlagen

2.1.1 Extraktion ergebnisorientierter FLOW-Prinzipien

Zielorientierte Sicht der FLOW-Prinzipien
Der Stand der bisherigen FLOW-Forschung ist auf einem akademischen Level sehr ausgeprägt, für eine konkrete Verwendung als „Motor“ und Leitidee eines Ratgebers, der effizient und zielorientiert Nutzen bringen soll, sind jedoch noch Abgrenzungen und fest umrissene Regeln zu definieren, die als Grundlage für eine einheitliche Anwendung von FLOW dienen können.

Betrachtung der aktuellen FLOW-Forschung und verwandter SE-Bereiche
Die FLOW-Forschung bietet verschiedene Ansätze, zum Beispiel zu *Tailoring* und *Analyse*. Es stehen neuerdings erste Informationsfluss-Muster zur Verfügung [GE2007PK]. Es ist zu untersuchen, wie diese (ggf. gemeinsam mit „FLOW-fremden“ Ansätzen) genutzt werden können, um Metriken zu erstellen oder FLOW für Aufgaben wie *Projektplanung*, *Teaming* oder *Capability-Matching* einzusetzen. Welche weiteren Themen des Softwareengineering sollten in die Entwicklung eines Ratgebers einbezogen werden? Könnte hier FLOW mit Softwarequalität verknüpft werden, ist eine Verbindung mit psychologischen Aspekten (wie Gruppendynamik) denkbar?

Gelingen soll der *Brückenschlag zwischen Forschung und Praxis*. Welche FLOW-Prinzipien und bisherige Forschungsansätze hierfür ausgewählt wurden wird im Folgenden dargestellt und begründet.

Spezielle Anforderungen vom Standpunkt dieser Arbeit
Die Anforderung, eine praktische Anwendbarkeit der FLOW-Theorie zu erreichen, bringt die Stichworte *Praxisrelevanz* und *Realisierbarkeit* in die Diskussion. Und selbst wenn eine praxisrelevante Thematik gefunden wird und ein diesbezüglicher Ratgeber technisch umsetzbar wäre, muss die *Zweckmäßigkeit* kritisch und realistisch betrachtet werden.

Zum Beispiel: Würde ein Ratgeber, der das zweifellos relevante Thema *Projektplanung* aufgreift, sich mit den Möglichkeiten der FLOW-Darstellung umsetzen lassen? Und wenn, wäre es vorstellbar, dass ein entsprechendes Tool wirklich zur Anwendung käme, um Informationsfluss-optimierte Projektpläne in FLOW-Notation zu erstellen?

Im Verlauf der Untersuchungen wurde entschieden, dass dies nicht wahrscheinlich ist und die verlockende Aufgabe „Tailoring bei der Projektplanung“ und ähnlich ehrgeizige Anwendungen keinen lohnenswerten Hintergrund für diese Arbeit bieten. Die Anzahl der Variablen,

die zu berücksichtigen wären, ist zu hoch, der Kontext zu ausgedehnt für die wirkungsvolle Anwendung von FLOW. Der *Tailoring*-Gedanke, der als ein zentrales Thema der FLOW-Forschung „Softwareengineering nach Maß“ zum Ziel hat, muss auf eine überschaubarere Umgebung angewendet werden und sich auf die Stärken von FLOW konzentrieren. Diese liegen klar in der Darstellung von abgeschlossenen Situationen und Zusammenhängen sowie deren Analyse. Der FLOW-Pattern-Katalog [GE2007PK], der zuletzt als ein Ergebnis der FLOW-Forschung präsentiert wurde, könnte hier gezielt zur Anwendung kommen.

Der Gedanke, diese Aspekte zusammenzuführen, wird in dieser Arbeit verfolgt und in der Ratgeber-Methode zum Ziel gebracht: Während der Untersuchungen (in Abschnitt 2.1.4) wird die Anwendbarkeit des Pattern-Katalogs diskutiert. Als erste Ergebnisse werden (in Abschnitt 2.3.2) *Problem*situationen als die Situationen gewählt, in denen der Ratgeber zur Anwendung kommen soll. Das Verfahren der Ratgeber-Methode beschreibt schließlich (in Kapitel 3.1.3) eine detaillierte Anwendung der FLOW-Techniken zum situationsbezogenen Prozess-Tailoring.

2.1.2 Anpassung der FLOW-Notation

Die Aufgabenstellung fordert zu einer Untersuchung der Erweiterungsmöglichkeiten der FLOW-Notation auf, sofern dies für den Ratgeber sinnvoll oder notwendig erscheint. Um die Eignung der Notation zu prüfen wurde sie in Interviews (siehe Kapitel 2.2) vorgestellt und aktiv verwendet, um Situationen zu beschreiben. Hierbei stellte sich heraus, dass die vorhandenen Notationselemente keiner Erweiterung bedürfen. Sie wurden lediglich mit anderen Notationen kombiniert, die in diesem Umfeld ohnehin gebräuchlich sind: Im Rahmen der Ratgeber-Methode wird zur Darstellung von Prozess-Zusammenhängen auf SADT (*Structured Analysis and Design Technique*, vgl. [SADT]) zurückgegriffen (3.1.2 *„Schritt 2: „Flussaufwärts“ - Suche im Prozess“*), und zur Analyse einzelner Situationen werden FLOW-Signaturen [vgl. WCSQ05] verwendet (3.1.3 *„Schritt 3: „Beratung“ - Abfragen von Signaturen“*).

2.1.3 Erstellung spezieller FLOW-Metriken

Es wurden verschiedene Methoden zur Quantifizierung von Informationsflüssen in FLOW-Diagrammen erwogen. Für die Bewertung der Situation in der Ratgeber-Methode wurden jedoch keine Metriken in diesem Sinne verwendet. Vielmehr wird eine qualitative Einordnung vorgenommen. Hier kommt die Idee des *Pattern-Matching* zur Anwendung, die durch die Betrachtung des FLOW-Pattern-Katalogs angeregt wurde: Abweichungen von wünschenswerten „guten“ Situationen werden als „schlecht“ beurteilt und führen zu Verbesserungsvorschlägen. Dieses Vorgehen ermöglicht auch eine Automatisierung und wird in Kapitel 3.1.3 *„Mit Hilfe von Fragen zu Darstellung und Analyse der Ist-Situation“* erläutert. Die Umsetzung verwendet keine der naheliegenden (und zunächst angedachten) notations-basierten Gewichtungen oder führt zu einer quantitativen Messbarkeit anhand eines „FLOW-Scores“ für Situationen. Stattdessen stellte es sich als praktikabler heraus, mittels Fragen zu helfen, eine „Ist“-Situation in FLOW darzustellen und Abweichungen vom „Soll“ aufzudecken. So wird eine qualitative Bewertung ermöglicht.

2.1.4 Neue FLOW-Muster und Abstraktionsebenen

Diskussion der praktischen Anwendbarkeit des FLOW-Pattern-Katalogs

Der von Xiaoxuan Ge vorgestellte Katalog von FLOW-Patterns [GE2007PK] wirft (bei aller Gültigkeit der darin beschriebenen Muster) die Frage auf, wie eine praktische Nutzung aussehen kann.

Die Bezeichnung „FLOW-Patterns“ ist gewählt in Anlehnung an die berühmten Design-Patterns von Gamma et al. [GAMMA94], wie im Abschlussvortrag zur Masterarbeit erklärt wurde. Tatsächlich ist dieser Begriff jedoch eher irreführend. Im Gegensatz zu Gammas Patterns, die wie „Schnittmuster“ als Vorlagen für das Vorgehen in bestimmten Situationen genutzt werden können, sind die in diesem Katalog vorgestellten Muster vielmehr als FLOW-„Signaturen“ zu verstehen: Es wird beschrieben, wie man sie erkennt, nicht, wie man vorgehen muss, um einen bestimmten Muster-Zustand zu erreichen (dies ist zudem auch nur bei wenigen aufgeführten Mustern überhaupt wünschenswert; nur 9 von 22 werden als „positiv“ klassifiziert, die meisten Muster schildern problematische Situationen).
Die hier gesammelten Beschreibungen könnten also – wie Fingerabdrücke – zum Vergleich genutzt werden, um mittels automatisiertem Pattern-Matching zu analysieren, ob eine vorliegende Situation ein FLOW-Muster enthält. Es werden in einigen Fällen komplementäre Muster genannt, deren Anwendung die Wirkung eines „negativen“ Musters umkehren kann. Jedoch kann der Katalog nicht ohne weiteres als eine Sammlung von Vorlagen zur FLOW-effizienten Gestaltung einer Situation herangezogen werden.

Sicher ist die Kenntnis des Katalogs und der darin beschriebenen Muster wertvoll. Beispielsweise hat ein Projektleiter, der sich die FLOW-Sicht auf die Prozesse seines Projekts in dieser Form zu Eigen gemacht hat, insbesondere die Möglichkeit, Muster wiederzuerkennen oder absichtlich anzuwenden oder zu vermeiden. Die Gültigkeit oder Richtigkeit der dort beschriebenen Muster soll hier nicht in Frage gestellt werden. In den Händen einer Person, die noch nicht mit den enthaltenen Mustern und der Denkweise von FLOW vertraut ist, bietet der Katalog jedoch keine zielorientierte Anwendungsmöglichkeit und ist somit nur bedingt von Nutzen.

Dies liegt vor allem daran, dass die im Katalog beschriebenen Muster in einer praktischen Situation bereits bekannt sein müssen und nicht effektiv „nachgeschlagen“ werden können. Denn zum einen ist es einfach unrealistisch, dass ein Nutzer in einer schwierigen Projektsituation zum Katalog greift und versucht, ein passendes Muster zu identifizieren. Diese Einschätzung bestätigen die meisten der Gesprächspartner, die in Interviews (Kapitel 2.2) befragt wurden. Die Nutzung des (Online-)Katalogs ist eher eine didaktische Maßnahme zur Information und Ausbildung von „FLOW-Awareness“.
Man kann sagen: *„Wer FLOW bereits so weit verinnerlicht hat, dass er die FLOW-Denkweise des Katalogs* (z.B. zur Erkennung von Problemmustern und Einsatz komplementärer Lösungsmuster) *anwenden kann, wird dies nicht mehr mit dem Katalog tun“*. Kurz, wer bereits FLOW-Experte ist, für den hat sich der Nutzen des Katalogs bereits ergeben.
Zudem gibt es Muster, die in der Praxis nur schwer aufzudecken sind, da sie zum Beispiel (wie die Muster „fehlende Erfahrung“ oder „Dead Document“) das Einzeichnen von fehlenden Informationsflüssen im FLOW-Diagramm voraussetzen. Eben das Fehlen dieser Flüsse macht hier aber das Problem aus, und warum sollte ein Nutzer ein Diagramm zeichnen, dessen Erstellung ihm keine neue Erkenntnis bringt, sondern voraussetzt, dass er bereits weiß, was er darstellen will?

Abstraktion und Detailliertheit

Bevor als Folgerung aus dieser Kritik der praktischen Anwendbarkeit des Pattern-Katalogs ein Lösungsansatz vorgestellt wird, sollen zwei Kernbegriffe in diesem Kontext geklärt werden. Die Muster des FLOW-Pattern-Katalogs sind nämlich auf der einen Seite sehr abstrakt und beschreiben Situationen allgemeingültig. Hierfür erfolgt eine „Übersetzung“ in eine Modellierung, wodurch das Auftauchen abundanter und der Wegfall präterierter Attribute eine Rolle bei der Verwendung spielen. Andererseits findet die FLOW-Analyse auch sehr detailliert und auf einer Betrachtungshöhe statt, die so nah am Informationsflussgedanken ist, dass sie voraussetzt, dass der Anwender bereits über FLOW-Verständnis verfügt. Beides steht einer praktischen Anwendung im Wege.

Anwendung auf „praxisnaher“ Analysestufe

Aufbauend auf diesen Überlegungen wurde der Ansatz entwickelt, die Anwendung der FLOW-Pattern-Idee praxisnah auf einer anderen *Analysestufe* zu beginnen und von dort aus an die detaillierte Analyse heranzuführen. Eine solche Stufe läge dabei zunächst noch weit oberhalb der Sicht der FLOW-Patterns. Die Idee ist dabei, mögliche Anwender auf dieser praktischen Ebene in der Realität „abzuholen“ und bei der Transformation in die FLOW-Sicht zu unterstützen. Hierbei müsste FLOW weder bereits verinnerlicht noch überhaupt bekannt sein. Die Heranführung könnte möglicherweise durch eine automatisierte Erstellung von FLOW-Diagrammen geschehen. Bevor aber genaue Methoden vorgeschlagen werden können, ist es Aufgabe, eine geeignete Analysestufe zu finden und zu beschreiben.

„Standardsituationen“ als neue Muster

Wie oben ausgeführt haben die Muster im FLOW-Pattern-Katalog zum großen Teil eher den Charakter von „Signaturen“ als von „Vorlagen“. Ein Ratgeber würde aber eher von letzteren (Pattern im Sinne der „Gang of Four“ [GAMMA94]) profitieren. Schließlich geht es bei einer Beratung darum, einen Lösungsweg vorgeschlagen zu bekommen.

Es ist also ein Teilziel dieser Arbeit, neue Muster dieser Art zu finden und dabei auch andere Analysestufen zu berücksichtigen als die sehr detaillierte Ebene, die für die Muster im Pattern-Katalog gewählt wurde. Diese wurden zunächst als „Standardsituationen“ bezeichnet. In Interviews mit Praktikern (siehe Abschnitt 2.2) wurden Beispiele gesammelt.

Bewertung von notierten und nicht-notierten Tatsachen

Weiterhin wurde die Wichtigkeit von „nicht-notierten“ Tatschen im FLOW-Pattern-Katalog bereits angesprochen. Gemeint sind Erkenntnisse, die ausdrücklich durch das Fehlen von wichtigen Elementen in einem FLOW-Diagramm zustande kommen. Der Vergleich von „Ist“ mit „Soll“ scheint hier die Lösung zu sein, um das Fehlen von Elementen oder Zusammenhängen feststellen zu können.

Signaturen von Aktivitäten

Die angesprochenen Aspekte werden durch *Signaturen von Aktivitäten* zusammengebracht. Auf einer praxisnahen Analysestufe werden hier Muster aufbereitet und verglichen. Die Ratgeber-Methode zeigt gleichzeitig eine Möglichkeit auf, wie FLOW-Erkenntnisse angewendet und vermittelt werden können, ohne dass der Ratsuchende selbst über FLOW-Wissen verfügen muss. Erläutert wird der Ansatz und das genaue Vorgehen in Kapitel 3.1.3 *„Schritt 3: „Beratung“ - Abfragen von Signaturen“*.

2.2 Interviews

In diesem Abschnitt wird erläutert, warum im Rahmen der Forschung zu dieser Arbeit Interviews durchgeführt wurden (2.2.1 *„Sinn und Zweck der durchgeführten* Interviews*“*), auf welche Weise dies geschah (2.2.2 *„Durchführung“*) und wie dabei jeweils technischen Teilaufgaben (2.2.3 *„Erhobene Anforderungen“*) und theoretischen Teilaufgaben (2.2.4 *„Empirie: Informationsfluss-Muster“*) berücksichtigt wurden.

2.2.1 Sinn und Zweck der durchgeführten Interviews

Das Ergebnis der Arbeit soll eine praktische Anwendung der FLOW-Forschung sein. Frühzeitig eine Verbindung zur Praxis herzustellen war daher sinnvoll. So wurde bereits zu Beginn der Arbeit entschieden, externe Quellen in die Untersuchungen mit einzubeziehen.

Dies wurde in Form von Interviews umgesetzt. Diese halfen, theoretische Ideen und Thesen auf Relevanz und Machbarkeit zu überprüfen. Weiterhin wurden so gezielt technische und praktische Anforderungen des zu erstellenden Ratgebers erhoben. Nebenbei ergaben sich in einigen Gesprächen auch Ansätze in völlig neue Richtungen, die teilweise aufgegriffen wurden.

2.2.2 Durchführung

Gesprächspartner

Als Interviewpartner aus der Wirtschaft konnten u.a. Mitarbeiter der auf Softwareentwicklung spezialisierten Firmen sd&m, IBM, Accenture sowie Berner & Mattner gewonnen werden. Alle Gesprächspartner sind als Entwickler, Manager, Projekt- oder Teamleiter in einer bestimmten Rolle am Softwareentwicklungsprozess beteiligt. Die heterogene Zusammensetzung der Bereiche, in denen die Befragten tätig sind, bietet bewusst unterschiedliche Sichtweisen auf die Thematik. So war es möglich, neben der fachlichen und technischen Perspektive beispielsweise auch wirtschaftliche Aspekte zu berücksichtigen.

Die Gesprächspartner im Einzelnen waren, in alphabetischer Reihenfolge der Firmennamen:

Niklas Polke, Systems Analyst	Accenture Technology Solutions GmbH
Kai Tomaschewski, Senior Programmer	Accenture Technology Solutions GmbH
Michaela Bunke, Consultant	BeOne Hamburg GmbH
Gerd Brost, Softwareingenieur Test Lead	Berner & Mattner Systemtechnik GmbH
Lars Dahlke, Principal Consultant	Capgemini sd&m AG
Christian Heinecke, Principal Consultant	Capgemini sd&m AG
Dr. Berthold Lause, Geschäftsführer	HANSETRANS Holding GmbH
Norbert Pich, Senior Managing Consultant u. Senior IT Architect	IBM Deutschland GmbH
Dr. Johannes Schütt, Project Manager und Application Architect	IBM Deutschland GmbH
Prof. Dr. Jörg Hähner, Juniorprofessor für "Organic Computing"	Leibniz Universität Hannover, Fachgebiet System- und Rechnerarchitektur

Strukturierung des Vorgehens

Die Gespräche wurden in Einzelfällen telefonisch, nach Möglichkeit aber persönlich und zur besseren Vergleichbarkeit weitgehend nach einem einheitlichen Muster durchgeführt. Auf eine FLOW-Einführung folgten zunächst Fragen, die einen oder mehrere der oben genannten theoretischen Aspekte betrafen, unter anderem:

- Anwendbarkeit des FLOW-Pattern-Katalogs
- Finden einer praxisnahen Analysestufe
- Vorstellung ggf. Anpassung der Notation
- Finden neuer Muster
- Anwendung von Metriken zur Bewertung

Ein Ziel war jeweils die Einschätzung der betreffenden These durch die interviewten „Praktiker" bezüglich Praxisrelevanz und Machbarkeit. Auch der FLOW-Pattern-Katalog wurde vorgestellt und in Auszügen diskutiert. Weiterhin wurden standardmäßig bestimmte Fragen zu Anforderungen an einen Ratgeber gestellt (siehe Abschnitt 2.2.3).

Improvisation und Individualisierung der Gespräche

Auch Änderungsvorschläge und neue Ideen wurden besprochen. Bei den ersten Interviews wurde mit sehr offenen Fragen bewusst Raum für die freie Entfaltung von Ideen der Gesprächspartner gelassen. So kamen neben Verfeinerungen von bestehenden Thesen auch komplett neue Ansätze zustande. In der kurzen Einführung in FLOW wurde je nach Gegenüber eher Technisches hervorgehoben, z.B. in Bezug auf die Notation und Darstellungsmöglichkeiten, oder die inhaltlichen Analyse-Aspekte in den Vordergrund gestellt. Den Befragten wurden so FLOW-Aspekte ihres jeweiligen Wirkungsbereichs vorgestellt, was in fast allen Fällen zum Anlass genommen wurde, spontan mit dem FLOW-Gedanken zu experimentieren.[1]

Auswertung und erste Ergebnisse der Interviews

Aber auch Kritik an FLOW oder einzelnen Thesen kam zum Ausdruck. Diese wurde notiert und bei der Erarbeitung des Konzepts berücksichtigt. Es wurde beispielsweise geäußert, dass die Einführung der FLOW-Notation als weitere Kommunikationsform im Team in der Praxis eine fast unüberwindliche Hürde darstellen würde. Dies veranlasste dazu, auf die angedachten Erweiterungen der Notation zu verzichten, wie auch den Einsatz von FLOW-Diagrammen im Ratgeber neu zu überdenken. Die Darstellung des „Soll"-Ablaufs von Prozessen in Form einer „Roadmap" als Ergebnis des Ratgebers schied nach diesem Hinweis aus.

[1] Als eine Nebennotiz dieser Untersuchung kann festgehalten werden, dass FLOW von den Befragten allgemein sehr schnell verstanden und für sinnvoll befunden wurde. Die Denkweise, Prozesse aus Informationsfluss-Sicht zu betrachten, inspirierte die Gesprächspartner zu verschiedenen Gedankenexperimenten, u.a. zu *Teaming* und *Capability-Matching*. Die Notation wurde schnell angenommen und häufig mit Datenfluss-Diagrammen verglichen. Als großer Gewinn wurde der „Awareness-Effekt" bei der Beschäftigung mit der Thematik empfunden. Der FLOW-Pattern-Katalog stieß auf reges Interesse.

Mit Voranschreiten der Arbeit und zunehmendem Kenntnisstand wurde der Fokus der Fragen spezifischer. Den im Vorfeld aufgestellten Thesen bezüglich Analysestufen und Mustern (siehe Abschnitt 2.1.4) zufolge wurde von Anfang an versucht, in den Gesprächen „Standard-Situationen“ zu identifizieren, die FLOW-Mustern auf einer anderen Analysestufe als der des Pattern-Katalogs entsprechen könnten. Hierauf wird im Abschnitt 2.2.4 *„Empirie: Informationsfluss-Muster“* eingegangen.

