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

Tailoring von Gate-Netzwerken

Bachelorarbeit

im Studiengang Informatik

von

Christian Schulz

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr.-Ing. Christian Grimm
Betreuer: Dipl.-Math. Thomas Flohr

Hannover, 10. Oktober 2007

Danksagung

Ich bedanke mich bei Herrn Dipl.-Math. Thomas Flohr für
die hervorragende Betreuung meiner Bachelorarbeit.

Inhaltsverzeichnis

1. Einleitung

1.1. Hintergrund

In der Softwareentwicklung werden häufig Quality Gates eingesetzt. Quality Gates sind spezielle Meilensteine und dienen der Strukturierung und Qualitätssicherung. Sie befinden sich zwischen den einzelnen Phasen des Softwareentwicklungsprozesses. In den Quality Gates werden die Entscheidungen über die Fortsetzung eines Projektes gefällt. Je nach Projekt ist es notwendig, die Anzahl der Gates, die Anordnung der Gates, die möglichen Fortschrittsentscheidungen und die Aktivitäten, die zur Entscheidungsfindung und zur Definition von Kriterien führen, anzupassen. Die Anzahl und Anordnung der Gates sowie mögliche Entscheidungen bilden das so genannte Gate-Netzwerk. Den Vorgang des Anpassens nennt man Tailoring.

1.2. Aufgabe

Das Ziel dieser Arbeit ist die Erstellung eines Anpassungskonzeptes für Gate-Netzwerke. Basierend auf dem Projektkontext und formalisierten Erfahrungen soll ein Gate-Netzwerk mit Aktivitäten zur Entscheidungsfindung und zur Definition von Kriterien aufgebaut werden. Die Erfahrungen geben an, unter welchen Voraussetzungen welche Anzahl und Anordnung von Gates, Entscheidungen und Aktivitäten zu wählen sind.

Ein weiteres Ziel dieser Arbeit ist die Implementierung des Anpassungskonzeptes. Die Anwendung soll die Möglichkeit bieten den Projektkontext einzugeben und eine Übersicht des gesamten Gate-Netzwerkes zu liefern. Die Anwendung wird in Form einer Java-Swing-Anwendung implementiert.

Am Beispiel des V-Modell XT soll das Konzept und dessen Implementierung angewendet werden. Dafür erforderlich ist die Formalisierung des Erfahrungswissens aus dem V-Modell XT und Modellierung dessen Gate-Netzwerke.

Zusätzlich besitzt diese Arbeit einen optionalen Teil. Hier sollen die modellierten Gate-Netzwerke in das SOA-Me-System exportiert werden. Im SOA-Me-System besteht die Möglichkeit die Prozesse der Gate-Netzwerke zu durchlaufen. Grundfunktionalitäten wie Oberflächengenerierung und Authentifizierung stehen dort schon zur Verfügung. Eventuell sind im Rahmen dieser Arbeit noch Funktionalitäten in diese Infrastruktur hinzuzufügen.

1.3. Gliederung

Im zweiten Kapitel werden einige Grundlagen und Begriffe erklärt, die im Laufe der Arbeit Verwendung finden. Es wird das Quality-Gate-Konzept eingeführt und dessen Einsatz in der Softwareenticklung erläutert. Dazu gehören Quality Gates und auch Gate-Netzwerke. Des weiteren wird erklärt was Tailoring ist. Danach wird der Stage-Gate-Prozess von Cooper und das V-Modell XT vorgestellt.

Im dritten Kapitel wird das Anpassungskonzept definiert. Angefangen wird mit den einzelnen Elementen eines Gate-Netzwerkes. Es wird beschrieben was unter Projektmerkmalen zu verstehen ist, wie ein Gate-Netzwerk aufgebaut wird und welche Zusatzinformationen die Quality Gates erhalten. Danach wird der Aufbau der Tailoring-Regel oder auch Erfahrungsbasis erläutert. Am Ende des Kapitels werden die wichtigsten Anwendungsfälle aufgelistet und beschrieben.

Das vierten Kapitel ist die Beschreibung der Implementierung. Es werden die wichtigsten Pakete und Komponenten beschrieben und ein Klassendiagramm des Datenmodells erläutert. Darauf folgt eine kurze Erläuterung der graphischen Darstellung sowie des graphischen Exportes.

In Kapitel fünf wird das Konzept auf das V-Modell XT angewendet. Es wird gezeigt, wie die Erfahrungen aus dem Modell in die Implementation übernommen werden. Zusätzlich wird eine Heuristik gezeigt, die das Erzeugen der Erfahrungsbasis erleichtert.

Das Kapitel sechs enthält eine Zusammenfassung und bietet einen Ausblick auf mögliche Erweiterungen dieser Arbeit.

2. Grundlagen

In diesem Kapitel werden die allgemeinen Grundlagen dieser Arbeit behandelt. Erst wird das Quality-Gate-Konzept wiedergegeben. Daraufhin wird erläutert was Tailoring ist und warum es notwendig ist, dieses einzusetzen. Am Ende werden noch der Stage-Gate-Prozess von Cooper und das V-Modell XT beschrieben.

Die folgenden Definitionen sind angelehnt an die Arbeiten von Jens Greive[2], Christian Peucker[1] und Daniel Scholz[3].

2.1. Quality-Gate-Konzept

Zeit- und Erfolgsdruck sind ein ständiger Begleiter in der Softwareentwicklung. Um Risiken zu minimieren und um das Projekt zu strukturieren, wird das Quality-Gate-Konzept eingesetzt. Wesentlicher Bestandteil des Konzeptes ist die Benutzung von Quality Gates, die an besonderen Stellen im Entwicklungsprozess eingefügt werden. Quality Gates sind Entscheidungspunkte, in denen über die Fortsetzung des Projektes entschieden wird. Zielsetzung der Quality Gates ist die frühzeitige Erkennung von Fehlern und Abweichungen. Die Anzahl und Anordnung der Quality Gates vom Projektstart bis zum Projektende bilden das so genannte Gate-Netzwerk.

Im folgenden Abschnitt werden Quality Gates und Gate-Netzwerke näher definiert.

2.1.1. Quality Gate

Ein Quality Gate ist ein Instrument, um in Projekten ein Mindestmaß an Qualität zu sichern. Sie können als Meilensteine im Projekt oder auch, wie der Name schon sagt, als Tor zwischen den Phasen verstanden werden. In einem Quality Gate werden die Ergebnisse bzw. Dokumente der Vorphase überprüft. Die Anforderungen an die Ergebnisse werden die “Kriterien” genannt. Diese Kriterien stehen in Checklisten. Nach der Überprüfung der Ergebnisse, anhand der Kriterien, wird entschieden, ob die Entwicklung in die nächste Phase übergehen darf. Quality Gates besitzen eine Deadline. Die Deadline gibt an, zu welchem Zeitpunkt die Ergebnisse geliefert werden sollen.

Quality Gates erreichen das Mindestmaß an Qualität dadurch, dass die Entwicklung erst in die nächste Phase übergeht, wenn alle Anforderungen an die Ergebnisse erfüllt sind. Der Übergang in die nächste Phase ist aber nicht der einzig mögliche Ausgang den das Quality Gate haben kann. Es kann auch entschieden werden, dass die Vorphase wiederholt werden soll. Dieser Fall tritt ein, wenn die Ergebnisse nur geringe Mängel vorweisen und diese in absehbarer Zeit behoben werden können. Es wird eine neue Deadline gesetzt und dann das Quality Gate wiederholt. Desweiteren ist es möglich nach einem Quality Gate das komplette Projekt abzubrechen. Ein Abbruch erfolgt, wenn die Ergebnisse stark von den Anforderungen abweichen

und keine Aussicht auf ein Projekterfolg besteht. Eine abgeschwächte Form des Abbruches ist die Zurückstellung des Projektes. Dieses wird dann zu einem späteren Zeitpunkt fortgeführt. Die Zurückstellung kann auch für die Synchronisation von mehreren Projekten benutzt werden.

Durch den Einsatz von Quality Gates wird ein Projekt strukturiert. Die Zeitpunkte der Quality Gates werden idealerweise zum Projektbeginn festgelegt und somit erhält man eine zeitliche Übersicht über den gesamten Projektverlauf. Diese Übersicht dient der Risikominimierung, da man die benötigten Ressourcen besser einschätzen kann. Auch kann damit der Projektfortschritt gemessen werden.

Die eigentliche Entscheidungsfindung in einem Quality Gate findet auf Grundlage des Reviews statt. Ein Review ist eine gewisse Nachprüfung. Reviews können verschiedenartig gestaltet sein. Sie variieren zwischen sehr informell und sehr formal. Die Art und Weise, wie ein Review durchgeführt wird, kann abhängig sein vom Umfang der Ergebnisse oder auch von der Wichtigkeit der Ergebnisse in Bezug auf das Gesamtprojekt.

Quality Gates müssen nicht zwangsweise zwischen definierten Phasen eingesetzt werden. Sie können auch zwischen einzelnen Prozessschritten benutzt werden. Die Voraussetzung für ein Quality Gate ist das Vorhandensein von klar definierbaren und folglich auch überprüfbaren Ergebnissen.

Reviewtechniken

Bei der Prüfung der Ergebnisse für das Quality Gate können verschiedene Reviewtechniken genutzt werden. Die Reviewtechniken definieren den Ablauf der Prüfung. Im späteren Verlauf dieser Arbeit wird auf einige Techniken Bezug genommen. Diese sind die informellen Techniken “**Peer Review**” und “**Walkthrough**” und die formelle Technik “**formales Review**” (wird im Allgemeinen aber nur Review genannt).
Das Peer Review kommt von engl. “peer” - der Gleichgestellte. Bei der Prüfung sind keine personalverantwortlichen Vorgesetzten des Autors anwesend. Beim Peer-Review untersuchen die Gutachter die Ergebnisse in einer Sitzung und erstellen entsprechende Fehlerprotokolle. Anhand derer wird der Ausgang des Peer-Review beurteilt.
Beim Walkthrough stellt der Autor die von ihm verfassten Ergebnisse in einer Sitzung einem oder mehreren Gutachtern vor. Diese versuchen dabei Fehler oder Unstimmigkeiten aufzudecken. Die Gutachter entscheiden über den Ausgang der Prüfung.
Das Review (oder auch formelle Inspektion) ist die formellste Ausprägung einer Quality-Gate-Prüfung. Sie benötigt auch den größten zeitlichen und personellen Aufwand. Die Gutachter prüfen die Ergebnisse unabhängig von einander und notieren potentielle Fehler, Fragen und Kommentare. In einer Review-Sitzung wird dann, unter Leitung eines Moderators, über diese Notierungen diskutiert und die Entscheidung über den Ausgang der Prüfung getroffen.

Graphische Darstellung

Die folgende Abbildung veranschaulicht die graphische Darstellung eines Quality Gates.

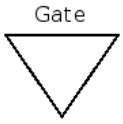


Abbildung 2.1.: *Quality Gate*

Ein Quality Gate wird in dieser Arbeit als ein gleichseitiges, auf der Spitze stehendes Dreieck dargestellt. Der Name des Quality Gates steht über dem Dreieck und bezieht sich sinngemäß auf eine abgeschlossene Aktivität (wie z.B. “System entworfen”) oder auf einen Zeitpunkt (wie z.B. “Fertig für Entwicklung”)

2.1.2. Gate-Netzwerke

Um einen Projektverlauf zu strukturieren, werden Gate-Netzwerke eingesetzt. Gate-Netzwerke werden definiert durch Anzahl und Anordnung der Quality Gates vom Projektstart bis zum Projektende. Sie ermöglichen also einen Überblick über das Projekt und erleichtern dadurch das Anpassen der Prozesse an den jeweiligen Projektkontext. Korrekter Weise müssten diese Netzwerke “Quality-Gate-Netzwerke” heißen. Im Laufe dieser Arbeit wird aber nur von Gate-Netzwerken die Rede sein.

Formale Definition

Ein Gate-Netzwerk N kann als Graph angesehen werden und ist ein 4-Tupel

$$N = (G, T, S, E),$$

wobei für die einzelnen Komponenten gilt:

- G ist eine endliche Menge von Quality Gates,
- $T \subseteq GxG$ ist eine endliche Menge von gerichteten Übergängen $T(u,v)$,
- $S \in G \setminus E$ ist der so genannte Startknoten,
- $E \in G \setminus S$ ist der so genannte Endknoten.

Die Elemente von T sind $(u,v) \in G$ und beschreiben einen gerichteten Übergang von einem Quality Gate u zu einem anderen v. Jedes Quality Gate in G besitzt mindestens einen ausgehenden Übergang, also einen Eintrag in u, und mindestens einen eingehenden Übergang, also einen Eintrag in v. Jedes Quality Gate aus G ist mit Hilfe der Übergänge vom Startknoten aus zu erreichen und von jedem Quality Gate aus G ist es möglich, mit Hilfe der Übergänge zum Endknoten E zu gelangen. Die Menge G enthält mindestens die zwei Elemente S und E.

Startknoten S und Endknoten E sind keine Gates im eigentlichen Sinne. Sie befinden sich in der Menge G damit die Operationen auf das Netzwerk vereinfacht werden. Für Startknoten und Endknoten gilt obige Definition mit einer Einschränkung. Der Startknoten S besitzt keine eingehenden Übergänge und somit *keinen* Eintrag in v und der Endknoten E besitzt keine

ausgehenden Übergänge, also *keinen* Eintrag in u. Desweiteren beziehen sie sich nicht, wie die Quality Gates, auf einen Entscheidungsprozess. Sie definieren lediglich den Start bzw. das Ende des Netzwerkes.

Beispiel:

G ={ Start, Anforderungen festgelegt, System entwickelt,
Komponenten gekauft, Projekt abgeschlossen, Ende}
S = Start
E = Ende
T ={ (Start, Anforderungen festgelegt),
(Anforderungen festgelegt, System entwickelt),
(Anforderungen festgelegt, Komponenten gekauft),
(Komponenten gekauft, System entwickelt),
(System entwickelt, System entwickelt),
(System entwickelt, Projekt abgeschlossen),
(Projekt abgeschlossen, Ende)}

Beschreibung an einem Beispiel

Die Gate-Netzwerke sind ein Modell eines Gate-Prozesses. Die folgende Abbildung enthält ein Gate-Prozess eines kleinen Projektes. Anhand diesem Projekt wird die Erstellung eines Gate-Netzwerkes, ausgehend von einem Gate-Prozess, gezeigt.

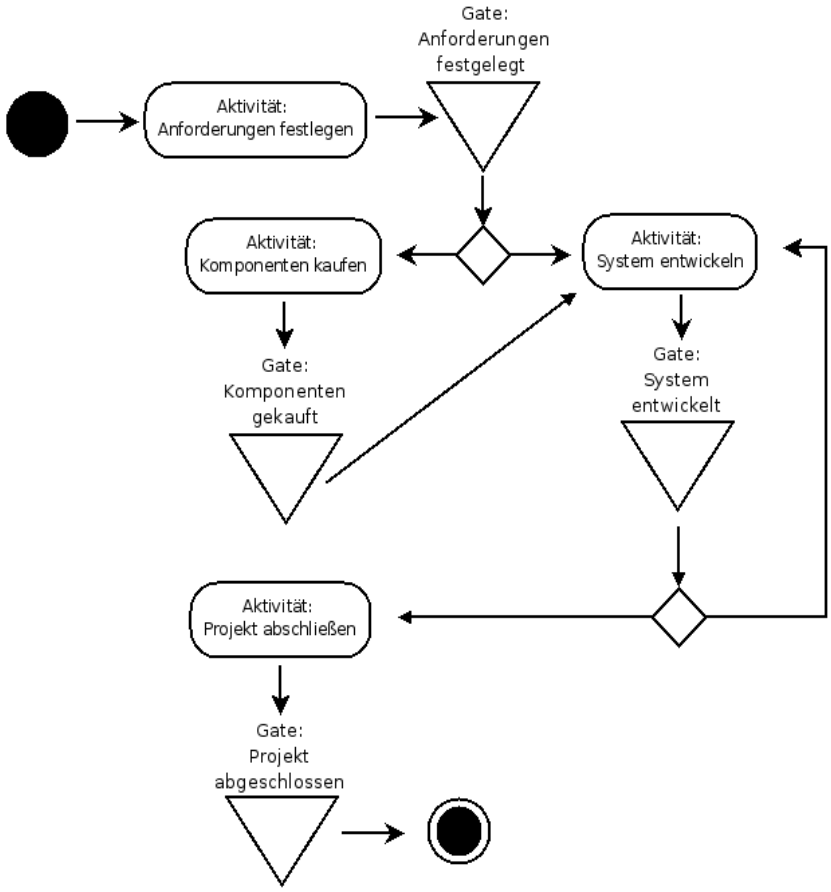


Abbildung 2.2.: *Gate-Prozess mit Aktivitäten und Quality Gates*

In dem Projekt werden erst die Anforderungen festgelegt. Diese werden danach geprüft. Bei der Prüfung wird entschieden, ob das System komplett selbst entwickelt werden soll oder ob zusätzliche Komponenten gekauft werden. Die Systementwicklung kann auch iterativ erfolgen. Beim Projektabschluss wird das Produkt an den Kunden übergeben.
Die Folgenden Regeln werden für die Überführung benötigt:

Regel 1: Der **Übergang**

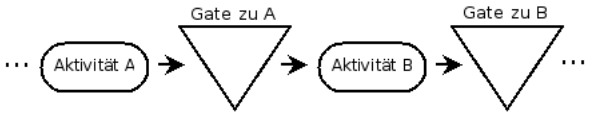


Abbildung 2.3.: *Übergang als Aktivität*

wird zu

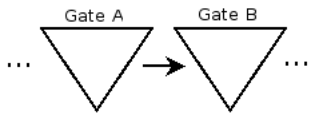


Abbildung 2.4.: *Übergang im Gate-Netzwerk*

Regel 2: Die **Verzweigung**

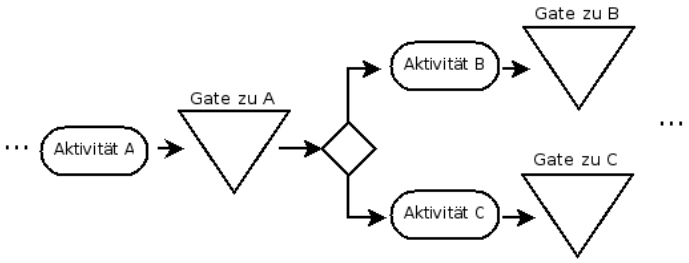


Abbildung 2.5.: *Verzweigung als Aktivität*

und die **Aufspaltung**

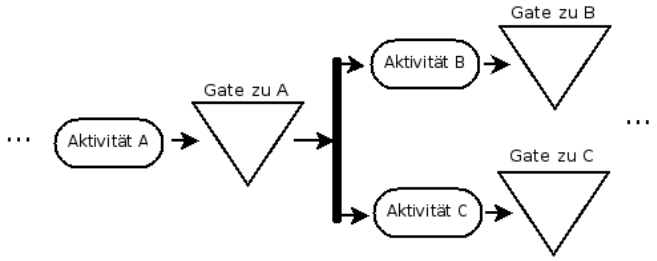


Abbildung 2.6.: *Aufspaltung als Aktivität*

werden zu

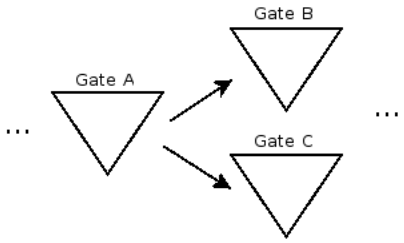


Abbildung 2.7.: *Verzweigung oder Aufspaltung im Gate-Netzwerk*

Regel 3: Die **Zusammenführung**

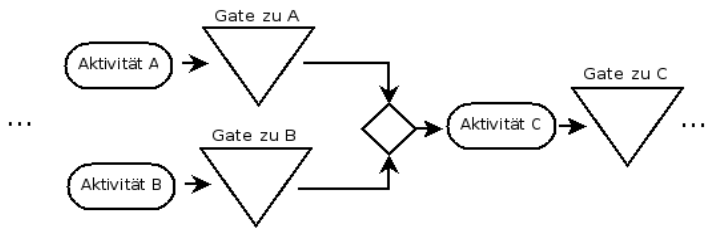


Abbildung 2.8.: *Zusammenführung als Aktivität*

und die **Synchronisation**

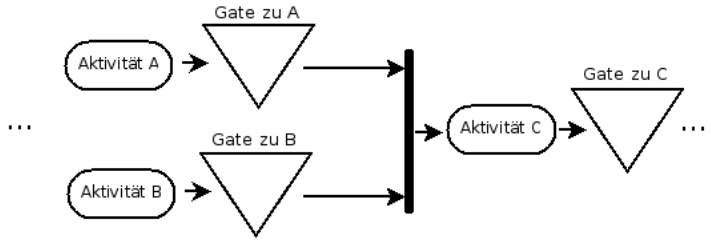


Abbildung 2.9.: *Synchronisation als Aktivität*

werden zu

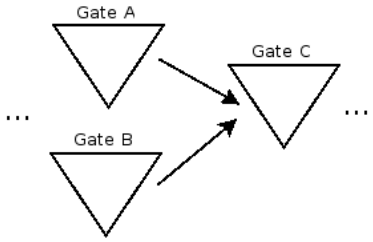


Abbildung 2.10.: *Zusammenführung im Gate-Netzwerk*

Mit Hilfe dieser Regeln ergibt sich für das Beispielprojekt folgendes Gate-Netzwerk.

