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

Analyse und Definition von verschiedenen FLOW-Modellen

Studienarbeit

im Studiengang Mathematik mit Studienrichtung Informatik

von

Elizaveta Sarkisyan

Prüfer: Prof. Dr. Kurt Schneider
Betreuer: Dipl.-Wirt.-Inform. Daniel Lübke

Hannover, 14. Februar 2006

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Während der verschiedenen Phasen der Softwareentwicklung werden viele unterschiedliche Dokumente erstellt. In Abbildung 1 ist ein typischer Grundriss eines Softwareentwicklungsprozesses dargestellt. Während der fünf Phasen werden neun verschiedene Dokumente erstellt.

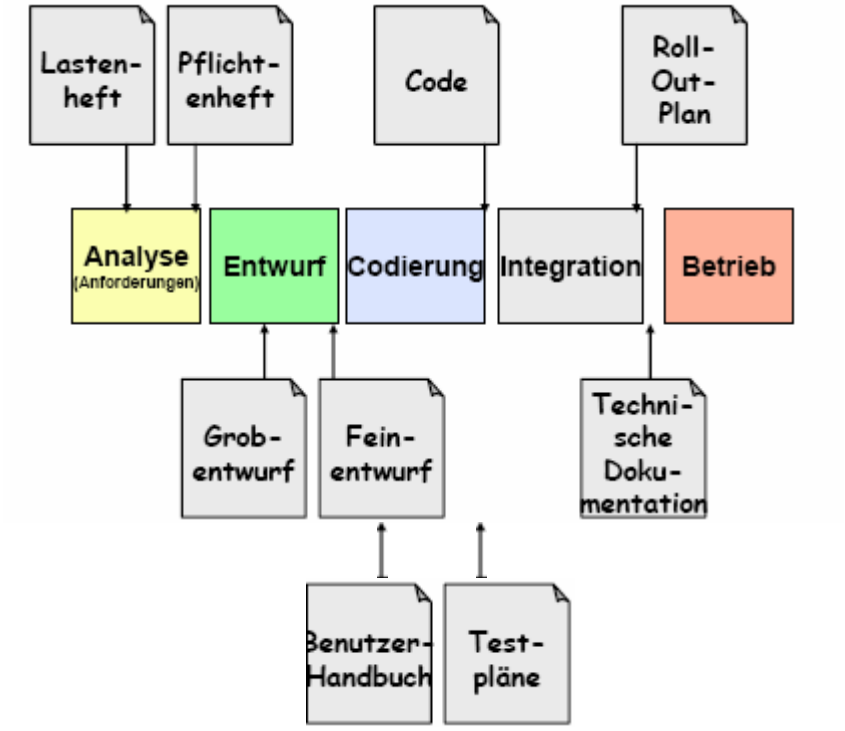


Abb.1: Beispielhafter Softwareentwicklungsprozess [1]

Oft werden noch mehr Dokumente erstellt und davon auch mehrere Versionen. Daher ist es in der Praxis unmöglich, alle Dokumente mittels vollständiger Reviews innerhalb einer vorgegebenen Zeitplanung zu prüfen. Die Vorteile der Reviews, wie z.B. frühere Fehlererkennung, sprechen jedoch dafür sie nicht einzuschränken. Es entsteht ein Dilemma, dessen Lösung nicht so trivial ist. Stünden mehr Zeit und mehr Mittel für die Projekte zur Verfügung, wäre dieses Problem gelöst. Doch aufgrund wirtschaftlicher Einschränkungen ist dies in der Praxis nicht gegeben. Qualität der Software vorzutäuschen oder zu ignorieren ist sicherlich kein Lösungsansatz.
FLOW ist ein Forschungsprojekt des Fachgebiets Software Engineering an der Universität Hannover, das sich mit diesem Problem beschäftigt. Die FLOW-Modelle, die im Rahmen dieses Projekts herausgearbeitet wurden, bieten an, eine realistische Lösung auf eine systematische Weise, wie z.B. Varianten von Qualitätstechniken gemäß Projektprioritäten (z.B. Flexibilität, Zuverlässigkeit usw., zu finden.

1.2 Aufgabenstellung

Im Rahmen dieser Studienarbeit werden vorhandene FLOW-Modellversuche analysiert und modifiziert. Verschiedene Konzepte und Notationen werden als ein allgemeines FLOW-Modell mathematisch definiert.
Zuletzt wird eine einfache Klassenbibliothek implementiert. Diese soll einen Grundstein zu Programmierung einer Software legen, die das Erstellen und Modifizieren des FLOW-Modells erlaubt.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit ist folgendermaßen strukturiert:

- Zunächst wird in Kapitel 2 das grundlegende Konzept der FLOW-Modelle dargestellt. Es werden FLOW-Kernvorstellungen genannt und anhand eines kleinen Beispiels die vorteilhafte Nutzung dieser Notation gezeigt.
- In Kapitel 3 fängt der Hauptteil dieser Arbeit an. Das FLOW-Modell besteht aus Objekten (Dokumente, Prozess, Menschen) und Beziehungen (Kommunikation) zwischen diesen Objekten. Deswegen wird das Modell als Graph modelliert. Zusätzlich wird eine graphische Notation bereitgestellt. Um diese Notation besser zu verstehen und auch definieren zu können, werden in diesem Kapitel zunächst die Grundlagen der Graphentheorie vorgestellt.
- Im nächsten Kapitel werden die vorhandenen Modellversuche analysiert und die mögliche Verschmelzung der Modelle diskutiert. Außerdem wird eine mögliche Verfeinerung des Prozesses und dadurch entstehenden Optionen der FLOW- Modelldarstellung vorgestellt.
- In Kapitel 5 wird anhand eines Klassendiagramms die Implementierung einer Klassenbibliothek gezeigt. Diese soll ein Fundament zu einer Benutzeroberfläche zum Erstellen und Modifizieren der FLOW-Modelle legen.
- Die Arbeit schließt mit Kapitel 6 mit einer Zusammenfassung und einem kurzen Ausblick.

2 Grundlegendes Konzept

FLOW ist ein Forschungsprojekt des Fachgebiets Software Engineering an der Universität Hannover.
Es dient der Anpassung (Verbesserung) der Qualitätstechniken und des Softwareentwicklungsprozesses durch Bereitstellung von Konzepten und einer graphischen Notation, die auf mehreren Kernvorstellungen beruht:

- die Konzentration auf Informationsflüsse innerhalb eines Projektes
- sowohl Datenflüsse aus der Kommunikation zwischen Menschen als auch Datenflüsse aus Dokumenten modellieren
  → die Art des Datenflusses wählen, die die Prioritäten des Projektes besser unterstützt
- die Erfahrung als Datenfluss betrachten

FLOW kann verwendet werden, um zu modellieren, zu analysieren, und zu helfen, komplette Softwareentwicklungsprozesse zu verbessern. Häufig wird es auf eine definierte Situation, einen Referenzprozess oder eine Teilmenge eines Projektes angewendet. [2]

FLOW bietet die Möglichkeit nur die Attribute des Originals zu erfassen, die dem Nutzer (z.B. Softwareentwickler) relevant erscheinen. Infolgedessen und weil FLOW eine Abstraktion eines Projektes ist, sprechen wir vom FLOW-Modell.
Dieses Modell bietet einen sehr guten und einfachen visuellen Vergleich (Suche nach Ähnlichkeiten und Abweichungen) von Prozessen/Mustern.