2.2.3 Erhobene Anforderungen

Im Standard-Teil der meisten Interviews wurden bestimmte Aspekte erörtert, die grundlegende Fragen dieser Arbeit betreffen. Es handelte sich um Punkte, die zu einer konkreten Anforderungserhebung für den zu entwerfenden Ratgeber beitrugen. Unter anderem diskutiert wurden die folgenden Fragen:

- Wozu wird der Ratgeber verwendet? Wobei soll er helfen?
- Wann, in welcher Projektphase und –situation?
- Wie sehen konkrete vorstellbare Einsatzszenarien aus?
- Wer kommt als Anwender des Ratgebers in Frage?
- Was für Einwände gibt es gegen die Anwendung von FLOW in dieser Form?

Die Praxiserfahrung der Befragten und deren Einschätzung von Wichtigkeit und Machbarkeit bestimmter Aspekte war von großer Bedeutung, um den wirklichen Fokus des Ratgebers festzulegen und eine Idee zu entwickeln, wie FLOW so zu einer praktischen Anwendung geführt werden kann.

Im „offenen“ Teil der Interviews wurden neue Ansätze skizziert und mit den Gesprächspartnern diskutiert. Die Bewertung der Praxisrelevanz bestimmter Situationen und das Entwickeln praxisnaher Vorgehensweisen waren hierbei wichtig für das in dieser Arbeit vorgestellte Konzept.

2.2.4 Empirie: Informationsfluss-Muster, Vorgehensmodelle und Aktivitäten

Die Interviews sollten unter anderem dazu genutzt werden, neue FLOW-Muster in irgendeiner Form aufzufinden. Ergebnisse aus vorangehenden Gesprächen wurden dabei zur Vorbereitung und Durchführung des nächsten Interviews herangezogen. Es wurden Beispiele für typische Projektsituationen gesammelt, von denen sich herausstellte, dass sie oft bereits Muster-Charakter haben: Hier werden Situationen beschrieben, die wiederholt und oft unter ähnlichen Bedingungen auftreten. Allen Beispielen ist eine Verbindung zu FLOW gemeinsam, da Informationsflüsse in irgendeiner Form eine Rolle spielen.

Auf die unterschiedlichen Gruppen, denen die Beispiele zugeordnet werden können, wird später detailliert eingegangen (siehe untenstehende Legende). Zunächst werden sie in der folgenden Tabelle zusammengefasst und beschrieben:

*)	**Bezeichnung** des Beispiels	**Beschreibung** ggf. mit Hinweis auf Bedingungen oder mögliche Auswirkungen
▲	Fehler (Bugs)	Die am häufigsten auftretende Problematik: In der erstellten Software sind Fehler vorhanden, die die ordnungsgemäße Ausführung verhindern oder erschweren.
▲	Produktqualität	Abweichungen bei Qualitätseigenschaften der Software allgemein (z.B. bezogen auf Benutzerfreundlichkeit: Der Workflow der Oberflächen ist stockend, oder Wartbarkeit: Der Code ist nur schwer lesbar).
▲	Performance	Spezielle Produkt-Qualitätseigenschaft, die so oft genannt wurde, dass sie gesondert aufgeführt wird.
▲	„Late Changes“	Späte Änderungen im Projekt, die nicht vorhergesehen wurden. Meist bezogen auf Anforderungen. Verursachen Kosten, Zeitdruck und oft zusätzliche Probleme.
▲	Nichteinhaltung des Projektplans	Einzelne Mitarbeiter, Teams oder Arbeitsgruppen schaffen es nicht, ihre Ziele *in-time* und/oder *in-the-money* zu erreichen.
▲	Zeitprobleme	Wiederholte Nichteinhaltungen des Projektplans in einem Bereich (oder zusammenfallende in mehreren) führen dazu, dass das gesamte Projekt eine gesetzte Deadline nicht einhalten kann.
◘	Ergebnisabweichungen	Zwei oder mehrere Arbeitsgruppen beschäftigen sich unter gleichen Vorgaben und Bedingungen mit derselben oder einer ähnlichen Aufgabe, die Ergebnisse weichen aber voneinander ab.
◘	Verschätzt!	Aufwand, Zeitbedarf und/oder Kosten für eine Aufgabe wurden falsch eingeschätzt.
◘	Stagnation	Änderungsbedarf wird festgestellt, Änderungen werden gefordert, aber wiederholt nicht umgesetzt und erneut beanstandet.
◘	Prozessmüdigkeit	Vorgeschriebene Prozess-Aktivitäten (z.B. Reviews, Dokumentation) werden nur noch nachlässig durchgeführt.

*)	**Bezeichnung** des Beispiels	**Beschreibung** ggf. mit Hinweis auf Bedingungen oder mögliche Auswirkungen
◊	„Hausmeister“	Ein Mitarbeiter führt Aufgaben nur im Rahmen des „Dienst nach Vorschrift“ aus. Es kommen keine Anregungen über den Tellerrand der Stellenbeschreibung hinaus, schlimmstenfalls werden wahrgenommene Fehler „großzügig übersehen“.
◊	„ungenauer Prüfer“	Ein Prüfer kann oder will seine Pflichten nicht ordentlich ausführen, die Gründlichkeit der Prüfungen leidet.
◊	verteiltes Team	Teams oder Teilteams sind räumlich getrennt, was zum Beispiel die Kommunikation erschwert, aber u.a. auch Cliquenbildung und Konkurrenzdenken fördert.
◊	„Inseldenken“	Es gibt keinen realen Kunden, sondern Anforderungen werden „am grünen Tisch“ für fiktive Stakeholder erhoben. Besonders nichtfunktionale Anforderungen werden so leicht vernachlässigt.
◊	mangelndes Qualitätsbewusstsein	Ein Mitarbeiter oder Teilteam stimmt nicht mit dem Rest des Teams darin überein, welche Qualitätsstandards eingehalten werden sollen.
◊	unterschiedliche Entwicklungsphilosophien	Im Team herrschen verschiedene Ansichten, welches die „richtige“ Art und Weise des Vorgehens zur Entwicklung ist.
◊	fehlende (Prozess-)Transparenz	Prozesse von Teilteams (speziell von fremden Abteilungen oder Firmen, die am Projekt beteiligt sind), sind nicht im Detail bekannt und werden nicht verstanden.
♦	Eitelkeit	Ein Mitarbeiter gesteht seine Fehler nicht ein oder beharrt gegenüber anderen (auch wider besseres Wissen) auf seinem Standpunkt, um sein Gesicht zu wahren.
♦	Rechthaberei /Profilneurose	Ein Mitarbeiter ist überzeugt, dass seine Ansicht/Arbeitsweise die bessere ist und versucht diese kompromisslos durchzusetzen.

Tabelle 1: Beispiele für Projektsituationen;

*) Die Symbole in der ersten Spalte beziehen sich auf die Klassifizierung, die in Kapitel 3.1.1 *„Probleme: Unterscheidung von Problematik, Situation und Ursache“* vorgenommen wird: ▲Problematik, ◘ Situation, ◊/♦ Ursachen

Eine Auswertung dieser Beispiele führte zu einer Klassifizierung von Problemen und im Folgenden zur Rückführung auf Software-Qualitätsaspekte – und damit zum ersten Schritt der Ratgeber-Methode. Der Ansatz, der als Ergebnis der in den Interviews gesammelten Problembeispiele entwickelt wurde, wird in Abschnitt 2.3.2 *„Ratgeber-Idee: vom Problem zur Ursache“* genauer erläutert.

Für die Auffindung der für die beschriebenen Probleme ursächlichen Prozessschritte wurden Prozess- und Vorgehensmodelle untersucht. Die Relevanz gängiger Modelle sowie die Möglichkeiten zum Tailoring wurden ebenfalls in den Interviews thematisiert. Die Ergebnisse und deren Anwendung in der Ratgeber-Methode werden in Abschnitt 3.1.2 *„Schritt 2: „Flussaufwärts“ - Suche im Prozess“* diskutiert.

Ebenso wurden häufige Situationen, die zur Lösung der problematischen Aktivitäten beitragen können, mit Interviewpartnern besprochen und verschiedene Vorgehensweisen gesammelt. Diese standen Pate für die „Soll“-Aktivitäten der Ratgeber-Methode, beschrieben in Abschnitt 3.1.3 *„Schritt 3: „Beratung“ - Abfragen von Signaturen“*.

2.3 Grundlegende Ergebnisse

2.3.1 Anforderungen an den Ratgeber

Die erhobenen Anforderungen, deren Ausarbeitung und spätere Umsetzung im Ratgeber stellen die Bearbeitung der „technischen Teilaufgaben" dar, wie sie in Abschnitt 1.2.1 beschrieben werden.

Ein Ratgeber, der konkret zur Problemlösung eingesetzt werden soll, muss einige Eigenschaften in jedem Fall aufweisen, auch wenn über die Form noch nichts weiter ausgesagt wird. Gleich, ob die Umsetzung in Form einer Sammlung von Vorgehensweisen oder Methoden oder einer Software erfolgt, muss ein Ratgeber folgende Anforderungen erfüllen, um einen effektiven Nutzen zu bringen:

- Eignung
  Der Ratgeber muss in der Lage sein, das gegebene Problem zu verstehen, muss also mit allen möglichen Parametern der Problemdomäne arbeiten können und diese auszuwerten verstehen.
- Verfügbarkeit
  Der Ratgeber soll schnell (d.h. je nach Aufgabenstellung mit angemessenem Aufwand) und möglichst jederzeit zu konsultieren sein.
- Dienlichkeit
  Es soll bei der Verwendung ein effektiver Mehrwert entstehen, der erhaltene Rat soll helfen, die jeweilige Aufgabe zu bewältigen.
- Eindeutigkeit
  Ein gegebener Rat sollte klar verständlich sein, kein „Orakelspruch" (der Verweis auf eine *Best Practice* kann dagegen durchaus ein sinnvoller Rat sein).
- Individualität
  Die Situation soll möglichst individuell behandelt werden. Ähnlich wie bei der Eindeutigkeit helfen allgemeine Weisheiten wenig, wenn es eine konkrete Situation zu bewältigen gilt.

Um diese Punkte in einer konkreten Instanz zu implementieren sind einige Rahmenbedingungen zu klären:

- Aufgabendefinition
  Die Situation(en), in der der Ratgeber eingesetzt werden soll und das konkrete Ziel der Verwendung (z.B. Planung, Problemlösung, Optimierung) müssen definiert werden.
- Rat-Typen
  Ratschläge können explizite Arbeitsanweisungen sein, wie man sich in einer Situation verhalten soll, aber auch „nur" aus dem Hinweis auf ein Problem oder eine Gefahr bestehen, deren man sich vor der Konsultierung des Ratgebers nicht bewusst war (*Awareness*). FLOW-Muster ermöglichen hierbei den Vergleich „guter" Situationen (*Best Practices*) mit der realen Ausführung, was das Aufzeigen von Abweichungen und Vorschlagen von Änderungen zulässt.

Als mögliche Benutzer des Ratgebers (oder: Ratsuchende) wurden allgemein diejenigen Personen identifiziert, die „im Rahmen ihrer Tätigkeit Entscheidungen im Softwareentwicklungsprozess treffen müssen und ein Interesse daran haben, dass Ihre Entscheidungen keine Probleme für das Projekt zur Folge haben“. Die Benutzer verwenden den Ratgeber dabei vornehmlich zur Analyse, um herauszufinden, warum ihre oder die Entscheidungen anderer eben *doch* zu Problemen geführt haben, um die (verborgenen) Ursachen zu finden, zu verstehen und künftig vermeiden zu können. Die Ratsuchenden sind keine FLOW-Experten, der Informationsfluss-Gedanke kann ihnen völlig fremd sein. Ein Nebeneffekt einer erfolgreichen Konsultierung des FLOW-Ratgebers soll eine gewisse *FLOW-Awareness* sein, die durch Erläuterungen der Zusammenhänge und Erklärungen der Arbeitsweise des Ratgebers erreicht wird.

Der Modellbegriff im Ratgeber

Um die Aufgabe zu erläutern, die der Ratgeber übernehmen soll, wird an dieser Stelle kurz auf die Modellbildung eingegangen, die hierfür nötig ist.

Modelle dienen zur Abstraktion wie auch zur Konkretisierung, die Modellierung bietet die Möglichkeit, etwas über eine modellierte Situation zu lernen. Um in der Lage zu sein, Verbesserungen vorzuschlagen, muss es daher Ziel sein, den Ratgeber präskriptive Modelle anfertigen zu lassen, die Handlungsweisen vorschreiben (im Sinne der *Design-Patterns*). Der Weg dahin führt über deskriptive Modelle, die „Ist“-Situationen beschreiben und mittels Pattern-Matching und ähnlichen Methoden FLOW-Muster finden.

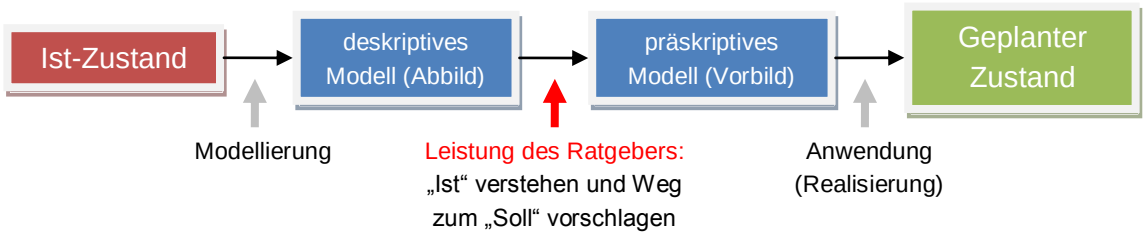


Abbildung 1: Verwendung von Modellen bei der Konzeptionierung des Ratgebers

Normalerweise wird auf diese Weise ein „exploratives“ Modell beschrieben (vgl. [LUDMOD02]). Am Modell kann dabei probiert werden, wie sich Änderungen auswirken, man überträgt die Änderungen aufs Original, wenn der Versuch am Modell erfolgreich war. Der zu erstellende Ratgeber stellt allerdings ein sogenanntes „Expertensystem“ dar. Hierbei muss nicht probiert werden, sondern bereits bekannte *Best Practices* können direkt herangezogen werden, um Änderungen vorzuschlagen. Diese können an einem präskriptiven Modell (bestenfalls in Form einer FLOW-Darstellung) der erwünschten Situation erläutert werden.

2.3.2 Ratgeber-Idee: vom Problem zur Ursache

Die Entwicklung der Ratgeber-Idee und die Untersuchung aller dafür notwendigen Hintergründe sowie die spätere Umsetzung im Ratgeber stellen die Bearbeitung der „theoretischen Teilaufgaben" dar, wie sie in Abschnitt 1.2.2 beschrieben werden.

Zunächst wurde als Folge der Interviews eine weitere Eingrenzung der Aufgaben des Ratgebers vorgenommen. Als „Projektsituationen", in denen der Ratgeber zur Anwendung kommen soll, wurden bestimmte Problemlagen gewählt. Probleme liegen hierbei grundsätzlich vor, wenn eine Abweichung von „Soll"- und „Ist"-Zustand festgestellt wird und sind zu unterscheiden von Aufgaben. Der Ratgeber soll dementsprechend nicht eingesetzt werden, um bestimmte Aufgaben anzugehen, wie das Besetzen von Teams oder das Planen von bestimmten Vorgängen. Vielmehr soll, wenn ein Problem festgestellt wird, dazu geraten werden, wie FLOW zur Analyse eingesetzt werden kann, um in einer Situation zur optimalen Reaktion zu gelangen.

Im Kapitel 2.1.4 *„Neue FLOW-Muster und Abstraktionsebenen"* wurden bereits Einwände gegen die praktische Anwendbarkeit der im FLOW-Pattern-Katalog vorgestellten Art von Mustern diskutiert. Ebenso wurde die Vermutung vorgestellt, es müsse andere „Analysestufen" oberhalb der abstrakten FLOW-Sicht gefunden werden, um mögliche Anwender an die FLOW-Pattern heranführen zu können.

Ausgehend von diesen Annahmen wurde die Suche nach neuartigen FLOW-Mustern in die Richtung von „typischen Problem-Situationen" gelenkt. Die Beschäftigung mit den gesammelten Praxisbeispielen (siehe *Tabelle 1* in Abschnitt 2.2.4, oben) bestätigte die Vermutung, dass diese tatsächlich auf verschiedenen Analysestufen einzuordnen sind (siehe Abschnitt 3.1.1, unten).

Dies wird im ersten Schritt des Ratgebers für die angedachte Heranführung an die FLOW-Thematik genutzt. Der Ratgeber hilft auf diese Weise, die Brücke zwischen den offensichtlichen *Symptomen* eines Problems und dessen verborgenen (Informationsfluss-)*Ursachen* zu schlagen.

Die praktische Anwendung von FLOW wird so ermöglicht, unter anderem werden die Muster des FLOW-Pattern-Katalogs anwendbar gemacht. Aber auch weitere Erkenntnisse der FLOW-Forschung werden im Rahmen der Methode genutzt, um eine Analyse des Softwareentwicklungsprozesses in Problemsituationen durchzuführen.

3 Ratgeber

Als Folge der betriebenen Forschung konnten die Ergebnisse zur Entwicklung eines Ratgeber-Konzepts herangezogen werden. Als Ziel dieser Arbeit soll dieses Konzept ausgearbeitet werden, so dass deutlich wird, wie eine entsprechende Ratgeber-Methode funktioniert. Die entsprechenden Hintergründe und Begründungen werden in Abschnitt 3.1 *„Das theoretische Konzept hinter dem Ratgeber“* erläutert. Weiterhin ist eine beispielhafte Implementierung gefordert, die in Abschnitt 3.2 *„Prototyp“* die Umsetzung der Methode demonstriert.

3.1 Das theoretische Konzept hinter dem Ratgeber

Der „weise Mann“ als Grundlage für den Aufbau

Als Gedankenmodell, um den Vorgang der Beratung zu veranschaulichen und um die Fähigkeiten des Ratgebers vorstellbar zu machen, wurde die Metapher vom „weisen Mann“ eingeführt. Dieser verfügt über viel (FLOW-)Fachwissen und Erfahrung. Er zeichnet sich durch routiniertes Vorgehen und präzise Anweisungen aus. Seinen Aussagen wird vertraut, und jeder Projektleiter würde sich wünschen, solch einen Berater „im Schrank stehen“ zu haben. Bei genauerem Hinsehen ist seine Arbeitsweise aber natürlich keine Zauberei, sondern Wissenschaft, und seine weisen Ratschläge sind keine Wunder, sondern Ergebnis genauer, erfahrungsbasierter Analysen.

Um den Ablauf einer Beratung nachzustellen, wurde auch in den durchgeführten Interviews bei Rollenspielen auf den „weisen Berater“ zurückgegriffen, wobei diese Rolle zwar über allgemeines Expertenwissen verfügt, der Ratgeber jedoch über die konkrete Situation, in der er konsultiert wird, anfangs nichts wissen kann.

Zunächst würde ein Berater daher versuchen, die Situation zu erfassen. Sobald er die Lage verstanden hat und – durch Einbringung seiner eigenen Erfahrung – vielleicht sowohl bereits ahnt, wonach er sucht, und ebenso, was ausgeschlossen werden kann, folgt eine entsprechend zielgerichtete Analyse der Arbeitsweise des Ratsuchenden. Findet der Berater hier Ansatzpunkte zur Verbesserung, wird er davon ausgehend eine Beratung durchführen.

Diesem Vergleich folgend sieht das entwickelte Konzept drei Phasen vor. Die erste dient dazu, die Situation des Ratsuchenden einzuordnen. Hierzu ist es notwendig, die Situation zunächst zu verallgemeinern, damit unter den vielen tausend Möglichkeiten eine erste „Orientierung“ möglich ist. Anschließend wird der Prozess untersucht, den der Ratsuchende zur Softwareentwicklung anwendet, dies geschieht z.B. im Wasserfallmodell „flussaufwärts“, wie auch der zweite Schritt überschrieben ist. Die gesammelten Informationen werden in der letzten Phase zur „Beratung“ genutzt.

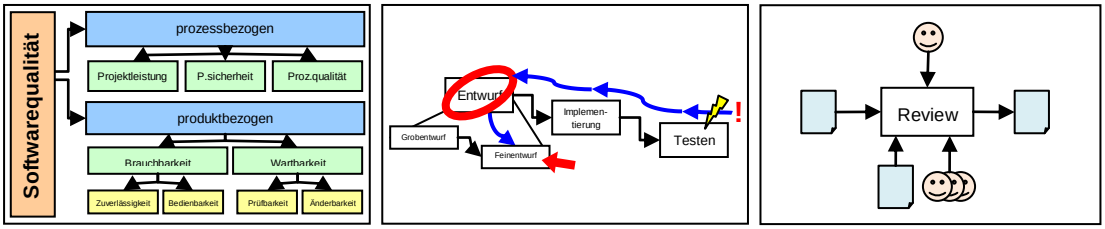


Abbildung 2: Übersicht der 3 Ratgeber-Phasen – Orientierung, Flussaufwärts, Beratung

Im Folgenden werden diese drei Haupt-Schritte des Ratgebers erläutert. Hierbei wird darauf eingegangen, wie im Rahmen dieser Arbeit entschieden wurde, dass der jeweilige Schritt notwendig wie er uns dem Ziel eines sinnvollen Ratschlags näherbringt.