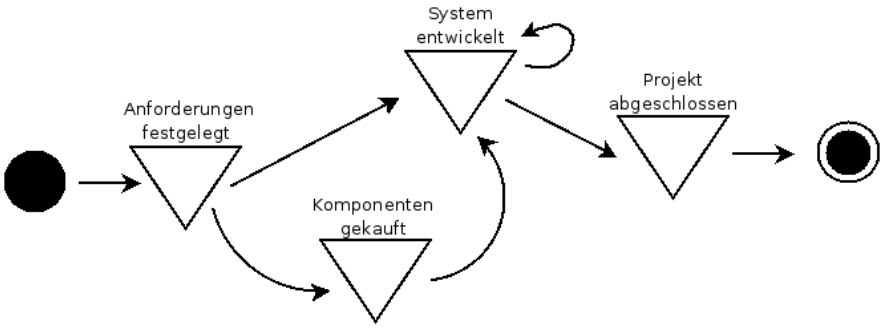


Abbildung 2.11.: *Gate-Prozess aus Abb.2.2 als Gate-Netzwerk*

Der Übergang auf sich selbst, wie bei “System entwickelt”, wird mit Regel 2 erstellt. Solch ein Übergang ist eine **Iteration** (siehe Tabelle 2.1).

Durch die Umwandlung des Gate-Prozesses zum Gate-Netzwerk gehen die zeitlichen Informationen von synchronisierten und parallelen Prozessen verloren. Dem Gate-Netzwerk ist nur noch zu entnehmen, welche Quality Gates vor einem anderen Quality Gate abgeschlossen werden müssen.

In der Folgenden Tabelle werden die Aussagen der einzelnen Komponenten des Gate-Netzwerkes näher erläutert.

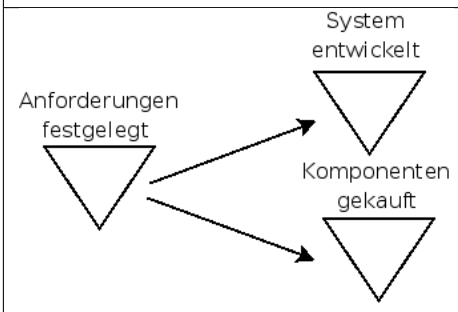
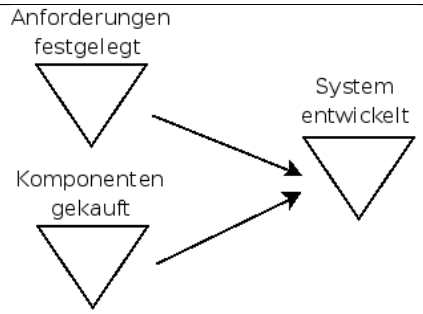
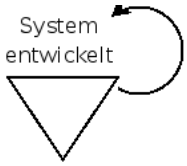
Graphische Darstellung	Bedeutung
	**Startknoten** - Er bezeichnet den Anfang des Netzwerkes und beinhaltet keine Aktivitäten.
	**Endknoten** - Er bezeichnet das Ende des Netzwerkes und beinhaltet keine Aktivitäten.
	**Quality Gate** - Das Quality Gates beinhaltet die Entscheidungsfindung. Es wird geprüft, ob alle Ergebnisse den Anforderungen (Kriterien) entsprechen.
	**Übergang** - Der mögliche Verlauf des Handlungsstranges. Während des Überganges erfolgen die Aktivitäten der Produkterzeugung
	**Verzweigung** - Der Handlungsstrang *kann* aufgespalten werden. Nach der Prüfung der Anforderungen wird entschieden, ob “Komponenten gekauft”, das “System entwickelt” oder auch beides zu erfolgen hat. Über den Zeitpunkt von “Komponenten gekauft” und “System entwickelt” ist nur zu erfahren, dass sie nach “Anforderungen festgelegt” erfolgen.
	**Zusammenführung** - An dieser Stelle werden die Handlungsstränge wieder zusammengeführt. Zeitlich liegen “Anforderungen festgelegt” und “Komponenten gekauft” vor “System entwickelt”.
	**Iteration** - Bei einer Iteration ist es möglich, auch nach Bestehen des Quality Gates die Aktivitäten für das Quality Gate zu wiederholen. Eine Iteration kann auch über eine Phase von mehreren Quality Gates erfolgen.

Tabelle 2.1.: *Bedeutung der einzelnen Komponenten*

Modelltheorie

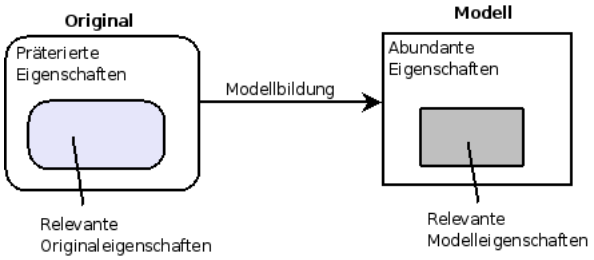


Abbildung 2.12.: *Gegenüberstellung von Original und Modell*

Bei der Modellbildung[4] wird ein Modell von einem Original erstellt. Dabei sind nur einige Eigenschaften des Originals von Interesse. Diese Eigenschaften nennt man *relevante* Originaleigenschaften. Alle anderen Eigenschaften nennt man *präterierte* Eigenschaften. Die relevanten Originaleigenschaften werden nun in das Modell abgebildet und bleiben dort als relevante Modelleigenschaften vorhanden. Desweiteren besitzt ein Modell zusätzliche Eigenschaften, welche nicht vom Original stammen. Diese zusätzlichen Eigenschaften werden *abundante* Modelleigenschaften genannt.

Um ein Modell zu charakterisieren, werden die Antworten auf folgende Fragen benötigt:

- Was ist das Original?
- Für wen ist es?
- Wozu dient es?
- Wann wird es benötigt?
- Welche Eigenschaften sind relevant?

Charakterisierung des Gate-Netzwerkes

Was ist das Original?: Das Original ist der Gate-Prozess eines Projektes mit Aktivitäten zur Produkterzeugung. Gefolgt von Aktivitäten zur Entscheidungsfindung.

Für wen ist das Modell?: Das Modell ist für das Planungsteam, welches den Gate-Prozess festlegt.

Wozu dient das Modell?: Das Modell zeigt die Beziehungen der einzelnen Quality Gates zueinander und dient als Planungsgrundlage für den festzulegenden Gate-Prozess.

Wann wird das Modell benötigt?: Das Modell wird in der Planungsphase (siehe Abschnitt 2.2) benötigt.

Welche Eigenschaften sind relevant?: Die relevanten Eigenschaften sind die Struktur des Gate-Prozesses mit den möglichen Wegen des Handlungsstranges vom Projektstart bis zum Projektende.

2.2. Phasen eines Projektes

Ein Projekt lässt sich in 3 Phasen einteilen:

- Startphase
- Planungsphase
- Durchführungsphase

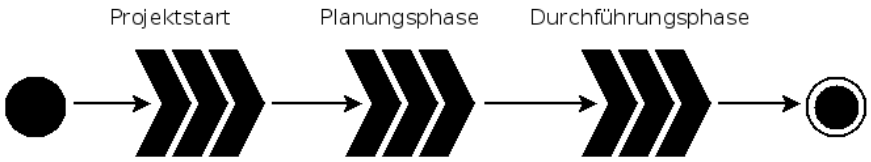


Abbildung 2.13.: *Phasen eines Projektes*

Das Projekt fängt mit der Startphase an. Während dieser Phase sind die Hauptakteure aus dem *Projektkernteam*. In diesem Team befinden sich vorrangig nur Personen aus dem Management. Während des Projektstarts werden die wirtschaftliche Machbarkeit und die Projektziele erörtert. Dann wird von dem Projektkernteam die Vorgehensstrategie festgelegt. Zu diesem Zeitpunkt geschieht das Tailoring (Eklärung siehe 2.3).

Das Ergebnis des Tailorings ist das Gate-Netzwerk für die darauf folgende Planungsphase. In dieser werden dann Aufwandsabschätzungen, Zeit-/Kostenpläne für den Gate-Prozess usw. erstellt.

In der Durchführungsphase oder auch Realisierungsphase wird mit der eigentlichen Entwicklung begonnen.

2.3. Tailoring

Aus dem Englischen übersetzt bedeutet “tailoring” oder “to tailor” “Schneiderei” bzw. “maßschneidern”. In der Softwareentwicklung bedeutet es das Anpassen eines Entwicklungsprozesses an den konkreten Projektkontext. Im Falle von Gate-Netzwerken werden die Netzwerke angepasst.

Das Anpassen ist nötig, denn nicht jedes Projekt ist gleich. Auch in ein und dem selben Unternehmen können sich bei ähnlichen Projekten die Anforderungen an das Ergebnis stark unterscheiden. Aktivitäten, die in einem Projekt benötigt werden, können in einem anderen Projekt überflüssig sein. Ein Beispiel wäre das auslassen des Feinentwurfes, da erst während der Implementierung verschiedene Entwürfe ausprobiert werden sollen. Ein anderes Beispiel wäre es, wenn in einem Projekt mehr und in einem anderen weniger Dokumentation verlangt wird. D.h. durch das Auslassen von überflüssigen Dokumenten und Aktivitäten werden Zeit und Kosten gespart. Zusätzlich kann man mit dem Tailoring risikoreiche Phasen aufteilen, in dem weitere Quality Gates eingesetzt werden, und man somit eine bessere Kontrolle über das Projekt erreicht.

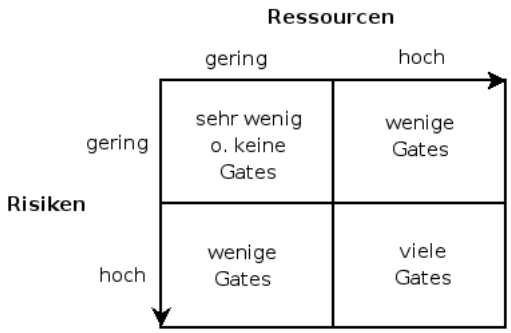


Abbildung 2.14.: *Ressourcen-Risiko-Matrix*

Sehr vereinfacht könnte man sagen: Je risikoreicher und ressourcenlastiger ein Projekt ist, desto mehr Gates sind von Vorteil.

Darin besteht auch der Hauptvorteil des Tailoring. Der Projektablauf ist besser an den Projektkontext angepasst. Durch das Auslassen von Quality Gates werden Ressourcen gespart und durch das Hinzufügen von Quality Gates an riskanten Stellen wird Risiko minimiert.

In der folgenden Abbildung ist der Ablauf vom Tailoring eines Gate-Netzwerkes zu sehen.

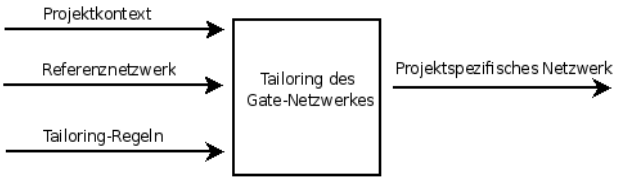


Abbildung 2.15.: *Tailoring-Prozess*

Das Tailoring benötigt mehrere Eingaben:

- Projektkontext
- Referenznetzwerk
- Tailoring-Regeln

Das Referenznetzwerk ist die Grundlage der Anpassungen. Mit Hilfe der anderen Eingabeparameter wird das Gate-Netzwerk an das konkrete Projekt angepasst.

Der Projektkontext sind die Informationen, die über das Projekt bekannt sind. Dazu gehören Informationen zu dem Projekttyp, also über die charakteristischen Merkmale des Projekts, auch die Ziele des Projekts und die technischen Anforderungen an die Software bzw. Hardware.

Ein weiterer Eingabeparameter sind die Tailoring-Regeln. Sie ergeben sich aus dem Wissen von Personen, die für das Tailoring verantwortlich sind, oder auch aus Vorgaben der Organisation.

Der letzte wichtige Eingabeparameter sind die Erfahrungen. Die Erfahrungen sind keine direkten Ausgaben der Tailoring-Aktivität. Sie sind Wissen aus vorhergehenden Projekten und

fließen als neue Tailoring-Regeln wieder mit ein. Diese Rückkopplung hilft dabei, dass durch das Tailoring erstellte Gate-Netzwerk schrittweise zu optimieren. Alle Eingabeparameter werden formal festgehalten.

2.3.1. Rollen beim Tailoring

Beim Tailoring werden folgende Rollen benötigt:

- Tailoring-Ersteller
- Tailoring-Ausführer
- Tailoring-Veränderer
- Projektplaner

Die Rolle des Tailoring-Erstellers initiiert die Erstellung der Tailoring-Regeln. Er besitzt Expertenwissen über Praktiken mit Gate-Netzwerken. Die Erstellung geschieht idealerweise nur ein einziges Mal für ein Unternehmen.

Der Tailoring-Ausführer besitzt Wissen über das aktuelle Projekt. Er kennt den genauen Projektkontext und startet auf dieser Basis das Tailoring. Er benötigt nicht so viel Erfahrung wie der Tailoring-Ersteller. Der Tailoring-Ausführer ist Mitglied des Projektkernteams(siehe Abschnitt 2.2) und führt das Tailoring in der Startphase aus.

Eine weitere Rolle ist der Tailoring-Veränderer. Er hat einen Überblick über die gesamten vorhergegangenen Projekte und dessen zugrundeliegenden Tailoring-Regeln. Er hat, wie der Tailoring-Ersteller, hochwertiges Wissen über Praktiken mit Gate-Netzwerken. Zusätzlich erweitert er sein Wissen fortlaufend. Der Tailoring-Veränderer ist derjenige, der die Tailoring-Regeln erweitert und optimiert. Diese Optimierungen werden hauptsächlich nach einem Projektende durchgeführt.

Der Projektplaner ist nicht direkt am Tailoring-Prozess beteiligt. Er benutzt das Ergebnis des Tailoring-Prozesses, das Gate-Netzwerk. Seine Aufgabe ist die Erstellung des eigentlichen Gate-Prozesses während der Planungsphase. Dafür wählt er die Aktivitäten aus, die ihm das Gate-Netzwerk als Möglichkeit vorgibt.

2.3.2. Vorteile des Tailoring

Der Hauptvorteil ist, dass der Projektablauf besser an den Projektkontext angepasst ist. Das Tailoring besitzt jedoch noch zwei weitere Vorteile.

Da die Tailoring-Regeln formal festgehalten werden, kann das Tailoring werkzeuggestützt ablaufen. Das Wissen muss nur einmalig durch den Tailoring-Ersteller in die Regeln übernommen werden und ist dann immer wieder schnell zugänglich. Der Tailoring-Ausführer erstellt mit dem Werkzeug zum Projektstart das Gate-Netzwerk. Dafür benötigt er nur den Projektkontext und keine weiteren hochwertigen Erfahrungen. Das erleichtert seine Arbeit.

Da das Wissen formal in den Regeln festgehalten ist, ist das Optimieren der Gate-Netzwerke durch den Tailoring-Veränderer besonders einfach. Er erweitert dafür lediglich die Tailoring-Regeln.

2.3.3. Arten des Tailoring

Durch die Betrachtung des Zeitpunktes des Tailoringvorganges entstehen zwei verschiedene Arten:

- Statisches Tailoring
- Dynamisches Tailoring

Das statische Tailoring wird im Rahmen der Projektinitialisierung durchgeführt. Diese geschieht in der Startphase. Das Ergebnis wird im Projekthandbuch formal festgehalten und dient als Basis für weitere Planungen. Das Tailoring, welches nach der Projektinitialisierung und damit *während* der Projektlaufzeit durchgeführt wird, wird dynamisches Tailoring genannt. Es wird eingesetzt, wenn späte oder unvorhersehbare Veränderungen im Projekt auftreten.

2.4. Prozessmodelle

In diesem Abschnitt werden zwei Modelle, in denen das Quality-Gate-Konzept angewandt wird, beschrieben. Es kann dabei zu einigen Überschneidungen kommen, da Quality-Gate-Prozesse vom Grundaufbau sehr ähnlich sind.

2.4.1. Stage-Gate-Prozess

Der Stage-Gate-Prozess von Cooper[5] ist ein Modell, welche für die Produktentwicklung eingesetzt werden kann. Es ist nicht strikt auf die Entwicklung von Software festgelegt. Der Stage-Gate-Prozess leitet die Entwicklung von der Produktidee bis zur Markteinführung. Ziel ist es, das Risiko des Scheiterns eines Projektes frühzeitig zu erkennen, um rechtzeitig gegenzusteuern. Der Prozess ist in Stages und Gates aufgeteilt.

Stages

Die Stages können auch als Phasen bezeichnet werden. Die Phasen bestehen aus parallelen und sequenziellen Aktivitäten, welche von Personen des Unternehmens ausgeführt werden. In jeder Phase werden Leistungen bzw. Ergebnisse geliefert. Diese Leistungen sind die Grundlage der Gates.

Gates

Vor den Stages (mit Ausnahme der Ideenfindung) befinden sich die Gates. Sie dienen als Entscheidungs- und Qualitätskontrollpunkte. Sie haben zu prüfen, ob alle Aufgaben der Vorphase erfüllt wurden und wie das weitere Vorgehen ist. Der Entscheidungsträger wird *Gate-Keeper* genannt. Um die Aufgabe zu erfüllen, benötigt man laut Cooper die folgenden drei Komponenten.

1.Deliverables: (Ergebnisse)
Im Vorfeld der Gates werden die Ergebnisse, die das Entwicklungsteam produziert, definiert. Alle Aktivitäten des Teams in der Vorphase müssen sich auf die Ergebnisse beziehen.

2.Criteria: (Kriterien)
Anhand der Kriterien werden die Ergebnisse der Vorphase beurteilt. Voraussetzung ist, dass die Kriterien für jeden Beteiligten einsehbar und verständlich sind. Sie werden für gewöhnlich in Checklisten formuliert. Desweiteren werden Kriterien in zwei Typen unterteilt. Zum Einen in “Must-meet” und zum Anderen in “Should-meet”. die “Must-meet”-Kriterien müssen auf jeden Fall bei der Überprüfung bestanden werden. Wenn nicht, führt dieses zu einem Abbruch des Projektes. “Should-meet”-Kriterien sind wünschenswerte Eigenschaften. Sie sind nicht projektkritisch und können daher auch unerfüllt bleiben.

3.Outputs: (Ausgaben)
Die Ausgaben bestehen aus den eigentlichen Entscheidungen, einem Plan über die in der nächsten Phase bereitgestellten Ressourcen und einer Auflistung der geforderten Ergebnisse für das nächste Gate. Laut Cooper gibt es vier mögliche Entscheidungen:

Go: Das Projekt geht in die nächste Phase über.

Kill: Das Projekt wird abgebrochen und es werden keine Ressourcen mehr investiert.

Hold: Das Gate gilt als bestanden, aber es fehlen Ressourcen für die nächste Phase. Erst wenn diese wieder zu Verfügung stehen wird das Projekt fortgeführt.

Recycle: Das Gate wurde nicht bestanden. Die Vorphase darf aber wiederholt werden und die Ergebnisse dürfen verbessert werden.

In der Zusammenfassung heißt das, dass die Gates die Ergebnisse als Eingabe erhalten. Dann werden die Ergebnisse in Bezug zu den Kriterien begutachtet und als Ausgabe liefert das Gate eine Entscheidung.

Darstellung des Prozesses

Die folgende Abbildung zeigt den Aufbau des Stage-Gate-Prozess nach Cooper. Danach werden darin enthalten Stages und Gates kurz erläutert.