So ist es z.B. möglich, eine technische Inspektion und einen Walkthrough durch einfache Visualisierung wie in Abb.2 und Abb.3 miteinander zu vergleichen:

Gemeinsamkeiten beider Prozesse sind dabei:
- Mindestens drei Gutachter sehen sich den Prüfling unabhängig an
- Steuerung durch Moderator
- Autor erläutert einen Ausschnitt des Prüflings

Im Vergleich zu der technischen Inspektion enthält das Walkthrough:
- Keine Checkliste
- Keine Einzelprotokolle
- Aber direktes Feedback an den Autor
- Protokoll

Durch die graphische Notation wird nicht nur Zeit beim Vergleichen gespart, sondern auch die Mühe die Unterschiede bzw. Gemeinsamkeiten zu merken.

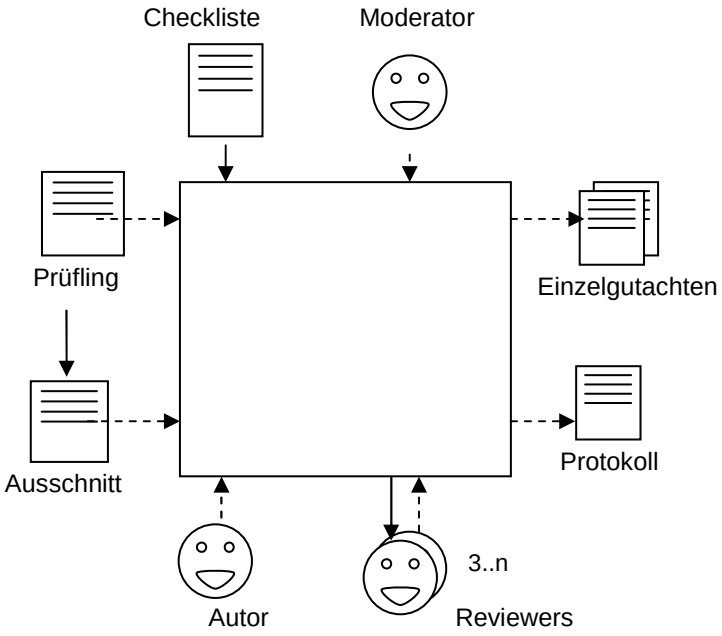


Abb.2: Technische Inspektion (vgl. [3])

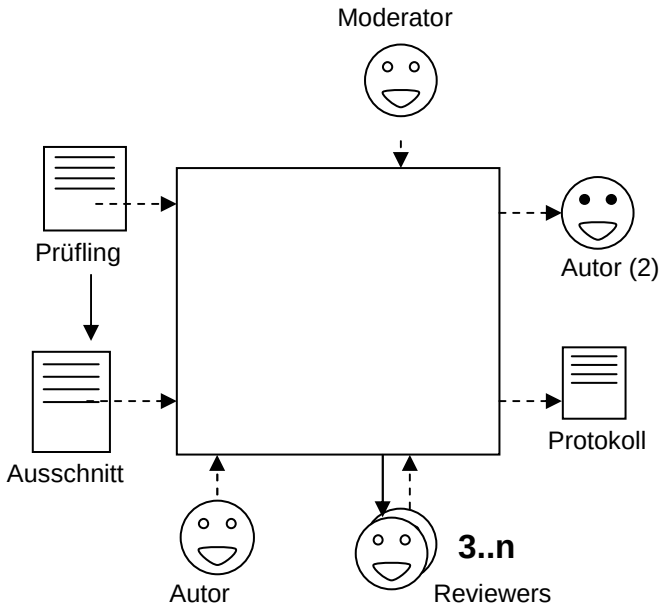


Abb.3: Walkthrough (vgl. [3])

Um auf unterschiedliche Prioritäten verschiedener Projekte eingehen zu können, wurden mehrere FLOW-Modelle entwickelt.

3 Definitionen

Das FLOW-Modell basiert auf der Graphentheorie. Anlässlich dessen werden beginnend die Grundlagen der Graphentheorie vorgestellt.

3.1 Exkurs zur Graphentheorie

Ein Graph besteht aus einer Menge von Punkten, zwischen denen eine Verbindung in der Form von Linien existiert. Die Punkte nennt man Knoten (oder auch Ecken) und die Linien nennt man meist Kanten(manchmal auch Bögen). Auf die Form der Knoten und Kanten kommt es im Allgemeinen dabei nicht an.
Man unterscheidet in der Graphentheorie vor allem zwischen ungerichteten und gerichteten Graphen sowie Graphen mit Mehrfachkanten und ohne Mehrfachkanten.

Ein Graph *G* ist ein Tupel (*V*, *E*), wobei *V* eine Menge von Knoten und *E* eine Menge von Kanten bezeichnet. Dabei ist *E* in

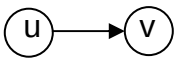
- ungerichteten Graphen ohne Mehrfachkanten eine Teilmenge aller 2-elementigen Teilmengen von *V*,
- gerichteten Graphen ohne Mehrfachkanten eine Teilmenge des kartesischen Produktes *V* x *V*,
- ungerichteten Graphen mit Mehrfachkanten eine Multimenge über der Menge aller 2-elementigen Teilmengen von *V*,
- gerichteten Graphen mit Mehrfachkanten eine Multimenge über dem kartesischen Produkt *V* x *V*,
- Hypergraphen eine Teilmenge der Potenzmenge von *V*.

Graphen, in denen zwei Knoten auch durch mehrere Kanten verbunden sein können, werden Multigraphen genannt. In einfachen Graphen ist solche Verbindung nicht erlaubt.
Hypergraphen sind solche, bei denen im Gegensatz zu einfachen Graphen Kanten mehr als nur zwei Knoten verbinden können. Um die Hypergraphen darzustellen, zeichnet man in der Regel eine Menge von Punkten, die den Knoten entsprechen, welche durch geschlossene Linien umkreist werden.

Ein gerichteter Graph G = (V, E) besteht

- aus einer endlichen, nicht leeren Menge V = $\{v_1,...,v_n\}$ von Knoten und
- einer Menge E $\subseteq$ V $\times$ V von geordneten Paaren e = (u,v), den Kanten .

Jede Kante (u,v) $\in$ E hat einen Anfangsknoten u und einen Endknoten v und damit eine Richtung von u nach v (u = v ist möglich). u und v sind Nachbarn, bzw. adjazent. (vgl. [5])

Graphische Darstellung: 

Oft erweitert man Graphen G=(V,E) zu knotengefärbten bzw. kantengefärbten Graphen, indem man das Tupel (V,E) ergänzt. Z.B. zu einem Tripel (V,E,f), wobei f eine Abbildung von E in die Menge der natürlichen Zahlen ist. Anschaulich gibt man jeder Kante damit eine Farbe. Ist f eine Abbildung von E/V, spricht man von Kantenfärbung/Knotenfärbung.

In vielen Bereichen der Wissenschaft und auch im alltäglichen Leben findet die Graphentheorie ihre Anwendung. Systeme, die aus Objekten und Beziehungen zwischen diesen Objekten bestehen, werden oft als Graphen modelliert.