In 2.3.2 *„Ratgeber-Idee: vom Problem zur Ursache“* wird erläutert, dass als „Situationen“, in denen der Ratgeber helfen soll, nicht das Angehen von Aufgaben gewählt wurde, sondern das Lösen von Problemen im Fokus der Anwendung steht. Gemeinsam mit Expertenwissen, das im Ratgeber ohne Zweifel zur Anwendung kommt (in der Metapher des „weisen Mannes“ entspricht es dessen Erfahrung), würde bei einer Umsetzung in Form einer Software ein Expertensystem entstehen. Die hier beschriebene Methode liefert Ansätze, um die hierfür notwendige „künstliche Intelligenz“ aufzubauen, da das Vorgehen im Prinzip aus einer Verknüpfung von Fragen an den Ratsuchenden (oder „Benutzer“) und einer Auswertung der Antworten besteht, umsetzbar z.B. in Form eines Entscheidungsbaums.

3.1.1 Schritt 1: „Orientierung“ - Klassifikation der Problematik

Zusammenhang von Problemen und Qualitätsaspekten

Probleme: Unterscheidung von Problematik, Situation und Ursache

Im Verlauf der geführten Gespräche wurde nach Mustern für „Problem-Situationen“ gesucht. Um die Interviewpartner nicht einzuschränken oder zu beeinflussen, sondern möglichst realistische und praxisrelevante Beispiele sammeln zu können, waren die entsprechenden Fragen bewusst offen formuliert. Alle Beispiele wurden zunächst unsortiert gesammelt und auch mit den Befragten diskutiert. Die Auswahl der für diese Arbeit verwendeten relevanten Beispiele findet sich in Abschnitt 2.2.4 *„Empirie: Informationsfluss-Muster, Vorgehensmodelle und Aktivitäten“*. Dort sind diese Beispiele auch im Einzelnen erläutert.

Untersuchungen der Praxis-Beispiele machten deutlich, dass die genannten „Problem-Situationen“ in eine Struktur einsortiert werden können, die die „Nähe“ zur Ursache des jeweiligen Problems darstellt.

Einige Beispiele befinden sich nämlich „an der Oberfläche“ und treten augenscheinlich und offensichtlich zu Tage, z.B. *Performance-Probleme*, *Programmfehler (Bugs)* oder *Zeitknappheit* und sogenannte *„Late Changes“*.

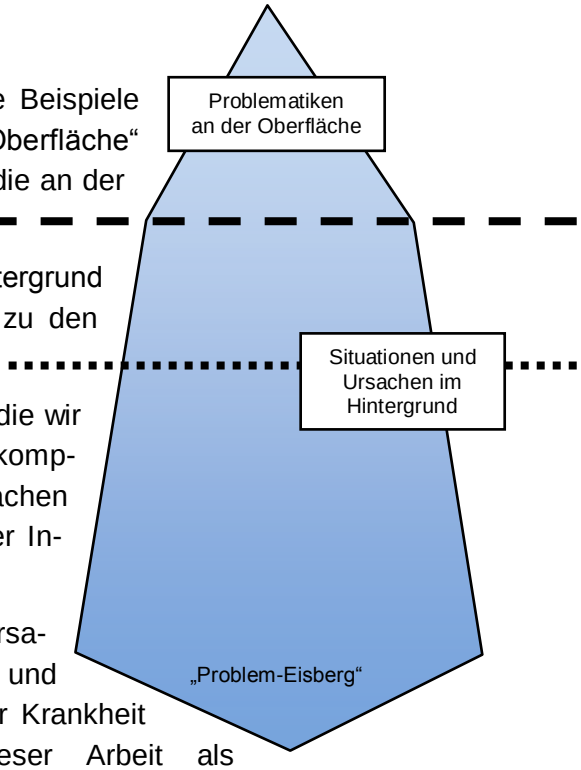
Andere der genannten Beispiele bezeichnen weniger einen tatsächlichen „Defekt“ oder einen Fehler, sondern mehr eine vorliegende Situation im Hintergrund, z.B. *Prozessmüdigkeit* oder *Stagnation*.

Wieder andere der gesammelten Problem-Situationen sind selbst Ursachen für spätere Problem-Situationen. Diese Beispiele befinden sich meist auf einer psychologischen Ebene und beziehen sich auf das Verhalten von Personen und Gruppen. Genannt wurden hier *„Eitelkeit“*, *„Rechthaberei“*, aber auch *„abweichende Entwicklungsphilosophien“* zwischen Teams und *„mangelndes Qualitätsbewusstsein“*.

Die Feststellung, dass diese Unterscheidung der gesammelten Beispiele vorliegt, und die Notwendigkeit, die Beispiele entsprechend auch sprachlich in der Diskussion unterscheiden zu können, führten zur Einführung vom Bild des „Problem-Eisbergs“, um die vorgefundene Hierarchie zu erläutern.

Die Eisberg-Metapher

Der oben genannten Feststellung folgend, dass einige Beispiele offenbare Schwierigkeiten beschreiben, die „an der Oberfläche“ auftreten, werden diese als *Problematiken* bezeichnet, die an der Spitze des Eisbergs zu finden sind.



Unter der „Wasseroberfläche“, also versteckt im Hintergrund dieser Problematiken, können *Situationen* liegen, die zu den Problematiken führen.

Problematiken wie Situationen selbst haben *Ursachen*, die wir am Fuße des Problem-Eisbergs finden und die beliebig kompliziert und detailliert sein können. Auch haben viele Ursachen keinen direkten Bezug zu FLOW, auch wenn sie später Informationsflüsse beeinflussen.

Die Gesamtheit der Verkettung einer oder mehrerer Ursachen, und der Situationen, zu denen sie führen können, und einer Problematik, in der sie sich wie Symptome einer Krankheit niederschlagen, bezeichnen wir im Rahmen dieser Arbeit als das *Problem*.

Nach dieser Unterscheidung und Begriffsklärung konzentriert sich die Arbeit im Folgenden auf bestimmte Bereiche. Nachdem alle Ebenen diskutiert wurden konnte ein Fokus festgelegt werden: Außer Acht gelassen werden hier demnach diejenigen Ursachen auf der feingranularsten, untersten Ebene, die oftmals ins Psychologische spielen, wie *Profilneurose*, *Eitelkeit* und *Existenzängste*. Sie sind zu allgemein und fachfremd für Betrachtungen in diesem Rahmen. Andere, wie die Ursachen-Muster *„Hausmeister“*, *„ungenauer Prüfer“* und *„Inseldenken“*, sind durchaus für die FLOW-Diskussion von Interesse. Sie liegen „weiter oben“ im Eisberg und näher an den Situationen, die als wiederkehrende Muster im Folgenden genauer betrachtet werden (siehe Abschnitt *„Ursachen- und Situations-Muster“*, unten).

Besondere Beachtung finden jedoch zunächst die Problematiken, da sie „an der Oberfläche“ der Probleme die „Spitze des Eisbergs“ bilden und als Symptome, die am vorliegenden Problem zuerst wahrgenommen werden, den ersten Ansatzpunkt für die methodische Untersuchung der Ursachen bieten.

Erste Unterscheidung von Problematiken

Sobald die obenstehende Definition von Problematiken vorgenommen war, wurde in den Interviews herausgearbeitet, dass diese sich danach unterscheiden lassen, welche Aspekte sie betreffen. Einige, wie *Performance-Schwierigkeiten* oder *Programmfehler (Bugs)*, beziehen sich eindeutig auf das Produkt. Andere betreffen den Entwicklungsprozess an sich, wie z.B. Probleme mit der Einhaltung von Deadlines (*Zeitknappheit*) oder Schwierigkeiten durch immer wieder erst spät auftretende Änderungen (*Late Changes*). Dies wird in der untenstehenden Darstellung veranschaulicht, die eine Zuordnung und Sortierung der gesammelten Beispiele zeigt.

Zuordnung der Situations-Beispiele zu den Ebenen des Eisbergs

Wie in Abschnitt 3.1.1 dargestellt lassen sich die gesammelten Beispiele den Ebenen des „Eisbergs“ zuordnen. Exemplarisch wurden in der Abbildung auch Verbindungen eingezeichnet, die Zusammenhänge zwischen Ursachen, Situationen und daraus resultierenden Problematiken deutlich machen. Die eingezeichneten Beziehungen ergeben sich dabei aus den konkreten Beispielen, die untersucht wurden, es sind noch weitere Zusammenhänge denkbar. Es zeigt sich aber bereits, dass insbesondere einige Ursachen sehr stark verknüpft sind. In der Darstellung wurde daher bei der Darstellung der Ursachen bereits unterschieden, und die grau gezeichneten Beispiele werden im Folgenden nicht betrachtet. Sie sind zu detailliert, größtenteils psychologisch begründet und können fast in jedem Fall irgendwie mit einem Projekt in Verbindung gebracht werden. Weiterhin sind die verbleibenden Ursachen-Beispiele bereits als Muster anzusehen (siehe unten: *„Ursachen-* und Situations-Muster*“*), da sie meist eine komplexere Vorbedingung der Problematik beschreiben.

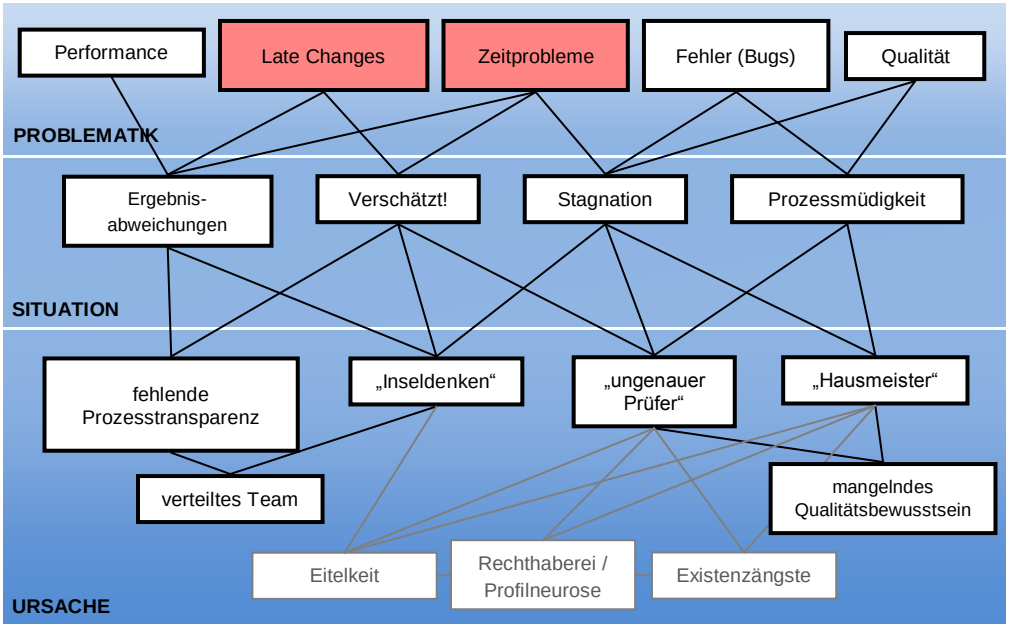


Abbildung 3: Einordnung und exemplarische Verknüpfung der Beispiele
(Erläuterungen siehe Liste in Kapitel 2.2.4)

Auf der Ebene der Problematiken zeigt sich eine weitere Unterscheidung, die im vorangegangenen Abschnitt bereits erläutert wurde. Die Beispiele *Late Changes* und *Zeitprobleme* sind farblich hervorgehoben, da sie sich von den anderen Beispielen auf dieser Ebene unterscheiden: Sie beziehen sich auf den Prozess der Software-Erstellung, nicht auf die Software als Produkt. Dieser Aspekt tritt hier zum ersten Mal in Erscheinung und wird im folgenden Abschnitt zum Anlass für eine genauere Diskussion genommen.

Zusammenhang mit Qualitätsaspekten

In der Unterscheidung von prozess- und produktbezogenen Problematiken wurde erstmals der Zusammenhang zwischen den Eigenschaften der betrachteten Problematiken mit Softwarequalitätsaspekten deutlich, wie sie z.B. in [LUD06] aufgeführt werden:

Abbildung 4: Unterscheidung von Software-Qualitätseigenschaften

Nicht nur ob sich Problematiken auf den Entwicklungsprozess oder das entwickelte Produkt beziehen unterscheidet sie, sondern auch der betroffene Qualitätsaspekt.

Auf den zweiten Blick ist dieser Zusammenhang auch nicht weiter verwunderlich, bedeuten die meisten Problematiken doch einfach Mängel im „Ist“ bezüglich eines bestimmten Qualitätsaspekts, eine Abweichung zwischen dem gewünschten „Soll“ und der erreichten Umsetzung. Jede Problematik, die an der „Wasseroberfläche“ als Spitze des Eisbergs zum Vorschein kommt, lässt sich somit einer Qualitätseigenschaft zuordnen.

Diese Möglichkeit zur Klassifikation der Problematik ist für den Ratgeber von enormer Bedeutung. Zunächst trägt sie dazu bei, die Situation überhaupt einordnen zu können. Es können tausende Situationen auftreten, die natürlich unmöglich einzeln und vollständig von einem Ratgeber erfasst und besprochen werden können. Die Verbindung mit Qualitätseigenschaften liefert hier eine sinnvolle Abstraktion und ermöglicht eine zwar noch allgemeine, aber doch schon zielgerichtete Bewertung.

Der Metapher des „weisen Manns“ (siehe oben) folgend soll zunächst versucht werden, eine grobe Einschätzung vorzunehmen, welcher Art das Problem ist. Hierbei ist die Bestimmung des betroffenen Qualitätsmerkmals eine große Hilfe. Allein die erste Unterscheidung: handelt es sich um eine Prozess- oder Produkteigenschaft, die betroffen ist? lässt schon eine Vielzahl von Möglichkeiten ausschließen und hilft bei der weiteren Beratung.

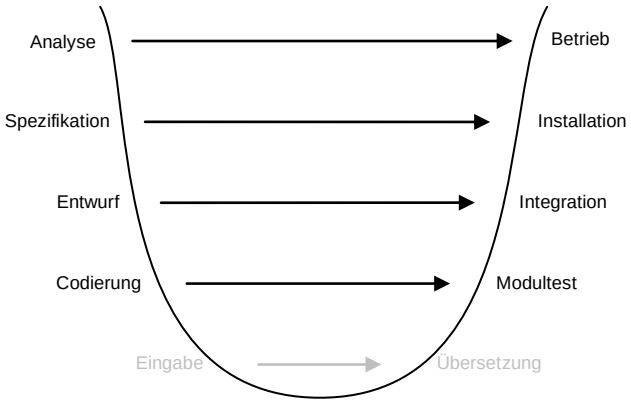


Abbildung 5: Die „Badenwannenkurve“ nach Ludewig – eine „etwas vereinfachte, dafür runde Darstellung des V-Modells“. Die Badewannenkurve stellt Verbindungen zwischen Prozessphasen her bezüglich des Auftretens von Problematiken und deren Ursachen

Die erste konkrete Hilfe besteht darin, dass unterschiedliche Qualitätsaspekte in unterschiedlichen Phasen des Projekts definiert werden und die Ursache eines Problems entsprechend eingegrenzt werden kann. Die sogenannte „Badewannenkurve“ [LUD06] beschreibt diesen Zusammenhang, der in gewisser Weise auch Hintergedanke des V-Modells ist: Hier werden Überprüfungen bestimmter Aktivitäten zu Projektbeginn in bestimmte Phasen gegen Projektende gelegt, da Entstehung und Ursachen bestimmter Probleme und Fehler in Zusammenhang stehen mit deren „Auftauchen“ bzw. Bemerken zu späteren Zeitpunkten.

So werden Bugs, die erst recht spät (bei der Implementierung) „eingebaut“ werden, beim Testen gefunden, nicht erhobene oder falsch umgesetzte Anforderungen fallen unter Umständen erst dem Kunden bei Akzeptanztests auf. Diese Zusammenhänge können genutzt werden, um vom betroffenen Qualitätsaspekt zu schließen auf die Phase, in der die Ursache zu suchen ist. Eine genaue Betrachtung der Zusammenhänge ergab, dass der Zeitpunkt, zu dem eine Problematik entdeckt wird, von Bedeutung für Rückschlüsse auf die Ursache ist. Am Beispiel von Qualitätsaspekten bezüglich der Brauchbarkeit wurden verschiedene Beispiele untersucht (siehe *Tabelle 2*).

Es zeigt sich, dass solche Qualitätseigenschaften weitgehend der Badewannenkurve folgen, die genau definierbar sind (wie Genauigkeit, Ausfallsicherheit, Effizienz etc.). Dies liegt daran, dass diese meist bewusst berücksichtigt und in den Anforderungen aufgeführt werden und entsprechend testbar sind. Fehler, die bei Modultests gefunden werden, sind sehr wahrscheinlich nicht auf Schwächen der Anforderungsanalyse zurückzuführen, denn dies würde bedeuten, dass die entsprechende Qualitätseigenschaft nicht als Anforderung erfasst wurde, somit würden in diesem Fall gar keine entsprechenden Modultests existieren. Im Entwurf und noch am ehesten in der Implementierung selbst sind hier die Ursachen zu vermuten.

Qualitäts-bereich	enthaltene Qualitätsmerkmale	*Problematik* wurde entdeckt bei…	*Ursache* wird vermutet bei… …Analyse	…Spezifikation	…Entwurf	…Codierung
Zuverlässigkeit	• Korrektheit • Genauigkeit • Ausfallsicherheit	…Betrieb/Abnahmetests				
		…Installation/Systemtests				
		…Integration				
		…Modultests				
Nützlichkeit	• Effizienz • Sparsamkeit • Leistungs-vollständigkeit	…Betrieb/Abnahmetests				
		…Installation/Systemtests				
		…Integration				
		…Modultests				
Bedienbarkeit	• Konsistenz • Verständlichkeit • Einfachheit	…Betrieb/Abnahmetests				
		…Installation/Systemtests				
		…Integration				
		…Usability-Tests				

Tabelle 2: Zusammenhänge zwischen Entdeckung einer Problematik und möglicher Lokalisation der Ursache, unter Berücksichtigung des betroffenen Qualitätsaspekts; je dunkler ein Feld der Matrix gefärbt ist, desto wahrscheinlicher ist ein Zusammenhang – im Bereich der Bedienbarkeit zeigen sich deutliche Abweichungen von der sonst bestätigten Aussage der „Badewannenkurve“

Aspekte, die oft als selbstverständlich vorausgesetzt und nicht ausdrücklich formuliert werden (wie Konsistenz, Verständlichkeit und Einfachheit) weichen dagegen vom „Badewannen-Modell“ ab: Probleme, die erst bei der Inbetriebnahme aufgefunden werden, können hier einmal in späteren Prozessschritten verursacht worden sein, da die Umsetzung der Anforderungen „verschleppt“ wurde. Zum anderen können in diesem Bereich auch bei Entwurf und Implementierung Fehler entstanden sein, die nicht früher entdeckt werden konnten, weil in vorangehenden Prozessschritten keine spezifischen Test-Aktivitäten vorgesehen wurden. Zusätzlich werden in Usability-Tests, die hier auch als Einzeltests mit ausgewählten Personen oder Personengruppen durchgeführt werden, Problematiken aufgedeckt, die Aspekte der Bedienbarkeit betreffen können, die zuvor nie erfasst wurden.

(Anmerkung: Es wurden nur Qualitätsaspekte der Gruppe „Brauchbarkeit“ betrachtet. An dieser Stelle wurde allerdings die Feststellung gemacht, dass die Qualitätsaspekte eines Bereichs bei einer entsprechenden Zuordnung offenbar zusammengefasst betrachtet werden können. Bei einer Einordnung weiterer Aspekte aus dem Bereich „Wartbarkeit“ und der

übergeordneten Gruppe „Prozessqualität“ kann das Rückschließen auf Zusammenhänge von Entdeckung und Ursache so möglicherweise vereinfacht werden.)

Wie wird mit diesen Zusammenhängen umgegangen und was bedeuten die gemachten Feststellungen für die Arbeitsweise des Ratgebers? Es ist zu betonen, welch wichtige Rolle für die gesamte Methode das Stellen der richtigen Fragen spielt (auf diesen Aspekt wird auch unten in Abschnitt 3.1.3 *„Mit Hilfe von Fragen zu Darstellung und Analyse der Ist-Situation“* eingegangen). Nach dem Zeitpunkt der Entdeckung einer Problematik zu fragen und den betroffenen Qualitätsaspekt zu bestimmen, ermöglicht an dieser Stelle bereits, das Problem „einzugrenzen“ und Vermutungen über seinen Ursprung anzustellen. Zudem können so Hintergründe in Erfahrung gebracht werden, die es erlauben, bestimmte Prozessschritte in den Mittelpunkt der weiteren Analyse zu rücken bzw. davon auszuschließen.

Ursachen- und Situations-Muster

Wie oben bereits angedeutet enthalten die zusammengestellten Beispiele bereits Muster. Dies wurde deutlich beim Sammeln der Beispiele, viele Beschreibungen wurden schon während der Interviews in Metaphern „verpackt“, die sich in den Gesprächen schnell ergaben. Auch mehrfach auftretende ähnliche Beschreibungen deuten darauf hin, dass es sich um Muster handelt.

Diese Erkenntnis zeigte, dass hier eine sinnvolle *Analysestufe* (vgl. Abschnitt 2.1.4) gefunden war, um darauf einen Einstieg für die Ratgeber-Methode aufzubauen: Die gesammelten, „muster-gütig“ in Erscheinung tretenden Situationen bieten eine Gelegenheit, den Ratsuchenden bei seinem Problem „in der Realität abzuholen“ und ihn bei der Analyse bis hin zu einer detaillierteren FLOW-Ebene zu begleiten, wie es zuvor gefordert wurde.