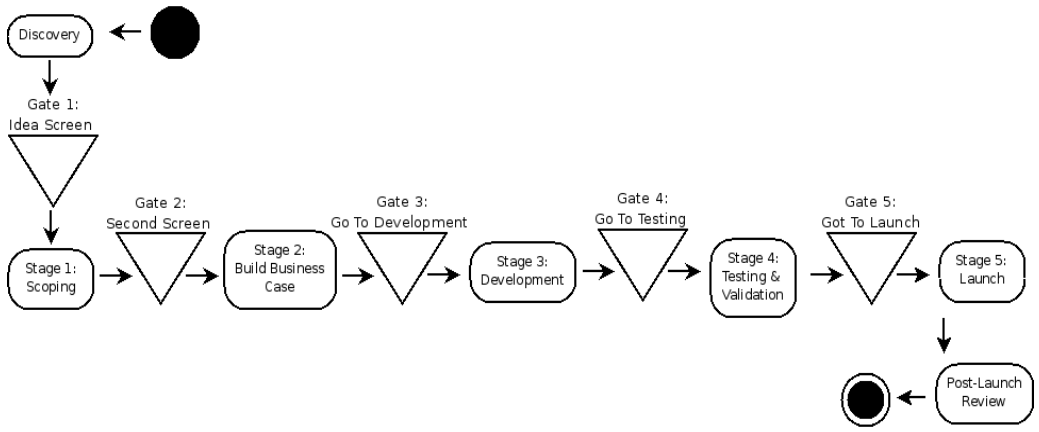


Abbildung 2.16.: *Stage-Gate-Prozess nach Cooper*

Discovery: (Ideenfindung)
In der Phase/Stage der Ideenfindung wird der Markt analysiert und eine Produktidee gesucht.

Idea Screen: (Prüfung der Idee)
In diesem Gate wird entschieden, ob die Produktidee Erfolgsaussichten hat.

Scoping: (Vorstudie)
Die Vorstudie besteht aus einer vorläufigen Marktbeurteilung. Es werden Marktgröße und auch Marktpotential untersucht. Dazu können auch Gespräche mit potentiellen Kunden gehören. Zusätzlich werden auch technische Produkteigenschaften untersucht.

Second Screen: (Prüfung der Vorstudie)
Dieses Gate ist im Wesentlichen eine Wiederholung von Gate "Idea Screen". Es ist eine strengere Prüfung der Kriterien des ersten Gates.

Building the Business Case: (Betriebswirtschaftliche Beurteilung)
Hier werden die Produktdefinition, ein detaillierter Projektplan und finanzielle Analysen erstellt.

Go to Development: (Fertig für Entwicklung)
Es werden die Produktdefinition und der Projektplan überprüft. Nach diesem Gate werden die Phasen kostenintensiver.

Development: (Entwicklung)
In der Realisierungsphase wird das eigentliche Produkt entwickelt. Das Ergebnis dieser Phase ist ein getesteter Prototyp. Dieser Prototyp kann auch schon potentiellen Kunden zur Beurteilung gegeben worden sein.

Go to Testing: (Fertig für Test & Validierung)
In diesem Gate wird geprüft, ob der Prototyp mit den Produktdefinitionen aus Gate "Go to Development" übereinstimmen.

Testing and Validation: (Test & Validierung)
Hier wird das Produkt selber und auch der Produktionsprozess getestet und analysiert. Die Kundenakzeptanz kann auch gemessen werden.

Go to Launch: (Fertig für Markteinführung)
Dies ist das letzte Gate vor der Markteinführung. Hier wird die Qualität der Aktivitäten in der "Testing and Validation"-Phase und deren Ergebnisse überprüft.

Launch: (Markteinführung)
In dieser Phase wird die Markteinführung durchgeführt.

Post-Launch Review: (Review nach Markteinführung)
Dieses Review erfolgt erst eine gewisse Zeit nach der Markteinführung. Es wird das gesamte Projekt analysiert. Es wird erörtert, was man aus diesem Projekt gelernt hat und was in späteren Projekten besser gemacht werden kann. Am Ende dieser Phase gilt das Projekt als beendet.

Die dritte Generation

In den vorhergehenden Abschnitten wurde der Stage-Gate-Prozess der zweiten Generation Beschrieben. Die erste Generation reicht bis in die 1960er Jahre zurück. Dort lag der Fokus auf technischen Kriterien und es wurden nur Reviews durchgeführt. In der zweiten Generation kamen Kriterien zur Marktsituation hinzu. Eine weitere wichtige Erweiterung war die Möglichkeit des steuernden und konstruktiven Eingriffes in den Entwicklungsprozess mit Hilfe der Gates.

Die Zweite Generation wurde von vielen Unternehmen erfolgreich aufgenommen und angepasst. Diese Anpassungen fasst Cooper in den sechs "F"-Konzepten zusammen und spricht dann von der dritten Generation.

Flexibility: (Flexibilität)
Flexibilität bedeutet, dass der Prozess nicht absolut bindend ist. Der Prozess soll flexibel an das Projekt angepasst werden. Dabei können Stages und Gates ausgelassen, gesplittet, kombiniert oder auch hinzugefügt werden. Diese Operationen werden abhängig vom Risiko-Level ausgeführt. Bei Projekten mit geringem Risiko wird der Prozess durch das Auslassen oder Kombinieren von Phasen beschleunigt. Bei Projekten mit hohem Risiko-Level wird der Prozess vergrößert. Somit kann häufiger auf das Projekt steuernd eingewirkt werden.

Das Konzept der Flexibilität beinhaltet die grundlegende Idee zum Anpassen der Gate-Netzwerke. Die Operationen, welche Cooper nennt, werden in dieser Arbeit zu den später noch folgenden Netzwerk-Operationen.

Die fehlenden fünf "F"-Konzepte haben keine besondere Bedeutung im Hinblick auf ein Anpassungskonzept für Gate-Netzwerke. Der Vollständigkeit wegen werden sie hier aufgelistet.

- Fuzzy-Gate

- Fluidity (Fluss)
- Focus (Fokus)
- Facilitation (Unterstützung)
- Forever green (Kontinuierliche Verbesserung)

2.4.2. V-Modell XT

Das V-Modell[6] ist ein Vorgehensmodell zum Planen und Durchführen von Projekten. Durch die Vorgabe konkreter, standardisierter Vorgehensweisen, zugehöriger Ergebnisse und verantwortlicher Rollen erhöht das V-Modell die Projekttransparenz. Es verbessert das Management von Projekten und erhöht nachhaltig die Erfolgswahrscheinlichkeit.

Das V-Modell hat einen militärischen Ursprung[9]. Die zivile Fassung wurde 1993 von der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt) fertiggestellt. Das "V-Modell 97" wurde 1997 veröffentlicht und ist eine Überarbeitung, welche seitdem für jegliche Softwareentwicklung in der Bundesverwaltung zur Anwendung empfohlen wurde.

Angesichts neuer Erkenntnisse in der Softwareentwicklung wurde das "V-Modell 97" wiederum Überarbeitet. In Jahre 2005 wurde das "V-Modell XT" veröffentlicht. Das XT steht dabei für Extrem Tailoring. Im "V-Modell XT" befinden sich Vorgehensbausteine, aus denen das konkrete Vorgehensmodell eines Projekts zusammengestellt wird. Dadurch wird das Anpassen an verschiedene Projekte und Organisationen ermöglicht.

Das "V" im V-Modell XT könnte als "Vorgehensmodell" interpretiert werden, steht aber eigentlich nur für die v-förmige Darstellung der Entscheidungspunkte in den Projektdurchführungsstrategien. Die Projektdurchführungsstrategien und Entscheidungspunkte werden später näher beschrieben.

Ziele

Mit der Benutzung des V-Modell XT sollen folgende Ziele erreicht werden:

Minimierung der Projektrisiken: Projektrisiken werden durch das V-Modell XT minimiert. Es werden standardisierte Vorgehensweisen festgelegt und die dazugehörigen Ergebnisse sowie die verantwortlichen Rollen werden beschrieben. Dadurch wird die Projekttransparenz gesteigert und die Projekte werden besser planbar. Die frühzeitige Erkennung von Planabweichungen wird gefördert und das ermöglicht wiederum eine frühzeitige Steuerung von Projekten.

Verbesserung und Gewährleistung der Qualität: Das V-Modell fordert eine frühzeitige Prüfung von Ergebnissen auf Vollständigkeit und Qualität. Durch die Vereinheitlichung von Produktinhalten werden die Ergebnisse besser lesbar, verständlicher und leichter zu überprüfen.

Eindämmung der Gesamtkosten: Durch die Standardisierung des V-Modells lassen sich, über den gesamten Projekt- und Systemlebenszyklus hinweg, der Aufwand und die Kosten besser einschätzen und steuern.

Verbesserung der Kommunikation aller Beteiligten: Dies wird erreicht durch die Benutzung einer einheitlichen Terminologie zwischen Nutzer, Auftraggeber, Auftragnehmer und Entwickler.

Aufbau des V-Modell XT

Das V-Modell XT ist so aufgebaut, dass es in verschiedenen Projektkonstellationen eingesetzt werden kann. Es beinhaltet eine Menge von Bausteinen und vordefinierten Strukturen. Die Bausteine und Strukturen werden abhängig von den Projekteigenschaften miteinander kombiniert und legen so den Ablauf eines Projektes fest.

Für die projektspezifische Anpassung des V-Modell XT werden folgende Elemente benötigt:

Projektmerkmale: Die Projektmerkmale charakterisieren ein konkretes Projekt. Diese Merkmale beinhalten alle Projekteigenschaften. Die wichtigsten Merkmale für das V-Modell sind der Projektgegenstand und die Projektrolle. Der Projektgegenstand ist das Ergebnis, welches im Projekt erarbeitet werden soll und die Projektrolle sagt aus, welche Rolle das Projekt im Zusammenspiel mit anderen Projekten übernimmt.

Projekttypen: Das V-Modell XT kennt vier Projekttypen “Systementwicklungsprojekt (Auftraggeber)”, “Systementwicklungsprojekt (Auftragnehmer)”, “Systementwicklungsprojekt (Auftraggeber/ Auftragnehmer)” und “Einführung und Pflege eines organisationsspezifischen Vorgehensmodells”. Der letztere Typ wird in dieser Arbeit nicht näher betrachtet.
Die Projekttypen sind abhängig von der Projektrolle und dem Projektgegenstand.

Vorgehensbausteine: Die Vorgehensbausteine sind die wesentlichen Einheiten des V-Modell XT. Vorgehensbausteine sind modular und können voneinander abhängig sein. Ein Baustein ist eine konkrete Aufgabe und umfasst dabei diejenigen Produkte, Aktivitäten und Verantwortlichkeiten, die für die Erfüllung dieser Aufgabenstellung relevant sind und damit inhaltlich zusammengehören. Zeitliche Abfolgen werden hier nicht festgelegt.

Entscheidungspunkte: Ein Entscheidungspunkt stellt einen Zeitpunkt dar. Zu diesem Zeitpunkt wird eine Projektfortschrittsentscheidung getroffen. Entscheidungspunkte definieren eine Menge von fertigzustellenden Produkten, über die Entscheidungen getroffen werden. Diese Punkte sind also Meilensteine und im Hinblick auf das Quality-Gate-Konzept stellen sie Quality Gates dar. Der Projektfortschritt kann auf diese Weise gemessen werden.

Projektdurchführungsstrategien: Projektdurchführungsstrategien definieren einen zeitlichen und organisatorischen Rahmen für das Projekt. Das V-Modell XT stellt für jeden Projekttyp mindestens eine Projektdurchführungsstrategie zur Verfügung. Die Projektmerkmale bestimmen die genauere Auswahl.

Die Projektdurchführungsstrategien bringen die für das Projekt relevanten Entscheidungspunkte in eine Reihenfolge. Die Strategie hat die Möglichkeit Entscheidungspunkte (bzw. Sequenzen von Entscheidungspunkten) mehrmals zu durchlaufen. So entstehen Iterationen. Es ist möglich, dass Handlungstränge verzweigen und parallel durchgeführt werden und danach wieder zusammen geführt werden. Desweiteren können mehrmals dieselben Entscheidungspunkte an unterschiedlichen Stellen in der Strategie verwendet werden.
Im Hinblick auf diese Arbeit stellen die Projektdurchführungsstrategien das Gate-Netzwerk dar.

Zusammenfassung

Das Tailoring im V-Modell XT geschieht also auf Basis der Projektmerkmale. Sie bestimmen den Projekttypen und die Projektdurchführungsstrategie. Der Projekttyp wiederum gibt die möglichen Vorgehensbausteine und Entscheidungspunkte vor. Die Projektdurchführungsstrategie legt den zeitlichen Rahmen fest.

Das statische Tailoring ist bis zum Entscheidungspunkt “Projekt definiert” abgeschlossen und die Anzahl und Anordnung aller Entscheidungspunkte ist im Projekthandbuch formal festgehalten. Es besteht die Möglichkeit spätere Änderungen vorzunehmen. Diese sind dann Teil des dynamischen Tailorings.

3. Konzept

Das Konzept soll das Tailoring von Gate-Netzwerken ermöglichen. Im Abschnitt 2.3 wurden die Eingabeparameter Projektkontext, Referenznetzwerk und Tailoring-Regeln für das Tailoring genannt. In diesem Konzept ist der Projektkontext durch die "Projektmerkmale" gegeben. Das Referenznetzwerk sind so genannte "Basis-Netzwerke". Die Tailoring-Regeln werden mit "Bedingungen" angegeben.

Das Ergebnis des Tailorings ist eine Menge von Gate-Netzwerken. Sie sind erweiterte Basis-Netzwerke und werden abhängig von der Bedingung in Verbindung mit den Projektmerkmalen erstellt. Die Projektmerkmale werden durch den Tailoring-Ausführer eingegeben. In den folgenden Abschnitten werden die Elemente zum Gate-Netzwerk-Aufbau noch näher erläutert.

Die Basis für das Tailoring ist in XML-Form gespeichert. Für alle folgenden Tailoring-Komponenten wird jeweils diese Form angegeben. Der XML-Aufbau der kompletten Regel-Basis ist in Abschnitt 3.5 beschrieben.

3.1. Projektmerkmal

Um das Tailoring für ein bestimmtes Projekt auszuführen, müssen für dieses Projekt beschreibende Informationen vorhanden sein. Diese Informationen werden Projektmerkmale genannt. In dem Konzept gibt es zwei verschiedene Typen von Merkmalen. Zum Einen gibt es Merkmale, die einen Wert aus einer *Liste* von Werten annehmen können. Zum Anderen gibt es Merkmale, die einen Wert aus einem *Bereich* von Werten annehmen können. Diese Typen wurden definiert als:

Selection: Wert aus einer Liste (alphanumerisch).

Range: Wert aus einem Bereich (numerisch).

Bei der Selection wird eine Liste von möglichen Werten vorgegeben. Diese Werte sind Bezeichnungen und können Buchstaben sowie auch Zahlen enthalten. Der "Tailoring-Ausführer" wählt den für sein Projekt passenden Wert aus. Eine Besonderheit der Selektion ist der *neutrale* Wert. Dieser Wert kann in der Selektionsliste mit vorgegeben werden. Wird nun vom "Tailoring-Ausführer" dieser Wert ausgewählt, hat dies die gleiche Bedeutung, als würde er alle Werte der Liste gleichzeitig auswählen.

Beispiel Selection 1:

Name des Merkmals: Hohe Risiken
Liste von Werten: { Ja, Nein}

Beispiel Selection 2: (mit neutralem Wert definiert als "-")

Name des Merkmals: Projektgegenstand
Liste von Werten: { -, Eingebettetes System, Komplexes System, Hardware-System}

Beim Typ Range ist der Bereich numerisch. Es wird eine minimaler und maximaler Wert definiert. Zur leichteren Benutzung gibt es noch eine Schrittweite. Die Schrittweite ermöglicht es dem "Tailoring-Ausführer" komfortabel den Wert des Merkmales auszuwählen. Der Wert sollte zwischen minimalem und maximalem Wert liegen.

Beispiel Range:

Name des Merkmals: Quellcodezeilen
Minimaler Wert: 100
Maximaler Wert: 1000000
Schrittweite: 1000

In der folgenden Box ist die XML-Syntax der Definition der Beispiele dargestellt:

```
<propertydefinition neutralvalue="−">
  <property name="Hohe␣Risiken" type="selection">
    <attribute name="Ja"/>
    <attribute name="Nein"/>
  </property>
  <property name="Projektgegenstand" type="selection">
    <attribute name="−"/>
    <attribute name="Eingebettetes␣System"/>
    <attribute name="Komplexes␣System"/>
    <attribute name="Hardware−System"/>
  </property>
  <property name="Quellcodezeilen" type="range" minvalue="100" maxvalue="1000000" step="1000"/>
</propertydefinition>
```

3.2. Bedingung

Eine Bedingung ist eine Überprüfung, ob ein Merkmal ein bestimmten Wert hat. Sie besitzt 3 Eingabeparameter:

Merkmalname: Name des Merkmals, welches überprüft werden soll.

Operator: Der Vergleichsoperator gibt an, wie das Merkmal überprüft werden soll. Folgende Werte sind möglich:("=", "<", "<=", ">", ">=")

Wert: Mit welchem Wert verglichen werden soll.

Diese Eingabeparameter stehen in der so genannten **Variable**.

Bei den Bedingungen für die verschiedenen Merkmalstypen (siehe 3.1)gibt es einige Einschränkungen. Ist das Merkmal vom Typ Selection, ist nur der Operator "=" erlaubt und der Wert darf alphanumerisch sein. Beim Typ Range sind alle fünf Vergleichsoperatoren möglich, aber der Wert darf nur numerisch sein.

Da das Tailoring nicht nur von einzelnen Werten von Merkmalen abhängt, sondern auch von Kombination mehrerer, sind auch die Bedingungen kombinierbar. Für die Kombination werden die drei Funktionen “UND” (Konjunktion), “ODER” (Disjunktion) und “NICHT” (Negation) benutzt. Die Funktionen stammen aus der Aussagenlogik und mit ihnen ist jede mögliche Aussage darstellbar. Daraus folgt, dass jede Aussage von Merkmalen durch Kombination von Bedingungen prüfbar ist.

Eine Bedingung ist *positiv* oder *erfüllt*, wenn die Aussage der Bedingung oder der Kombination der Bedingungen Wahr ist.

3.2.1. Präfixnotation

Die Kombination der Bedingung wird in Präfixnotation dargestellt, da diese leichter zu parsen (Einlesen in die Applikation) ist. Die allgemein gebräuchliche Schreibweise von Ausdrücken ist die Infixnotation. Dort steht der Befehl zwischen den Parametern. Beispiel (mit Befehl UND sowie den Parametern A und B):

Ist **(A UND B)** erfüllt?

In Präfixnotation werden die Befehle vor alle Parameter gestellt.

Ist **(UND A B)** erfüllt?

In dieser Arbeit sind die Befehle die Funktionen UND(Parameter_1, .., Parameter_N), ODER(Parameter_1, .., Parameter_N) und NICHT(Parameter). Die Parameter können wiederum Funktionen sein oder auch die Variablen mit den drei Eingabeparameter für die Bedingung.

Beispiel einer Bedingung:

Frage: Ist das Projekt kein “Komplexes System” und hat es weniger als 10000 Quellcodezeilen?

Bedingung(“BedA”):
Merkmalname: “Projektgegenstand”
Operator: “=”
Wert: “Komplexes System”

Bedingung(“BedB”):
Merkmalname: “Quellcodezeilen”
Operator: “<”
Wert: 10000

Bedingung als Kombination: **UND(NICHT(BedA) , BedB)**

In XML sieht die kombinierte Bedingung des Beispiels folgendermaßen aus:

```
<condition>
  <and>
    <not>
      <variable propertyname="Projektgegenstand" operator="=" attribute="Komplexes␣System" />
    </not>
    <variable propertyname="Quellcodezeilen" operator="<" attribute="10000" />
  </and>
</condition>
```

3.3. Quality Gate

Die grundlegenden Elemente des Gate-Netzwerkes sind die Quality Gates. Für das Konzept werden die Quality Gates um zusätzliche Informationen, die so genannten Metainformationen, erweitert. Die Informationen beziehen sich zum Einen auf den Entscheidung-Prozess (*Review*) und zum Anderen auf die Erstellung der Entscheidungsgrundlage (*Kriterienerstellung*).

Für diese Metainformationen wird auch die graphische Darstellung vom Quality Gate erweitert. Das Dreieck wird in vier Bereiche unterteilt und diese enthalten Freiräume für atomare Informationen.