3.2 FLOW-Graph

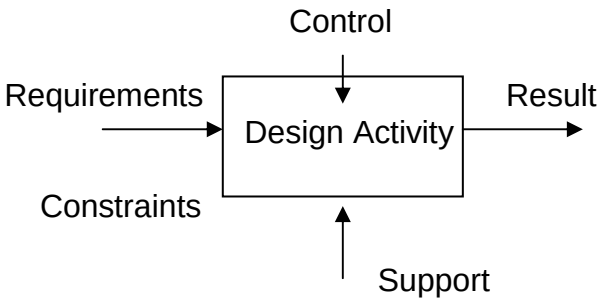
Das FLOW – Modell ist ein gerichteter Graph ohne Mehrfachkanten G=(V,E,a,f):

$V = P \cup D \cup M$ ist die Knotenmenge mit

- P : Menge der Knoten, die Prozesse darstellen
- D : Menge der Knoten, die Dokumente darstellen
- M : Menge der Knoten, die Menschen (Gutachter, Moderator, Protokollant, Autor usw.) darstellen

$E \subseteq V \times V \times B$ ist die Menge der gerichteten Kanten zwischen den Knoten. Hier ist

- B : Menge der Benutzungsart, die in die Prozesse reinfliest (Control, Support, Input/Output)
- e = (v, w, b) mit v als Startknoten und w als Endknoten von e und $b \in B$.



Die Kanten werden als Pfeilen dargestellt, wobei die Art der Pfeile modellabhängig ist.

$a : V \rightarrow \mathbb{N} \times T \times N$ ist eine Abbildung

- $\mathbb{N}$: die Menge der natürlichen Zahlen. Anschaulich gibt man jedem Knoten damit eine Anzahl. Bei Prozessen ist die Anzahl immer 1.
- T : Typ des Knotens. Abhängig von der Knotenmenge kann es z.B. sein:
  - P: LIDs, Walkthrough, Review, Quality Gate

- D: Checkliste, Lastenheft, Prüfling, Protokoll, Entwurf

- M: Autor, Moderator, Gutachter, Protokollant

N : Name bzw. Bezeichnung des Knotens (z. B. Frau Meyer bzw. 1.1). Dies ist keine vordefinierte Menge, sondern ein String der eingegeben werden kann. Soll das Modell allgemein gehalten werden, tragen die Knoten keine Namen.

$f : V \rightarrow A \times F$ eine Abbildung

A : Art der Darstellung einer Kante ($\longrightarrow$, $\dashrightarrow$)

F : Farbliche Darstellung einer Kante

Darstellung der Knoten :

	Knotenmenge	Anzahl = 1	Anzahl > 1
Prozess	P		
Dokument	D		
Mensch	M		

Die Darstellung der Kanten ist modelabhängig. Diese wird bei Beschreibung der Modelle im Kapitel 4 explizit vorgestellt.

4 Modelle

Die verschieden graphischen Notationen der FLOW-Modelle, die durch Darstellung der Datenflüsse anhand der Kanten entstehen, ermöglichen den Softwareentwicklern auf unterschiedliche Prioritäten verschiedener Projekte einzugehen.
Die Modelle werden immer in derselben Struktur vorgestellt: Zuerst wird eine beispielhafte Abbildung dargestellt. Im Folgenden werden diese kurz charakterisiert. Anschließend werden mögliche Vor- und Nachteile beschrieben.
Dies geschieht anhand eines Beispiels G=(V,E,a,f), in dem ist:

$V = P \cup D \cup M$

$P = \{p_1\}$
$M = \{m_1, m_2, m_3, m_4\}$
$D = \{d_1, d_2, d_3, d_4\}$

$a : V \rightarrow \mathbb{N} \times T \times N$

$a(p_1)$ = (1, Review, null)
$a(m_1)$ = (1, Moderator, null)
$a(m_2)$ = (3, Gutachter, null)
$a(m_3)$ = (1, Autor, null)
$a(m_4)$ = (1, Protokollant, null)
$a(d_1)$ = (1, Checkliste, null)
$a(d_2)$ = (1, Prüfling, null)
$a(d_3)$ = (1, Ausschnitt, null)
$a(d_4)$ = (1, Protokoll, null)

null ist ein neutrales Element aus der Menge N, der eingesetzt wird falls der Knoten kein Name enthält.

$E = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$

$e_1 = (d_1, p_1,$ Control)
$e_2 = (m_1, p_1,$ Control)
$e_3 = (m_2, p_1,$ Input/Output)
$e_4 = (p_1, m_2,$ Input/Output)
$e_5 = (m_3, p_1,$ Input/Output)
$e_6 = (m_3, d_2,$ Input/Output)
$e_7 = (d_2, d_3,$ Input/Output)
$e_8 = (m_4, d_4,$ Input/Output)
$e_9 = (d_3, p_1,$ Support)

Die Abbildung f, die die Kantenfärbung darstellt, ist modellabhängig und wird bei jeweiligen Modelle definiert.

4.1 Modell „Quellenabhängige Kante“

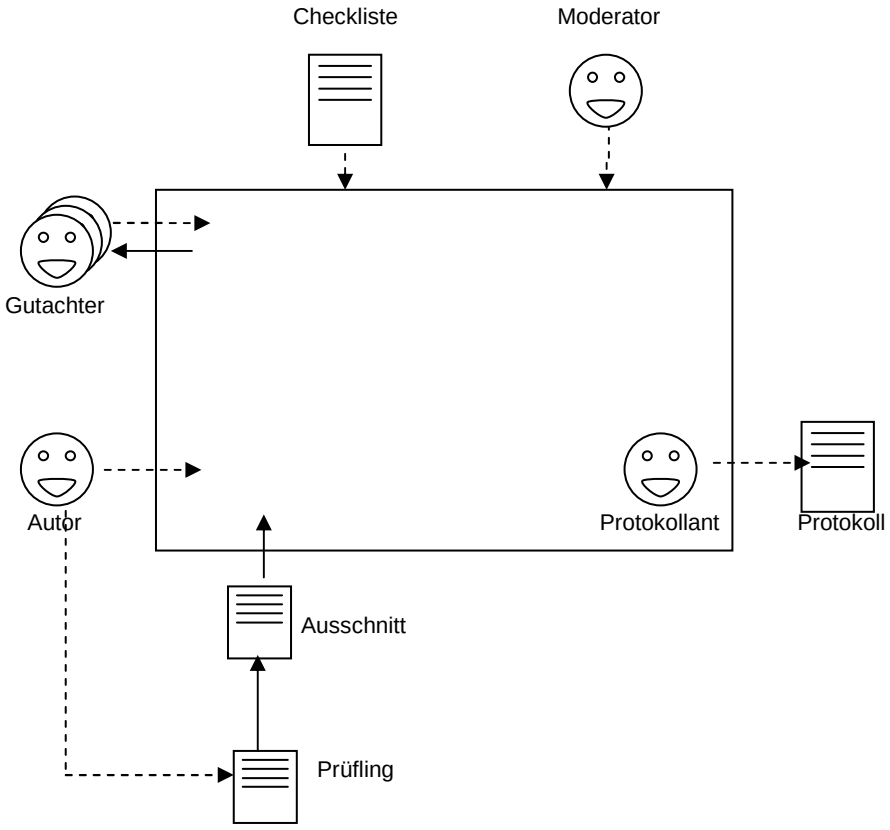


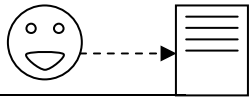
Abb.4: Beispiel für Modell „Quellenabhängige Kante“ (vgl. [3])

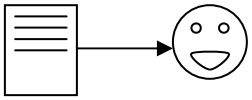
$f : V \rightarrow A \times F$

$f(e_{1}) = (0, \text{schwarz})$
$f(e_{2}) = (1, \text{schwarz})$
$f(e_{3}) = (1, \text{schwarz})$
$f(e_{4}) = (0, \text{schwarz})$
$f(e_{5}) = (1, \text{schwarz})$
$f(e_{6}) = (1, \text{schwarz})$
$f(e_{7}) = (0, \text{schwarz})$
$f(e_{8}) = (1, \text{schwarz})$
$f(e_{9}) = (0, \text{schwarz})$

wobei 0 für durchgezogene Kante steht und 1 für gestichelte.
Schwarz ist die neutrale Farbe.