Wie aus den bisherigen Informationen Rückschlüsse auf das weitere Vorgehen gezogen werden können, veranschaulicht die folgende Grafik:

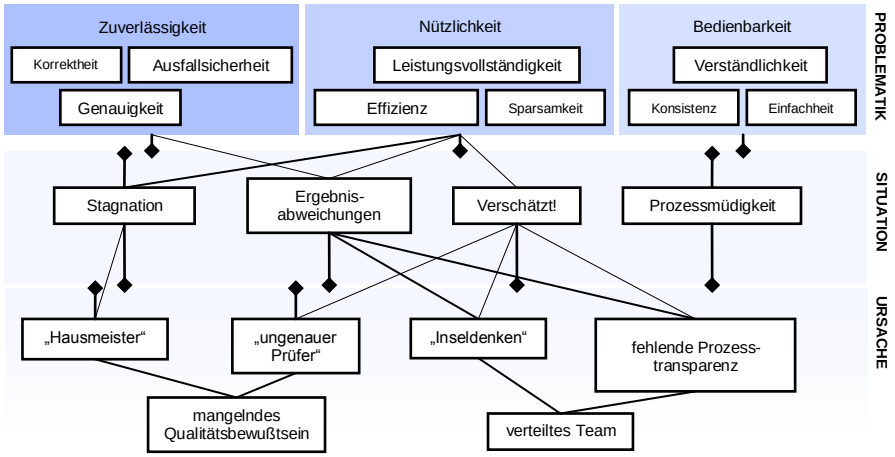


Abbildung 6: Zusammenhänge zwischen Qualitätsbereichen und Situations- bzw. Ursachen-Mustern

Nachdem die Klassifizierung der vorliegenden Problematik geholfen hat, die betroffene Qualitätseigenschaft und damit die grobe Richtung des Problems einzuordnen (Verbindung Problematik → Qualitätsbereich), können Situationen benannt werden, die im Vorfeld der Problematik typisch sind (Verbindung Qualitätsbereich → Situationsmuster) und auf Ursachen hindeuten, die wiederrum Anlass für das Auftreten dieser Situationen sein können (Verbindung Situationsmuster → Ursache). So kann möglicherweise bereits eine Identifizie-

rung der tatsächlichen Ursache erreicht werden, auf jeden Fall aber eine weitere Einordnung und damit eine wichtige Vorbereitung des weiteren Vorgehens: Situations-Muster weisen den Weg in eine bestimmte Richtung für die Folgende Untersuchung, da sich jedes Muster im *Kontext bestimmter Aktivitäten* bewegt (Verbindung Situationsmuster → Aktivitätskontext). Dies wird bereits an den folgenden vier Bespielen deutlich:

Situations-Muster	Aktivitäts-Kontext	Erläuterung
Stagnation	Prüfungen	Als *Stagnation* wird das wiederholte Anmahnen von Fehlern bezeichnet, die immer wieder vorgelegt und nicht bearbeitet werden. Das Feststellen der Fehler setzt Prüfungsaktivitäten voraus. Diese *Prüfungen* sind zu untersuchen, so könnten zum Beispiel die erstellten Änderungsanweisungen missverständlich sein, oder vielleicht werden bei der Durchführung selbst grundsätzliche Fehler gemacht.
Ergebnisabweichungen	Begegnungen	Als *Ergebnisabweichungen* wurden Situationen zusammengefasst, in denen verschiedene Teams oder Mitarbeiter unter gleichen Bedingungen unterschiedliche oder nicht zueinander passende Ergebnisse produzieren. Dies geschieht, wenn „aneinander vorbei" geredet wird und keine Abstimmung stattfindet. Alle (als *Begegnungen* zusammenzufassenden) Aktivitäten, bei denen Mitarbeiter zur Abstimmung zusammenkommen sind daher auf korrekte Durchführung zu untersuchen, beispielsweise Planungssitzungen, Reviews, Änderungsausschüsse.
Verschätzt!	Beurteilungen	Die Situation *Verschätzt!* liegt vor, wenn die Einschätzung eines Zeit- oder Kosten-Aufwands falsch eingeschätzt wurden. Ein Team oder einzelne Mitarbeiter haben die Lage nicht richtig beurteilt. Unabhängig davon, ob ein Team in einer Sitzung oder einzelne Verantwortliche die *Beurteilung* durchgeführt haben ist die jeweilige Aktivität zu überprüfen.
Prozessmüdigkeit	Bürokratie	*Prozessmüdigkeit* äußert sich in Aktivitäten, die nur oberflächlich und „der Form halber" durchgeführt werden und ist überall zu finden, wo Vorschriften als *Bürokratie* aufgefasst werden können. Sowohl die bereits erwähnten Prüfungen und Begegnungen gehören dazu, als auch vorgegebene Prozesse zur Dokumentation und ähnliche Anweisungen.

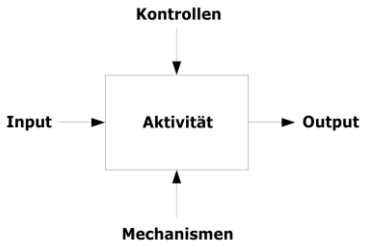
Tabelle 3: Zusammenhänge zwischen Situationsmustern und Aktivitätskontexten;
diese Schlussfolgerungen werden im dritten Schritt der Ratgeber-Methode genutzt, wenn die hier identifizierten Aktivitäten analysiert werden (3.1.3 *„Schritt 3: „Beratung" - Abfragen von Signaturen*").

3.1.2 Schritt 2: „Flussaufwärts“ - Suche im Prozess

Nachdem zuvor zumindest die Problematik klassifiziert wurde und somit eine Eingrenzung der unüberschaubaren Menge von Ansatzmöglichkeiten vorgenommen wurde, dient der nächste Schritt wieder dazu, die Beratung möglichst nah an die individuelle Situation des Benutzers heranzuführen. Dies erfolgt durch die Vorgabe einer weitgehend allgemeinen Methode, durch die der Benutzer bei der Untersuchung seines Entwicklungsprozesses angeleitet wird. Hierbei ist es nicht notwendig, dass dem Ratgeber der vorliegende Prozess genau bekannt ist. Es werden allgemeingültige Anweisungen zum Vorgehen gegeben, die der Benutzer leicht an seine individuelle Situation anpassen kann. Ziel dieses Schrittes ist es, systematisch diejenige Aktivität zu identifizieren, die mit der größten Wahrscheinlichkeit die Ursache für das aufgetretene Problem birgt.

Die Grundregeln für das Vorgehen sind dabei einfach und leicht nachzuvollziehen. Aufbauend auf der *Structured Analysis and Design Technique* (vgl. [SADT]) wird der Softwareentwicklungsprozess als eine Aneinanderreihung von Funktionen oder Aktivitäten betrachtet.

Diese Aktivitäten hängen zusammen über die In- und Outputs, die in ihnen verarbeitet bzw. erstellt werden.



Zunächst sind die Aktivitäten gleichzusetzen mit den Schritten des verwendeten Vorgehensmodells.

Wenn der Output eines dieser Schritte als fehlerhaft erkannt wurde, ist die einfache Frage, die der Benutzer mit Blick auf die entsprechende Aktivität beantworten muss, ob es möglich oder sogar wahrscheinlich ist, dass der Fehler auch bereits im Input der Aktivität enthalten war.

Entscheidet der Benutzer, dass dies wahrscheinlich ist, so kann der aktuelle Schritt nicht die Ursache des Fehlers sein, und der vorangehende Schritt ist stattdessen zu untersuchen.[2]

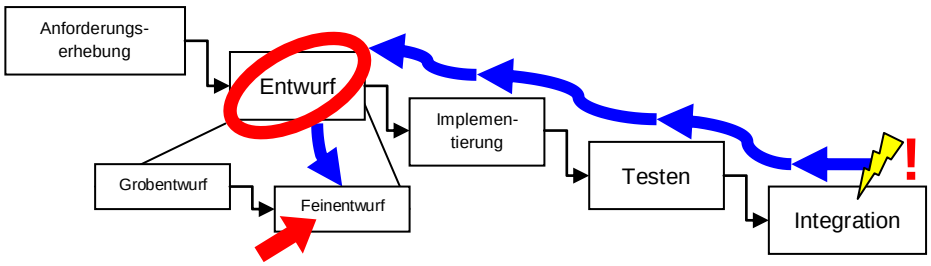


Abbildung 7: Flussaufwärtsschwimmen im Prozess und Verfeinerung einer Aktivität

Ist dies aber auszuschließen oder nicht sicher, wird die Aktivität „verfeinert“ untersucht. Es wird die Detaillierungs-Notation von IDEFØ verwendet, um diesen Vorgang zu erläutern. Die Aktivität wird hierbei in einzelne „Unter“-Aktivitäten aufgeteilt, die wiederum über die In- und Outputs in Verbindung stehen (wobei der erste Input und der letzte Output mit denen der

[2] Dies wurde in Interviews auch als das „Weiterschieben des Schwarzen Peters“ bezeichnet. Im Wasserfallmodell dargestellt liegt es nahe, dieses Vorgehen mit „Flussaufwärtsschwimmen“ zu vergleichen. Beide Metaphern wurden zur Beschreibung und als Merkhilfen beibehalten.

verfeinerten Aktivität übereinstimmen). Die Suche wird auf dieser Ebene fortgesetzt, entsprechend der „Flussaufwärts“-Metapher beginnend beim letzten Schritt der Verfeinerung.

Es ist möglicherweise nötig, die In- und Outputs genauer zu benennen, um sie von denen zu unterscheiden, die die zur Verfeinerung herangezogenen Aktivitäten verbinden. Der Verfeinerungs-Vorgang kann wiederholt durchgeführt werden, immer motiviert durch die Hinterfragung von In- und Output der einzelnen Aktivitäten auf Behaftung durch Fehler. Wenn keine weitere Verfeinerung mehr möglich ist, wurde die Aktivität gefunden, die den Ursprung des Fehlers beinhaltet (Details und Einschränkungen siehe Prototyp, Abschnitt 3.2 *„Schritt 2: Eine Anleitung zur Ursachensuche flussaufwärts“*).

3.1.3 Schritt 3: „Beratung“ - Abfragen von Signaturen

Nachdem die Aktivität im Prozess identifiziert wurde, die die Ursache für ein Problem birgt, soll sie für eine FLOW-Analyse genutzt werden. Dieser Schritt bedient sich dabei unter anderem der Darstellung der Signatur der Aktivität. Hierdurch soll eine einfache „Gegenüberstellung“ von Benutzer und Situation erreicht werden, in der dem Benutzer ein Teil „seines Prozesses“ (in eine einfach verständliche Notation übersetzt) zur weiteren Analyse präsentiert wird.

Vorab: Begriffsklärungen und Definitionen zu FLOW-Signaturen

Entsprechend der Definition von Schneider und Lübke [WCSQ05] ist die FLOW-Signatur einer Aktivität eine Sammlung von Interfaces, die einen Block mit den umgebenden Objekten verbinden, den Informationsspeichern. Jedes Interface ist dabei die Sammlung sämtlicher FLOWs zu und von einem der Objekte. Ein FLOW wird definiert durch seine Quelle und Senke, die unidirektionale Richtung des Flusses sowie Flow-Typ (*Information* oder *Erfahrung*) und -Aggregatzustand (*fest* oder *flüssig*). Zusätzlich zum Begriff Informationsspeicher möchte ich (in Anlehnung an die Terminologie des *Rational Unified Process* [s. RUP]) die synonym zu verwendende Bezeichnung Artefakt einführen.

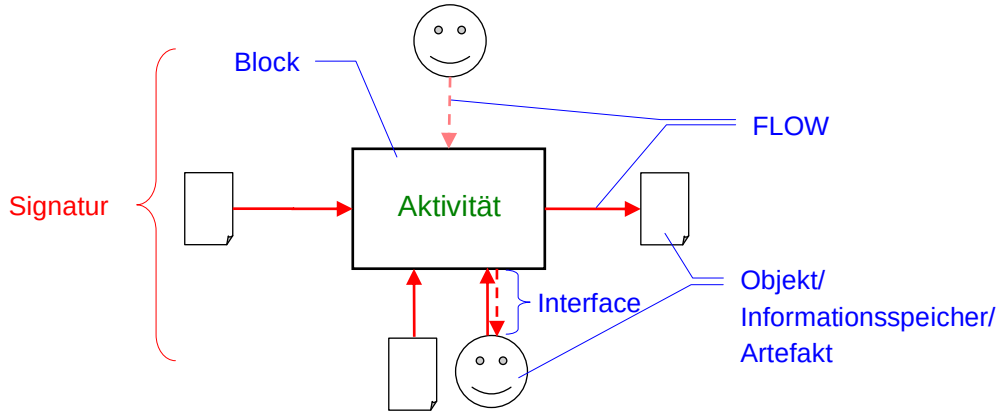


Abbildung 8: FLOW-Signatur einer Aktivität

Hinweis: Im Gegensatz zur IDEFØ-Notation [s. SADT], die neben den „Outputs“ einzig unterhalb der Aktivität bei den „Mechnisms“ auswärts gerichtete Pfeile als sogenannte „Calls“ zulässt, haben von der Aktivität ausgehende FLOWs in diesem Zusammenhang eine andere Bedeutung. Ein „Call“ ist der Verweis auf einen anderen Block und dessen Verfeinerungen, die den aktuellen detailliert darstellen. Ausgehende FLOWs in der Darstellung von Signaturen bezeichnen Informationsflüsse oder Erfahrungsgewinne, die aus der Aktivität entstehen, aber nicht Ziel (und damit „Output“) der Aktivität, sondern eher Nebenprodukte sind.

a. Mit Hilfe von Fragen zu Darstellung und Analyse der Ist-Situation

Die erwünschte Gegenüberstellung wird erreicht, ohne dass der Ratsuchende diese Darstellung selbst anfertigen oder über fundierte Kenntnisse der Notation verfügen muss. Hintergrund ist, dass wichtige Elemente der Darstellung – sogar ganze FLOW-Muster – verloren gehen bzw. nicht erfasst werden, wenn Diagramme manuell erstellt werden (siehe Abschnitt 2.1.4 *„Diskussion der praktischen Anwendbarkeit des FLOW-Pattern-Katalogs“*).

Stattdessen soll die Tatsache, dass sich Standard-Situationen systematisch auswerten lassen, ausgenutzt werden, indem Fragen gestellt werden, um die Parameter der betreffenden Aktivität abzufragen und die Ist-Situation auf verschiedenen Ebenen zu untersuchen. Zunächst bietet allein die FLOW-Darstellung der Aktivität an sich Analysemöglichkeiten aus der Informationsfluss-Perspektive. Im Hintergrund, für den Benutzer zunächst verborgen, können über Fragen auch bereits Informationen über die Aktivität gesammelt werden, die von Notation und Darstellung nicht erfasst werden.

Wie schon in den ersten, in den vorangegangenen Kapiteln beschriebenen Schritten der Ratgeber-Methode ist das Stellen der richten Fragen das Mittel zum Zweck, um nach diesem dritten Schritt den Ratgeber in die Lage zu versetzen, hinreichend spezifische Ratschläge anzubieten.

Fragen, die zu stellen sind, um die konkrete „Ist“-Signatur einer Aktivität zu erhalten, unterscheiden sich zunächst danach, wie sie gewonnen werden und was durch sie erreicht werden soll.

b. Erster Typ von Fragen: Aus Standard-Signaturen abgeleitete Fragen

Der erste Fragen-Typ ergibt sich aus der zugrundeliegenden Standard-Situation, die ermittelt wurde. Hierbei wird eine Situation, die als „Soll“ bekannt ist, ausgenutzt. Davon ausgehend, dass die üblichen Interfaces und Artefakte in der „Ist“-Situation ebenfalls alle vorhanden sein könnten, werden diese entsprechend abgefragt. Mit welchem Hintergedanken die Fragen jeweils formuliert werden, kann sich allerdings unterscheiden.

Die folgenden Beispiele beziehen sich auf die Signatur einer Standard-Prüfung:

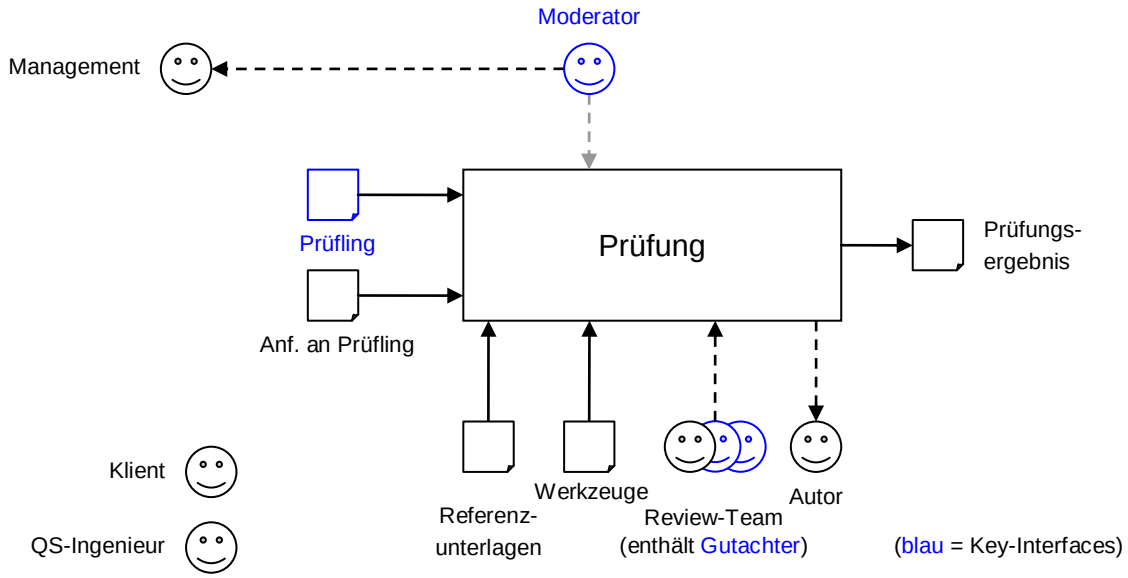


Abbildung 9: FLOW-Signatur einer Standard-Prüfung

Fragen zur Validierung

Einfache erste Fragen ergeben sich hier durch die Notwendigkeit, zunächst zu klären, ob die angenommene Situation wirklich zutreffend ist. Hierzu könnten verschiedene Signaturen „vorgeschlagen" und dem Benutzer zur Auswahl angeboten werden. Dies erfordert aber, dass dieser sich bereits mit der Notation auseinander gesetzt hat und ermöglicht ihm, später zu „schummeln", da er weiß oder ahnt, welche Antworten von ihm erwartet werden. Die Frage *„Welche der folgenden Signaturen beschreibt Ihre Aktivität am besten?"* ist also naheliegend, aber nicht die optimale Vorgehensweise. Stattdessen können *Key-Interfaces*, also Schlüsselelemente der Signatur abgefragt werden. Für alle Prüfungen sind das zumindest Moderator, Prüfling und Gutachter. Wenn Fragen nach dem Vorhandensein dieser Elemente ver*nein*t werden, kann davon ausgegangen werden, dass man sich mit der Vermutung, eine Prüfungs-Aktivität vor sich zu haben, auf dem Holzweg ist. Hierbei handelt es sich um Fragen zur *Validierung*.

Fragen zur Individualisierung

Weitere Fragen sind dazu gedacht, die Aktivität mit den Interfaces auszustatten, die tatsächlich vorhanden sind. Wir bezeichnen diese Phase als *Individualisierung*. Abfragen der Standard-Interfaces ermöglicht es, diese durch die tatsächlich vorhandenen und verwendeten zu ersetzen. An dieser Stelle sind oft Erklärungen und Beispiele nötig. In unserem Beispiel kann nach verwendeten Referenzunterlagen gefragt werden. Gleichzeitig mit einer Definition („Referenzunterlagen helfen, zu überprüfen, ob die Anforderungen, die an den Prüfling gestellt wurden, eingehalten wurden.") können mögliche Unterlagen genannt werden („Referenzunterlagen sind z.B. Checklisten, Prüfpläne, Fragenkataloge oder Musterdokumente") um deutlich zu machen, was hier gemeint ist. Wichtig ist, dass es dem Benutzer klar wird, wenn nach Artefakten gefragt wird, die in seiner Aktivität vorkommen – auch dann, wenn sie vielleicht anders bezeichnet werden. Es soll eine weitgehend vollständige Ausstattung der FLOW-Signatur mit den verwendeten Informationsspeichern erreicht werden, um ein möglichst genaues deskriptives Modell der Situation zu erhalten.

Fragen zur Verifikation

Die Fragen der dritten Kategorie betreffen die *Verifikation* der Aktivität. Sie hinterfragen gezielt kritische Zusammenhänge und liefern Anhaltspunkte, ob die Aktivität korrekt ausgeführt wird. Für Aktivitäten lassen sich Regeln erstellen (siehe Abschnitt „Regeln zur Ergänzung der Abfrage von Signaturen"), die an dieser Stelle überprüft werden können. Dies kann indirekt durch Fragen geschehen, die später keinen unmittelbaren Beitrag zur korrekten Darstellung der Aktivität leisten, deren Beantwortung aber Einblicke in die Durchführung der Aktivität und die internen Abläufe bietet. Beispielsweise gilt für technische Reviews eine Vielzahl von Regeln, und oft lassen sich *Best Practices* formulieren, die eingehalten werden sollten.

Einige dieser Regeln lassen sich bereits mit den oben geschilderten Fragen zur Individualisierung klären, so zum Beispiel die grundlegenden Aussagen „der Autor darf nie mit dem Prüfling verwechselt werden" und „der Autor darf nicht Teil des Review-Teams sein". Wenn während der Individualisierung die Frage nach dem Prüfling mit „Autor" beantwortet wird oder dieser in der Liste der Gutachter genannt wird, sind diese Regeln offensichtlich verletzt.