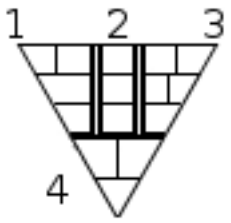


Abbildung 3.1.: *Erweitertes Quality Gate mit den 4 Bereichen*

Damit die weiteren Erläuterungen übersichtlicher sind, wurde auf die Darstellung des Quality-Gate-Namens verzichtet.

3.3.1. Metainformationen

In den folgenden Abschnitten werden die Metainformationen, deren graphische Darstellung, deren Darstellung in XML-Form und deren Bedeutung näher erläutert.

Kriterienerstellung

Die ersten drei Bereiche beziehen sich auf die Erstellung der Entscheidungsgrundlage. Diese Grundlage sind die Kriterien. Wie schon in Abschnitt 2.1.1 erwähnt, sind die Kriterien in Checklisten hinterlegt und beschreiben die Anforderungen an die Ergebnisse der Vorphase. Die Erstellung dieser Kriterien kann auf die unterschiedlichste Art und Weise geschehen.

Zeitpunkt der Erstellung
Für den Zeitpunkt der Erstellung der Kriterien gibt es vier mögliche Werte:

Startphase (Start “S”)

Planungsphase (Planning “P”)

Durchführungsphase und vor dem Gate (Realisation Before “B”)

Durchführungsphase und im Gate (Realisation In “I”)

Beispiel:

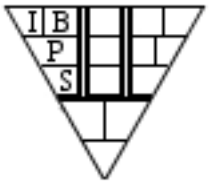


Abbildung 3.2.: *Quality Gate mit Metainformationen zum Erstellungszeitpunkt*

Das Beispiel Quality Gate enthält alle vier möglichen Werte des Zeitpunktes der Kriterienerstellung. Das bedeutet für den Projektplaner, welcher das Gate-Netzwerk benutzt, um den Gate-Prozess zu definieren, dass ihm alle Möglichkeiten zur Verfügung stehen. Er kann definieren, dass die Kriterien für dieses Quality Gate schon in der Startphase festgelegt werden, oder dass die Kriterien in Planungs- bzw. Durchführungsphase festgelegt werden. Mit “*vor dem Gate*” ist der Zeitpunkt eines früheren Quality Gates gemeint. Somit kann abgebildet werden, dass in einem Quality Gate die Kriterien für das nächste Quality Gate erstellt werden. Die Option “*im Gate*” ermöglicht die Erstellung der Kriterien direkt während der Entscheidung. Die dabei erstellten Kriterien sind im seltensten Fall ein Auslöser für eine negative Entscheidung, da sie bei der Produkterzeugung noch nicht bekannt waren. Sie haben Dokumentationscharakter oder sind gerade im Zuge der Prozessoptimierung entstanden und werden für spätere Projekte benötigt. Die Angaben des Zeitpunktes sind keine exklusiv-ODER-Verknüpfung. Der Projektplaner kann bei obigem Beispiel einen Teil der Kriterien für das Quality Gate in der Planungsphase, einen anderen Teil der Kriterien in der Durchführungsphase vor dem Gate erstellen lassen und zu den anderen Zeitpunkten keine Erstellungen festlegen.

Die Definition des Zeitpunktes XML-Form:

```
<creationtime planning="true" start="true" realisation_before="true" realisation_in="true"/>
```

Systematik der Erstellung
Die Systematik oder auch Vorgehensweise der Kriterienerstellung ist eine weitere Metainformation des Quality Gates. Hierbei wird charakterisiert, wie man bei der Erstellung vorgeht. Die drei möglichen Werte sind:

Systematisch (Systematic “S”)

Intuitiv (Intuitive “I”)

Unstrukturiert (Unstructured “U”)

“Systematisch” benötigt mehr Arbeitsaufwand als “Intuitiv”. Und “Intuitiv” benötigt wiederum mehr Arbeitsaufwand als “Unstrukturiert”.

Beispiel:

Abbildung 3.3.: *Quality Gate mit Metainformationen zur Systematik*

Je risikoreicher und ressourcenlastiger ein Projekt ist, desto wichtiger sind klar definierte und für alle beteiligten verständliche Kriterien. In einem Unternehmen haben sich mit der Zeit Vorgehensweisen zur Kriterienerstellung etabliert und werden formal festgehalten. Werden diese Vorgehensweisen genutzt, wird dies als systematisch angesehen. Systematisch können auch Vorgehensweisen aus so genannten “Best Practices” sein. Ein Beispiel wäre hier GQM. **G**oal **Q**uestion **M**etric[10] ist eine systematische Vorgehensweise zur Erstellung spezifischer Qualitätsmodelle und lässt sich als Baumstruktur sehen. Als Wurzel steht das Ziel (Goal), das über die Knoten (Questions) zu den Blättern (Metric) verfeinert wird. Weniger formell ist die intuitive Kriterienerstellung. Hierbei wird auf das Spezialwissen und die Erfahrungen des Kriterienerstellers zurückgegriffen. Die intuitive Vorgehensweise wird bei Quality Gates mit wenig Risiko eingesetzt und erfordert weniger Aufwand als die systematische Vorgehensweise. Aus Gründen der Flexibilität wurde dem Konzept noch die unstrukturierte Vorgehensweise hinzugeführt. Diese wird eingesetzt, wenn wenig oder kein Erfahrungswissen zur Kriterienerstellung vorhanden ist, oder auch bei sehr kleinen Projekten.
Die Metainformationen im Beispiel sagen aus, dass der Projektplaner die Möglichkeit hat einige Kriterien systematisch und andere intuitiv erstellen zu lassen. Er hat auch die Möglichkeit festzulegen, dass alle Kriterien systematisch erstellt werden müssen.

Die Definition der Vorgehensweise XML-Form:

```
<systematic systematic="true" intuitive="true" unstructured="true"/>
```

Individualität der Erstellung
In einem Unternehmen werden die Kriterien nicht immer wieder neu erstellt. Über mehrere Projekte hinweg entstehen Listen von Kriterien, welche immer wieder benutzt werden. Diese Listen dienen als Basis für die Kriterienerstellung. Im Folgenden wird diese Basis als Katalog bezeichnet. Idealerweise gibt es für jedes Quality Gate einen Katalog. Die Individualität ist nun die Art und Weise der Nutzung dieser Kataloge zum Definieren der Kriterien. Die Folgenden Werte sind möglich:

Fester Katalog (Defined Catalog “D”)

Auswahl aus Katalog (Catalog Select “S”)

Erweiterter Katalog (Catalog Append “A”)

Auswahl aus erweitertem Katalog (Catalog Change “C”)

Ohne Katalog (Without Catalog “W”)

Beispiel:

Abbildung 3.4.: *Quality Gate mit Metainformationen zur Individualität*

Mit “Fester Katalog” ist ein bestimmter und unveränderter Katalog mit Kriterien für das Quality Gate gemeint. Bei der Auswahl aus einem Katalog wird ein Katalog vom Projektplaner festgelegt, aus dem der Ersteller der Kriterien einzelne Kriterien auswählen muss. Beim erweiterten Katalog werden Kriterien hinzugefügt. Mit “Ohne Katalog” kann festgelegt werden, dass kein Katalog vorgegeben ist.
Die Aussage von “Auswahl aus erweitertem Katalog” (C) ist *nicht* genau die gleiche Aussage wie aus der Kombination von “Auswahl aus Katalog” (S) mit “Erweiterter Katalog” (A). Hat ein Quality Gate die Metainformation (A) und (S), kann der Projektplaner die Kriterien aus einem Katalog auswählen lassen und auch erweitern lassen. Er hat aber auch die Möglichkeit das Auswählen *nicht* zuzulassen. Somit müssen die Kriterien des festgelegten Kataloges komplett übernommen werden und können zusätzlich erweitert werden. Die Möglichkeit hat der Projektplaner mit der Metainformation (C) nicht. Desweiteren sind nicht alle Kombinationen sinnvoll. Die Kombination aus (A),(S) und (C) hat die gleiche Aussage wie die Kombination aus nur (A) und (S).

Die Definition der Individualität XML-Form:

```
<individuality defined_cat="true" catalog_sel="true" catalog_app="true" catalog_sel_app="true" without_cat="true"/
    >
```

Review

Die vierte Gruppe von Metainformationen bezieht sich auf den Entscheidungsprozess des Quality Gates das so genannte Review. Ein Review kann verschiedenartig ausgeprägt sein. Es kann zwischen sehr informell (unstrukturiert) und sehr formal (d.h. gut strukturiert und geregelt) variieren. Mögliche Werte der Metainformation sind:

Peer Review (Peer Review “P”)

Walkthrough (Walkthrough “W”)

Technisches Review (Formal Inspection “I”)

Beispiel:

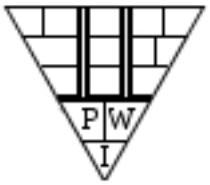


Abbildung 3.5.: *Quality Gate mit Metainformationen zum Review*

Die Techniken wurden in Abschnitt 2.1.1 bereits beschrieben. Im obigen Beispiel hat der Projektplaner die Möglichkeit eine von dem 3 Reviewtechniken für das Quality Gate auszuwählen.

Die Definition Reviews XML-Form:

```
<techniques formalinspection="true" walkthrough="true" peerreview="true"/>
```

Standard Metainformationen

In dieser Arbeit werden folgende Metainformationen als Standard festgelegt.

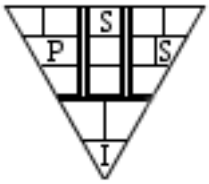


Abbildung 3.6.: *Standard-Metainformationen vom Quality Gate*

Im ersten Bereich steht das “P” für Planungsphase, das bedeutet, dass die Kriterien für das Quality Gate währende der Planungsphase festgelegt werden. Das “S” im zweiten Bereich sagt, dass dafür eine systematische Vorgehensweise benutzt wird. Das “S” im dritten Bereich sagt aus, dass die Grundlage für alle Kriterien schon in Katalogen definiert sind und aus denen werden nur noch einzelne Kriterien ausgewählt. Auf Grund des “I” im vierten Bereich wird für das Quality Gate ein formles Review durchgeführt.

Die Standardmetainformationen besitzen in jedem der vier Bereiche nur einen Eintrag. Es sind aber auch mehr möglich. Wenn in einem Bereich mehrere Einträge vorhanden sind, kann der Projektplaner einen Auswählen oder auch eine Kombination wählen (z.B. nur einen Teil der Kriterien in der Planungsphase festlegen lassen und den anderen Teil später).

3.4. Aufbau eines Gate-Netzwerkes

Für den Aufbau eines Gate-Netzwerkes werden zunächst die Grundlegenden Elemente, die Quality Gates, benötigt. Sie werden mit Metainformationen erweitert. Danach werden dann aus den Quality Gates die Basis-Netzwerke zusammengestellt. Die Basis-Netzwerke werden dann abhängig von den Bedingungen (Siehe Abschnitt 3.2) zu Gate-Netzwerken erweitert.

3.4.1. Definition der Quality Gates

Die erste Aufgabe bei der Erstellung von Quality Gates ist die Deklaration des Quality-Gate-Namens. Zusätzlich müssen die Bezeichnungen des Start- und Endknotens des Gate-Netzwerkes definiert werden. Die Definition geschieht in der so genannten Gate-Liste. Die folgende Box zeigt die XML-Form der Liste aus dem Beispiel aus Abschnitt 2.1.2:

```
<gatelist startnodename="Start" endnodename="Ende">
  <gate name="Projekt␣abgeschlossen"/>
  <gate name="Anforderungen␣festgelegt"/>
  <gate name="System␣entwickelt"/>
  <gate name="Komponenten␣gekauft"/>
</gatelist>>
```

Die zweite Aufgabe bei der Erstellung von Quality Gates ist das Hinzufügen von Metainformationen. Die Metainformationen werden in zwei Schritten hinzugefügt. Beim ersten Schritt werden die Informationen zur Kriterienerstellung und im zweiten Schritt die Informationen des Reviews hinzugefügt. In beiden Schritten werden die so genannten **Alternativen** genutzt.

Alternativen

In einem Unternehmen können die gleichen Quality Gates (gleiche Bezeichnung) in verschiedenen Projekten mit unterschiedlichem Projektkontext auch unterschiedliche Metainformationen besitzen. Das Konzept unterstützt diese Eigenschaft mit der Einführung von alternativen Quality-Gates.

Dafür werden die Metainformationen der Quality Gates in zwei Bereichen definiert. Der eine Bereich ist für die Voreinstellungen (default) und der andere ist für die Alternativen (alternative).

Die Default-Metainformationen müssen für jedes Quality Gate angegeben werden. Um abhängig von Projektkontext verschiedene Alternativ-Metainformationen hinzuzufügen, werden Bedingungen eingesetzt.

In dem Konzept wird die Definition der Metainformationen für die Kriterienerstellung (criteria) getrennt von der Definition der Metainformationen für das Review (review). Die beiden folgenden Beispiel sollen das “Default-Alternativ-Konzept” verdeutlichen.

Beispiel:
Die folgende Box zeigt die XML-Form der Definition der Kriterienerstellung ohne Alternativen. Bei jedem Quality Gate werden die Kriterien systematisch in der Planungsphase erstellt,

indem sie aus einem Katalog ausgewählt werden.

```
<criteria>
  <default>
    <gate name="Anforderungen␣festgelegt">
      <individuality catalog_sel="true"/>
      <creationtime planning="true" />
      <systematic systematic="true"/>
    </gate>
    <gate name="System␣entwickelt">
      <individuality catalog_sel="true"/>
      <creationtime planning="true" />
      <systematic systematic="true"/>
    </gate>
    <gate name="Komponenten␣gekauft">
      <individuality catalog_sel="true"/>
      <creationtime planning="true" />
      <systematic systematic="true"/>
    </gate>
    <gate name="Projekt␣abgeschlossen">
      <individuality catalog_sel="true"/>
      <creationtime planning="true" />
      <systematic systematic="true"/>
    </gate>
  </default>
</criteria>
```

Beispiel:
Die Definition der Review-Metainformationen wird analog zu den Metainformationen der Kriterienerstellung angelegt. Das folgende Beispiel enthält zusätzlich auch Alternativen. Standardmäßig wird bei jedem Quality Gate eine formale Inspektion durchgeführt, also ein formales Review gemacht. Wenn das Projekt weniger als 5000 Quellcodezeilen hat, wird bei dem Quality Gate “System entwickelt” auf das aufwändige formale Review verzichtet und es wird nur ein Walkthrough durchgeführt.

```
<review>
  <default>
    <gate name="Komponenten␣gekauft">
      <techniques formalinspection="true"/>
    </gate>
    <gate name="System␣entwickelt">
      <techniques formalinspection="true"/>
    </gate>
    <gate name="Projekt␣abgeschlossen">
      <techniques formalinspection="true"/>
    </gate>
    <gate name="Anforderungen␣festgelegt">
      <techniques formalinspection="true"/>
    </gate>
  </default>
  <alternative>
    <condition>
      <variable propertyname="Quellcodezeilen" operator="<" attribute="5000" />
    </condition>
    <gate name="System␣entwickelt">
```

```
      <techniques walkthrough="true"/>
    </gate>
  </alternative>
</review>
```

Es besteht die Möglichkeit, dass ein Quality Gate mehrere verschiedene Alernativen hat. Für die Gate-Netzwerke wird dann nur die *letzte*(in Reihenfolge der Regel-Datei) Alternative mit positiver Bedingung eingesetzt. Es werden sozusagen die Metainformationen des Quality Gates überschrieben.

3.4.2. Erstellung der Basis-Netzwerke

Der nächste Schritt für den Aufbau von Gate-Netzwerken ist die Definition von Basis-Netzwerken. Sie können im Hinblick auf das Tailoring als die Referenznetzwerke angesehen werden. Basis-Netzwerke können wiederum auf anderen Basis-Netzwerken basieren. Jedes Basis-Netzwerk, welches *nicht* auf einem anderen Basis-Netzwerk basiert, hat das "Default-Basis-Netzwerk" als Grundlage. Es besteht nur aus dem Start- und Endknoten ohne Quality Gates.

Abbildung 3.7.: *Default-Basis-Netzwerk mit Start- und Endknoten*

Basis-Netzwerke werden mit Netzwerk-Operationen erstellt.

Operationen auf Netzwerken

Mit Hilfe von Netzwerk-Operationen werden die Basis-Netzwerke aus Quality Gates aufgebaut. Mit diesen Operationen geschieht das Anpassen der Gate-Netzwerke an den Projektkontext. Im folgenden werden die möglichen Operationen erläutert.

SplitGateLinear
Mit SplitGateLinear wird in dem Gate-Netzwerk ein Quality Gate (Quelle) durch zwei aufeinander folgende Quality Gates ersetzt. Das erste neue Quality Gate erhält alle eingehenden Übergänge der Quelle und das zweite neue Quality Gate erhält alle ausgehenden Übergänge der Quelle.

Definition:

```
splitGateLinear(<zu ersetzende Gate>,
                <erstes neues Gate>,
                <zweites neues Gate>);
```

Beispiel:

```
splitGateLinear(Entwurf erstellt,
                Grobentwurf erstellt,
                Feinentwurf erstellt);
```

Das Netz

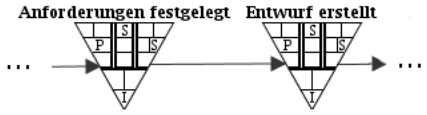


Abbildung 3.8.: *Teil-Netzwerk mit Quality Gate für den Entwurf*

wird zu:

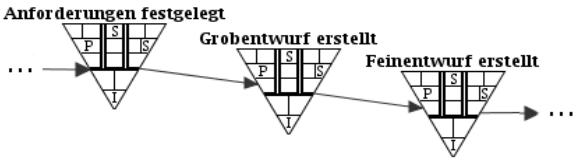


Abbildung 3.9.: *Teil-Netzwerk mit zwei Quality Gates für den Entwurf*

Darstellung in XML-Form:

```
<splitgatelinear sourcegate="Entwurf␣erstellt" firstgate="Grobentwurf␣erstellt" secondgate="Feinentwurf␣erstellt"/>
```

JoinGates

Das genaue Gegenteil von SplitGateLinear ist JoinGates. Es werden zwei direkt aufeinander folgende Quality Gates (A und B) durch ein neues Quality Gate ersetzt. JoinGates darf nur ausgeführt werden, wenn Quality Gate A nur Quality Gate B als einzigen Nachfolger und Quality Gate B nur Quality Gate A als einzigen Vorgänger hat.

Definition:

```
joinGates(<erstes zu ersetzende Gate>,
          <zweites zu ersetzende Gate>,
          <neues Gate>);
```

Beispiel:

```
joinGates(Grobentwurf erstellt,
          Feinentwurf erstellt,
          Entwurf erstellt);
```

Das Netz (siehe 3.9) wird zu: (siehe 3.8)

Darstellung in XML-Form:

```
<joingates firstgate="Grobentwurf␣erstellt" secondgate="Feinentwurf␣erstellt" targetgate="Entwurf␣erstellt"/>
```

SplitGateParallel

Bei SplitGatePrarallel wird in dem Gate-Netzwerk ein Quality Gate (Quelle) in zwei unabhängig voneinander liegende Quality Gates aufgeteilt. Die neuen Quality Gates erhalten alle eingehenden und ausgehenden Übergänge der Quelle.

Definition:

```
splitGateParallel(<zu ersetzende Gate>,
                  <erstes neues Gate>,
                  <zweites neues Gate>);
```

Beispiel:

```
splitGateParallel(System entwickelt,
                  System entwickelt,
                  Komponenten gekauft);
```

Das Netz

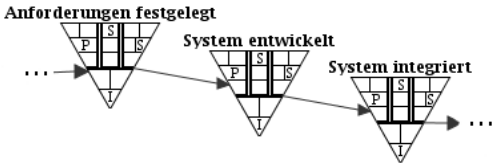


Abbildung 3.10.: *Teil-Netzwerk ohne gekaufte Komponenten*

wird zu:

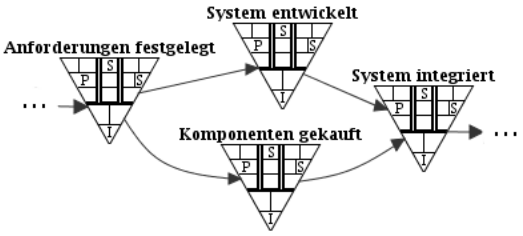


Abbildung 3.11.: *Teil-Netzwerk ohne gekaufte Komponenten*

Darstellung in XML-Form:

```
<splitgateparallel sourcegate="System␣entwickelt" firstgate="Komponenten␣gekauft" secondgate="System␣
    entwickelt"/>
```

AddAfterGate

Mit der Operation AddAfterGate wird ein neues Quality Gate hinter einem vorhandenen Quality Gate eingefügt. Das neue Quality Gate übernimmt alle ausgehenden Übergänge des vorhandenen Quality Gates und es wird ein Übergang vom vorhandenen zum neuen Quality Gate erstellt.