Die Darstellung der Kanten hängt nur von dem Startknoten ab:

Die Information kommt von den Menschen (verbal): 

Die Information kommt aus einer dokumentierten Quelle: 

Vorteil: Die Darstellung der Kanten ist sehr übersichtlich, lenkt nicht von Inhalt ab, trägt jedoch eine Information in sich.

4.2 Modell „Zwischenprozesse“

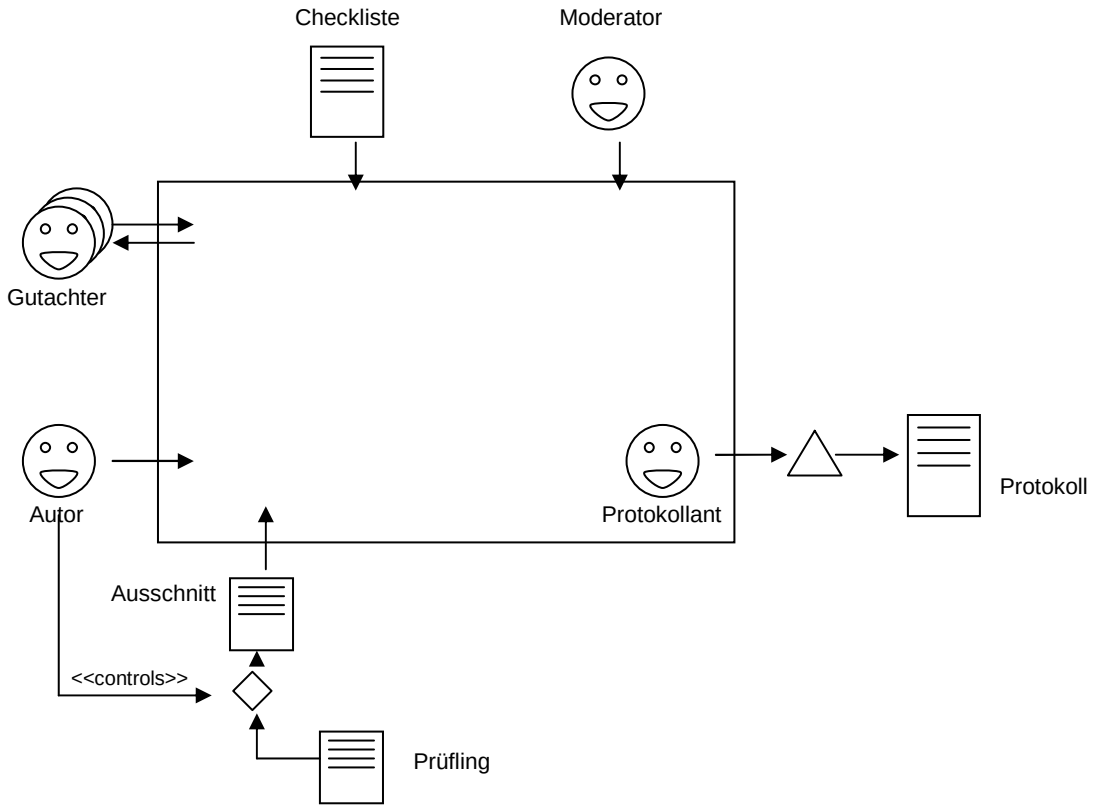


Abb.5: Beispiel für Modell „Zwischenprozesse“ (vgl. [3])

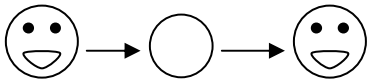
$f : V \rightarrow A \times F$

$f(e_{1}) = (0, \text{schwarz})$
$f(e_{2}) = (0, \text{schwarz})$
$f(e_{3}) = (0, \text{schwarz})$
$f(e_{4}) = (0, \text{schwarz})$
$f(e_{5}) = (0, \text{schwarz})$
$f(e_{6}) = (0, \text{schwarz})$
$f(e_{7}) = (5, \text{schwarz})$
$f(e_{8}) = (3, \text{schwarz})$
$f(e_{9}) = (0, \text{schwarz})$

wobei 0 für durchgezogene Kante steht. Schwarz ist die neutrale Farbe.

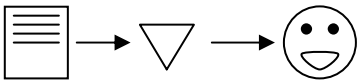
Die Darstellung der Kante hängt von Start- und Endknoten ab:

Zwischen Menschen:
f(e) = (2,schwarz)

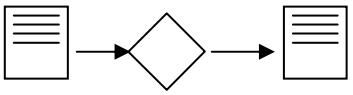


Zwischen Mensch und Dokument:
f(e) = (3,schwarz)

Zwischen Dokument und Mensch:
f(e) = (4,schwarz)



Zwischen zwei Dokumenten:
f(e) = (5,schwarz)



Kanten die den Prozess als Black Box mit den anderen Knoten verbinden werden als einfache Pfeile dargestellt, da man den Start- oder Endknoten nicht genau kennt.

Die Menge der Knoten V wird erweitert um eine neue Menge der Zwischenknoten Z.
Also $V' = V \cup Z$. Die Zwischenknoten sind eine durch Start- und Endknoten beschränkte Art von Prozessen.

Ein Beispiel der graphischen Notation des Modells „Zwischenprozesse“ ist in Abb. 5 dargestellt. Bei der Analyse dieses Modells wurde festgestellt:

Vorteil: Man kann diese Knoten wie Prozesse bzgl. der Kanten behandeln und damit auch Art der Informationsflüsse zu unterscheiden. Außerdem könnte man falls notwendig eine Verzweigung „ innerhalb“ der Pfeilen darstellen. Durch diese Darstellung entsteht die Möglichkeit die Kanten mit der Zusatzinformation zu ergänzen.

Nachteil: Diese Notation kann schnell unübersichtlich werden. Dieses Modell ist zu komplex. Eine Kante muss in Allgemein hier als Hyperkante behandelt werden.

Aufgrund dieser Nachteile, wurde die Entscheidung getroffen, dieses Modell nicht zu benutzen. Sollte die Notwendigkeit in solche Zwischenknoten bestehen, die dann nicht durch Kanten dieser Art, sondern als Prozess – Knoten einführen. Für den Fall, dass diese Entscheidung verworfen wird, wird im Folgenden dieses Modell in der Verbindung mit den anderen diskutiert.

Um den Inhalt der Informationsflüssen darzustellen benutzt man Modell „Drei-Farben“.