Andere Regeln sind nicht so einfach zu überprüfen. Diese betreffen meist die *Durchführung* der Aktivität an sich. Ludewig erinnert in [LUD06] beispielsweise daran, dass beim technischen Review „Prüfung" und „Korrektur" dringend voneinander getrennt werden müssen:

„Die Entwicklung oder Diskussion von Lösungen ist nicht Aufgabe des Review-Teams.“ Ob dies eingehalten wird, kann direkt nachgefragt werden, es können aber auch Artefakte hinterfragt werden, die der Benutzer als „Outputs“ anführt und deren Zweck nicht bekannt ist. Wird zum Beispiel ein Ergebnis-Dokument mit der Bezeichnung „korrigierte Software“ angegeben, ist klar, dass im Prüfungs-Prozess eine Korrektur ausgeführt wurde. Problematisch ist an dieser Stelle die Automatisierung, siehe dazu Abschnitt *„Klassifizierung von Artefakten“*.

c. Zweiter Typ von Fragen: Aus dem Prozesskontext abgeleitete Fragen

Der dem Abfragen von Signaturen vorangehende Schritt, der das „Flussaufwärtsschwimmen“ im Prozessmodell beinhaltet, bietet bereits wertvolle Informationen, die helfen, den richtigen Kontext für die zu beschreibende Aktivität zu bestimmen. Insbesondere der Zeitpunkt im Prozessverlauf gibt einige Anhaltspunkte, wonach es sich zu fragen lohnt.
Wenn zum Beispiel eine Review-Aktivität betrachtet werden soll, und bekannt ist, dass diese zeitlich zu Beginn des Projekts in der Anforderungserhebung angesiedelt ist, dann ist es sehr wahrscheinlich, dass es sich bei dem „Prüfling“ noch nicht um erstellten Quellcode, sondern um die erhobenen Anforderungen handelt.
Hier ist die Frage angebracht und sogar wichtig, ob der Klient als Gutachter ins Review-Team eingeladen wurde. Die Annahme, dass zu diesem Zeitpunkt eigentlich noch kein Code bestehen kann, führt zu weiteren Fragen, so kann zum Beispiel einfach danach gefragt werden. Wenn daraufhin *doch* Quellcode als zu überprüfen angegeben wird können weitere Schlussfolgerungen gezogen werden.

d. Vergleich und Einsatz der Fragen-Typen

Die Fragen, die zu stellen sind, haben also verschiedene Hintergründe sowohl bezüglich ihrer Generierung als auch ihrer Ziele. Worin bestehen diese Unterschiede und was ist zu beachten?

Zunächst scheinen sich auch für den kontext-abhängigen Fragentyp Fragen zur Validierung, Individualisierung und Verifikation erstellen zu lassen, wie es für Standard-Signaturen abgeleitete Fragen der Fall ist. Dies trifft bei genauerer Betrachtung nur teilweise zu, wie hier erläutert werden soll.

So dient die im vorhergehenden Abschnitt genannte Frage nach Quellcode in einer Situation, die eigentlich keinen Quellcode als Artefakt vorsieht, an dieser Stelle nicht zur Validierung der gewählten Standardsituation. Es wäre ein Zirkelschluss, aus dem Prozess-Kontext etwas zu vermuten und dann den Fehler beim Kontext zu suchen, wenn die Vermutung nicht zutrifft.

Zur Verdeutlichung: Werden nach der Annahme einer Standard-Situation Fragen zur Bestimmung von deren Parametern gestellt, und Key-Interfaces tauchen nicht auf, lässt dies den Schluss zu, dass die angenommene Standard-Situation nicht vorliegt. Hier haben die Fragen eine Validierungs-Funktion, wie oben beschrieben.
Der Kontext im Prozess allerdings, insbesondere die zeitliche Einordnung, ist dagegen durch die vorangehenden Schritte der Methode schon gesichert. Bestenfalls wurde die ausgewählte Standardsituation zudem bereits auf die genannte Weise validiert. Vermutungen, die jetzt angestellt werden, betreffen nicht die Wahl der Standardsituation an sich, sondern beziehen sich auf deren genaue Ausführung und Details.
Angenommen also, dass entgegen der „Standard-Vermutung“ (nämlich dass zum Zeitpunkt der Anforderungserhebung noch keine Software zu prüfen ist) *doch* Software vorgelegt wird,

trägt diese Erkenntnis, gewonnen durch entsprechende Kontext-motivierte Fragen, zur Individualisierung bei: Wenn in dieser Situation Software „auftaucht“, wird vermutlich Prototyping betrieben. Dies ist eine Elicitation-Technik, und deren Anwendung ist ein wichtiger Hinweis. Aus denselben Gründen wie oben ist die Verifikation des Vorgehens durch Fragen, die aus dem Kontext generiert wurden, in Bezug auf die vorliegende Standard-Situation nicht möglich. Weiterhin müsste ein „Soll“ vorliegen, das mit einem „Ist“ verglichen werden kann, um eine Verifikation durchführen zu können. Dies ist gegeben im Fall der Standard-Situation, die erreicht werden soll. Der Prozesskontext ist nicht abgegrenzt und überschaubar genug, um eine „Soll“-Durchführung vorherzusagen. Somit können keine Fragen aus dem Kontext gewonnen werden um die Durchführung zu verifizieren.

Zusammenfassend ergibt sich also für das Zusammenspiel der unterschiedlichen Frage-Typen folgendes Bild: Zuerst sind Fragen aus der Standard-Signatur der angenommenen „Ist“-Situation zu erstellen, die die Validierung der Annahme erlauben. Um die Signatur anschließend zu individualisieren können weitere Signatur-basierte Fragen hinzukommen, unterstützend kann für weitere Fragen hier der Prozess-Kontext herangezogen werden. Um die Durchführung zu verifizieren wird wieder auf Fragen zurückgegriffen, die aus der „Soll“-Signatur und zugehörigen Regeln zu gewinnen sind.

Wie Fragen aus Signaturen und zugehörigen Regeln gewonnen werden, wird im Folgenden beschrieben.

e. Generierung von Fragen aus Standardsituationen

„Standardsituationen“ können alle möglichen Situationen im Verlauf eines Projekts sein, die positive „Soll“-Zustände sind, also erstrebenswert – z.B. *Best Practices*. Diese werden durch FLOW-Signaturen dargestellt und zur Fragengenerierung ausgewertet. Einen vollständigen und aussagekräftigen Fragenkatalog zu erstellen, der möglichst viele Situationen erfasst und miteinander in Abhängigkeit setzt, um damit eine breite „Datenbank“ für die Entscheidungen des Ratgebers anbieten zu können, ist eine umfangreiche und zeitaufwändige Aufgabe. Für diesen Vorgang ergeben sich den speziellen Eigenschaften einer jeden untersuchten Situation entsprechende Aspekte, die nicht in eine feste Methode zur Extraktion der „richtigen“ Fragen gepresst werden können.

So werden entsprechend den in Abschnitt *b* unterschiedenen Fragen-Typen aus der Signatur Fragen zur Validierung, Individualisierung und Verifikation gewonnen, sowie aus dem Kontext weitere Fragen zur Individualisierung (siehe Abschnitt *c*). Es gilt aber auch Gesichtspunkte zu berücksichtigen, die nicht in eine Vorlage einzuordnen sind und vom Analysten intuitiv und allein anhand seines Wissens um die Eigenschaften der untersuchten Standardsituation mit einzubeziehen sind. Entsprechende Analysen könnten daher in Workshops mit mehreren Spezialisten durchgeführt werden, um möglichst viele wichtige Aspekte zu berücksichtigen.

An einem Beispiel soll hier eine mögliche Signaturanalyse zur Gewinnung von Fragen demonstriert werden; dieses Vorgehen kann als Vorlage für spätere Untersuchungen weiterer Aktivitäten dienen. Als Aktivitätskontext (siehe 3.1.1 *„Ursachen- und Situations-Muster“*) wurden die *Prüfungen* gewählt.

Zusammenfassung von Aktivitäten und Zuordnung zu einem Kontext

Es wurden viele verschiedene Standardsituationen gesammelt, die sich als Signaturen von Aktivitäten darstellen lassen. Oberbegriffe sind hier zum Beispiel *Änderungsmanagement*, *Elicitation*, *Risikomanagement* und *Qualitätssicherung*.

Aktivitäten wie Workshops, Interviews, Reviews lassen sich diesen übergeordneten Gruppen mehrfach zuordnen. Wird ein bestimmter Kontext betrachtet, begrenzt dies die Auswahl. Im Falle der hier zu betrachtenden *Prüfungen* auf die „analytischen Maßnahmen“ aus dem Bereich der Qualitätssicherung [LUD06].

Die Vielzahl der hier denkbaren Aktivitäten beherrscht man bei der Analyse zunächst durch eine Zuordnung zu Gruppen. Es wurden zunächst *Schulungen, Audits, Softwaretests* und *–Prüfungen* unterschieden. Die zugehörigen Aktivitäten sondern sich durch die Eigenarten der jeweiligen Durchführung voneinander ab. Zudem werden die einen zur Verbesserung und Sicherung der *Prozess*qualität eingesetzt und die anderen an *Milestones* zur Sicherstellung der *Produkt*qualität durchgeführt.

Aus dieser Unterscheidung ergibt es sich für den Kontext „Prüfungen“, im Folgenden die Gruppe der *Software-Prüfungen* zu betrachten. Hierzu gehören Aktivitäten, die sich von den Signaturen her ähneln, aber nach dem betriebenen Aufwand unterscheiden lassen: Von *Stellungnahme*, *Walkthrough* und *Durchsicht* bis zu *Inspection* und *Technischem Review*.

Sammlung von Artefakten des gewählten Kontext

Diese Aktivitäten werden nun als *Software-Prüfungen* gemeinsam betrachtet. Möglicherweise ist bei der Beratung noch nicht klar, welche der Aktivitäten genau „auf dem Prüfstand“ ist,

daher empfiehlt es sich, zunächst alle Artefakte zu sammeln, die im betrachteten Bereich zum Einsatz kommen. Über „Key-Interfaces“ (s. Abschnitt *b*) kann dann das Vorliegen einer bestimmten Aktivität (sei es als zu erreichendes „Soll“ oder abzufragendes „Ist“) bestimmt werden. Für die betrachteten *Software-Prüfungen* ergeben sich folgende Artefakte:

Informationsspeicher	Beschreibung
Autor	Verfasser des Prüflings
Gutachter	Experte, der zur Beurteilung des Prüflings teilnimmt
Klient	Kundenvertreter
Manager	meist nicht-technisch-verantwortlicher Vorgesetzer
Moderator	Spezialist, der die Prüfung leitet
Notar	protokolliert die Prüfung
Projektleiter	u.a. technisch Verantwortlicher
QS-Ingenieur	Qualitätssicherungs-Ingenieur, Organisatorisch verantwortlich für die Qualitätssicherung
Review-Team	aktiv an Prüfung teilnehmende Personen (nicht der Autor!)
Tester	führt Software-Tests durch
Änderungsauftrag	verbindlicher Auftrag, Änderungen am Prüfling durchzuführen
Checkliste	Hilfsmittel zur ordentlichen Durchführung der Prüfung
Fragenkatalog	Hilfsmittel zur ordentlichen Durchführung der Prüfung
Musterdokument	Vorlage zum Vergleich mit dem Prüfling
Prüfling	zu prüfendes Artefakt
Prüfplan	Hilfsmittel zur ordentlichen Durchführung der Prüfung
Referenzunterlagen	Vorlage zum Vergleich
Richtlinien	Hilfsmittel zur ordentlichen Durchführung der Prüfung
Spezifikation	enthält Anforderungen an den Prüfling
Stellungnahme/Beurteilung	Aussage eines einzelnen Gutachters oder Sammlung der Aussagen des Review-Teams nach der Prüfung
Werkzeug	technisches Hilfsmittel zur Durchführung der Prüfung

Tabelle 4: Informationsspeicher im Aktivitätskontext „Prüfungen“

Klassifizierung von Artefakten

Die Artefakte lassen sich wiederrum zu Informationsträger-Gruppen zusammenfassen und gegebenenfalls sogar hierarchisieren. Diese Klassifizierung muss von Hand vorgenommen werden und dient dazu, die Artefakte zusammenzufassen. So können während der Beratung Vorschläge gemacht oder Beispiele genannt werden. Zudem können „Erwartungen“ überprüft werden, die je nach Kontext abweichen können (siehe oben genanntes Beispiel: In einer Prüfung während der Anforderungsanalyse wird üblicherweise keine Software geprüft, wenn doch, liegt die Spezialprüfung „Prototyping“ vor).

Die oben genannten Informationsspeicher wurden wie folgt zusammengefasst:

Informationsspeicher-Gruppe	enthält
Vorgabe	*Oberklassen*: Prüfling, „Ist“-Resultat *Instanzen*: Programmcode, Entwurfsbeschreibung, Anforderungen, Spezifikation, Dokumentation
Werkzeug	Fragenkatalog, Checkliste, Prüfplan, Musterdokument
Referenzunterlagen	Richtlinien, Spezifikation, Beispielimplementierung
Prüfungsergebnis	Stellungnahme/Beurteilung, Änderungsauftrag

Tabelle 5: Informationsspeicher-Gruppen im Aktivitätskontext „Prüfungen“

Mit Hilfe der Zusammenfassungen lässt sich die Anwendbarkeit verallgemeinern. Die Artefakte werden nun in einer Signatur-Darstellung miteinander kombiniert, wobei weitere Zusammenfassungen und Umbenennungen vorgenommen werden können (erwähnenswerte Rollen oder einzelne Artefakte sind hier verkürzend „am Rande“ dargestellt, es sind verschiedene Ausprägungen dieser Aktivität denkbar, in denen zum Beispiel ein Kundenvertreter mitwirkt oder der Projektleiter mit einbezogen wird; diese speziellen Instanzen ergeben sich aus dem genauen Zusammenhang):

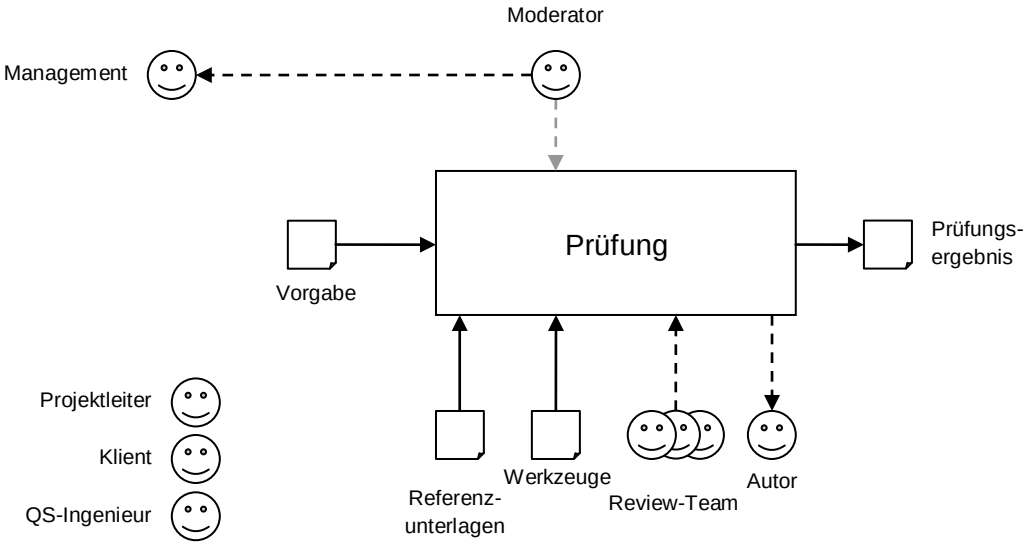


Abbildung 10: Vereinigung der Artefakte aller Prüfungen in einer Standard-Signatur

Erste Fragen

Aus dieser allgemeinen, „übergeordneten“ Signatur ergeben sich die ersten Fragen nach den beteiligten Informationsspeichern und deren Rollen. Die tatsächlich vorliegende Aktivität wird dabei durch die nach und nach sichtbaren detaillierten Zusammenhänge erkennbar, insbesondere durch die Key-Interfaces. Die wichtigsten Artefakte werden mehr oder weniger direkt erfragt, wobei bestimmte Fragen natürlich nur gestellt werden, wenn vorangehende Antworten dies zulassen:

Informationsspeicher	mögliche Frage(n)
Moderator	Wer stellt sicher, dass die Teilnehmer die Prüfung gründlich vorbereiten? Wer lädt zur Prüfung ein? Leitet ein speziell ausgebildeter Moderator die Prüfung? Überwacht der Moderator die formale Ausführung?
Vorgabe	Handelt es sich bei dem Prüfling um (Programmcode/ Entwurfsbeschreibung/Anforderungen/Dokumentation)? In welcher Form liegt der Prüfling vor? Wird der Prüfling vom Autor vorgestellt?
Review-Team	Wie viele Personen bilden das Review-Team?
Gutachter	Wie viele Gutachter nehmen teil? Ist der Moderator auch Gutachter? Ist der Autor auch Gutachter?
Notar	Gibt es einen Protokollführer für die Prüfung?
Tester	Werden Tester befragt?
Referenzunterlagen	Welche Form von Referenzen gibt es, gegen die der Prüfling geprüft wird (Richtlinien, Spezifikation)?
Prüfungsergebnis	Welche Ergebnisse werden erstellt (Stellungnahme/Beurteilung, Änderungsauftrag)?
Werkzeug	Welche Werkzeuge werden eingesetzt, um die Prüfung zu steuern (Fragenkatalog, Checkliste, Prüfplan, Musterdokument)?

Tabelle 6: Informationsspeicher im Aktivitätskontext „Prüfungen“ und entsprechende Fragen

Aus speziellen Rollen/Informationsspeichern ergeben sich weitere Fragen, die nur in gewissen Fällen (zum Beispiel bei Vermutung einer bestimmten Ursache oder einem bestimmten zeitlichen Kontext im Projekt) gestellt werden:

Informationsspeicher	mögliche Frage(n)
QS-Ingenieur	Ist ein Mitarbeiter speziell verantwortlich für die Qualitätssicherung? Welche Rolle spielt er bei der Prüfung (Moderator, Gutachter, Organisator)?
Autor	Nimmt der Autor an der Prüfung teil? In welcher Rolle (Verteidiger des Prüflings, stiller Zuhörer, …)?
Klient	Nimmt ein Kundenvertreter an der Prüfung teil? In welcher Rolle (Gutachter, stiller Zuhörer, Berater, …)?
Projektleiter	Nimmt ein technisch Verantwortlicher (Projektleiter) an der Prüfung teil? Wird ein technisch Verantwortlicher (Projektleiter) über die Ergebnisse der Prüfung informiert?

Tabelle 7: Weitere Informationsspeicher spezieller Fälle des Aktivitätskontextes „Prüfungen“

f. Regeln zur Ergänzung der Abfrage von Signaturen

An dieser Stelle wird bereits deutlich, dass die Fragen nicht nur die bloße Existenz bestimmter Informationsspeicher und deren Rolle in der Aktivität „abklopfen“, sondern auch bereits aus deren Kontext motiviert sind. So wird nicht nur nach positiven Sachverhalten gefragt, auch unerwünschte Möglichkeiten werden erwogen und gezielt überprüft. Um die Erstellung solcher Fragen nicht vollkommen der Intuition des Analysten zu überlassen ist es sinnvoll, auf *Regeln* einzugehen, die zu bestimmten Aktivitäten gelten.

Am Beispiel des *Technischen Reviews* soll dies erläutert werden. Für diese Form der Prüfung sind Regeln besonders deutlich festgelegt. Anhand einer Auswahl nach [LUD06] wird deutlich, welche Prüfungen sich ergeben, indem nach „Regelverstößen“ gesucht wird:

Regel	daraus abgeleitete Prüfung(en)
Alles, was gefordert wurde, muss auch geprüft werden.	Gibt es zu jedem zu prüfendem Aspekt („Ist“) eine Quelle für ein gegenüber zu stellendes „Soll“?
Der Autor darf nicht über den Prüfling urteilen, und somit nicht Teil des Review-Teams sein.	Wird der Autor als Gutachter eingesetzt?
Werden Anforderungen geprüft, muss ein Kundenvertreter am Review teilnehmen.	Worum handelt es sich beim Prüfling? Ist ggf. ein Klient Teil des Review-Teams?
Im Review wird nur geprüft, Korrekturen werden weder vorgenommen noch diskutiert.	Werden als Prüfungsergebnisse Änderungsvorlagen oder geänderter Code genannt?
Befunde werden nicht in Form von Anweisungen an den Autor formuliert.	Haben erstellte Änderungsaufträge detaillierten „Lösungscharakter“, so dass die Gutachter zu Entwicklern werden?
Der QS-Ingenieur sollte nicht dem Projektleiter unterstellt sein, sondern an eine höhere Ebene berichten.	An wen berichtet der QS-Ingenieur?
Der Moderator organisiert das Review, lädt dazu ein und führt es durch.	Wer organsiert, lädt ein und moderiert?
Eine Review-Sitzung ist auf zwei Stunden beschränkt.	Wie lange dauert eine Sitzung?
Der Moderator bricht oder sagt die Prüfung ab, wenn sie nicht erfolgreich durchgeführt werden kann (aufgrund mangelnder Vorbereitung oder Teilnahme).	Werden entsprechende Prüfungen durchgeführt? Kommt es häufiger zu entsprechenden Abbrüchen/Absagen? Ist die Vorbereitung und Teilnahme gut? Wird trotz schlechter Bedingungen nicht abgebrochen oder abgesagt?
Prüfling und Autor sind nicht dasselbe! Der Autor darf nicht angegriffen werden und weder sich noch den Prüfling verteidigen.	Werden Prüfling und Autor sauber getrennt? Muss der Autor seinen Code vorstellen oder verteidigen?
Rollen werden nicht vermischt. Insbesondere darf der Moderator nicht gleichzeitig als Gutachter fungieren.	Kommt es zu aufgeweichten Grenzen zwischen den Rollen? Nimmt der Moderator als Gutachter teil?
Allgemeine Stilfragen außerhalb der Richtlinien dürfen nicht diskutiert werden.	Sind Richtlinien vorhanden? Wird auf darauf geachtet, dass nur entsprechende Fragen diskutiert werden?
Jeder Gutachter erhält genügend Zeit, seine Befunde angemessen zu präsentieren.	Passt das Verhältnis von Gutachtern/Dauer eines Reviews? Welche Räumlichkeiten/Hilfsmittel werden zur Verfügung gestellt?
Ohne Konsens der Gutachter darf nicht fortgefahren werden. Die Unterschrift aller Teilnehmer auf dem Protokoll am Ende des Reviews ist erforderlich.	Kommt es dazu, dass Gutachter übergangen werden? Was sind die Gründe?