Definition:

```
addAfterGate(<Gate vor der Einfügeposition>,
             <einzufügende Gate>);
```

Beispiel:

```
addAfterGate(Anforderungen festgelegt,
             System implementiert);
```

Das Netz

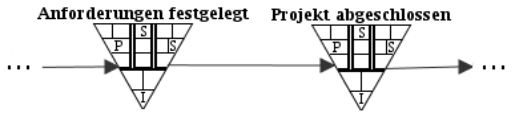


Abbildung 3.12.: *Teil-Netzwerk ohne eigenständiges Quality Gate während der Implementierung*

wird zu:

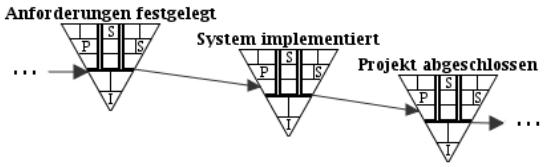


Abbildung 3.13.: *Teil-Netzwerk mit eigenständigem Quality Gate während der Implementierung*

Darstellung in XML-Form:

```
<addaftergate posgate="Anforderungen festgelegt" inputgate="System implementiert"/>
```

AddBeforeGate

Mit der Operation AddBeforeGate wird ein neues Quality Gate vor einem vorhandenen Quality Gate eingefügt. Das neue Quality Gate übernimmt alle eingehende Übergänge des vorhandenen Quality Gates und es wird ein Übergang vom neuen zum vorhandenen Quality Gate erstellt.

Definition:

```
addBeforeGate(<Gate hinter der Einfügeposition>,
              <einzufügende Gate>);
```

Beispiel:

```
addBeforeGate(Entwurf erstellt,
              System entwickelt);
```

Das Netz

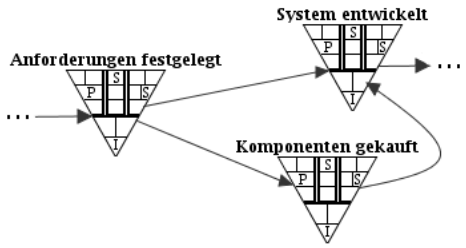


Abbildung 3.14.: *Teil-Netzwerk ohne Entwurfs-Quality-Gate*

wird zu:

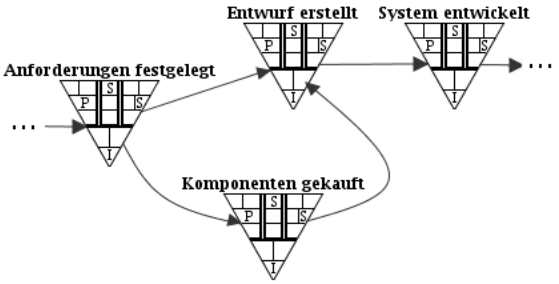


Abbildung 3.15.: *Teil-Netzwerk mit Entwurfs-Quality-Gate*

Darstellung in XML-Form:

```
<addbeforegate inputgate="Entwurf␣erstellt" posgate="System␣entwickelt" />
```

AddBetweenGates
Die Operation AddBetweenGates fügt ein neues Quality Gate zwischen zwei andere Quality Gates (A und B) ein. Es wird ein Übergang von A zu dem neuen Quality Gate und ein Übergang vom neuen Quality Gate zu B erzeugt. **Wichtig**: Existiert zwischen A und B bereits ein Übergang, wird dieser *nicht* entfernt. Ist dies aber gewünscht, muss der Übergang mit Delete-Loop explizit gelöscht werden.

Definition:

```
addBetweenGates(<Gate vor der Einfügeposition>,
                <einzufügende Gate>,
                <Gate hinter der Einfügeposition>);
```

Beispiel:

```
addBetweenGates(Anforderungen festgelegt,
                Komponenten gekauft,
                System entwickelt);
```

Das Netz

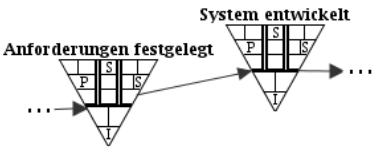


Abbildung 3.16.: *Teil-Netzwerk ohne gekaufte Komponenten*

wird zu:

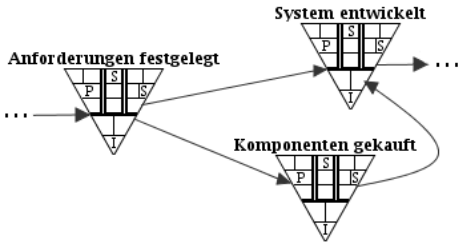


Abbildung 3.17.: *Teil-Netzwerk mit gekauften Komponenten*

Darstellung in XML-Form:

```
<addbetweengates inputgate="Komponenten gekauft" fromgate="Anforderungen festgelegt" togate="System
    entwickelt"/>
```

InsertLoop
Mit InsertLoop werden Iterationen erstellt. Es wird ein Übergang von einem Quality Gate (A) zu einem anderen Quality Gate (B) hinzugefügt. Wie im Beispiel können A und B das gleiche Quality Gate sein. Mit der gleichen Operation ist es auch möglich, einen Quality-Gate-Übersprung zu modellieren. Dabei wird ein Übergang zum “übernächsten”Quality Gate eingefügt.

Definition:

```
insertLoop(<Startgate des Übergangs>,
           <Zielgate des Übergangs>);
```

Beispiel:

```
insertLoop(System entwickelt,
           System entwickelt);
```

Das Netz

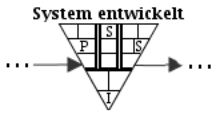


Abbildung 3.18.: *Teil-Netzwerk mit Quality Gate*

wird zu:

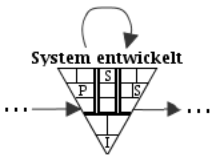


Abbildung 3.19.: *Teil-Netzwerk mit Quality Gate und einer Iteration auf sich selbst*

Darstellung in XML-Form:

```
<insertloop fromgate="System␣entwickelt" togate="System␣entwickelt"/>
```

DeleteLoop
DeleteLoop ist das Gegenteil von Insertloop. Es wird ein Übergang aus dem Gate-Netzwerk entfernt.

Definition:

```
deleteLoop(<Startgate des Übergangs>,
           <Zielgate des Übergangs>);
```

Beispiel:

```
deleteLoop(System entwickelt,
           System entwickelt);
```

Das Netz (siehe 3.19) wird zu: (siehe 3.18)

Darstellung in XML-Form:

```
<deleteloop fromgate="System␣entwickelt" togate="System␣entwickelt"/>
```

RemoveGate
Mit RemoveGate wird ein Quality Gate aus dem Gate-Netzwerk entfernt. Dabei wird jeder eingehende Übergang mit jedem ausgehenden Übergang verbunden.

Definition:

```
removeGate(<zu löschende Gate>);
```

Beispiel:(ohne Bild)

```
removeGate(Komponenten gekauft);
```

Darstellung in XML-Form:

```
<removegate gate="Komponenten␣gekauft"/>
```

ReplaceGate
Bei ReplaceGate wird ein Quality Gate (Alt) durch ein anderes Quality Gate (Neu) ausgetauscht. Das Quality Gate Neu erhält alle eingehenden und ausgehenden Übergänge von Quality Gate Alt.

Definition:

```
replaceGate(<zu ersetzende Gate>,
            <neues Gate>);
```

Beispiel:(ohne Bild)

```
replaceGate(Projekt abgeschlossen,
            Markteinführung erfolgt);
```

Darstellung in XML-Form:

```
<replacegate oldgate="Projekt␣abgeschlossen" newgate="Markteinfuehrung␣erfolgt"/>
```

Komplettes Basis-Netzwerk

Als Beispiel für ein komplettes Basis-Netzwerk wird das Gate-Netzwerk aus Abschnitt 2.1.2 genommen. Das Netzwerk basiert auf dem Default-Basis-Netzwerk, da keine Basis angegeben wurde.

Darstellung in XML-Form:

```
<basenetwork name="Basisnetz" basedon="">
    <addaftergate posgate="Start" inputgate="Anforderungen␣festgelegt"/>
    <addaftergate posgate="Anforderungen␣festgelegt" inputgate="System␣entwickelt"/>
    <addaftergate posgate="System␣entwickelt" inputgate="Projekt␣abgeschlossen"/>
    <insertloop fromgate="System␣entwickelt" togate="System␣entwickelt"/>
    <addbetweengates inputgate="Komponenten␣gekauft" fromgate="Anforderungen␣festgelegt" togate="System␣
        entwickelt"/>
  </basenetwork>
```

Benutzung der Netzwerk-Operationen

Die Netzwerk-Operationen erlauben es, dass man mit unterschiedlichen Kombinationen der Operationen identische Netzwerke erzeugt. Die Netzwerk-Operationen sollten möglichst übersichtlich kombiniert werden (evtl. mit Leerzeilen zur Strukturierung und Kommentaren). Auch sollten Operationen den Sinn ihrer Benutzung wiederspiegeln. D.h warum wurde gerade diese Operation an dieser Stelle gewählte. Dies erleichtert die Aufgabe des Tailoring-Veränderers.

3.4.3. Erstellung der Gate-Netzwerke

Ein Gate-Netzwerk basiert *immer* auf einem Basis-Netzwerk. Jedes Gate-Netzwerk besitzt eine Bedingung (siehe 3.2) und kann einen Erweiterungsteil haben. Die Bedingung ist eine Voraussetzung für das Gate-Netzwerk. Wenn die Bedingung positiv ist, wird das Gate-Netzwerk dem Tailoring-Ausführer angezeigt. Im Erweiterungsteil kann das Basis-Netzwerk weiter verändert werden. Dafür werden die gleichen Operationen, wie bei den Basis-Netzwerken eingesetzt.

Beispiel:
Das Basis-Netzwerk “Basisnetz” beinhaltet kein Quality-Gate “Test” hinter “System entwickelt”. Wenn für das Projektmerkmal “Test des Systems” durch den Tailoring-Ausführer die Eigenschaft “Ja” eingegeben wurde, wird das Basis-Netzwerk um das Quality-Gate “Test” erweitert und dann als Gate-Netzwerk “Netzwerk mit Test” ausgegeben. Bei “Nein” wird das Gate-Netzwerk “Netzwerk ohne Test”, welches keine Erweiterungen zum Basis-Netzwerk enthält, ausgegeben.

Darstellung in XML-Form:

```
<networkdefinition>
  <basenetwork name="Basisnetz">
    <addaftergate posgate="Start" inputgate="Anforderungen␣festgelegt"/>
    <addaftergate posgate="Anforderungen␣festgelegt" inputgate="System␣entwickelt"/>
    <addaftergate posgate="System␣entwickelt" inputgate="Projekt␣abgeschlossen"/>
  </basenetwork>
  <gatenetwork displayname="Netzwerk␣ohne␣Test" basedon="Basisnetz">
    <condition>
      <variable propertyname="Test␣des␣Systems" operator="=" attribute="Nein" />
    </condition>
  </gatenetwork>
  <gatenetwork displayname="Netzwerk␣mit␣Test" basedon="Basisnetz">
    <condition>
      <variable propertyname="Test␣des␣Systems" operator="=" attribute="Ja" />
    </condition>
    <extension>
      <addaftergate posgate="System␣entwickelt" inputgate="System␣getestet"/>
    </extension>
  </gatenetwork>
</networkdefinition>
```

Es besteht die Möglichkeit, dass mehrere Gate-Netzwerke positive Bedingungen haben. Ist dies der Fall, werden auch alle diese Gate-Netzwerke ausgegeben und dem Tailoring-Ausführer angezeigt.

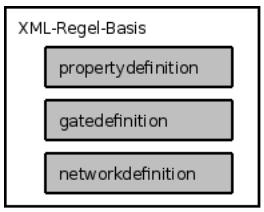
3.5. Regel-Basis

Die Regel-Basis wird in einer XML-Datei gespeichert. Deren Aufbau ist im Folgenden Abschnitt beschrieben. Im Anhang befindet sich zusätzlich die Dokumenttypdefinition[11] (DTD) für die XML-Datei. Eine Dokumenttypdefinition beschreibt die Struktur eines Dokumentes, indem die Reihenfolge, die Verschachtelung der Elemente und die Art des Inhaltes von Attributen festgelegt werden.

3.5.1. Aufbau der XML-Datei

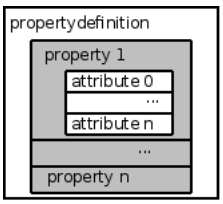
Das Wurzelelement der Datei (in dieser Arbeit **qganetta**) hat ein Attribut *version*. Dort ist die Versionsnummer oder Bezeichnung der Regel-Basis festgehalten.

Die XML-Datei ist in drei Blöcke unterteilt.



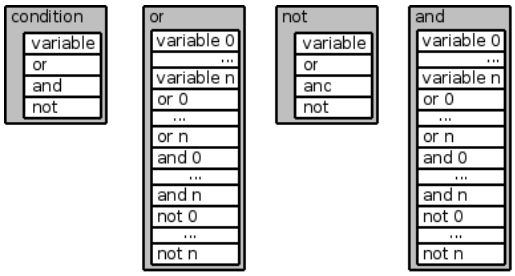
Propertydefinition

Das Element **propertydefinition** hat das Attribut *neutralvalue* (siehe 3.1). Innerhalb Propertydefinition sind ein oder mehrere **property**-Elemente. Attribute von Property sind *name,type,minvalue,maxvalue,step*. Der Typ hat die möglichen Werte “range” und “selection”. Bei Typ Selektion enthält das Property-Element mehrere **attribute**-Elemente. Den Attribute-Elementen wird ein Name (*name*) gegeben. Jedes Attribute ist ein Eintrag in der Werteliste.



Bedingung

Eine Bedingung wird an mehreren Stellen der XML-Datei genutzt und befindet sich in dem **condition**-Element. Wenn eine Bedingung durch die Projekteigenschaften positiv wird, werden auch die auf das Condition-Element folgenden Elemente (der gleichen Hierarchiestufe) beachtet. Andernfalls werden diese Überlesen. Das Condition-Element ist eine Kombination von **variable**-, **or**-, **not**- und **and**-Elementen.

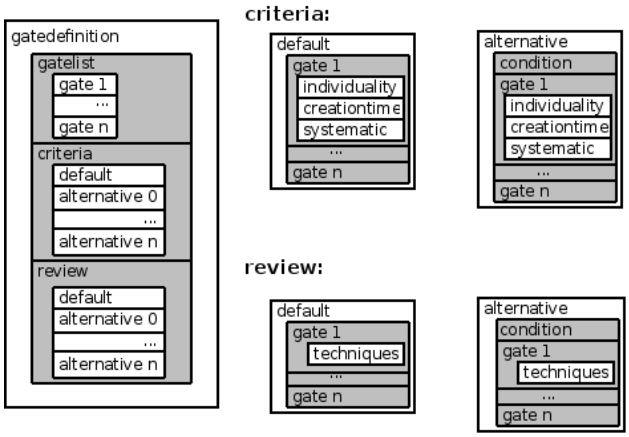


Die einzelnen Bedingungen stehen in dem Variable-Element und nur dieses hat Attribute. Es hat *propertyname* für die Merkmalsbezeichnung, *operator* für den Vergleichsoperator und *attribute* für den Vergleichswert.

Wichtig: Für den Vergleichsoperator sind folgende Werte möglich:(“=”, “<”, “<=”, “>”, “>=”). Sie müssen in XML maskiert werden. Maskiert lauten die Werte folgendermaßen: (“=”, “<”, “<=”, “>”, “>=”)

Gatedefinition

Das **gatedefinition**-Element besitzt drei weitere Elemente.



Im Element **gatelist** werden alle Quality Gates deklariert. Hier wird zuerst über die Attribute *startnode* und *endnode* die Bezeichner des Startknoten und des Endknoten festgelegt. Die Gatelist enthält mehrere **gate**-Elemente über deren Attribut *name* der Namen jedes Quality Gates festgelegt wird.
Unter dem Element **criteria** werden die Default- und die Alternativ-Metainformationen für die Kriterienerstellung definiert. Die Metainformationen wurden unter Abschnitt 3.3.1 und 3.4.1 näher beschrieben. Das Element **default** enthält mehrere Gate-Elemente (jedes steht für ein Quality Gate, welches über das Attribut *name* eindeutig identifiziert wird). An dieser Stelle sind auch unter dem Gate-Element weitere Elemente, wie **individuality**, **creationtime** und **systematic**. Das Individuality-Element hat die Attribute *catalog_sel* (für Auswahl aus Katalog), *catalog_app* (für Katalog mit Erweiterungen), *catalog_sel_app* (für Auswahl aus Katalog mit Erweiterungen), *defined_cat* (für fester definierter Katalog) und *without_cat* (für ohne Katalog).

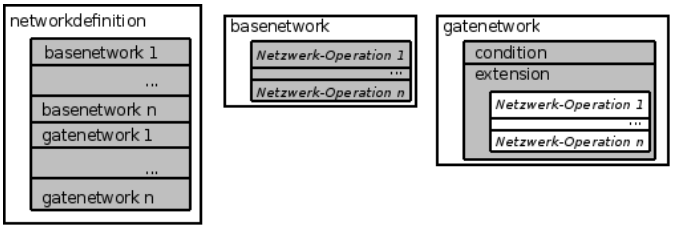
Das Creationtime-Element hat die Attribute *start* (für Projektstart), *planning* (für Planungsphase), *realisation_before* (für vor dem Gate in der Durchführungsphase) und *realisation_in* (für während des Gates). Das Systematic-Element hat die Attribute systematic (für systematischen Vorgehen), intuitive (für intuitives Vorgehen) und unstructured (für unstrukturiertes Vorgehen). Wenn eines dieser Attribute den Wert **true** enthält, ist diese Metainformation gesetzt.

Das **alternative**-Element ist fast genauso aufgebaut, wie das Default-Element. Es steht nur am Anfang zusätzlich immer eine Bedingung (siehe 3.2 und 3.5.1). Ist diese Bedingung positiv, werden die folgenden Gate-Elemente gelesen und die Metainformationen übernommen. Es kann beliebig viele Alternativen geben und jede Alternative kann die Metainformation für beliebig viele Quality Gates definieren.

Der Aufbau des **review**-Elementes ist analog zu dem Criteria-Element. Nur die Stelle der Metainformationen ist anders. Dort befindet sich das **techniques**-Element und es hat die Attribute *peerreview* (für das Peer Review) *walkthrough* (für das Walk Through) und *formalinspection* (für das technische Review). Wenn eines dieser Attribute den Wert **true** enthält, ist diese Metainformation gesetzt.

Networkdefinition

In dem Element **networkdefinition** werden alle Basis- und Gate-Netzwerke definiert.



Unter dem Networkdefinition-Element befinden sich mindestens ein **basenetwork**-Element und mindestens ein **gatenetwork**-Element. Das Basenetwork-Element hat die Attribute *name* (für die Bezeichnung den Netzes) und *basedon* (Basis-Netzwerk, auf dem Aufgebaut wird). Ist kein *basedon* angegeben wird auf dem Default-Basis-Netzwerk (siehe 3.4.2) aufgebaut. In dem Basenetwork-Element befinden sich die einzelnen Netzwerk-Operationen (siehe 3.4.2), die das Netzwerk aufbauen. Das Gatenetwork-Element hat als Attribute *displayname* (Zur Anzeige in der Applikation) und auch *basedon*. Hier ist das *basedon*-Attribut ein Pflichtattribut. In dem Gatenetwork-Element befindet sich erst ein **condition**-Element und dann ein optionales **extension**-Element. In dem Extension-Element befinden sich wiederum die Netzwerk-Operationen, welche das Basis-Netzwerk(über *basedon* referenziert) erweitern. Wenn die Bedingung im Condition-Element positiv ist, wird die Erweiterung ausgeführt und das entstandene Gate-Netzwerk mit angezeigt. Ist die Bedingung positiv, aber keine Erweiterung angegeben, so ist das angezeigte Gate-Netzwerk eine Kopie seines Basis-Netzwerkes.

3.6. Anwendungsfälle

In den Folgenden Abschnitten werden einige Anwendungsfälle kurz erläutert. Die Fälle betreffen alle die Erstellung/Veränderung von Tailoring-Regeln und die Benutzung der Regeln. Es wird erläutert wer diese Aufgabe erledigt und welche Schritte er durchführen muss.