4.3 Modell „Drei-Farben“

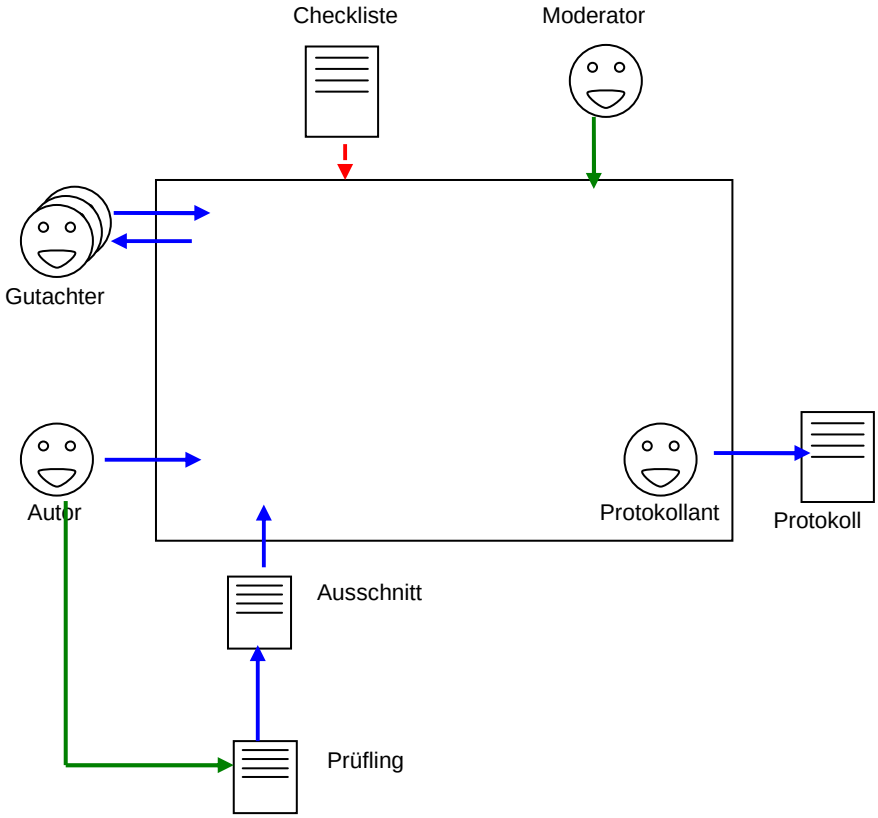
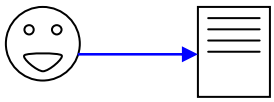


Abb.6: Beispiel für Modell „Drei-Farben“ (vgl. [3])

$f : V \rightarrow A \times F$

$f(e_1) = (0, \text{rot})$
$f(e_2) = (0, \text{grün})$
$f(e_3) = (0, \text{blau})$
$f(e_4) = (0, \text{blau})$
$f(e_5) = (0, \text{blau})$
$f(e_6) = (0, \text{grün})$
$f(e_7) = (0, \text{blau})$
$f(e_8) = (0, \text{blau})$
$f(e_9) = (0, \text{blau})$

Hier werden die Kanten als durchgehende Pfeile dargestellt, die aber in Abhängigkeit von Inhalt verschieden gefärbt sind:

Projektinformation	Blau	
Erfahrung	Grün	

Grundwissen Rot

Wie in Abb.6 als Beispiel dargestellt sieht man, dieses Modell bringt einen Vorteil aber auch Nachteile mit sich.

Vorteil: Datenflussinhalte werden berücksichtigt. Anhang dieses Modells kann man sofort sehen an welcher Stelle die Erfahrung eingesetzt werden muss.

Nachteile: Die Anschauung zwischen wem der Datenfluss stattfindet, entfällt. Dies ist besonders relevant für die Datenflusse, die aus dem, als Black Box dargestelltem, Prozess rausfliesen.
Durch farbige Darstellung entstehen zusätzliche Ansprüche an die eventuell erforderliche Endgeräte sowie Drucker, Kopierer.

Um die Vorteile aller Modelle zu nutzen und Nachteilen dabei möglichst zu beseitigen, wurde versucht die Modelle zu mischen. Im Folgenden wird auf die Vorteile gemischter Modelle sowie dabei entstehende Probleme eingegangen.

4.4 Modell „Quellenabhängige Kante + Zwischenprozesse“

In Modell „Quellenabhängige Kante“ und „Zwischenprozesse“ tragen die Kanten die gleiche Informationsart. Das ist der Grund, warum die Implementation des Mischmodells nur dann möglich ist, wenn man für jede Kante eine Abfrage einbauen würde, welches Modell genutzt werden soll. Das würde zu unnötig vielen Schritten führen und sehr aufwendig sein.
Somit soll man sich von Anfang an für ein der Modellen entscheiden.
Da Modell „Zwischenprozesse“ verworfen wurde, wird diese Kombination nicht weiter betrachtet.

4.5 Modell “Quellenabhängige Kante + Drei-Farben“

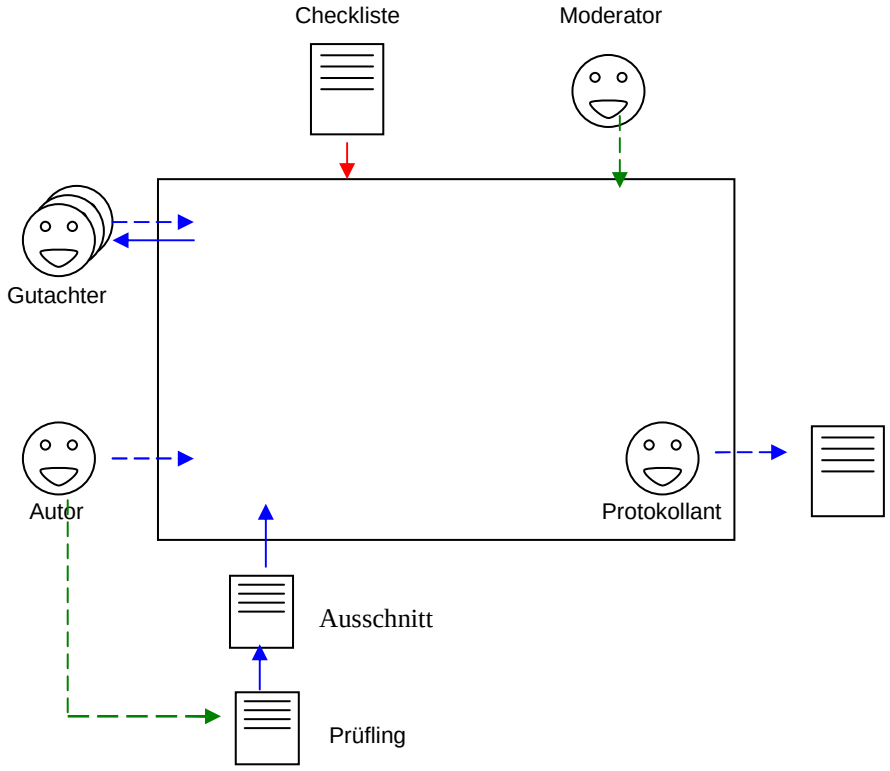


Abb.7: Beispiel für Modell “Quellenabhängige Kante + Drei-Farben“ vgl. [3])

$f : V \rightarrow A \times F$

$f(e_{1}) = (0, \text{rot})$
$f(e_{2}) = (1, \text{grün})$
$f(e_{3}) = (1, \text{blau})$
$f(e_{4}) = (0, \text{blau})$
$f(e_{5}) = (1, \text{blau})$
$f(e_{6}) = (1, \text{grün})$
$f(e_{7}) = (0, \text{blau})$
$f(e_{8}) = (1, \text{blau})$
$f(e_{9}) = (0, \text{blau})$

wobei 0 für durchgezogene Kante steht und 1 für gestichelte.

Aufgrund der verschiedenen Informationsarten, die in Modellen „Quellenabhängige Kante“ und „Drei-Farben“ als Kanten dargestellt werden, tritt hier keine Kollision zwischen Modellen auf. Außerdem der Nachteil des Modells „Drei-Farben“, dass Anschauung zwischen wem der Datenfluss stattfindet, entfällt, wird beseitigt. Die beiden Modelle erweitern einander.
Die Zusammennutzung ist vorteilhaft und ist Beispielhaft in der Abb.7 dargestellt.