Tabelle 8: Regeln zu *Technischen Reviews* und entsprechende Prüfungen

Hier ist absichtlich die Rede von „Prüfungen“, nicht von „Fragen“. Einige der Regeln lassen sich auch anders prüfen als durch Fragen an den Benutzer. Zum Beispiel ist die Rollentrennung verletzt, wenn an einen Mitarbeiter mehrere Rollen vergeben werden, es lässt sich überprüfen, ob die Rollen von „den richtigen“ Personen besetzt werden (der Autor darf nicht Gutachter sein) und 20 Gutachter werden in 2 Stunden Sitzung sicher nicht „angemessen“ zu Wort kommen.

g. Erteilen von Ratschlägen

Es ist mit diesen Fragen möglich, die vorliegende Aktivität mit den tatsächlich verwendeten Informationsspeichern auszustatten und darzustellen. Gleichzeitig kann validiert werden, um welche Aktivität es sich handelt, und mit einer „Muster“-Darstellung verglichen werden. So ist es möglich, Abweichungen und daraus resultierende Verbesserungsvorschläge anzuzeigen. Diese basieren dann sowohl auf dem direkten Vergleich der Signaturen als auch aus den Regeln, die hinter der Mustersignatur stehen.

Viele der mit den Regeln angesprochenen Aspekte sind schon sehr nah am FLOW-Gedanken, wenn zum Beispiel der Fluss von Informationen zwischen dem QS-Beauftragten und seinen Vorgesetzten untersucht wird, Informationsflüsse zwischen Personen oder Rollen aufgedeckt werden, die absichtlich voneinander getrennt werden sollten oder die Forderung nach „fester“ Dokumentation durch einen Notar gestellt wird.

Durch die Darstellung der Signaturen mittels Elementen der FLOW-Notation zeigt sich, dass die Analyse auf dieser Ebene wieder in die Nähe der FLOW-Pattern gelangt. Viele Ratschläge, die sich aus der bloßen Gegenüberstellung der „Ist“- mit der „Soll“-Situation ergeben, können über die Darstellung vermittelt werden. Die Erklärung der Hintergründe (z.B. warum sich ein Informationsfluss zwischen zwei Rollen befinden sollte, weshalb bestimmte Informationen nicht bei einer Person zusammenlaufen dürfen) rundet die Beratung ab und macht die Ratschläge verständlich und ermöglicht die Nachhaltigkeit, eine gewisse „FLOW-Awareness“ und einen Lerneffekt.

3.2 Prototyp

Die exemplarische Umsetzung muss sich wegen der zeitlichen Begrenzung der Bearbeitungszeit auf ausgewählte Aspekte beschränken. Einzig Schritt 2 der Methode beschreibt eine sehr allgemeine Vorgehensweise und unterliegt keinen Kürzungen. Schritt 1 eines vollständigen Ratgebers, nach dem vorgestellten Konzept entwickelt, kann dagegen theoretisch auf Qualitätseigenschaften aus allen Bereichen angewendet werden. Gewählt wurde für diese Beispielimplementierung der Qualitätsbereich „Brauchbarkeit“, da in den Interviews Problematiken aus diesem Bereich besonders häufig diskutiert wurden. Außerdem wurde eine bestimmte Menge von Situationen als besonders relevant befunden, die sogenannten „Prüfungen“, wozu Signaturen ausgewählter Aktivitäten für Schritt 3 der Ratgeber-Methode untersucht wurden.

Als Folge der Beschränkungen können nicht alle Situationen, die in der Wirklichkeit auftreten auf mit der hier beschriebenen Ratgeber-Instanz untersucht werden. Auf Erweiterungsmöglichkeiten und –ansätze wird in Kapitel 4.2.1 *„Weitere Ursachen- und Situations-Muster“* eingegangen.

Schritt 1: Hilfsmittel zur Orientierung

Fragenkatalog zur Einordnung der Problematik

① In welchem Bereich äußert sich das Problem?

Welche Aussage trifft eher zu:

Die festgestellte Problematik…

(1) ○ …bezieht auf die entwickelte Software.
(Beispiele: Effizienz oder Genauigkeit der Software, Lesbarkeit des Codes)

(2) ○ …betrifft einen Aspekt des Entwicklungsprozesses.
(Beispiele: Entwicklungsgeschwindigkeit, Planungssicherheit, Projektklima)

Bestimmung der betroffenen Software-Qualität: Produkt- oder Prozessqualität,
weiteres Vorgehen bei Auswahl von Möglichkeit (1):

② Welcher Qualitätsaspekt Ihres Produkts ist betroffen?

Welche Aussage trifft eher zu:

Die festgestellte Problematik äußert sich in…

(A) ○ …der Brauchbarkeit der entwickelte Software.
(Beispiele: Genauigkeit, Sparsamkeit, Verständlichkeit)

(B) ○ …deren Wartbarkeit.
(Beispiele: Testbarkeit, Strukturiertheit des Codes, Geräteunabhängigkeit)

Bestimmung des Qualitäts-Bereichs : Brauchbarkeit oder Wartbarkeit,
weiteres Vorgehen bei Auswahl von Möglichkeit (A):

③ In wie fern ist die Brauchbarkeit Ihres Produkts ist eingeschränkt?

Welche Aussage trifft eher zu:

(a) ○ Die Software ist nicht korrekt in bezug auf ihre Spezifikation.

(b) ○ Die Resultate der Software weichen von mathematisch korrekten Resultaten ab.

(c) ○ Die Software erbringt des Öfteren nicht die erwartete Funktion.

(d) ○ Die Software braucht erheblich mehr Rechenzeit, als minimal erforderlich wäre.

(e) ○ Die Software braucht erheblich mehr Speicherplatz und/oder andere Betriebsmittel, als minimal erforderlich wäre.

(f) ○ Die Software erbringt nicht alle geforderten Leistungen.

(g) ○ Die Software verhält sich gegen den Benutzer in ähnlichen Situationen nicht ähnlich (dies ist bezogen auf Bedienung, Fehlermeldungen, Dateiformate usw.).

(h) ○ Benutzer sind nicht in der Lage, rasch zu verstehen, wie sie mit der Software umgehen müssen.

(i) ○ Benutzern erscheint der allgemeine Aufbau der Software zu kompliziert.

Bestimmung des betroffenen Qualitäts-Bereichs, hierbei entspricht: Wahl (a)-(c) = *Zuverlässigkeit*, (d)-(f) = *Nützlichkeit*, (g)-(i) = *Bedienbarkeit*, siehe untenstehende Abbildung.

④ Zu welchem Zeitpunkt im Projektverlauf ist die Abweichung festgestellt worden?

(I) ○ früh: Bei der Durchführung von Modultests.

(II) ○ später: Während der Integration im Kundensystem.

(III) ○ noch später: In der Installation (bei Systemtests).

(IV) ○ sehr spät: Im Betrieb (vom Kunden).

Bestimmung des Zeitpunkts, zu dem die Problematik entdeckt wurde. Auswertung Wahl (I)-(IV) zum Einstieg in den Prozess in Schritt 2: Entsprechend untenstehender Abbildung.

Hilfsmittel zur Auswertung des Fragenkatalogs

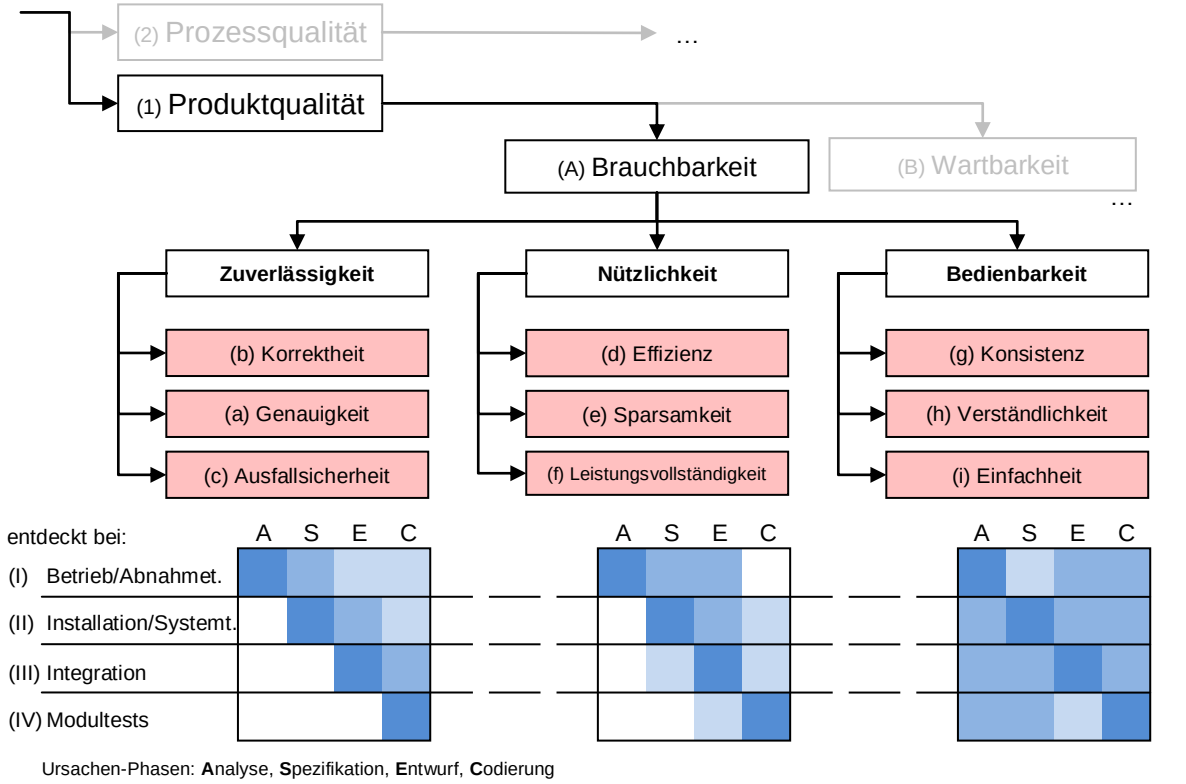


Abbildung 11: Schlüssel zur Auswertung der Antworten

Festhalten der Ergebnisse:

Betroffener Qualitätsbezug	produktbezogen / prozessbezogen			
Betroffener Qualitätsbereich 1	Brauchbarkeit / Wartbarkeit			
Betroffener Qualitätsbereich 2	Zuverlässigkeit / Nützlichkeit / Bedienbarkeit			
Betroffener Qualitätsaspekt	Korrektheit / Genauigkeit / Ausfallsicherheit / Effizienz / Sparsamkeit / Leistungsvollständigkeit / Konsistenz / Verständlichkeit / Einfachheit			
Zeitpunkt der Feststellung	Betrieb / Installation / Integration / Modultest			
Zu untersuchender Prozessbereich	A	S	E	C

Tabelle 9: Vorlage zum Festhalten der Ergebnisse zum „Prozessbereich“

Auswahl von Situations- und ggf. Ursachenmustern

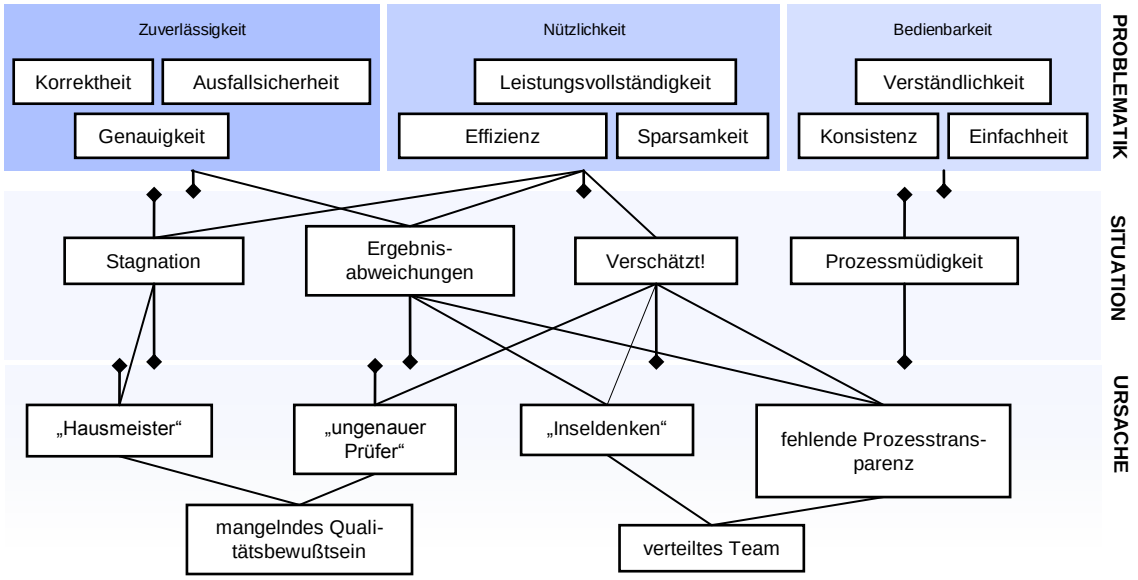


Abbildung 12: Problematik-Muster-Zuordnungsschlüssel; von oben nach unten zu lesen, über „direkte“ und „Universalverbindungen“ wird eine Wertung der Verknüpfungen erreicht

Festhalten der Ergebnisse:

Zur Bewertung wird die Summe der Gewichte der vorhergehenden Knotens addiert mit den Bewertungen aller Verknüpfungen, über die der aktuelle Knoten erreicht wird.		Bewertung	
		direkt = 2	universal = 1
Mögliche Situationsmuster:	☐ Stagnation		
	☐ Ergebnisabweichungen		
	☐ Verschätzt!		
	☐ Prozessmüdigkeit		
Mögliche Ursachenmuster:	☐ „Hausmeister“		
	☐ „ungenauer Prüfer“		
	☐ „Inseldenken“		
	☐ fehlende Prozesstransparenz		
Mögliche Hintergründe:	☐ mang. Qualitätsbewusstsein		
	☐ verteiltes Team		
zu untersuchender Aktivitätskontext:	☐ Prüfungen: Signaturen von Software-Prüfungen		
	☐ Begegnungen (zB. Änderungsausschuss, Schlichtung, …)		
	☐ Beurteilungen (zB. Grobentwurf-Schätzung, Angebot, …)		
	☐ Bürokratie (zB. Dokumentationsprozess, Reviews, …)		

Tabelle 10: Vorlage zum Festhalten der Ergebnisse zum „Aktivitätskontext“

Schritt 2: Eine Anleitung zur Ursachensuche flussaufwärts

Vorbedingungen

Zu untersuchender Prozessbereich (siehe Schritt 1):	A	S	E	C

Arbeitsanweisungen

I. Der jeweils verwendete Entwicklungsprozess wird grob in die Bereiche „Anforderungsanalyse“, „Spezifikation“, „Entwurf“ und „Codierung“ („ASEC-Modell“) unterteilt. Ähnliche oder nicht aufgeführte Schritte können dabei zusammengefasst werden (z.B. *System-Entwurf* mit *Programm-Entwurf*, *Machbarkeitsstudie* mit *Anforderungsanalyse*, *Codierung* mit *Testen*), später werden diese Zusammenfassungen wieder getrennt.

II. Entsprechend der in Schritt 1 gegebenen Empfehlung kann die Untersuchung auf bestimmte Schritte des Prozesses beschränkt werden.

III. Es wird „flussaufwärts“, also vom letzten relevanten Schritt in der Ausführungsreihenfolge, bis an den Anfang des Prozesses vorgegangen.

IV. Zu jedem Schritt sind zwei Fragen zu stellen:

1. *War der Fehler im ‚Output‘ dieses Schritts enthalten?*
    a. Ist dies der Fall ist der Fehler in oder vor diesem Schritt entstanden. Es wird fortgesetzt mit der zweiten Frage.
    b. Ist dies nicht der Fall ist der Fehler erst nach diesem Schritt entstanden. Kommt dies bereits zu Beginn der Untersuchung vor und wurden Schritte übersprungen sind diese wieder einzubeziehen.
2. *War der Fehler im ‚Input‘ dieses Schritts enthalten?*
    a. Wenn ja, ist der aktuell betrachtete Schritt nicht an der Entstehung des Fehlers beteiligt, sondern hat ihn vom vorhergehenden Schritt „geerbt“. Als nächstes ist der vorhergehende Schritt mit beginnend mit Frage 1 zu betrachten.
    b. Ist dies nicht der Fall ist der Fehler im aktuell betrachteten Schritt entstanden. Der aktuelle Schritt muss verfeinert werden (entsprechend dem Vorgehen, wie es bei [SADT] beschrieben wird). Der verfeinernde „Unterprozess“ wird ebenso untersucht wie ab II., wiederrum beginnend mit dem letzten Schritt.

Beschränkungen

Die Untersuchung endet spätestens, wenn die Prozessschritte nicht weiter verfeinert werden können. Dann befindet man sich in der Regel auf der Ebene von Aktivitäten. Es sollte bis hier hin eine Aktivität gefunden worden sein, die zu dem in Schritt 1 vermuteten Aktivitätskontext gehört.

Wurde keine passende Aktivität gefunden, kann es sein, dass die Klassifizierung in Schritt 1 falsch ist. Falsche Vermutungen können zudem durch unbekannte Situationsmuster entstehen, der Aktivitätskontext ist in diesem Fall auszuweiten.

In jedem Fall sollte der Bereich für die Ursache des Problems sich auf diese Weise einschränken lassen. Schritt 3 kann allerdings nur nach Bestimmung eines zu untersuchenden Aktivitätskontextes ausgeführt werden. Andernfalls muss die Beratung hier enden.

Schritt 3: Instrumente zur Beratung im Aktivitätskontext „Prüfungen“

Hilfsmittel für die Untersuchung Aktivitäts-Kontexts

Allgemeine Signatur des Aktivitäts-Kontextes

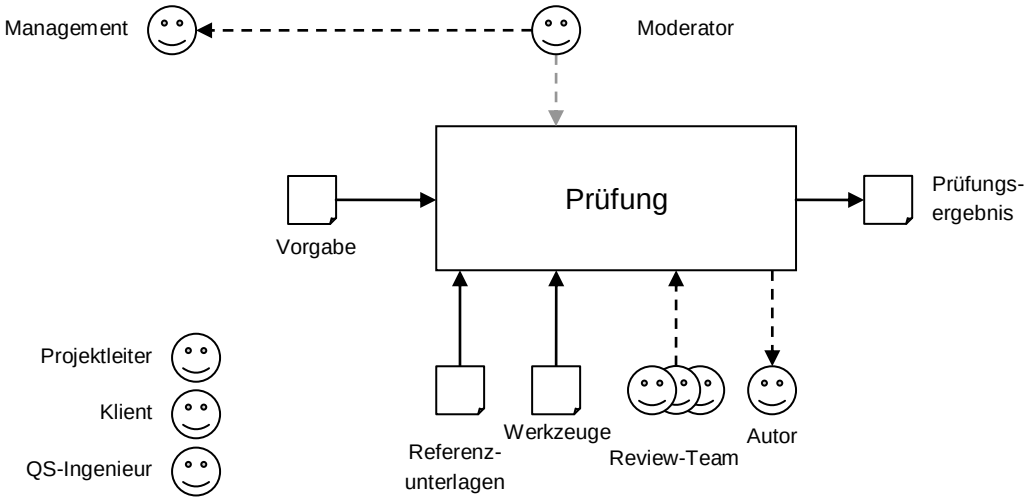


Abbildung 13: Allgemeine FLOW-Signatur des Aktivitäts-Kontextes „Prüfung“

Aus der allgemeinen Signatur des Aktivitäts-Kontextes abgeleitete Fragen

Informations-speicher	Frage(n)
Moderator	Wer lädt zur Prüfung ein? * Wer stellt sicher, dass die Teilnehmer die Prüfung gründlich vorbereiten? * Leitet ein speziell ausgebildeter Moderator die Prüfung? In welcher Rolle? * ☐ Überwacht die formale Ausführung. ☐ Nimmt als Gutachter teil.
Vorgabe	Handelt es sich bei dem Prüfling um… ** ○ …Programmcode? ○ …Entwurfsbeschreibung? ○ …Anforderungen? ○ …Dokumentation? In welcher Form liegt der Prüfling vor? ** ☐ Quellcode ☐ Diagramme/Skizzen ☐ Texte/Dateien Wird der Prüfling vom Autor vorgestellt?*
Prüfungs-ergebnis	Welche Ergebnisse werden erstellt? ** ☐ Stellungnahme/Beurteilung ☐ Änderungsauftrag ☐ andere: …

Informations-speicher		Frage(n)
	Review-Team	
	Gutachter	Wie viele Gutachter nehmen teil? */** Ist der Moderator auch Gutachter? Ist der Autor auch Gutachter? Nimmt ein Kundenvertreter als Gutachter teil? *
	Notar	Gibt es einen Protokollführer für die Prüfung? * In welcher Form wird das Protokoll geführt?
	Tester	Nehmen Tester an der Prüfung teil? */**
	Referenz-unterlagen	Welche Form von Referenzen gibt es, gegen die der Prüfling geprüft wird? ** ☐ Richtlinien ☐ Spezifikation
	Werkzeuge	Welche Werkzeuge werden eingesetzt, um die Prüfung zu steuern? ** ☐ Fragenkatalog ☐ Checkliste ☐ Prüfplan ☐ Musterdokument
	QS-Ingenieur	Ist ein Mitarbeiter speziell verantwortlich für die Qualitätssicherung? * Welche Rolle spielt er bei der Prüfung? * ☐ Moderator ☐ Gutachter ☐ Organisator ☐ andere: …
	Autor	Nimmt der Autor an der Prüfung teil? In welcher Rolle? *** ☐ Verteidiger des Prüflings ☐ Gutachter ☐ stiller Zuhörer
	Klient	Nimmt ein Kundenvertreter an der Prüfung teil? In welcher Rolle? * ○ Gutachter ○ stiller Zuhörer ○ Berater
	Projektleiter	Nimmt ein technisch Verantwortlicher (Projektleiter) an der Prüfung teil? Wer wird über die Ergebnisse der Prüfung informiert? */** ☐ Projektleiter (technisch Verantwortlicher) ☐ Manager (organisatorisch Verantwortlicher) ☐ Entwickler (technische Sachbearbeiter)

Anmerkungen:

* Hier muss ggf. eine neue Rolle erfasst werden.