Erstellen der Regel-Basis

Zu diesem Zeitpunkt sind noch keine Tailoring-Regeln vorhanden. Der Tailoring-Ersteller ist derjenige, der die Regel-Basis erstellt. Dafür durchläuft er folgende Schritte:

Schritt 1: Identifizieren der Projektmerkmale, welche für das Tailoring ausschlaggebend sind (siehe 3.1).

Schritt 2: Übertragen dieser Projektmerkmale in den Block Propertydefinition (siehe 3.5.1) der Regel-Basis

Schritt 4: Identifizieren der einzelnen Quality Gates mit deren Metainformationen und gegebenenfalls deren Alternativen.

Schritt 5: Übertragen der Quality Gates in den Block Gatedefinition (siehe 3.5.1)der Regel-Basis

Schritt 6: Identifizieren der Gate-Netzwerke mit deren Bedingungen.

Schritt 7: Aus den Gemeinsamkeiten der verschiedenen Gate-Netzwerke die Basis-Netzwerke erstellen

Schritt 8: Basis-Netzwerke und Gate-Netzwerke mit den einzelnen Erweiterungen in die Regel-Basis übernehmen (siehe 3.5.1)

Nach der Erstellung der Regel-Basis können Gate-Netzwerke vom Tailoring-Ausführer modelliert werden.

Hinzufügen von Projektmerkmalen

Wenn schon eine Regel-Basis vorhanden ist, ist es die Aufgabe des Tailoring-Veränderers diese Regeln zu erweitern/optimieren. Wenn ein neues Projektmerkmal hinzugefügt werden soll, sind folgende Schritte nötig:

Schritt 1: Identifizieren des Types des Projektmerkmals (siehe 3.1).

Schritt 2a: (Typ: Range) Identifizieren des maximalen bzw. des minimalen Wertes sowie der komfortablen Schrittweite.

Schritt 2b: (Typ: Selection) Identifizieren der möglichen Werte.

Schritt 3: Übertragen dieser Projektmerkmale in den Block Propertydefinition (siehe 3.5.1) der Regel-Basis

Wenn das Projektmerkmal hinzugefügt wurde, kann es in den Bedingungen der Regel-Basis genutzt werden (siehe 3.5.1).

Erstellung von Bedingungen

Der Tailoring-Ersteller und der Tailoring-Veränderer benötigen Bedingungen für die Voraussetzung der alternativen Quality Gates und der Gate-Netzwerke. Für die Erstellung benötigen sie folgende Schritte:

Schritt 1: Identifizieren der benötigten Projektmerkmale.

Schritt 2: Erzeugen der Bedingung in Präfixnotation (siehe 3.2.1).

Schritt 3: Übertragen der Bedingung an die benötigte Stelle der Regel-Datei.

Hinzufügen von Quality Gates

Im Zuge der Optimierung ist es möglich, dass Quality Gates zum Gate-Netzwerk hinzugefügt werden sollen. Diese Aufgaben übernimmt der Tailoring-Veränderer. Dafür befolgt er diese Schritte:

Schritt 1: Identifizieren der Metainformationen und deren Alternativen des Quality Gates (siehe 3.3.1).

Schritt 2: Übertragen der Quality-Gate-Informationen in den Block Gatedefinition (siehe 3.5.1) der Regel-Basis.

Schritt 3: Einbinden des Quality Gates mit Hilfe der Netzwerk-Operationen (siehe 3.4.2) an die gewünschte Stelle.

Hinzufügen von Gate-Netzwerken

Wenn ein neues Gate-Netzwerk hinzugefügt werden soll, hat der Tailoring-Veränderer folgende Schritte zu beachten:

Schritt 1: Wenn das neue Gate-Netzwerk auch neue Projekteigenschaften benötigt, müssen diese erst eingefügt werden.

Schritt 2: Wenn das neue Gate-Netzwerk neue Quality Gates oder auch neue Alternativen besitzt, müssen diese erst eingefügt werden.

Schritt 3: Identifizieren der Bedingung des neuen Gate-Netzwerkes.

Schritt 4: Überprüfen, ob das neue Gate-Netzwerk auf einem schon vorhandenen Basis-Netzwerk basieren kann. Wenn ja, werden nur die Erweiterungen in das Gate-Netzwerk übernommen (siehe 3.4.3). Wenn nein, muss zusätzlich ein neues Basis-Netzwerk erstellt (siehe 3.4.2).

Es ist möglich, dass das neue Gate-Netzwerk (Neu) auf einem schon vorhandenen Gate-Netzwerk (Alt) basiert. Wenn dies der Fall ist, sollten die Netzwerk-Operationen aus dem extension-Element des Gate-Netzwerkes (Alt) herausgenommen und damit ein neues Basis-Netzwerk erstellt werden. Danach basieren beide Gate-Netzwerke (Neu und Alt) auf dem neuen Basis-Netzwerk. Gate-Netzwerk (Alt) hat dann keine Erweiterungen, sondern nur noch die alte Bedingung und Gate-Netzwerk (Neu) enthält die neue Bedingung aus Schritt 3 und die Netzwerk-Operationen zur Erweiterung.

Ausführen des Tailorings

Der Tailoring-Ausführer benutzt die Applikation zur Erstellung der Gate-Netzwerke.

Schritt 1: Auswahl der Regel-Basis (üblicherweise gibt es in einem Unternehmen nur eine aktuelle Version)

Schritt 2: Aus den Vorgegebenen möglichen Projektmerkmalen die auswählen, welche zum aktuellen Projekt passen

Schritt 3: Es werden eine Menge passender Gate-Netzwerke angezeigt. Der Tailoring-Ausführer speichert diese Netze (in dieser Arbeit als Grafik).

Schritt 4: Übergabe der Gate-Netzwerke an den Projektplaner

Der Projektplaner nutzt diese Gate-Netzwerke als Ausgangspunkt der Projektplanung.

4. Realisierung

In diesem Kapitel wird die Realisierung des Tailoring-Konzeptes beschrieben. Erst wird darauf eingegangen welche Software eingesetzt wurde. Dann wird der Aufbau der Realisierung beschrieben. Am Ende wird auf die Benutzung der Applikation eingegangen.

Die entwickelte Applikation hat den Namen “QGaNetTA” bekommen und das steht für “Quality Gate-Network Tailoring Assistent”.

Die Applikation wird in der Programmiersprache Java (version 1.6) programmiert und benutzt Java-Swing für die Benutzeroberfläche. Als Java-Entwicklungsumgebung wird “eclipse” eingesetzt. Sie ist die zur Zeit bekannteste Entwicklungsumgebung für Java.

Gate-Netzwerke werden als gerichtete Grafen angesehen. Zur grafischen Darstellung dieser Grafen wird die Open-Source-Bibliothek JGraph[8] verwendet. Sie beinhaltet Java-Swing-Komponenten, welche in die Applikation eingebunden werden.
Am Anfang der Entwicklung wurde zur grafischen Darstellung der Gate-Netzwerke noch Graphviz[7] eingesetzt. Graphviz ist ein Open-Source-Programmpaket zur Visualisierung gerichteter und ungerichteter Grafen. Der Vorteil von Graphviz ist, dass es Algorithmen zur automatischen Anordnung der Knoten des Grafen enthält und somit eine grafische Darstellung der Grafen mit möglichst wenigen Überschneidungen erzeugt. Der Nachteil ist, dass es ein externes Programmpaket ist und es sich nicht komfortabel in eine Java-Applikation einbinden lässt. Zusätzlich besitzt Graphviz *keine* Funktion zum Zeichen der benutzerdefinierten Knoten (Quality Gates). Wegen der Nachteile wurde Graphviz nicht weiter eingesetzt.
JGraph bietet dagegen die Möglichkeit zum Zeichen der benutzerdefinierten Knoten. Es bietet zwar auch Algorithmen zur automatischen Anordnung der Knoten, aber *nicht* in der kostenlosen Version. In der Applikation wird dafür nur ein rudimentärer Algorithmus implementiert und der Benutzer übernimmt die Aufgabe der übersichtlichen Anordnung.

4.1. Package-Übersicht

Die Applikation wurde in verschiedene Packages aufgeteilt. Dabei wurde die grafische Oberfläche vom Datenmodell und von anderen Aufgaben getrennt. Dies erhöht die Wartbarkeit der Anwendung. In der folgenden Abbildung sind die Beziehungen der einzelnen Packages zueinander dargestellt. Für jedes Package wird der Inhalt und die Aufgaben näher erläutert.

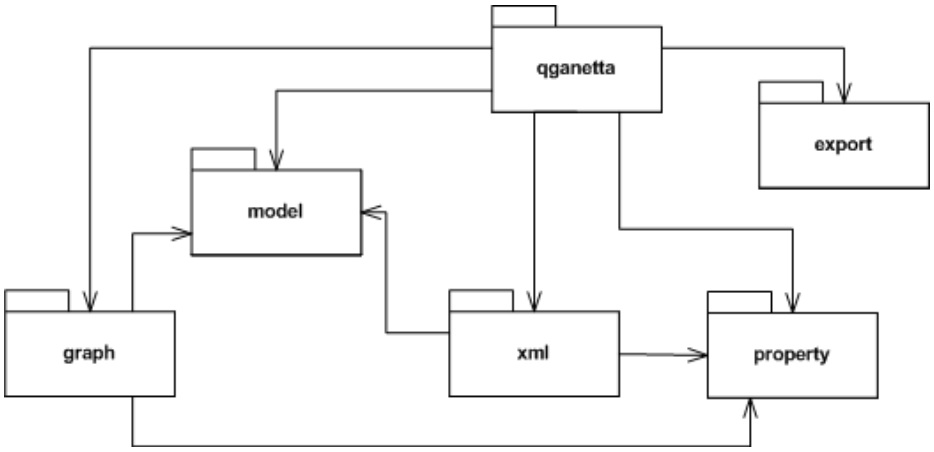


Abbildung 4.1.: *Beziehungen der Packages zueinander*

Package qganetta
Qganetta ist der Hauptteil der Applikation. Hier befindet sich die *main*-Methode zum Starten der Anwendung. Desweiteren befinden sich in dem Package die Klassen für die grafische Benutzeroberfläche und der Benutzerinteraktion.

Package model
Im Package model befinden sich die Klassen für die Datengrundlage der Gate-Netzwerke mit Quality Gates und deren Metainformationen. Es sind auch die Methoden zur Manipulation der Gate-Netzwerke und Quality Gates vorhanden. Eine deutlichere Übersicht ist in 4.2 zu sehen.

Package graph
Hier sind die Klassen für die grafische Darstellung der Gate-Netzwerke enthalten. Dabei werden die Swing-Komponenten der JGraph-Bibliothek überschrieben und an die Anforderungen des Konzeptes angepasst. Das Package graph benötigt als Datengrundlagen die Packages model und property.

Package xml
Im Package xml werden die Daten aus der XML-Regel-Datei eingelesen. Die Klassen hier kennen den Aufbau/ Struktur dieser Datei und übernehmen den Inhalt in das Datenmodell. Hier werden die Gate-Netzwerke abhängig vom Projektkontext erzeugt.

Package property
Dieses Package ist ein Unter-Package von model. Hier befindet sich das Datenmodell der Projektmerkmale. Zum Einen ist das die Liste mit allen möglichen Projektmerkmalen aus der Regel-Basis und zum Anderen sind das die vom Tailoring-Ausführer ausgewählten Projektmerkmale.

Package export
Im Package export ist nur die Methode zum grafischen Export der Gate-Netzwerke enthalten.

4.2. Klassenübersicht des Datenmodells

Das Datenmodell der Gate-Netzwerke befindet sich im Package model. Das folgende Klassendiagramm zeigt die wichtigsten Elemente dieses Datenmodells.

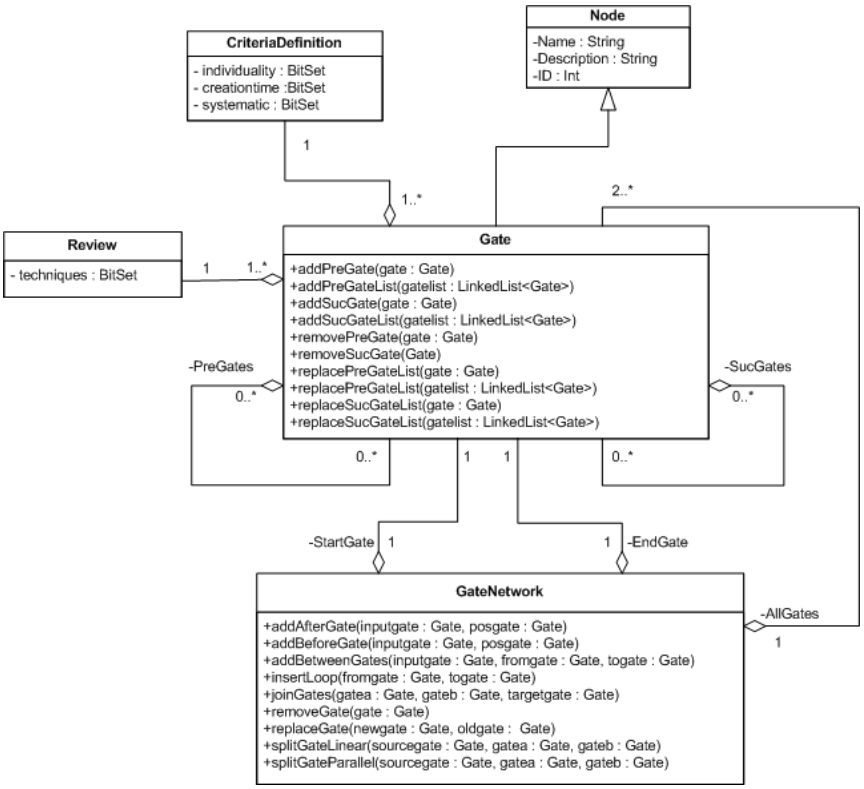


Abbildung 4.2.: *Klassendiagramm des Gate-Netzwerkes*

Zentraler Punkt in diesem Datenmodell ist das Gate. Dieses repräsentiert ein Quality Gate. Die Metainformationen des Quality Gates stehen in Review und CriteriaDefinition von der Klaase Gate. Die Variable *techiques* in Review und die Variablen *individuality*, *creationtime* und *systematic* in CriteriaDefinition enthalten die einzelnen gesetzten Werte der Metainformationen. Ein Gate besitzt zusätzlich noch eine Liste von Vorgänger-Gates (*PreGates*) und eine Liste von Nachfolger-Gates (*SucGates*). Die Nachfolger repräsentieren die Übergänge in dem Gate-Netzwerk. Die Vorgänger sind für das Gate-Netzwerk nicht nötig, aber sie vereinfachen die Netzwerk-Operationen. Die Methoden von Gate beziehen sich nur auf das Gate bzw. die Vorgänger und Nachfolger.
In GateNetwork werden der Startknoten (*StartGate*) und der Endknoten (*EndGate*) festgelegt. Die Variable *AllGates* beinhaltet eine Referenz auf alle Gates des Netzwerkes und wird in den Netzwerk-Operationen zum identifizieren der Gates benötigt. Die Methoden in GateNetwork sind die definierten Netzwerk-Operationen des Konzeptes. Sie verändern die Struktur des Gate-Netzwerkes.

4.3. Grafische Darstellung

In diesem Abschnitt werden alle Benutzeroberflächen der Applikation beschrieben. Gleichzeitig wird der Ablauf der Benutzung der Applikation dargestellt.

4.3.1. Benutzeroberfläche

Nach dem Start der Applikation muss die Regel-Basis geladen werden. Dafür erscheint ein Öffnen-Dialog. Die zu öffnende Datei muss eine komplette Regel-Basis, wie im Konzept(3.5) festgelegt, beinhalten.

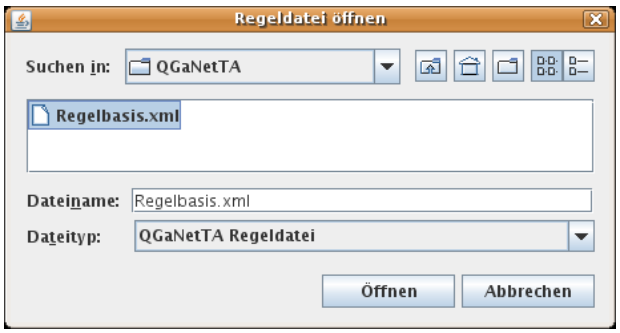


Abbildung 4.3.: *Dialog zum öffnen der Regel-Basis*

Nach dem Laden der Regel-Basis werden die Projektmerkmale eingegeben. Der Dialog zum Eingeben der Projektmerkmale zeigt alle möglichen Projektmerkmale und deren Werte aus der Regel-Basis. Die für das aktuelle Projekt passenden Werte müssen ausgewählt werden. Mit “Ok” werden die Einstellungen übernommen.

Abbildung 4.4.: *Dialog zur Eingabe der Projektmerkmale*

Nach dem Bestätigen der Projektmerkmale wird das Hauptfenster der Applikation angezeigt. Im Folgenden werden die einzelnen Bereiche und deren Benutzung näher erläutert.

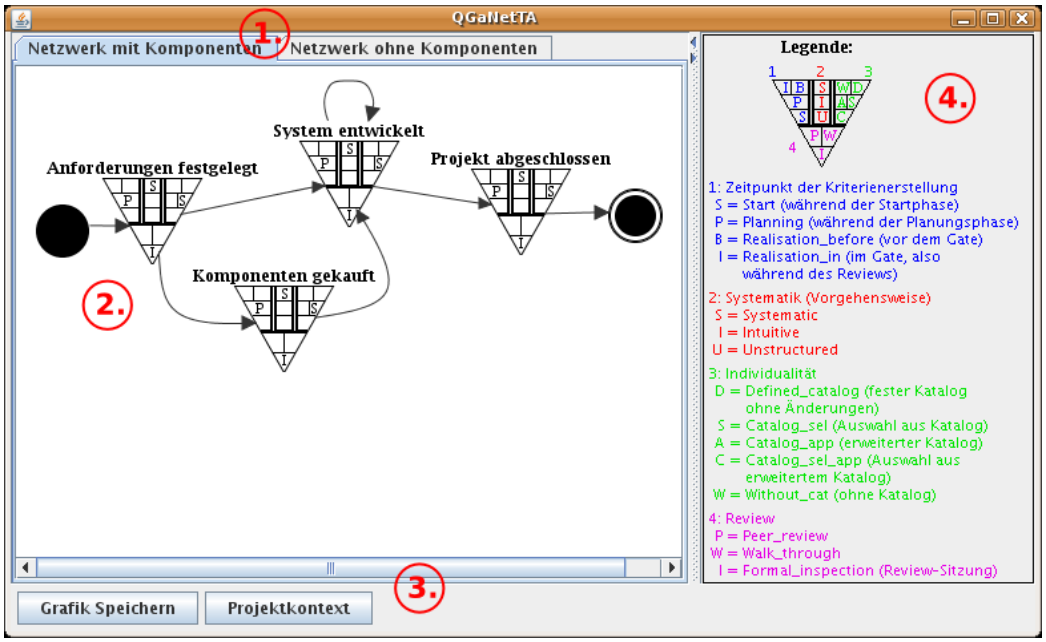


Abbildung 4.5.: *Hauptoberfläche der Applikation*

Bereich 1: Für jedes Gate-Netzwerk, welches eine positive Bedingung hat, wird hier ein neues Register erstellt. Der angezeigte Titel stammt vom Attribut *displayname* des Gate-Netzwerkes.

Bereich 2: In diesem Bereich wird das Gate-Netzwerk angezeigt. Die einzelnen Quality Gates können mit der Maus an die gewünschte Position verschoben werden. Dabei muss die linke Maustaste gedrückt gehalten werden. Die Übergänge zwischen den Quality Gates sind zuerst gerade Linien. Durch einen Rechtsklick mit der Maus auf einen Übergang werden Wegpunkte hinzugefügt. Diese Punkte können dann auch mit der Maus an die gewünschte Position verschoben werden.

Bereich 3: Am unteren Rand der Applikation sind die zwei Button “Grafik Speichern” und “Projektkontext”. Beim Aufruf von Projektkontext wird der Dialog zur Eingabe der Projektmerkmale geöffnet. Hier können die Einstellungen nochmals verändert werden. Nach der Bestätigung werden die Gate-Netzwerke neu modelliert. Beim Aufruf von Grafik Speichern wird das aktuell zu sehende Gate-Netzwerk als Grafik gespeichert. Es erscheint ein Speichern-Dialog zum festlegen des Namens bzw. des Speicherortes. Die erstellte Grafik wird im nächsten Abschnitt erläutert (siehe 4.3.2).