4.6 Modell „Zwischenprozesse+ Drei-Farben“

Die Modellvariante „Zwischenprozesse+ Drei-Farben“ist analog zu Modell “Quellenabhängige Kante + Drei-Farben“ aufgebaut.

Die Modelle „Quellenabhängige Kante“ bzw. „Zwischenprozesse“ und „Drei-Farben“ können nach Belieben teilweise oder vollständig gemischt werden.

Durch ausgewähltes Mischen ist es möglich die Aufmerksamkeit auf einen bestimmten Informationsinhalt zu lenken oder einfach nicht relevante Information weg zulassen und damit bessere Übersicht zu bekommen.
Möchte man einen Informationsinhalt besonders hervorheben, z. B. Erfahrung, dann erweitert man z.B. das Modell „Quellenabhängige Kante“ nur um eine Farbe, in diesem Fall grün, und lässt andere Kanten schwarz wie in Abb.8.

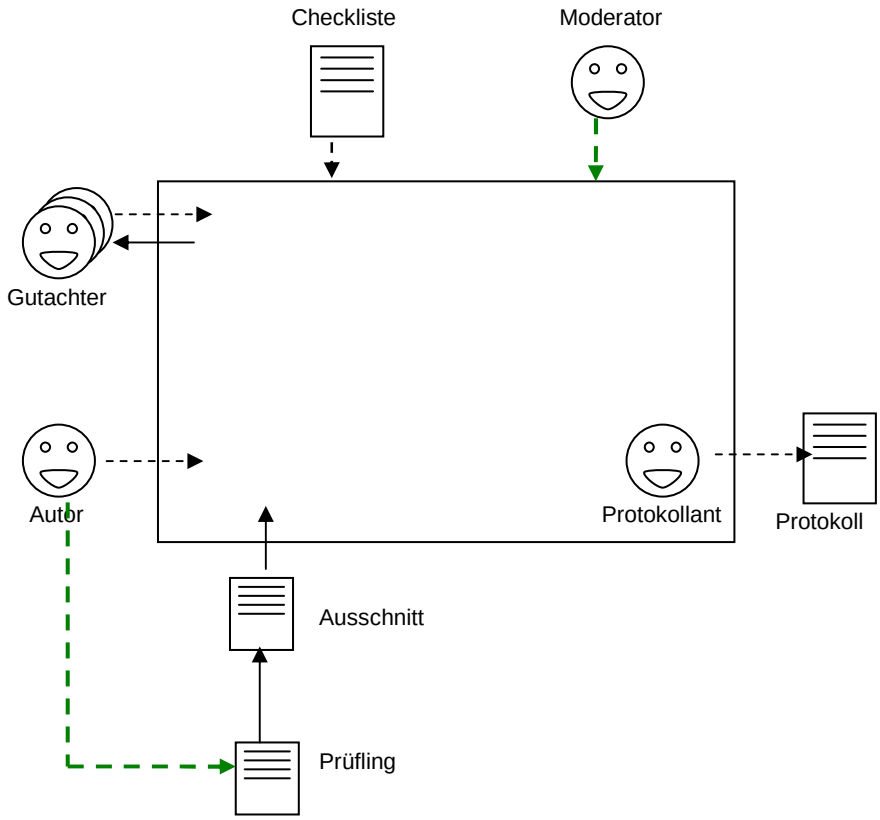


Abb.8: Beispiel für Modell “Quellenabhängige Kante + Drei-Farben“ (vgl. [3])

Somit enthält man 3 verschiedene FLOW- Modelle: Modell „Quellenabhängige Kante“, Modell „Drei-Farben“ und Modell “Quellenabhängige Kante + Drei-Farben“.

4.7 Prozessverfeinerung

Eine weitere optionale Darstellung des FLOW- Modells entsteht durch Prozessverfeinerung. Es besteht die Möglichkeit den Prozess, der vorher als Blackbox dargestellt wurde, zu durchleuchten. Dies kann nützlich sein, um z. B. während der Planung eines Projektes zu überlegen, was in der Blackbox geschieht und es darzustellen. Dabei können Verzweigungen innerhalb der Pfeile, wie in der Abb.9 dargestellt ist, entstehen. Ist es der Fall, gibt es verschieden Möglichkeiten dieses Problem zu lösen. Diese sind:

- In einem bestehenden Diagramm
  - Verzweigung „innerhalb“ der Pfeile
    Dies ist möglich durch Hypergraphen zu erreichen, jedoch sind diese sehr schwer zu implementieren und die dahinter stehende Theorie ist komplexer.
  - Einführung eines neutralen Knoten: Die Kanten gehen dann von ihm aus.
  - Lösche die zu „verzweigende“ Kante und führe gleich mehrere Kanten von dem Ursprungsknoten aus.
- Sich von Anfang an entscheiden was innerhalb/außerhalb der Black – Box bleibt
  → keine Verzweigung nötig.

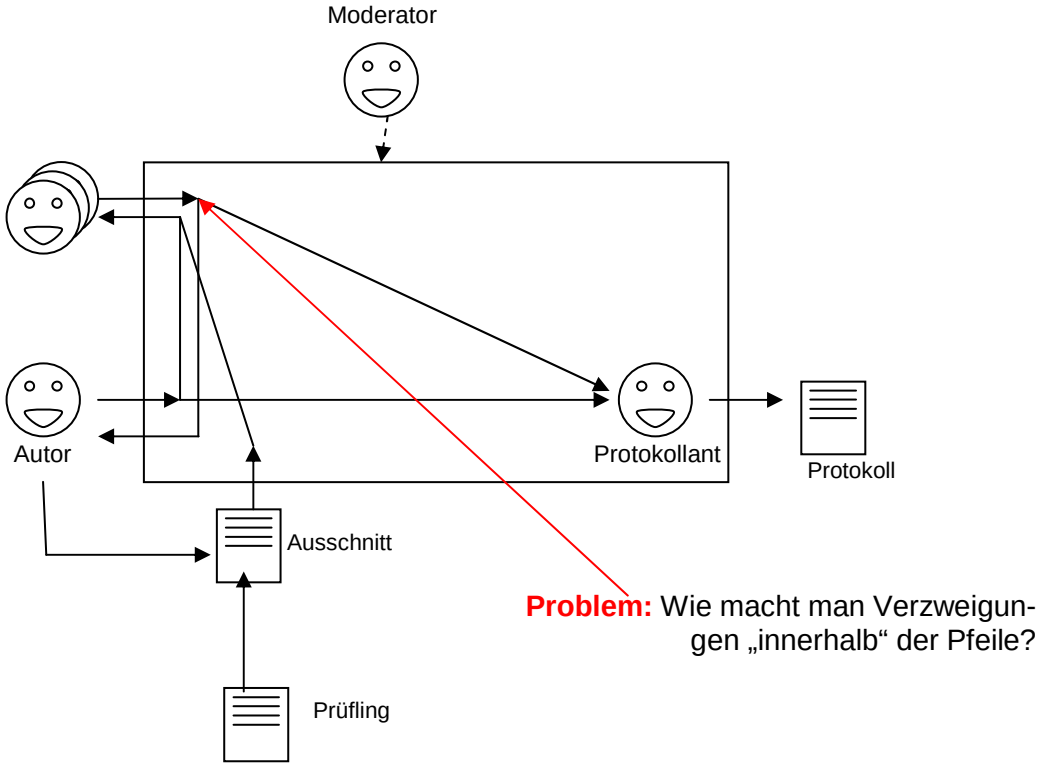


Abb. 9: Blackbox –Verfeinerung (vgl. [4])

Die oben beschriebenen Modelle sind allgemein gehalten und eignen sich sehr gut bei der Planung eines Prozesses.
Führt man dann dieses Prozess durch (z.B.: Review wie in Abb.10), verändert sich einwenig das allgemeine Modell. Es sind nicht mehr irgendwelche Menschen, sondern identifizierte Personen mit Namen.