** Mehrere Instanzen sind möglich, diese müssen jeweils namentlich erfasst werden.

*** Erfragen von Hintergründen, drückt sich nicht unbedingt sichtbar in der Darstellung aus.

Key-Interfaces bestimmter Prüfungen des Aktivitäts-Kontextes

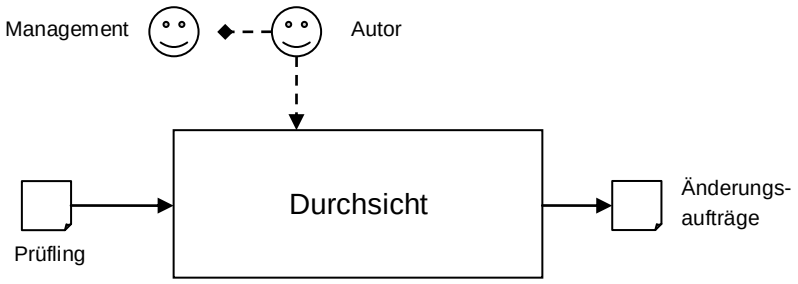


Abbildung 14: FLOW-Signatur der Aktivität *Durchsicht*

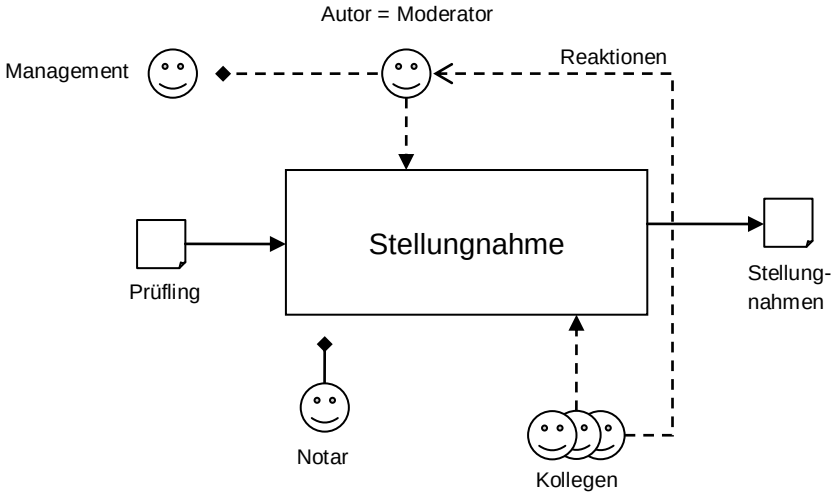


Abbildung 15: FLOW-Signatur der Aktivität *Stellungnahme*

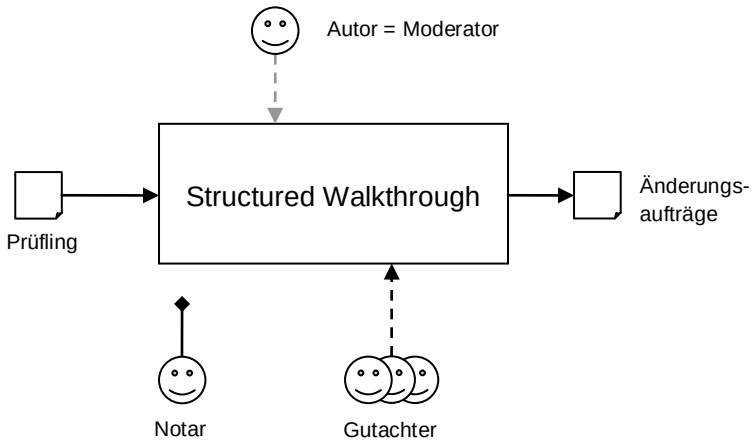


Abbildung 16: FLOW-Signatur der Aktivität *Structured Walktrough*

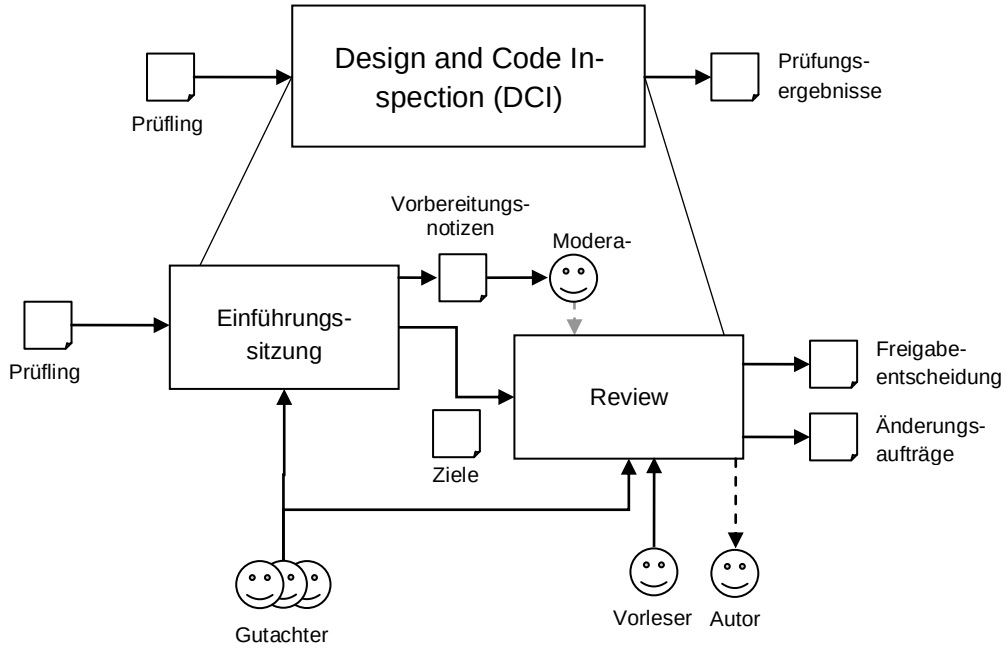


Abbildung 17: FLOW-Signatur der Aktivität *Design and Code Inspection*

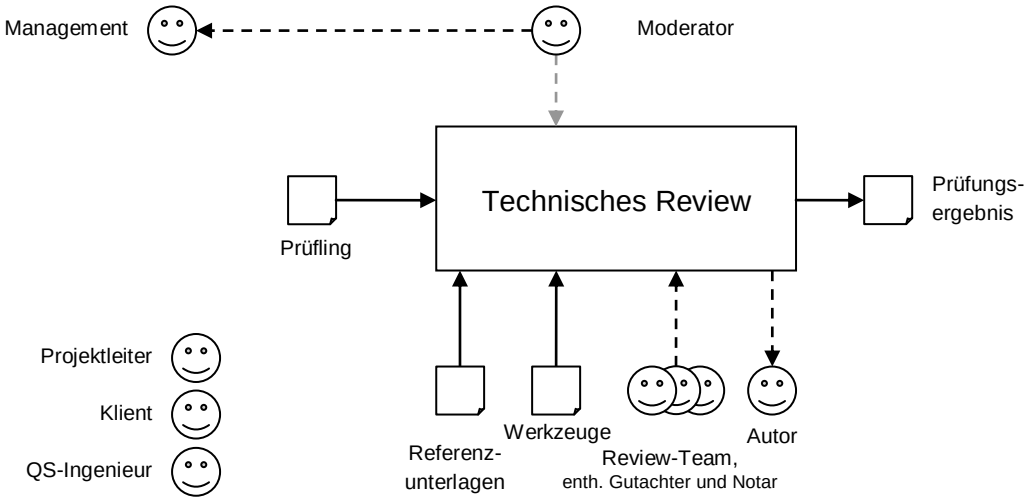


Abbildung 18: FLOW-Signatur der Aktivität *Technisches Review*

Technisches Review: Zusätzliche Fragen, abgeleitet aus Regeln

Review-Regel	Frage(n) bzw. Prüfung(en):
Trennung von Prüfung und Lösung	Werden Lösungsvorschläge für Änderungsaufträge erstellt?
Review-Sitzung dauert max. 2 Stunden	Wie lange dauert eine Sitzung?
Was gefordert wurde, muss auch geprüft werden.	Vergleichen von Vorgaben und Referenzunterlagen.
Jeder Gutachter soll angemessen gehört werden.	Vergleichen der Dauer, der zu prüfenden Artefakte und der Anzahl der Gutachter.

Beispiel-Konsultierung des Ratgebers

Die Anwendung des Ratgebers soll an einem durchgängigen Beispiel demonstriert werden. Die geschilderte Situation und die handelnden Personen sind fiktiv, das Beispiel lehnt sich jedoch an tatsächliche Situationen an, die während der Interviews diskutiert wurden. Ebenso wie im Ratgeber wurde in den Gesprächen der Weg von der Problematik zur Ursache nachgezeichnet. Hierbei wurden die Mittel der Ratgeber-Methode eingesetzt, also zunächst hauptsächlich Fragen vor dem Hintergrund einer FLOW-Perspektive.

*In den Filialen der **Oberbank AG** wird täglich ein Kassenabschluss durchgeführt. Die Kassierer jeder Filiale führen hierzu eine Kassenaufnahme durch. Der tatsächliche Bestand wird dann mit dem rechnerischen Bestand verglichen, der aus der Summe aller Buchungssätze des Tages hervorgeht. Sobald die Kasse „stimmt", übermittelt jede Filiale den Kassenabschluss an den Hauptrechner in der Zentrale, wo ein Gesamtabschluss durchgeführt wird und etwaige Kassendifferenzen ausgebucht werden.*

*Die **Supersoft GmbH** hat den Auftrag erhalten, dieses Verfahren in einer Software weitgehend zu automatisieren. Der verantwortliche Projektleiter ist Paul Leiter, dessen Team das Softwareprojekt kürzlich abschließen und an den Kunden ausliefern konnte. In 350 Filialen wurde die neue Software installiert und mittlerweile seit 3 Wochen zum Kassenabschluss eingesetzt. Jetzt erreichen Paul Leiter Beschwerden von Verantwortlichen der **Oberbank AG**. Das Programm liefert zwar zuverlässige, korrekte Zahlen, und die Kollegen in den Geschäftsstellen konnten auch schnell in die Bedienung eingewiesen werden – allerdings benötigt die Konsolidierung der Buchungssätze in jeder Filiale bis zu 10 Minuten. Die Übertragung der Daten an den Zentralserver dauert sogar noch länger. Der Feierabend der Mitarbeiter verzögert sich so nicht unwesentlich.*

Eine Überprüfung ergibt, dass das Hauptproblem darin technisch begründet liegt, dass der Kassenabschluss in allen Filialen gleichzeitig *durchgeführt wird.*

Paul Leiter ist zunächst ratlos. Denn natürlich wurde dieser Aspekt im Vorfeld bedacht, eine Überlastung des Zentralservers sollte die Software automatisch durch ein ausgeklügeltes Warteschlagen-System verhindern. Ein großer Teil des Entwicklungsaufwands war auf die Vermeidung genau dieser Problematik ausgerichtet gewesen. Die einzelnen Routinen waren zudem speziell auf Effizienz getestet worden. Er fragt sich, wie diese Problematik dennoch auftreten konnte und konsultiert den FLOW-Ratgeber, um den Ursachen auf den Grund zu gehen. Er beantwortet zunächst die Fragen des ersten Schrittes.

Im Ratgeber werden nach dem ersten Schritt folgende Ergebnisse festgehalten:

Betroffener Qualitätsbezug	produktbezogen			
Betroffener Qualitätsbereich 1	Brauchbarkeit			
Betroffener Qualitätsbereich 2	Nützlichkeit			
Betroffener Qualitätsaspekt	Effizienz			
Zeitpunkt der Feststellung	Betrieb			
Zu untersuchender Prozessbereich	A	S	E	C

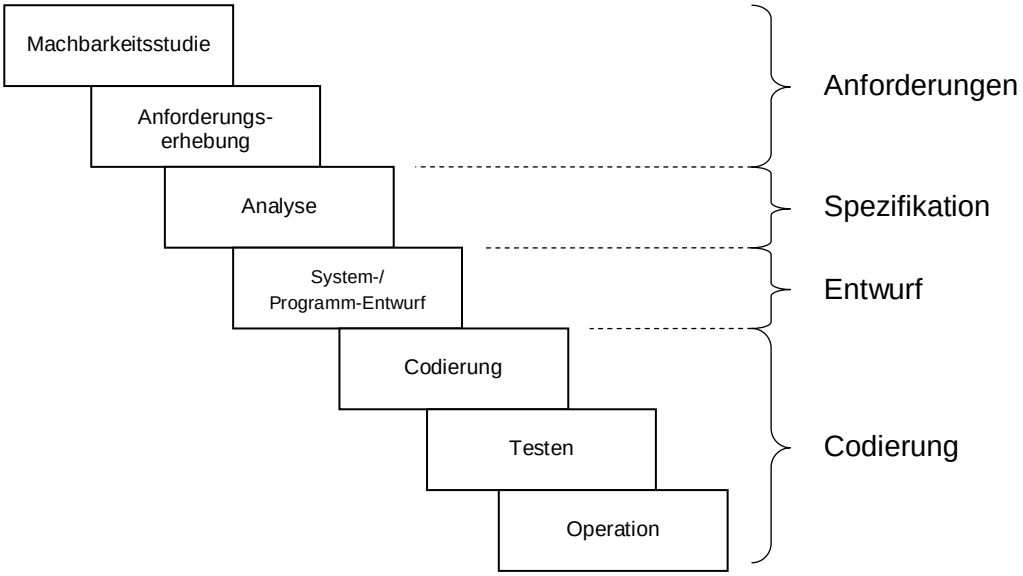
Diese Ergebnisse ermöglichen weiter, den Aktivitätskontext auf „Prüfungen“ festzulegen, denn dem Ratgeber bietet sich folgendes Bild:

(vgl. *Abbildung 12*, Seite 46) Zur Bewertung wird die Summe der Gewichte der vorhergehenden Knoten addiert mit den Bewertungen aller Verknüpfungen, über die der aktuelle Knoten erreicht wird.		Bewertung	
		direkt = 2	universal = 1
Mögliche Situationsmuster:	☑ Stagnation	0+2+1=<u>3</u>	
	☑ Ergebnisabweichungen	0+2=2	
	☑ Verschätzt!	0+2=2	
	☑ Prozessmüdigkeit	0+1=1	
Mögliche Ursachenmuster:	☑ „Hausmeister“	(3+2+2+1)+2+(4·1)=<u>14</u>	
	☑ „ungenauer Prüfer“	(3+2+2+1)+2+(4·1)=<u>14</u>	
	☑ „Inseldenken“	(2+2)+2+2=8	
	☑ fehlende Prozesstransparenz	(2+2)+2+2=8	
Mögliche Hintergründe:	☑ mang. Qualitätsbewusstsein	(14+14)+2+2=<u>32</u>	
	☑ verteiltes Team	8+8+2+2=20	
zu untersuchender Aktivitätskontext:	☑ Prüfungen: Signaturen von Software-Prüfungen		
	☐ Begegnungen (zB. Änderungsausschuss, Schlichtung, …)		
	☐ Beurteilungen (zB. Grobentwurf-Schätzung, Angebot, …)		
	☐ Bürokratie (zB. Dokumentationsprozess, Reviews, …)		

Der erste Schritt, die Orientierung, ist damit abgeschlossen. Mitgenommen in den nächsten Schritt wird der Fokus auf *Software-Prüfungen*, die im Bereich *Anforderungen* bis *Entwurf* zu untersuchen sind. Eher am Anfang des Entwicklungsprozesses werden dabei vor allem Situationen wie *Stagnation* vermutet. *„Hausmeister“*-Verhalten, *„ungenaue Prüfer“* oder *mangelndes Qualitätsbewusstsein* könnten eine Rolle spielen. Diese Ergebnisse werden an dieser Stelle bereits teilweise erläutert, um den Benutzer bei der Durchführung des nächsten Schrittes zu unterstützen.

Der zweite Schritt beginnt mit der Information des Benutzers, wie er die Suche nach Ursachen in seinem Entwicklungsprozess durchzuführen hat und wo er beginnen soll.

*Paul Leiter skizziert zunächst den Entwicklungsprozess, der in der **Supersoft GmbH** zur Anwendung kommt. Dieser folgt dem erweiterten Wasserfallmodell, das er entsprechend den Anweisungen des Ratgebers auf die vorgegebenen Prozessschritte „ASEC“ verteilt:*



Die Suche im Prozess, das „Flussaufwärtsschwimmen“, soll nun durchgeführt werden. Paul Leiter nimmt erstaunt den Hinweis des FLOW-Ratgebers zur Kenntnis, er könne vermutlich alle Schritte der Codierung *überspringen. Damit würde er beim* Entwurf *beginnen. Er traut dieser Aussage nicht und schließt die* Codierung *noch in die Suche mit ein.*

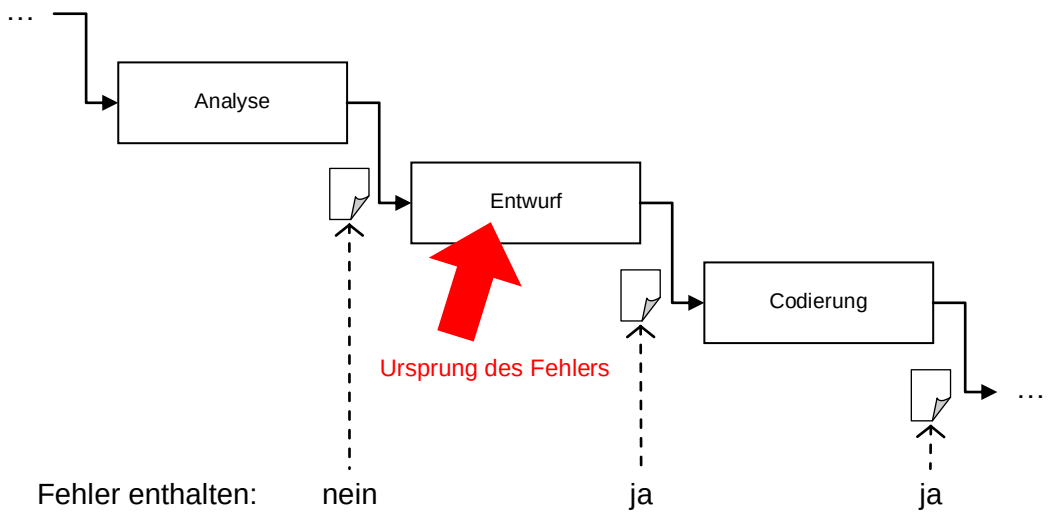
Der Benutzer muss sich nicht hundertprozentig an die Einschätzungen des Ratgebers halten, wenn er an den Hinweisen zweifelt.

Er stellt sich die vom Ratgeber vorgegebene Frage: „War der Fehler im ‚Output‘ dieses Schritts enthalten?“ – dies kann er für die Codierung klar mit ‚ja‘ beantworten, schließlich macht die erstellte Software Probleme. Auch ob „der Fehler im ‚Input‘ dieses Schritts bereits vorgegeben war“, wie die nächste Frage lautet, kann er sofort bejahen. Er weiß, dass bei der Implementierung keine zusätzlichen Fehler entstanden sein können, da alle durchgeführten Modultests erfolgreich ausgeführt werden konnten. „Wenn, dann sind diese Tests fehlerhaft…“, überlegt er, „demnach wäre der Fehler in dem Schritt zu suchen, in dem diese Tests erstellt wurden“.