Bereich 4: An der Rechten Seite ist eine Legende dargestellt. Sie erklärt kurz, welche Bedeutung die Metainformationen in den Quality Gates haben. An der linken Seite der Legende sind zwei kleine Pfeile. Mit diesen Pfeilen kann die Legende ausgeblendet bzw. wieder eingeblendet werden.

4.3.2. Grafik-Export

Der Grafik-Export wird durch “Grafik Speichern” ausgelöst. Dabei wird das aktuelle Gate-Netzwerk in eine Bilddatei des Types “Portable Network Graphics” (kurz PNG) umgewandelt. Das exprotierte Bild kann nicht wieder in die Applikation eingelesen werden. Das Bild wird nun näher erläutert.

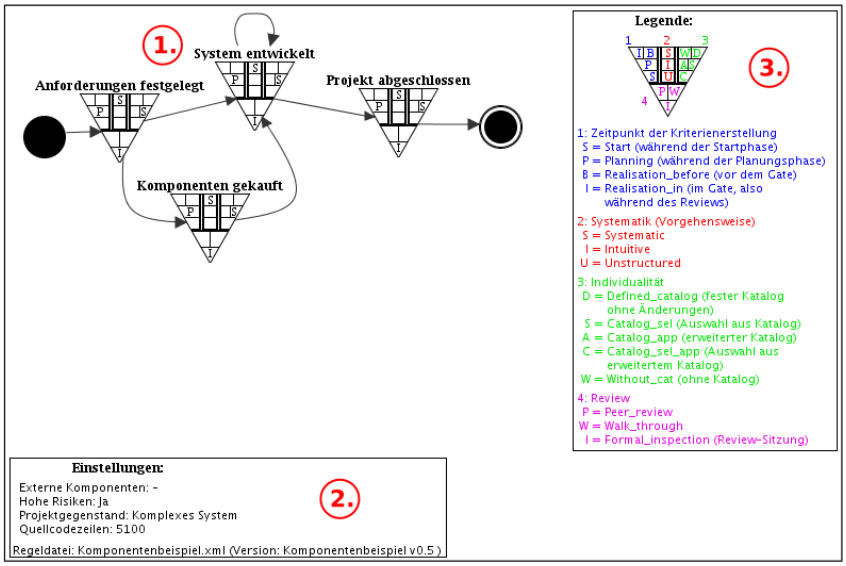


Abbildung 4.6.: *Bildausgabe eines Gate-Netzwerkes*

Bereich 1: Hier ist die Abbildung des Gate-Netzwerkes zu sehen. Sie ist identisch mit der Abbildung in der Applikation.

Bereich 2: Unter dem Gate-Netzwerk wird eine weitere Legende eingefügt. Dort stehen alle Projektmerkmale und deren gewählte Werte. Desweiteren steht dort der Name der benutzten XML-Regel-Datei sowie deren Versionsbezeichnung aus dem Attribut *version* des Wurzelelementes.

Bereich 3: Die rechte Legende ist wiederum identisch mit der Legende der Applikation.

5. Konzept angewandt auf das V-Modell XT

Das Konzept wird in diesem Abschnitt exemplarisch auf das V-Modell XT (siehe 2.4.2) angewandt. Dafür muss das Erfahrungswissen aus diesem Modell in das Konzept des "Tailoring von Gate-Netzwerken" überführt werden. Es ist also eine Regel-Basis zu erstellen.

Für die Regel-Basis sind zur Zeit noch keine Grundlagen vorhanden. Dies bedeutet, dass die Aufgaben des Tailoring-Erstellers zu leisten sind. In den Anwendungsfällen (siehe 3.6) des Konzeptes sind die einzelnen Schritte zur Erstellung der Regel-Basis Aufgeführt. Diese Schritte werden im weiteren Verlauf vollzogen.

In dieser Arbeit wird das Erstellen der Gate-Netzwerke des V-Modell XT nur an zwei Gate-Netzwerken durchgeführt. Die Regel-Basis für die fehlenden Gate-Netzwerke ist auf der CD zu dieser Arbeit zu finden.

5.1. Identifizieren der Projektmerkmale

Im ersten Schritt werden die Projektmerkmale identifiziert. Das V-Modell XT definiert eine Reihe von Projektmerkmalen. Nicht alle Projektmerkmale bzw. ihre Werte sind für das Tailoring der Gate-Netzwerke von Bedeutung. Die ausschlaggebenden Projektmerkmale sind:

Projektgegenstand: Dieses Merkmal kann einen Wert aus einer Liste von Werten annehmen. Daraus folgt, dass das Merkmal Projektgegenstand vom Typ "Selection" ist. Folgende Werte sind möglich:

- Eingebettetes System
- HW-System
- Komplexes System
- SW-System
- Systemintegration

Projektrolle: Projektrolle ist auch von Typ "Selection" und hat folgende Werte:
(AG = Auftraggeber, AN = Auftragnehmer)

- AG mit einem AN
- AG mit mehreren AN
- AN ohne Unterauftragnehmer
- AN mit Unterauftragnehmer
- AG/AN ohne Unterauftragnehmer

- AG/AN mit Unterauftragnehmer

Systemlebenszyklusausschnitt: (Typ “Selection”)

- Entwicklung
- Wartung und Pflege
- Weiterentwicklung und Migration

Fertigprodukte: (Typ “Selection”)

- Ja
- Nein

Hohe Realisierungsrisiken: (Typ “Selection”)

- Ja
- Nein

Zu jedem Merkmal wird noch der “neutrale” Wert (siehe 3.1) in die Liste eingetragen. Die Definition der Merkmale wird nun in die XML-Regel-Datei übernommen. Der Inhalt der Datei ist im Anhang dieser Arbeit zu finden.

5.2. Identifizieren der Quality Gates

Das V-Modell XT definiert Entscheidungspunkte. Diese sind gleichzusetzen mit den Quality Gates. Folgende 18 sind für diese Arbeit von Bedeutung:

- Projekt genehmigt
- Projekt definiert
- Anforderungen festgelegt
- Projekt ausgeschrieben
- Angebot abgegeben
- Projekt beauftragt
- Iteration geplant
- System spezifiziert
- System entworfen
- Feinentwurf abgeschlossen
- Systemelemente realisiert

- System integriert
- Lieferung durchgeführt
- Projektfortschritt überprüft
- Abnahme erfolgt
- Projekt abgeschlossen
- Gesamtprojekt aufgeteilt
- Gesamtprojektfortschritt überprüft

Das Konzept benötigt noch Start- bzw. Endknoten. Diese werden als “Start” und “Ende” hinzugefügt.

Für die Quality Gates müssen danach die Metainformationen definiert werden. Das V-Modell XT macht keine konkreten Aussagen zur Kriterienerstellung und zur Art der Entscheidungsfindung. Es definiert lediglich, das die Projektfortschrittsentscheidung durch einen “Lenkungsausschuss” getroffen werden. Dieses kommt der “formalen Inspektion” am nächsten. Aus diesen Gründen werden für alle Quality Gates des V-Modell XT die “Standard Metainformationen” (siehe 3.3.1) verwendet. Demzufolge sind auch keine alternativen Quality Gates in der Regel-Basis vorhanden.

Anschließend werden die Quality-Gate-Definitionen in die Regel-Basis (siehe Anhang) übernommen.

5.3. Identifizieren der Gate-Netzwerke

Im V-Modell XT repräsentieren die Projektdurchführungsstrategien die Gate-Netzwerke. Folgende Strategien existieren im V-Modell XT: (für die Arbeit irrelevanten Strategien wurden ausgelassen)

- Vergabe und Durchführung eines Systementwicklungsprojektes (AG)
- Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG)
- Inkrementelle Systementwicklung (AN)
- Komponentenbasierte Systementwicklung (AN)
- Agile Systementwicklung (AN)
- Wartung und Pflege von Systemen (AN)
- Inkrementelle Systementwicklung (AG/AN)
- Komponentenbasierte Systementwicklung (AG/AN)
- Agile Systementwicklung (AG/AN)

- Wartung und Pflege von Systemen (AG/AN)

Die weiteren Schritte werden nur an den beiden ersten Strategien veranschaulicht. Die Erstellung der Gate-Netzwerke aus den restlichen Strategien erfolgt analog dazu.

Nun müssen die Bedingungen für jede Strategie extrahiert und in Präfixnotation umgewandelt werden. Im V-Modell XT sind folgende Bedingungen aufgelistet:

Vergabe und Durchführung eines Systementwicklungsprojektes (AG): Ist gültig, wenn die Merkmale eine Kombination von folgenden Werten haben.

Projektgegenstand: Eingebettetes System, HW-System, Komplexes System, SW-System, Systemintegration

Projektrolle: AG mit einem AN

Daraus folgt diese Bedingung in Präfixnotation:

```
AND(OR(Projektgegenstand = “Eingebettetes System”,
       Projektgegenstand = “HW-System”,
       Projektgegenstand = “Komplexes System”,
       Projektgegenstand = “SW-System”,
       Projektgegenstand = “Systemintegration”),
    Projektrolle = “AG mit einem AN”)
```

Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG): Ist gültig, wenn die Merkmale eine Kombination von folgenden Werten haben.

Projektgegenstand: Eingebettetes System, HW-System, Komplexes System, SW-System, Systemintegration

Projektrolle: AG mit mehreren AN

Daraus folgt diese Bedingung in Präfixnotation:

```
AND(OR(Projektgegenstand = “Eingebettetes System”,
       Projektgegenstand = “HW-System”,
       Projektgegenstand = “Komplexes System”,
       Projektgegenstand = “SW-System”,
       Projektgegenstand = “Systemintegration”),
    Projektrolle = “AG mit mehreren AN”)
```

Das Gate-Netzwerk für “Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG)” ist:

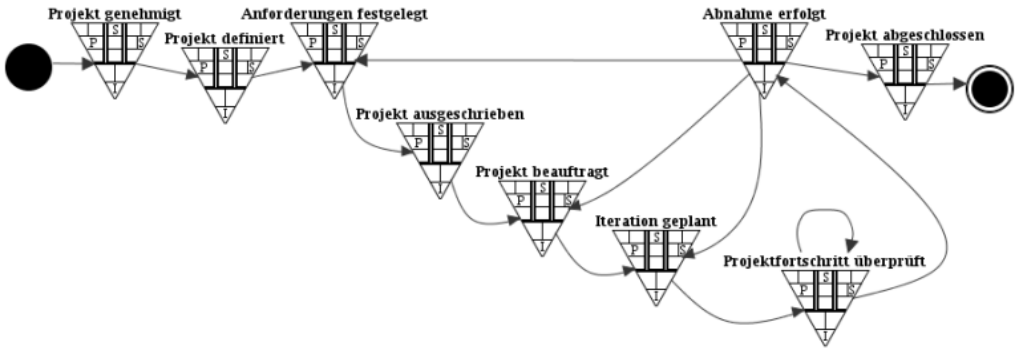


Abbildung 5.1.: *Gate-Netzwerk für Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG)*

Das Gate-Netzwerk für “Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG)” ist:

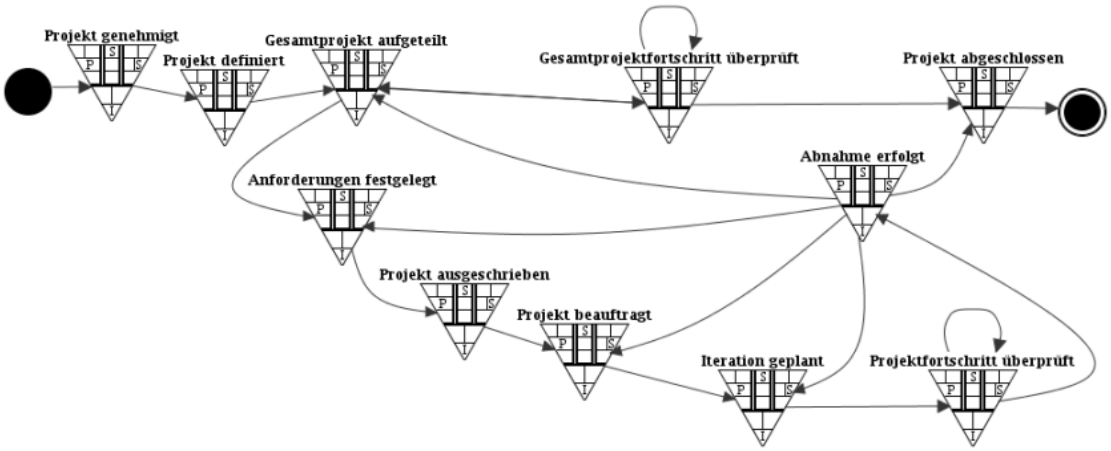


Abbildung 5.2.: *Gate-Netzwerk für Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG)*

5.4. Basis-Netzwerke erstellen

Aus den beiden oben gezeigten Gate-Netzwerken müssen jetzt die Gemeinsamkeiten in Basis-Netzwerke überführt werden. Das eliminiert Redundanzen aus der Regel-Basis und erleichtert die Wartung durch den Tailoring-Veränderer.

Während dieser Arbeit hat sich eine **Heuristik** zum Erzeugen der Basis-Netzwerke ergeben. Eine Heuristik ist ein Lösungsverfahren für ein Problem. Die gefundene Lösung ist dabei nicht zwangsweise die bestmögliche Lösung, aber die Wahrscheinlichkeit, eine optimale Lösung zu finden, ist sehr hoch.

Ausgangsbasis für das Lösungsverfahren sind die vorhandenen Gate-Netzwerke. Es werden folgende Schritte ausgeführt:

Schritt 1: Am Anfang wird in jedem Gate-Netzwerk ein Pfad vom Startknoten bis zum Endknoten markiert. Dieser Pfad sollte möglichst alle Knoten (Quality Gates) des Netzwerkes beinhalten. Kein Knoten darf dabei mehrmals vorkommen. Die definierten Eigenschaften eines Gate-Netzwerkes garantieren, dass solch ein Pfad immer existiert.

Schritt 2: Nun werden alle Pfade miteinander verglichen. Zuerst werden aus allen Pfaden diejenigen Knoten gelöscht, welche nicht auch in jedem anderen Pfad vorhanden sind. Jeder Pfad besitzt jetzt die gleiche Länge und die gleichen Knoten (evtl. in einer anderen Reihenfolge).

Schritt 3: Für jeden Index(Position der Knoten im Pfad) werden die Schritte 4, 5 und 6 durchgeführt. Angefangen wird am Startknoten eines jeden Pfades.

Schritt 4: Alle aktuellen Knoten werden verglichen. Drei Fälle können auftreten:

- Sind alle Knoten der aktuellen Position gleich, wird Schritt 4 mit dem nächsten Index wiederholt.
- Exitstiert die aktuelle Position in einem der Pfade nicht wird zum Schritt 7 übergegangen.
- Es sind unterschiedliche Knoten vorhanden. Der Knoten, der sich am häufigsten an der aktuellen Position befindet, bleibt bestehen. Er wird sich als **X** gemerkt. Weiter mit Schritt 5.

Schritt 5: In allen Pfaden **P**, wo sich **X** nicht an der aktuellen Position befindet, befindet sich **X** an späteren Positionen. Jetzt werden in diesen Pfaden **P** alle Knoten von der aktuellen Position bis zur Position *vor* **X** gelöscht.

Schritt 6: Jeder in Schritt 5 gelöschte Knoten wird auch in allen anderen Pfaden gelöscht. Nun befindet sich **X** in jedem Pfad an der gleichen Position. Weiter bei Schritt 4 mit dem nächsten Index.

Schritt 7: Die Erstellung ist zu Ende. Das gesuchte Basis-Netzwerk entspricht dem Pfad vom Startknoten bis zum vorletzten Index.

Die Qualität dieser Heuristik hängt stark von der Wahl der Pfade ab. Die entstandenen Basis-Netzwerke sind “einfach” strukturiert und besitzen keine Verzweigungen. Die Basis-Netzwerke sollten nach der Verwendung der Heuristik zusätzlich noch bearbeitet werden, so dass sie eine sinngemäße Einheit bilden. Dadurch wird die spätere Arbeit des Tailoring-Veränderers erleichtert.

Das Lösungsverfahren wird auf die oben genannten Gate-Netzwerke angewendet. Die folgende Abbildung visualisiert das Ergebnis:

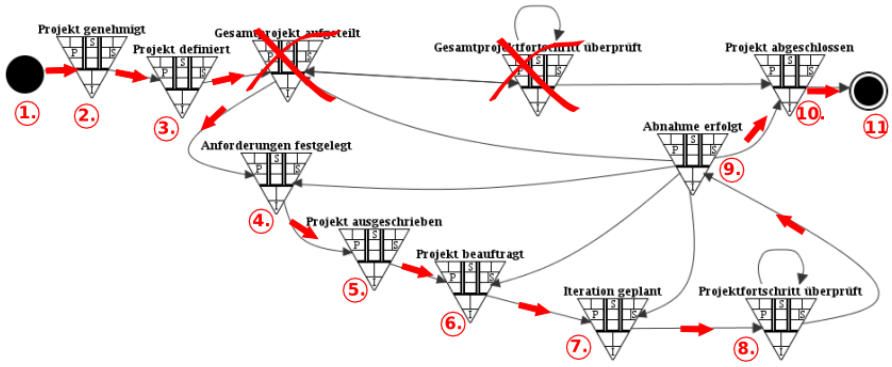


Abbildung 5.3.: *Pfad durch das Gate-Netzwerk*

Die 11 Knoten (Quality Gates) aus der Abbildung bilden das neue Basis-Netzwerk. Sie können mit der Netzwerk-Operation “AddAfterGate” eingebunden werden. Desweiteren sind einige Übergänge in beiden Gate-Netzwerken identisch. Diese werden mit “InsertLoop” eingefügt.

Darstellung des Basis-Netzwerkes in XML:

```
<basenetwork name="Systementwicklung␣AG" >
  <addaftergate posgate="Start" inputgate="Projekt␣genehmigt"/>
  <addaftergate posgate="Projekt␣genehmigt" inputgate="Projekt␣definiert"/>
  <addaftergate posgate="Projekt␣definiert" inputgate="Abnahme␣erfolgt"/>
  <addaftergate posgate="Abnahme␣erfolgt" inputgate="Projekt␣abgeschlossen"/>
  <addaftergate posgate="Projekt␣definiert" inputgate="Anforderungen␣festgelegt"/>
  <addaftergate posgate="Anforderungen␣festgelegt" inputgate="Projekt␣ausgeschrieben"/>
  <addaftergate posgate="Projekt␣ausgeschrieben" inputgate="Projekt␣beauftragt"/>
  <addaftergate posgate="Projekt␣beauftragt" inputgate="Iteration␣geplant"/>
  <addaftergate posgate="Iteration␣geplant" inputgate="Projektfortschritt␣ueberprueft"/>

  <insertloop fromgate="Abnahme␣erfolgt" togate="Anforderungen␣festgelegt"/>
  <insertloop fromgate="Abnahme␣erfolgt" togate="Projekt␣beauftragt"/>
  <insertloop fromgate="Abnahme␣erfolgt" togate="Iteration␣geplant"/>
  <insertloop fromgate="Projektfortschritt␣ueberprueft" togate="Projektfortschritt␣ueberprueft"/>
</basenetwork>
```

Da die Projektdurchführungsstrategie “Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG)” auf “Vergabe und Durchführung eines Systementwicklungsprojektes (AG)” nur aufbaut, ist in diesem Fall das Gate-Netzwerk von “Vergabe und Durchführung eines Systementwicklungsprojektes (AG)” identisch mit dem Basis-Netzwerk. Die Definition dieses Gate-Netzwerkes beinhaltet nur eine Bedingung und keine Erweiterungen.