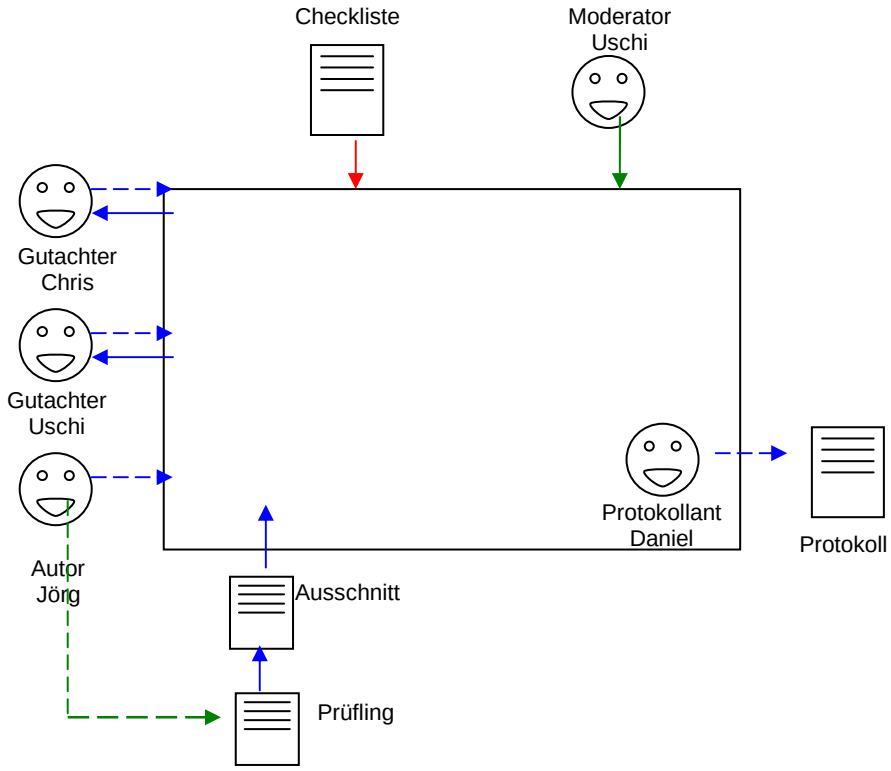


Abb.10: Review(vgl.[3])

Hier ist:

$V = P \cup D \cup M$

$P = \{p_1\}$
$M = \{m_1, m_{21}, m_{22}, m_{23}, m_3, m_4\}$
$D = \{d_1, d_2, d_3, d_4\}$

$a : V \rightarrow \mathbb{N} \times T \times N$

$a(p_1) = (1, Review, null)$
$a(m_1) = (1, Moderator, null)$
$a(m_{21}) = (1, Gutachter, null)$
$a(m_{22}) = (1, Gutachter, null)$
$a(m_{23}) = (1, Gutachter, null)$
$a(m_3) = (1, Autor, null)$
$a(m_4) = (1, Protokollant, null)$
$a(d_1) = (1, Checkliste, null)$
$a(d_2) = (1, Prüfling, null)$

$a(d_3)$ = (1, Ausschnitt, null)
$a(d_4)$ = (1, Protokoll, null)

$E = \{ e_1, e_2, e_{31}, e_{32}, e_{33}, e_{41}, e_{42}, e_{43}, e_5, e_6, e_7, e_8, e_9\}$

e_1 = (d_1, p_1, Control)
e_2 = (m_1, p_1, Control)
e_{31} = (m_{21}, p_1, Input/Output)
e_{32} = (m_{22}, p_1, Input/Output)
e_{33} = (m_{23}, p_1, Input/Output)
e_{41} = (p_1, m_2, Input/Output)
e_{42} = (p_1, m_2, Input/Output)
e_{43} = (p_1, m_2, Input/Output)
e_5 = (m_3, p_1, Input/Output)
e_6 = (m_3, d_2, Input/Output)
e_7 = (d_2, d_3, Input/Output)
e_8 = (m_4, d_4, Input/Output)
e_9 = (d_3, p_1, Support)

$f : V \rightarrow A \times F$

$f(e_1)$ = (0, rot)
$f(e_2)$ = (1, grün)
$f(e_{31})$ = (1, blau)
$f(e_{32})$ = (1, blau)
$f(e_{33})$ = (1, blau)
$f(e_{41})$ = (0, blau)
$f(e_{42})$ = (0, blau)
$f(e_{43})$ = (0, blau)
$f(e_5)$ = (1, blau)
$f(e_6)$ = (1, grün)
$f(e_7)$ = (0, blau)
$f(e_8)$ = (1, blau)
$f(e_9)$ = (0, blau)

Dabei ist

- $\forall v \in V$: $a(v) = (1, t, n)$ mit $t \in T$, $n \in N$.
  Hier: die Anzahl von Knoten und Kanten verdreifacht sich. Anstelle eines Knotens m_2 mit $a(m_2)$ = (3, Gutachter, null) entstehen drei Knoten: m_{21}, m_{22}, m_{23}. Sowie aus dazugehörigen Kanten e_3 und e_4 entstehen e_{31}, e_{32}, e_{33}, e_{41}, e_{42}, e_{43}.
- Kommt eine Person mehrmals vor (Moderator = Gutachter = Uschi) werden dafür trotz dem mehrere Knoten angelegt um:
  - Die Allgemeine Struktur nicht neu definieren zu müssen
  - Leichte allgemeine Prozesse mit konkreten Projekten vergleichen zu können

Durch Modifizieren eines allgemeinen Prozesses zu einem konkreten verfeinerten Prozess entsteht weitere optionale Darstellung eines FLOW-Modells.
Die durch Anpassen einen gegebenen Prozesses an die Bedürfnisse eines konkreten Projektes (Tailoring) entstehende Einschränkungen werden sofort

sichtbar. Mit Hilfe dieser graphischen Notation können sogar die Tailoring- Regel ausgearbeitet werden.

5 Implementierung einer Klassenbibliothek

Nach dem die FLOW-Modelle analysiert und definiert wurden, wurde eine Benutzungsoberfläche entwickelt, die das Erstellen und Modifizieren der Modelle ermöglicht. Dazu wurde das MVC-Entwurfsmodell, dass ein Programm in drei Einheiten strukturiert, benutzt: Modell, View und Controller Um dieses Architekturmuster zu realisieren, wurde das Observer Pattern benutzt.
Die Modell-Einheit repräsentiert die Kernfunktionalität und besteht aus mehreren Klassen: *Kante*, *FlowModel* und *AbstraktKnoten* mit den Unterklassen *Prozess*, *Dokument* und *Mensch*. Diese wurden als eine Klassenbibliothek implementiert, wie in Abb.11 abgebildet. Die Modell-Klassen sind Observable.
Eine Kante e=(v,w,b) wird in der Klasse *Kante* erstellt durch getStartKnoten() = v, getEndKnoten() = w und getBenutzungsart() = b. Die Färbung der Kanten erfolgt mit den Methoden getDarstellungsart()/setDarstellungsart() und getKantenfarbe()/setKantenfarbe().
Aufgrund unterschiedlicher Symbole für die verschiedenen Knotenarten wurden 3 Unterklassen der Klasse AbstraktKnoten implementiert. Infolge der Vereinbarung, dass die Anzahl des Prozesses immer nur eins ist, wird die Methode setAnzahl() in der Klasse Prozess überschrieben. Es wurden für alle vorhandenen Attribute aus den Definitionen in Kapitel 3 entsprechende Methoden erstellt. Zusätzlich werden für die Erstellung eines Knotens in der AbstraktKnoten-Klasse Kantenlisten erstellt, um die doppelverketteten Listen als Datenstruktur zu erreichen.
Die Klasse FlowModel mit Methoden zum Erstellen und Modifizieren eines FLOW-Modells hat die Funktion eines Container.