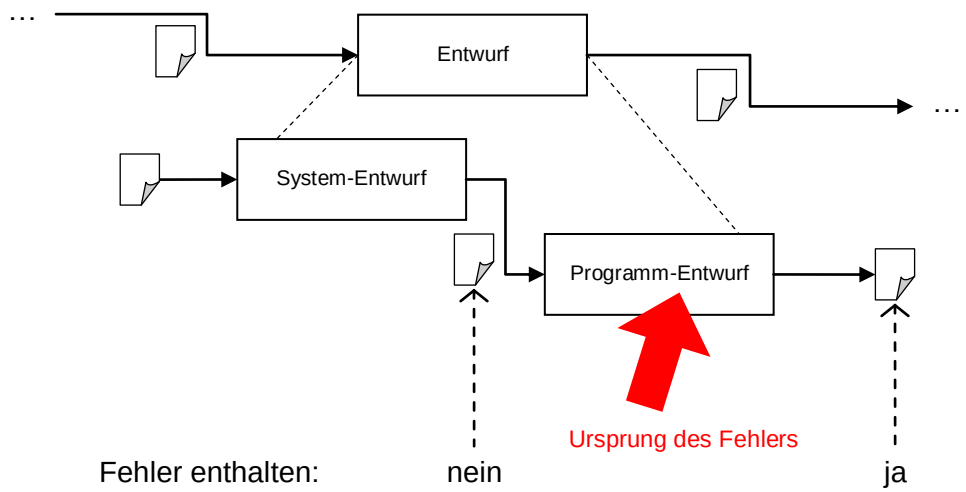
Überrascht stellt er fest, dass er nun die Codierung *als Fehlerquelle selbst eliminiert hat, die Voraussage des Ratgebers war also zutreffend. Er wendet sich dem* Entwurf *zu. Wieder wird dieselbe Frage nach Fehlern im ‚Output‘ dieses Schritts gestellt, entsprechend seiner vorhergehenden Überlegung beantwortet er sie wiederum mit ‚ja‘: Er geht davon aus, dass zu diesem Zeitpunkt die Fehler bereits „gemacht waren“. Die nächste Frage, den ‚Input‘ betreffend, kann er nicht auf Anhieb beantworten. Der Ratgeber rät, die entsprechenden ‚Inputs‘ durchzusehen, also die ‚Outputs‘ des vorhergehenden Schritts.*

Der Ratgeber gibt Anleitungen, wenn der Benutzer nicht sicher ist, wie er verfahren soll, wo er beispielsweise benötigte Informationen findet.

Paul wirft einen Blick auf die Dokumente, die aus der Analyse *entstanden sind und erinnert sich beim Blättern in der Spezifikation: Hier wurde der Aspekt ‚Effizienz' besprochen und das Problem der vielen gleichzeitigen Server-Verbindungen bereits vorhergesehen. Somit muss der Fehler später entstanden sein, indem dieser Aspekt wieder vernachlässigt wurde. Er beantwortet die Frage, ob der aktuelle Schritt bereits fehlerhafte Inputs vom Vorgänger übernommen hat, entsprechend mit „nein" und hat damit als zu untersuchenden Prozessschritt den* Entwurf *identifiziert:*



Der Entwurf gliedert sich im erweiterten Wasserfallmodell auf in System- *und* Programm-Entwurf*, weshalb Paul Leiter eine entsprechende Verfeinerung nach der Anleitung des Ratgebers vornimmt und die Ursache weiter „einkreist":*



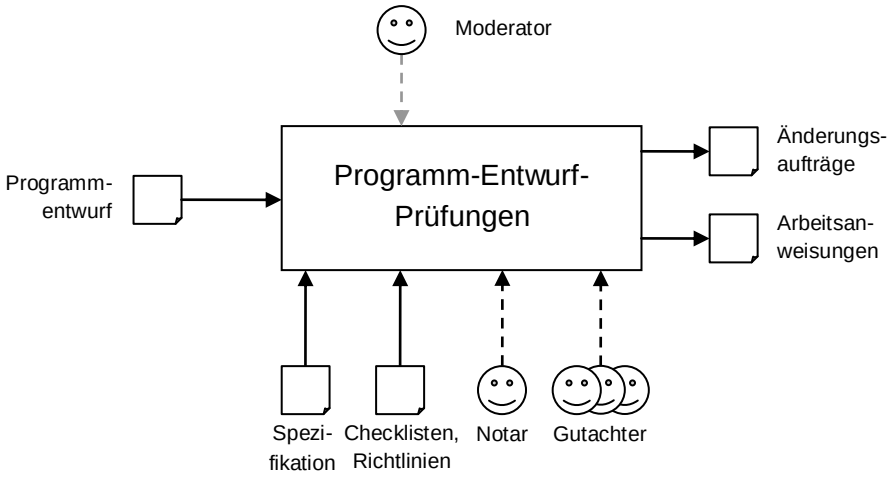
Der Ratsuchende wurde hier dazu gebracht, sich mit den Abläufen in seinem Entwicklungsprozess auseinanderzusetzen. Nach Abschluss dieses zweiten Schrittes sollte der Bereich, in dem der Fehler zu vermuten ist, zumindest sinnvoll eingeschränkt sein.

Der dritte Schritt verbindet nun die Erkenntnisse, die bisher gewonnen werden konnten und fragt den Benutzer nach Prüfungs-Aktivitäten, die im identifizierten Prozessschritt vorgenommen wurden.

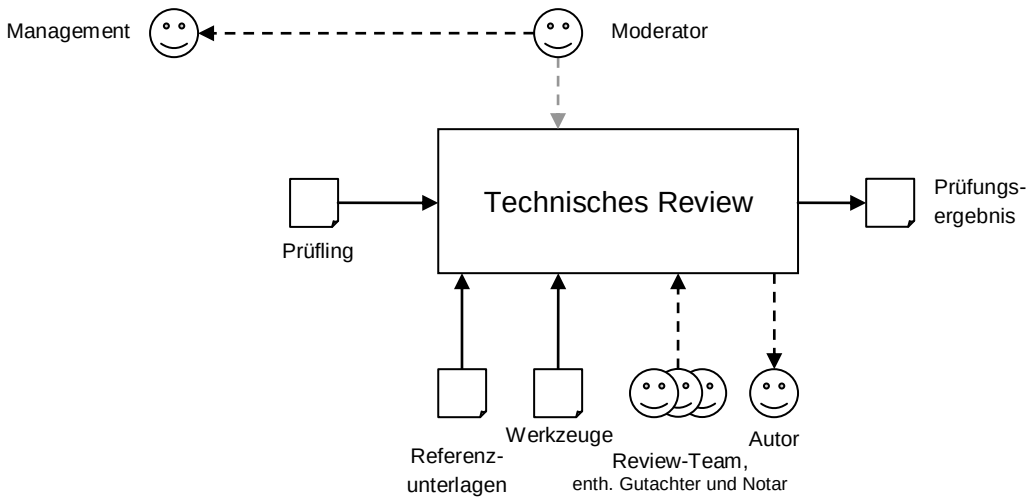
Paul Leiter bekommt den Hinweis, dass die Prüfungs-Situationen im benannten Prozessschritt untersucht werden sollten, da es möglich ist, dass hier Informationen „verschleppt" werden. Es sollen eine Reihe Fragen beantwortet werden, die die Ausführung von Prüfungen in diesem Abschnitt beleuchten sollen.

Einige Fragen kann er nicht alleine beantworten, da er nicht an Prüfungen teilnimmt. Er zieht einen Mitarbeiter hinzu, der am Entwurf mitgewirkt hat.

Die Befragungen ergeben die Beteiligung der folgenden Informationsspeicher: Ein *Moderator* organisierte und leitete die Sitzungen, in diesem Bereich wurden *Programm-Entwürfe* gegen die *Spezifikation* geprüft, *Checklisten* und *Richtlinien* wurden als Werkzeuge der Prüfung eingesetzt, *Änderungsaufträge* und *Arbeitsanweisungen* als Ergebnisse produziert. Weiterhin bestand das *Review-Team* aus vier Personen, die Sitzungen dauerten 1,5 Stunden und wurden *protokolliert*. Mit diesen Informationen kann der Ratgeber sich ein Bild der „Ist"-Situation machen, deren als FLOW-Signatur etwa so aussehen würde:



Hierin sind die meisten Key-Interfaces von *Technischen Reviews* enthalten:



Es kann also davon ausgegangen werden, dass die durchgeführten Prüfungen *Technische Reviews* „sein sollen“. Die Abweichungen gegenüber vollwertigen Reviews sind in der Gegenüberstellung zu erkennen und werden dem Benutzer präsentiert:

1. Der Autor des Prüflings wird nicht in die Prüfung integriert. Die erstellten Änderungsaufträge werden später vom ihm bearbeitet, es ist daher kritisch, dass er an der Sitzung teilnimmt und versteht, warum diese Änderungen notwendig sind.
2. Es werden Arbeitsanweisungen formuliert. D.h. die Gutachter werden zu Entwicklern, Korrektur und Prüfung werden nicht sauber getrennt. Dies ist ein Verstoß gegen eine Review-Regel (siehe z.B. [LUD06], S. 261f.), der zu Problemen führen kann.
3. Es ist keine Verbindung zum Management erkennbar. Reviews werden üblicherweise von einem Linienvorgesetzten oder Projektleiter gemanagt, der den Auftrag zur Erstellung des Prüflings gegeben hat und somit auch die Verantwortung für die Freigabe des Prüflings trägt.

Diese Abweichungen werden mit entsprechenden Lösungsvorschlägen (den Autor zu integrieren, in Prüfungen keine Korrekturen vorzunehmen und das Management einzubinden) dem Benutzer aufgezeigt und *erklärt*.

Die Darstellung der Abweichungen macht Paul Leiter klar, dass die Reviews in der Supersoft GmbH nicht so vorbildlich abliefen, wie er dachte. Ihm wird beim Lesen der Prüfungsprotokolle bewusst, dass einige Entwürfe nicht nur mehrfach geprüft wurden, sondern dass auch oft wiederholt dieselben Fehler angemahnt wurden.

Im Gespräch mit seinen Mitarbeitern erkennt er, dass die nicht eingebundenen Autoren häufig die Änderungen nicht umgesetzt haben, da sie nicht verstanden haben, warum die geforderten Änderungen notwendig waren. Es wird deutlich, dass die Gutachter einen besseren Überblick über das Gesamtkonzept und so mögliche Problemquellen früher erkannt hatten. Für die Architekten waren die Zusammenhänge nicht ersichtlich. Da sie nicht zu den Reviews eingeladen worden waren, konnten sie nicht nachvollziehen, wieso sie bestimmte Stellen ihres Entwurfs ändern sollten, die ihrer Meinung nach perfekt funktionierten. So kam es zur Umsetzung fehlerhafter Entwürfe, die erst im Zusammenspiel die von den Gutachtern befürchteten Auswirkungen auf die Effizienz hatten. Verschiedene Programmteile, die einzeln gut funktionierten (und so auch alle Modultests bestehen konnten), behinderten sich im Betrieb gegenseitig.[3]

Paul Leiter versteht nun, wie es zu der vorliegenden Problematik kommen konnte und weiß, wie er die Reviews in seinem Softwareentwicklungsprozess verbessern kann. Er weiß jetzt, wie es zu „Stagnation“ bei Prüfungen kommen kann und wird diese künftig verhindern.

[3] Hinweis: Auch wenn die Hintergrundsituation frei erfunden ist, handelt es sich bei der beschriebenen Ursache um ein reales Beispiel, das in den Interviews mehrfach (von unterschiedlichen Gesprächspartnern) geschildert wurde.

4 Bewertung und Ausblick

4.1 Bewertung

Obwohl der Informationsflussgedanke nachvollziehbar und naheliegend ist, fällt es schwer, eine ausdrückliche „Gebrauchsanweisung“ zu formulieren. Genau dies war aber die Aufgabe dieser Arbeit, FLOW sollte zu einer praktischen Anwendung geführt werden.

Es war unklar, ob eine Anleitung zu bestimmten FLOW-Techniken die Antwort wäre, oder vielmehr mit Hilfe von Ergebnissen der FLOW-Forschung eine eigene Technik zu entwickeln sein würde. Die Lösung, die in dieser Arbeit vorgestellt wird, verbindet beide Gedanken. So kommen mit den FLOW-Signaturen existente Techniken praktisch zur Anwendung. Wichtiger ist aber wohl die Erkenntnis, FLOW-Ideen und mittels Informationsflussforschung entwickelten *Best Practices* zu verwenden, um die Lücke zwischen Theorie und Praxis zu schließen.

Dies geschieht über das Konzept des FLOW-Ratgebers, der an „normalen“ Situationen in Softwareprojekten ansetzt und versucht, alltägliche Problematiken auf ihre Informationsfluss-Ursachen zurückzuführen.

Während der Arbeit hat sich die Vermutung bestätigt, dass die in [GE2007] eingeführten FLOW-Patterns bislang auf einer unzugänglichen Ebene präsentiert werden und von einem Ratgeber nur indirekt genutzt werden können. FLOW zu einer praktischen Anwendbarkeit zu führen setzt zunächst praktische, lebensnahe Muster voraus. Die Verbindung zu den FLOW-Patterns ist anschließend herzustellen, denn eins dieser „heranführenden Muster“ kann wieder etliche solche FLOW-Patterns enthalten.

Allgemein steckt FLOW als „Motor“ überall in diesem Ratgeber-Konzept. Die resultierenden Vorgehensweisen sind immer „FLOW-motiviert“. Wie aber die Metapher vom weisen Mann verdeutlicht, muss das verwendete FLOW-Wissen nicht unbedingt immer an der Oberfläche liegen. Auch wenn Ratschläge genannt werden, die mit Informationsflüssen direkt nichts zu tun zu haben scheinen, sind die Gründe, warum sie in diesem Ratgeber auftauchen, auf jeden Fall von Untersuchungen aus der FLOW-Perspektive beeinflusst.
Entsprechende Erfahrungen wie *Best Practices* einfließen zu lassen ist sogar selbst Teil des FLOW-Gedankens. Ein Beispiel sind die Review-Regeln, die natürlich schon vor dieser Arbeit verfasst wurden, hier aber als Grundlage für das erstellen von Ratschlägen am Ende der Beratung dienen. Für die Regel „Trennung von Prüfung und Korrektur“ gilt unter anderem die Begründung, dass die Rollen von Entwickler und Prüfer getrennt werden müssen, damit der „ändernde Prüfer“ nicht zu den Entwicklern überläuft und anschließend nicht mehr nach Fehlern sucht, sondern die Korrektheit seiner Entscheidung verteidigt ([LUD06]).
Was sonst steckt hinter dieser Überlegung als der FLOW-Gedanke, der das optimale Fließen von Informationen zwischen den Informationsspeichern zum Ziel hat. Es lässt sich sogar verschwommen eine Mischung der FLOW-Pattern „Stille Post“ und „Person als Senke“ dahinter erkennen: Der „ändernde Prüfer“ nimmt Informationen auf und fängt an, diese einerseits durch seine sich wandelnde Rolle falsch zu verstehen und sie andererseits zum Schutz seiner Entwicklung nicht mehr weiterzugeben. Eine solche Konstellation ist mit FLOW-Pattern schwer darstellbar, daher wurde in dieser Arbeit die Wichtigkeit einer neuen Analysestufe aufgezeigt, auf der „heranführende Muster“ die Verbindung zwischen realen Problematiken und der FLOW-Theorie herstellen.

4.2 Ausblick und Weiterführung der Forschung

4.2.1 Weitere Ursachen- und Situations-Muster

In den Interviews zu dieser Arbeit konnten erst einige wenige Beispiele gesammelt werden. Weitere „heranführende Muster“ auf der Ebene der Situationen und Ursachen zu identifizieren und in die Methode einzubinden ist entscheidend für eine Erweiterung und Verbesserung des Ratgebers. So kann dessen Leistungsfähigkeit erhöht und der Beratungsspielraum erweitert werden: Die Schlussfolgerungen an dieser Stelle und die späteren Ratschläge können umso präzisier und differenzierter ausfallen, von je mehr möglichen Situationen und Ursachen und deren Verknüpfungen der Ratgeber Kenntnis hat. Das englische „knowledge“ (mit „Kenntnis“ statt dem üblichen „Wissen“ übersetzt) zeigt, wie die ihm bekannten Muster hier die „Knowledge Base“ einer künstlichen Intelligenz hinter einem automatisierten Ratgeber darstellen.

4.2.2 Umsetzungen des Ratgebers

Der Ratgeber besteht im Moment als „körperlose Methode“. Wie bereits an mehreren Stellen in der Arbeit erwähnt, ist eine Umsetzung des Ratgebers in Form einer Software als ein sogenanntes *Expertensystem* durchaus denkbar. Der Prototyp zeigt, dass die Methode grundsätzlich keine Elemente enthält, die nicht in einer Software realisiert werden könnten. An bestimmten Punkten wäre so eine Umsetzung sogar wünschenswert, zum Beispiel beim Pattern-Matching in Schritt 3 könnte die Gegenüberstellung automatisiert werden, auch die Wissensdatenbank mit den Verknüpfungen von Problematiken, Situationsmustern und Aktivitäts-Kontexten lässt sich in der EDV erfassen. Eine Benutzeroberfläche könnte die Bedienbarkeit verbessern, indem z.B. das Prozessmodell des Benutzers einmalig erfasst und in das „ASEC-Modell“ übertragen wird, so dass es bei künftigen Verwendungen zur Verfügung steht.

4.2.3 Zusammenhang von Prozess- und Produkt-Qualitätsaspekten

Es konnte nur ein Teil der Qualitätsaspekte abgedeckt werden, die den Hintergrund für die Klassifizierung von Problematiken bilden. Ein Gedanke, der während dieser Arbeit leider nicht weiter verfolgt werden konnte, betrifft die Zusammenhänge zwischen Produkt- und Prozessqualität. Es war teilweise extrem schwierig, die in den Interviews gesammelten Beispiel-Problematiken den Qualitätsaspekten eindeutig zuzuweisen. Es scheint hier eine mehrstufige Abhängigkeit vorzuliegen, durch die *Prozess*-Qualitäten, die Problematiken „in Mitleidenschaft“ ziehen, sich im Endeffekt auf Qualitätseigenschaften des *Produkts* auswirken. Für eine Einordnung über mehrere Bereiche hinweg hätte es aber einer größeren Menge von Situationsmustern und Beispielen bedurft, die nicht mehr zu erheben waren. In dieser Richtung bietet sich vielleicht eine Möglichkeit, die Kategorien zur Unterscheidung von Problematiken sozusagen „mehrdimensional“ zu erweitern.

5 Verzeichnisse

5.1 Verwendete Literatur

[KRS2004] Hans-Bernd Kittlaus, Christoph Rau und Jürgen Schulz; *Software-Produkt-Management – Nachhaltiger Erfolgsfaktor bei Herstellern und Anwendern*; Springer, 2004 (ISSN: 1439-5428)

[GE2007] Ge, Xiaoxuan; *FLOW Patterns: Beschreibung und Diskussion von Informationsflussmustern in der Softwareentwicklung*; Fachgebiet für Software Engineering der Leibniz Universität Hannover, 2008

[GE2007PK] Ge, Xiaoxuan; *FLOW-Pattern-Katalog zur Masterarbeit*; Fachgebiet für Software Engineering der Leibniz Universität Hannover, 2008

[WCSQ05] Kurt Schneider und Daniel Lübke; *Systematic Tailoring of Quality Techniques*; Fachgebiet für Software Engineering der Leibniz Universität Hannover, 2005

[LUDMOD02] Jochen Ludewig; *Modelle im Software Engineering – eine Einführung und Kritik*; M. Glinz, G. Müller-Luschnat (Hrsg.): Modellierung 2002 Tutzing, 25.-27. März 2002

[LUD06] Jochen Ludewig und Horst Lichter; *Software Engineering - Grundlagen, Menschen, Prozesse, Techniken*; Dpunkt Verlag, 2006 (ISBN: 3-89864-268-2)

[GAMMA94] Erich Gamma et al.; *Design Patterns – Elements of Reusable Object-Oriented Software;* Addison-Wesley Professional, 1994 (ISBN 0-2016-3361-2)

[RUP] International Business Machines Corporation; Website: *Rational Unified Process – Proven best practices for software and systems delivery and implementation and effective project management;* http://www-306.ibm.com/software/awdtools/rup, November 2008

[SADT] Structured Analysis and Design Technique, Quellen im Web z.B. bei Wikipedia unter http://en.wikipedia.org/wiki/Structured_Analysis_and_Design_Technique, November 2008

[*] T. DeMarco; *Structured Analysis and Systems Specification*; Prentice Hall, 1978 (ISBN: 0-13-854380-1)

[*] Allen H. Dutoit und Bernd Brügge; *Objektorientierte Softwaretechnik*; Pearson Studium, August 2004 (ISBN: 978-3-8273-7082-2)

[*] Tarek Abdel-Hamid und Stuart E. Madnick; *Software Project Dynamics – An Integrated Approach*; Prentice Hall, 1994 (ISBN: 0-13-822040-9)

[*] Kurt Schneider; *Ausführbare Modelle der Software-Entwicklung*; vdf Hochschulverlag, 1994 (ISBN: 3-7281-2148-7)

[*] Thomas Funke et al.; *Softwareentwicklung mit ISO 9000 in mittelständischen Unternehmen*; Springer, 2000 (ISBN: 3-540-66754-7)

[*] Bob Hughes und Mike Cotterell; Software Project Management; McGraw-Hill, 2002 (ISBN: 0-07-709834-X)

[*] verwendete Literatur ohne Verweise im Text

5.2 Abbildungen

Unbeschriftete Abbildungen

Beschriftete Abbildungen

Abbildungen ohne Quellenangabe sind selbst erstellt.

5.3 Tabellen