Darstellung in XML:

```
<gatenetwork displayname="Systementwicklungsprojekt␣(AG)␣mit␣einem␣AN" basedon="Systementwicklung␣AG
      ">
  <condition>
    <and>
      <or>
        <variable propertyname="Projektgegenstand" operator="=" attribute="Systemintegration" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="HW−System" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="Eingebettetes␣System" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="Komplexes␣System" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="SW−System" />
      </or>
      <variable propertyname="Projektrolle" operator="=" attribute="AG␣mit␣einem␣AN" />
    </and>
  </condition>
</gatenetwork>
```

Die Definition des Gate-Netzwerkes “Vergabe und Durchführung mehrerer Systementwicklungsprojekte (AG)” beinhaltet zusätzlich zur Bedingung auch Erweiterungen.
Darstellung in XML:

```
<gatenetwork displayname="Systementwicklungsprojekt␣(AG)␣mit␣mehreren␣AN" basedon="Systementwicklung␣
      AG">
  <condition>
    <and>
      <or>
        <variable propertyname="Projektgegenstand" operator="=" attribute="Systemintegration" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="HW−System" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="Eingebettetes␣System" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="Komplexes␣System" />
        <variable propertyname="Projektgegenstand" operator="=" attribute="SW−System" />
      </or>
      <variable propertyname="Projektrolle" operator="=" attribute="AG␣mit␣mehreren␣AN" />
    </and>
  </condition>
  <extension>
    <addaftergate inputgate="Gesamtprojekt␣aufgeteilt" posgate="Projekt␣definiert"/>
    <addbetweengates inputgate="Gesamtprojektfortschritt␣ueberprueft" fromgate="Gesamtprojekt␣aufgeteilt"
          togate="Projekt␣abgeschlossen"/>

    <insertloop fromgate="Gesamtprojektfortschritt␣ueberprueft" togate="Gesamtprojektfortschritt␣ueberprueft"/>
    <insertloop fromgate="Gesamtprojektfortschritt␣ueberprueft" togate="Gesamtprojekt␣aufgeteilt"/>
    <insertloop fromgate="Abnahme␣erfolgt" togate="Gesamtprojekt␣aufgeteilt"/>
  </extension>
</gatenetwork>
```

Dies war der letzte Schritt zur Erstellung der Regel-Basis. Sie kann nun mit der Applikation genutzt werden, um die Gate-Netzwerke zu modellieren. Der komplette Inhalt dieser Regel-Basis ist im Anhang zu finden.

6. Fazit

In diesem Abschnitt werden die Ergebnisse dieser Arbeit noch einmal zusammengefasst und bewertet.

6.1. Zusammenfassung

Ziel dieser Arbeit war die Erstellung eines Konzeptes, welches das Tailoring von Gate-Netzwerken ermöglicht. Die dritte Generation des Stage-Gate-Prozesses von Cooper lieferte die Ideen zur Optimierung von Gate-Netzwerken. Aus diesen Ideen entstanden die Netzwerk-Operationen des Konzeptes.

Mit Hilfe der Basis-Netzwerke und Gate-Netzwerke der Regel-Basis, ist es möglich verschiedene Quality-Gate-Netzwerke der Unternehmen formal festzuhalten. Die Projektmerkmale und Bedingungen ermöglichen das Anpassen der Gate-Netzwerke. Diese werden in der Regel-Basis auch formal festgehalten. Das Konzept bietet die Möglichkeit, jederzeit das Tailoring zu optimieren und weitere Erfahrungen einfließen zu lassen.

Zusätzlich wurden die Quality Gates um Metainformationen erweitert. Sie definieren, wie die Entscheidungsgrundlage erstellt wird und in welcher Weise die Entscheidungsfindung des Quality Gates abläuft. Auch ist es möglich, die Metainformationen abhängig vom Projektkontext zu definieren.

Während der Erstellung des Konzeptes ist eine Heuristik zum Finden von Basis-Netzwerken aus einer Menge von Gate-Netzwerken entstanden. Die Heuristik ist möglicherweise noch optimierbar, da sie nur lineare Basis-Netzwerke erzeugt.

Das Konzept sollte exemplarisch das Tailoring von Gate-Netzwerken des V-Modell XT ermöglichen. Diese Aufgabe konnte das Konzept leisten und die Ergebnisse sind auf der CD zu dieser Arbeit zu finden.

Desweiteren war es Aufgabe, das Konzept in einer Java-Applikation zu realisieren. Die erstellte Applikation ermöglicht die Eingabe eines Projektkontextes und liefert abhängig davon eine Liste von passenden Gate-Netzwerken. Die Gate-Netzwerke können als Grafik exportiert werden.

Ein Manko der Applikation ist die automatische grafische Anordnung der Gate-Netzwerke. Um die Gate-Netzwerke übersichtlich anzuordnen, ist die Interaktion des Benutzers erforderlich.

Der optionale Teil dieser Arbeit, das Exportieren der modellierten Gate-Netzwerke in das SOA-Me-System, wurde nicht bearbeitet.

6.2. Ausblick

Durch das Benutzen des Konzeptes ist aufgefallen, dass eine weitere Netzwerk-Operation Redundanzen aus der Regel-Datei eliminieren könnte. Diese Netzwerk-Operation müsste das Einfügen von Sub-Netzen in die Gate-Netzwerke ermöglichen. Dafür müsste zusätzlich eine Möglichkeit geschaffen werden, diese Sub-Netze auch zu definieren.

Eine zusätzliche mögliche Erweiterung betrifft die Projektmerkmale des Konzeptes. Der Merkmalstyp “Range” unterstützt zur Zeit nur numerische Werte. Es wären aber auch Einträge wie “Gut - Normal - Schlecht - Sehr Schlecht” denkbar. Also Werte einer Ordinalskala, die in Relation zueinander verglichen werden können.

Die während dieser Arbeit entstandene Heuristik könnte in späteren Arbeiten optimiert werden. Aus folgenden Gründen stellt sich aber die Frage, ob eine Optimierung wirklich nötig ist. Die Heuristik erzeugt möglicherweise Basis-Netzwerke mit Quality Gates, welche sinngemäß nicht zusammen gehören. Dies erfordert wiederum Anpassungsarbeit des Tailoring-Erstellers bzw. des Tailoring-Veränderers. Die sinngemäße Trennung ist möglicherweise nicht automatisierbar. Desweiteren wird die Größe der Gate-Netzwerke nicht übermäßig ansteigen, da mehr Quality Gates auch mehr Arbeitsaufwand benötigen. Eine Person mit einem gewissen Maß an Erfahrung ist höchstwahrscheinlich der Heuristik überlegen.

Die automatische übersichtliche grafische Anordnung (Layout) der Gate-Netzwerke könnte in späteren Arbeiten implementiert werden. Dafür müsste ein Algorithmus erstellt werden, der die Quality Gates und deren Übergänge so anordnet, dass möglichst wenig Überschneidungen entstehen. Eine mögliche Erweiterung wäre auch die Speicherung des Layoutes zu jedem Gate-Netzwerk in einer externen Datei.

Die Regel-Basis des V-Modell XT besitzt über 800 Textzeilen. Solch große Regel-Dateien werden unübersichtlich und sind mit einem Text-Editor schlecht zu bearbeiten. Eine Erweiterung dieser Arbeit könnte also ein Editor für die Regel-Basis sein.

A. Anhang

XML-Regel-Basis-Datei vom V-Modell XT: (komplette Regel-Basis ist auf CD)

```
<?xml version="1.0" encoding="UTF−8"?>
<!DOCTYPE qganetta SYSTEM "GaNetTA.dtd">
<qganetta version="VModell−XT␣0.9">
<propertydefinition neutralvalue="−">
 <property name="Projektgegenstand" type="selection">
  <attribute name="−"/>
  <attribute name="HW−System"/>
  <attribute name="Eingebettetes␣System"/>
  <attribute name="Komplexes␣System"/>
  <attribute name="SW−System"/>
  <attribute name="Systemintegration"/>
 </property>
 <property name="Projektrolle" type="selection">
  <attribute name="−"/>
  <attribute name="AG␣mit␣einem␣AN"/>
  <attribute name="AG␣mit␣mehreren␣AN"/>
  <attribute name="AN␣ohne␣Unterauftragnehmer"/>
  <attribute name="AN␣mit␣Unterauftragnehmer"/>
  <attribute name="AG/AN␣ohne␣Unterauftragnehmer"/>
  <attribute name="AG/AN␣mit␣Unterauftragnehmer"/>
 </property>
 <property name="Systemlebenszyklusausschnitt" type="selection">
  <attribute name="−"/>
  <attribute name="Entwicklung"/>
  <attribute name="Wartung␣und␣Pflege"/>
  <attribute name="Weiterentwicklung␣und␣Migration"/>
 </property>
 <property name="Fertigprodukte" type="selection">
  <attribute name="Ja"/>
  <attribute name="Nein"/>
  <attribute name="−"/>
 </property>
 <property name="Hohe␣Realisierungsrisiken" type="selection">
  <attribute name="Ja"/>
  <attribute name="Nein"/>
  <attribute name="−"/>
 </property>
</propertydefinition>

<gatedefinition>
 <gatelist startnodename="Start" endnodename="Ende">
    <gate name="Projekt␣genehmigt"/>
    <gate name="Projekt␣definiert"/>
    <gate name="Abnahme␣erfolgt"/>
    <gate name="Projekt␣abgeschlossen"/>
    <gate name="Anforderungen␣festgelegt"/>
```

```
    <gate name="Projekt␣ausgeschrieben"/>
    <gate name="Projekt␣beauftragt"/>
    <gate name="Iteration␣geplant"/>
    <gate name="Projektfortschritt␣ueberprueft"/>
    <gate name="Gesamtprojekt␣aufgeteilt"/>
    <gate name="Gesamtprojektfortschritt␣ueberprueft"/>
  </gatelist>
  <criteria>
   <default>
    <gate name="Projekt␣genehmigt">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Projekt␣definiert">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Abnahme␣erfolgt">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Projekt␣abgeschlossen">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Anforderungen␣festgelegt">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Projekt␣ausgeschrieben">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Projekt␣beauftragt">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Iteration␣geplant">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Projektfortschritt␣ueberprueft">
     <individuality catalog_sel="true"/>
     <creationtime planning="true" />
     <systematic systematic="true"/>
    </gate>
    <gate name="Gesamtprojekt␣aufgeteilt">
     <individuality catalog_sel="true"/>
```

```
      <creationtime planning="true" />
      <systematic systematic="true"/>
     </gate>
     <gate name="Gesamtprojektfortschritt␣ueberprueft">
      <individuality catalog_sel="true"/>
      <creationtime planning="true" />
      <systematic systematic="true"/>
     </gate>
    </default>
  </criteria>
  <review>
   <default>
    <gate name="Projekt␣genehmigt">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Projekt␣definiert">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Abnahme␣erfolgt">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Projekt␣abgeschlossen">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Anforderungen␣festgelegt">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Projekt␣ausgeschrieben">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Projekt␣beauftragt">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Iteration␣geplant">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Projektfortschritt␣ueberprueft">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Gesamtprojekt␣aufgeteilt">
     <techniques formalinspection="true"/>
    </gate>
    <gate name="Gesamtprojektfortschritt␣ueberprueft">
     <techniques formalinspection="true"/>
    </gate>
   </default>
  </review>
</gatedefinition>

<networkdefinition>
 <basenetwork name="Systementwicklung␣AG" >
  <addaftergate posgate="Start" inputgate="Projekt␣genehmigt"/>
  <addaftergate posgate="Projekt␣genehmigt" inputgate="Projekt␣definiert"/>
  <addaftergate posgate="Projekt␣definiert" inputgate="Abnahme␣erfolgt"/>
  <addaftergate posgate="Abnahme␣erfolgt" inputgate="Projekt␣abgeschlossen"/>
  <addaftergate posgate="Projekt␣definiert" inputgate="Anforderungen␣festgelegt"/>
```

```
  <addaftergate posgate="Anforderungen␣festgelegt" inputgate="Projekt␣ausgeschrieben"/>
  <addaftergate posgate="Projekt␣ausgeschrieben" inputgate="Projekt␣beauftragt"/>
  <addaftergate posgate="Projekt␣beauftragt" inputgate="Iteration␣geplant"/>
  <addaftergate posgate="Iteration␣geplant" inputgate="Projektfortschritt␣ueberprueft"/>

  <insertloop fromgate="Abnahme␣erfolgt" togate="Anforderungen␣festgelegt"/>
  <insertloop fromgate="Abnahme␣erfolgt" togate="Projekt␣beauftragt"/>
  <insertloop fromgate="Abnahme␣erfolgt" togate="Iteration␣geplant"/>
  <insertloop fromgate="Projektfortschritt␣ueberprueft" togate="Projektfortschritt␣ueberprueft"/>
 </basenetwork>
 <gatenetwork displayname="Systementwicklungsprojekt␣(AG)␣mit␣einem␣AN" basedon="Systementwicklung
       ␣AG">
  <condition>
   <and>
    <or>
     <variable propertyname="Projektgegenstand" operator="=" attribute="Systemintegration" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="HW−System" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="Eingebettetes␣System" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="Komplexes␣System" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="SW−System" />
    </or>
    <variable propertyname="Projektrolle" operator="=" attribute="AG␣mit␣einem␣AN" />
   </and>
  </condition>
 </gatenetwork>
 <gatenetwork displayname="Systementwicklungsprojekt␣(AG)␣mit␣mehreren␣AN" basedon="
       Systementwicklung␣AG">
  <condition>
   <and>
    <or>
     <variable propertyname="Projektgegenstand" operator="=" attribute="Systemintegration" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="HW−System" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="Eingebettetes␣System" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="Komplexes␣System" />
     <variable propertyname="Projektgegenstand" operator="=" attribute="SW−System" />
    </or>
    <variable propertyname="Projektrolle" operator="=" attribute="AG␣mit␣mehreren␣AN" />
   </and>
  </condition>
  <extension>
   <addaftergate inputgate="Gesamtprojekt␣aufgeteilt" posgate="Projekt␣definiert"/>
   <addbetweengates inputgate="Gesamtprojektfortschritt␣ueberprueft" fromgate="Gesamtprojekt␣aufgeteilt"
         togate="Projekt␣abgeschlossen"/>

   <insertloop fromgate="Gesamtprojektfortschritt␣ueberprueft" togate="Gesamtprojektfortschritt␣ueberprueft"/
         >
   <insertloop fromgate="Gesamtprojektfortschritt␣ueberprueft" togate="Gesamtprojekt␣aufgeteilt"/>
   <insertloop fromgate="Abnahme␣erfolgt" togate="Gesamtprojekt␣aufgeteilt"/>
  </extension>
 </gatenetwork>
</networkdefinition>
</qganetta>
```

Dokumenttypdefinition der XML-Regel-Basis-Datei:

```
<?xml version="1.0" encoding="UTF−8"?>
<!ELEMENT qganetta ((propertydefinition, gatedefinition, networkdefinition))>
<!ATTLIST qganetta
 version CDATA #REQUIRED
>
<!ELEMENT propertydefinition ((property+))>
<!ATTLIST propertydefinition
 neutralvalue CDATA #REQUIRED
>
<!ELEMENT property ((attribute∗))>
<!ATTLIST property
 type (selection | range) #REQUIRED
 name CDATA #REQUIRED
 maxvalue CDATA #IMPLIED
 minvalue CDATA #IMPLIED
 step CDATA #IMPLIED
>
<!ELEMENT attribute EMPTY>
<!ATTLIST attribute
 name CDATA #REQUIRED
>
<!ELEMENT gatedefinition ((gatelist, criteria, review))>
<!ELEMENT gatelist ((gate+))>
<!ATTLIST gatelist
 endnodename CDATA #REQUIRED
 startnodename CDATA #REQUIRED
>
<!ELEMENT criteria (default, alternative∗)>
<!ELEMENT review (default, alternative∗)>
<!ELEMENT default ((gate+))>
<!ELEMENT alternative ((condition, gate+))>
<!ELEMENT gate ((techniques | (individuality, creationtime, systematic))?)>
<!ATTLIST gate
 name CDATA #REQUIRED
>
<!ELEMENT techniques EMPTY>
<!ATTLIST techniques
 formalinspection (true | false) #IMPLIED
 peerreview (true | false) #IMPLIED
 walkthrough (true | false) #IMPLIED
>
<!ELEMENT individuality EMPTY>
<!ATTLIST individuality
 catalog_app (true | false) #IMPLIED
 catalog_sel (true | false) #IMPLIED
 catalog_sel_app (true | false) #IMPLIED
 defined_cat (true | false) #IMPLIED
 without_cat (true | false) #IMPLIED
>
<!ELEMENT creationtime EMPTY>
<!ATTLIST creationtime
 planning (true | false) #IMPLIED
 realisation_before (true | false) #IMPLIED
 realisation_in (true | false) #IMPLIED
 start (true | false) #IMPLIED
```

```
>
<!ELEMENT systematic EMPTY>
<!ATTLIST systematic
 intuitive (true | false) #IMPLIED
 systematic (true | false) #IMPLIED
 unstructured (true | false) #IMPLIED
>

<!ELEMENT networkdefinition ((basenetwork+, gatenetwork+))>
<!ELEMENT basenetwork ANY><!--ANY weil es zu viele Kombinatioen von Operationen gibt-->
<!ATTLIST basenetwork
 basedon CDATA #IMPLIED
 name CDATA #REQUIRED
>
<!ELEMENT gatenetwork ((condition, extension?))>
<!ATTLIST gatenetwork
 basedon CDATA #REQUIRED
 displayname CDATA #REQUIRED
>
<!ELEMENT extension ANY><!--ANY weil es zu viele Kombinatioen von Operationen gibt-->

<!ELEMENT condition (variable | and | not | or )>
<!ELEMENT and ANY><!--ANY weil es zu viele Kombinatioen gibt-->
<!ELEMENT or ANY><!--ANY weil es zu viele Kombinatioen gibt-->
<!ELEMENT not (variable | and | not | or)>
<!ELEMENT variable EMPTY>
<!ATTLIST variable
 attribute CDATA #REQUIRED
 operator CDATA #REQUIRED
 propertyname CDATA #REQUIRED
>

<!ELEMENT splitgateparallel EMPTY>
<!ATTLIST splitgateparallel
 sourcegate CDATA #REQUIRED
 firstgate CDATA #REQUIRED
 secondgate CDATA #REQUIRED
>
<!ELEMENT splitgatelinear EMPTY>
<!ATTLIST splitgatelinear
 sourcegate CDATA #REQUIRED
 firstgate CDATA #REQUIRED
 secondgate CDATA #REQUIRED
>
<!ELEMENT joingates EMPTY>
<!ATTLIST joingates
 targetgate CDATA #REQUIRED
 firstgate CDATA #REQUIRED
 secondgate CDATA #REQUIRED
>
<!ELEMENT addbetweengates EMPTY>
<!ATTLIST addbetweengates
 fromgate CDATA #REQUIRED
 inputgate CDATA #REQUIRED
 togate CDATA #REQUIRED
>
```

```
<!ELEMENT addaftergate EMPTY>
<!ATTLIST addaftergate
 inputgate CDATA #REQUIRED
 posgate CDATA #REQUIRED
>
<!ELEMENT addbeforegate EMPTY>
<!ATTLIST addbeforegate
 inputgate CDATA #REQUIRED
 posgate CDATA #REQUIRED
>
<!ELEMENT insertloop EMPTY>
<!ATTLIST insertloop
 fromgate CDATA #REQUIRED
 togate CDATA #REQUIRED
>
<!ELEMENT deleteloop EMPTY>
<!ATTLIST deleteloop
 fromgate CDATA #REQUIRED
 togate CDATA #REQUIRED
>
<!ELEMENT replacegate EMPTY>
<!ATTLIST replacegate
 oldgate CDATA #REQUIRED
 newgate CDATA #REQUIRED
>
<!ELEMENT removegate EMPTY>
<!ATTLIST removegate
 gate CDATA #REQUIRED
>
```

Literaturverzeichnis

[1] Christian Peucker,
Ein Anpassungskonzept basierend auf Bausteinen für Quality Gates,
Gottfried Wilhelm Leibniz Universität Hannover, Juni 2007

[2] Jens Greive,
Entwurf und Implementierung eines Systems zur netzbasierten Durchführung von Quality-Gates,
Gottfried Wilhelm Leibniz Universität Hannover, September 2005

[3] Daniel Scholz,
Entwurf und Implementierung von NetQGate mit J2EE,
Gottfried Wilhelm Leibniz Universität Hannover, November 2006

[4] Prof. Dr. Kurt Schneider,
Folien zur Vorlesung Grundlagen der Softwaretechnik,
Gottfried Wilhelm Leibniz Universität Hannover, WS 05/06

[5] Robert G. Cooper,
Winning at New Products: Accelerating the Process from Idea to Launch,
Basic Books, 2001

[6] Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung,
V-Modell XT Modelldokumentation,
http://www.v-modell-xt.de/, 2004

[7] Graph Visualization,
http://www.graphviz.org/, September 2007

[8] Java Graph Visualization,
http://www.jgraph.com/jgraph.html, September 2007

[9] V-Modell,
http://de.wikipedia.org/wiki/V-Modell, September 2007

[10] Goal Question Metric,
http://de.wikipedia.org/wiki/GQM, September 2007

[11] Dokumenttypdefinition,
http://de.wikipedia.org/wiki/Dokumenttypdefinition, September 2007

Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 10. Oktober 2007

_____

Christian Schulz