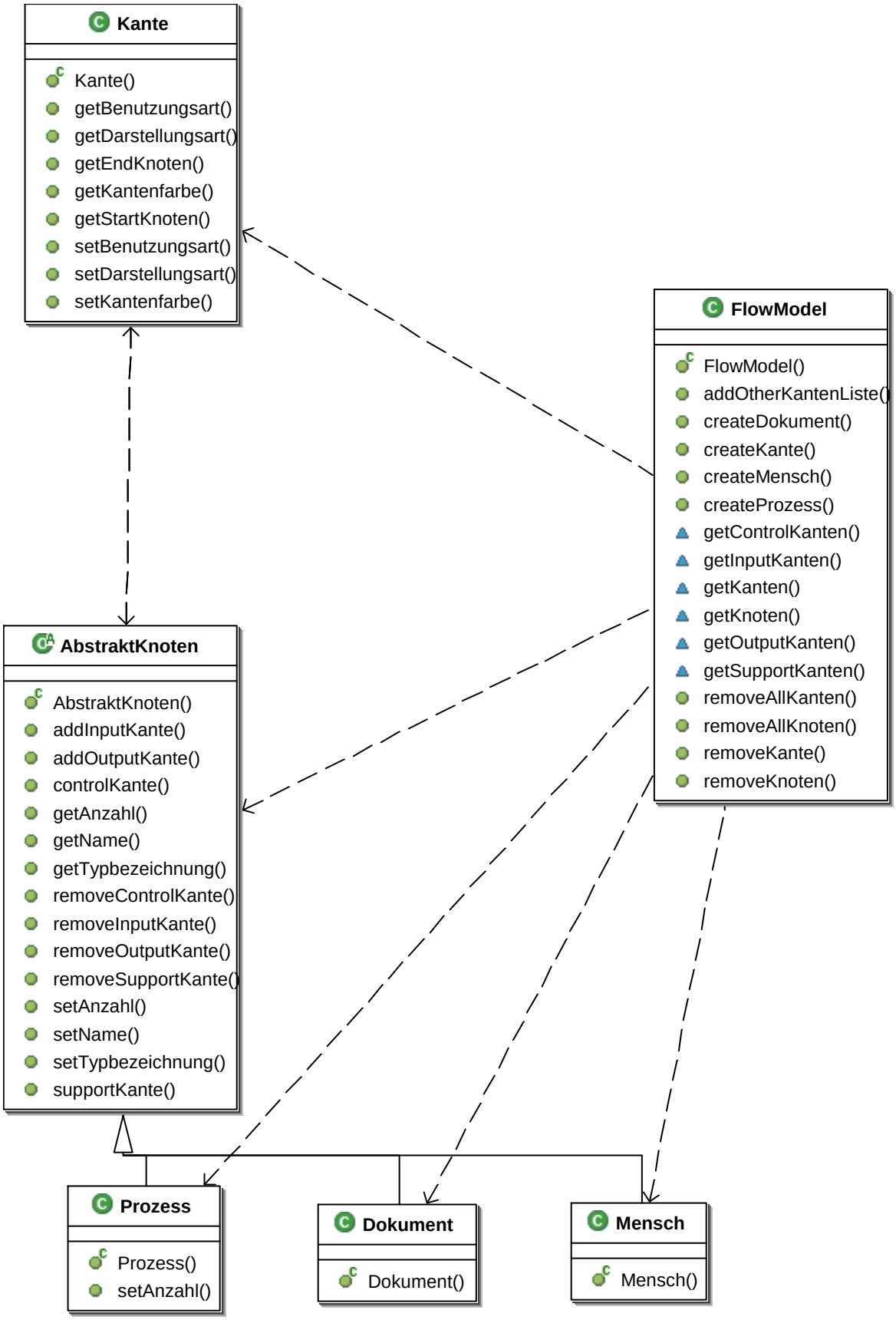


Abb. 11: Klassendiagramm

6 Zusammenfassung und Ausblick

Die Anwendung eines Software-Entwicklungsprozesses ist in den meisten Organisationen inzwischen zu einer Selbstverständlichkeit geworden. Jedoch können nicht alle Eigenheiten jedes Projektes im allgemeinen Prozess erfasst werden. Deswegen muss der Prozess dem konkretem Projekt angepasst werden. Um das zu Erleichtern und auch einen Prozess zu beschreiben eignet sich das FLOW-Modell. FLOW-Modelle, im Gegensatz zu anderen ähnlichen Notationen, berücksichtigen vor allem auch Erfahrungs- und andere Informationsflüsse.

Im Rahmen dieser Arbeit wurden drei durch Datenflussdarstellung verschieden Modelle ausgearbeitet um auf verschiedene Prioritäten der Projekte angehen zu können. Im ersten Modell „Quellenabhängige Kante“ wird dargestellt von welcher Quelle aus (dokumentiert oder verbal) der Datenfluss kommt. In zweitem Modell „Drei-Farben“ ist der Inhalt der Datenflusse wichtig. Die Erfahrung, Projektinformation und Grundwissen werden hier berücksichtigt. Das dritte Modell „Quellenabhängige Kante + Drei-Farben“ ist eine Kombination aus den ersten beiden Anhand eines Beispieles wurden Unterschiede dieser Modelle demonstriert.
Für die Implementierung einer Benutzeroberfläche, die das Erstellen sowie Modifizieren des FLOW-Modells gestattet, wurde für das MVC- Architekturmuster entschieden. Die Modell-Einheit wurde dann als Klassenbibliothek implementiert.

Bei der Weiterentwicklung sollen weitere MVC- Einheiten (View und Controller) implementiert werden.
Bevor das FLOW-Modell in industrieller Praxis eingesetzt werden kann, empfiehlt es, einige Experimente mit Studenten durchzuführen. Dadurch können theoretisch ausgearbeiteten Konzepte in der Praxis geprüft und eventuell (durch Nacharbeitung) verbessert werden.

7 Abbildungsverzeichnis

8 Literatur

[1 Kurt Schneider Daniel Lübke Thomas Flohr, 2005, Softwareentwicklung zwischen Disziplin und Schnelligkeit, Tele Kommunikation Aktuell, Ausgabe 06/'05, S. 5

[2] Kurt Schneider, Daniel Lübke: Systematic Tailoring of Quality Techniques Proceedings of the 3rd World Congress of Software Quality, 2005, Vol. 3

[3] Kurt Schneider, Daniel Lübke: Legacy Situation und Spektrum von Review-Varianten

[4] Kurt Schneider, Daniel Lübke: FLOW-Experimente, Brainstorming 14.10.2004

[5] Horst Sachs: Einführung in die Theorie der endlichen Graphen. (Carl Hanser Verlag München, 1971);
http://de.wikipedia.org/wiki/Graphentheorie