

**Prüfer:** Prof. Dr. rer. nat. Jochen Ludewig  
**Betreuerin:** Dipl. Inform. Patricia Mandl-Striegnitz

**Beginn am:** 08.05.2000  
**Beendet am:** 07.11.2000  
**CR-Klassifikation:** D.2.8, D.2.9, H.5.2, I.6.6

Studienarbeit Nr. 1780

## **Konzeption und Realisierung eines Auswertungswerkzeugs für SESAM-2**

Tilman Hampp

Institut für Informatik  
Universität Stuttgart  
Breitwiesenstr. 20 - 22  
D-70565 Stuttgart

## **Zusammenfassung**

Um praktische Erfahrung in der Durchführung von Software-Projekten zu vermitteln, verfolgt die Abteilung Software Engineering an der Universität Stuttgart den Ansatz, Projekte am Rechner zu simulieren. Das SESAM-2-System ist ein Projektsimulator, der alle wesentlichen Aspekte eines Projekts nachbildet. Einzige Ausnahme ist der Projektleiter, der das Projekt als Spieler am Simulator durchführt.

Damit bei Spielern ein Lerneffekt durch die Simulation erzielt wird, werden Projektsimulationen in ein Schulungskonzept eingebunden. Wichtiger Bestandteil einer Schulung ist eine Analyserunde. In dieser Analyserunde stellt der Tutor der Schulung das Vorgehen und die erzielten Ergebnisse der Projektsimulationen vor und diskutiert diese mit den Teilnehmern. Für diese Analyserunde wertet der Tutor die von den Teilnehmern als Spieler durchgeführten Projekte aus.

Ziel dieser Studienarbeit ist die Konzeption und Realisierung eines Auswertungswerkzeugs, das den Tutor bei der Analyse der Spielverläufe unterstützt. Am Simulator durchgeführte Projekte kann der Tutor mit dem Werkzeug anhand von Protokolldateien auswerten. Die Ergebnisse lassen sich anschaulich als Tabellen, Linien- oder Balkendiagramme oder Ganttcharts darstellen.

## Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>1</b>
1.1	Motivation	1
1.2	Aufgabenstellung	1
1.3	Übersicht über die weitere Arbeit	2
<b>2</b>	<b>Das SESAM-System</b>	<b>3</b>
2.1	Simulation von Softwareprojekten	3
2.2	Schulungen mit SESAM-2	5
2.3	Analyse der Spielverläufe	6
<b>3</b>	<b>Anforderungen an das Auswertungswerkzeug</b>	<b>8</b>
3.1	Eigenschaften und Vorgehen	8
3.2	Überblick über die Spezifikation	9
3.3	Funktionalität des Auswertungswerkzeugs	10
3.4	Anforderungen an die Qualität	37
<b>4</b>	<b>Architektur des Auswertungswerkzeugs</b>	<b>39</b>
4.1	Eigenschaften	39
4.2	Vorgehen	40
4.3	Ergebnis des Entwurfs	40
<b>5</b>	<b>Implementierung</b>	<b>47</b>
5.1	Vorgehen	47
5.2	Ergebnis der Implementierung	47
5.3	Erweiterungen	48
5.4	Metriken	49
<b>6</b>	<b>Bewertung und Rückblick</b>	<b>51</b>
6.1	Projektverlauf	51
6.2	Ausblick	53
<b>7</b>	<b>Begriffslexikon</b>	<b>54</b>
<b>8</b>	<b>Literaturverzeichnis</b>	<b>62</b>

## Abbildungsverzeichnis

Abbildung 1	Überblick über das SESAM-System (aus Reißing, 1996)	4
Abbildung 2	Nachrichten und Kommandos	11
Abbildung 3	Hauptfenster	13
Abbildung 4	Auszug aus einer Protokolldatei	14
Abbildung 5	Skizze eines Syntaxbaums	15
Abbildung 6	Eingabefenster für Bedingungen	16
Abbildung 7	Bedingungen und Syntaxbaum	17
Abbildung 8	Auswahl von Namen	18
Abbildung 9	Auswahl von Aliasnamen	19
Abbildung 10	Wahl von Datum und Parametern	20
Abbildung 11	Eingabe von Projektionen	20
Abbildung 12	Skizze für die Projektion	21
Abbildung 13	Beispiel für eine Ergebnistabelle	21
Abbildung 14	Eingabe von Optionen und Operationen	22
Abbildung 15	Beispiel für eine Gruppierung	23
Abbildung 16	Layout einer Tabelle	24
Abbildung 17	Tabellendiagramm	26
Abbildung 18	Aufwand für die Phasen	27
Abbildung 19	Beispiel der Nachrichten für ein Ganttchart	28
Abbildung 20	Bedingungen für ein Gantt-schema	30
Abbildung 21	Nachrichten für den Beginn einer Tätigkeit	31
Abbildung 22	Projektion für ein Gantt-diagramm	32
Abbildung 23	Beispiel für eine Ergebnistabelle	33
Abbildung 24	Layout für ein Gantt-diagramm	34
Abbildung 25	Gantt-diagramm Spezifikation und Entwurf	35
Abbildung 26	Gantt-diagramm der Mitarbeiter	36
Abbildung 27	Grobentwurf	43
Abbildung 28	Struktur der Daten	44
Abbildung 29	Zugriff auf die Daten	45
Abbildung 30	Auswertung der Syntaxbäume	46
Abbildung 31	Gesamtgröße in Zeilen Code	49
Abbildung 32	Größe der Module	50
Abbildung 33	Größe von Funktionen und Prozeduren	50
Abbildung 34	Meilensteine	51
Abbildung 35	Projektplan	52

# 1 Einführung

## 1.1 Motivation

Mit dem SESAM-Projekt (Ludewig, 1994) wird das Ziel verfolgt, die Ausbildung von Projektleitern um eine Simulationskomponente zu ergänzen. Der Simulator SESAM-2 bildet am Rechner ein vollständiges Softwareprojekt nach. Einzige Ausnahme ist der Projektleiter, der das Projekt am Rechner als Spiel durchführt.

In einer Schulung wird dieses Spiel durch eine Analyserunde ergänzt, in der die Ergebnisse der simulierten Projekte und das Vorgehen der Spieler von einem Tutor vorgestellt und mit den Spielern diskutiert werden (Mandl-Striegnitz, 2000a). Für diese Rückmeldungen des Tutors an die Spieler analysiert der Tutor den Verlauf der simulierten Projekte und die erzielten Resultate.

Um Daten über die im simulierten Projekt erzielten Resultate zu bekommen, beispielsweise Metriken der im Projekt entstandenen Dokumente, existiert bereits das Werkzeug Sesanalyzer. Informationen über den Spielverlauf, beispielsweise welcher Entwickler welches Dokument wann angefertigt hat, müssen von Hand aus einer Protokolldatei erhoben werden.

## 1.2 Aufgabenstellung

In dieser Studienarbeit sollte ein Auswertungswerkzeug konzipiert und realisiert werden, das den Tutor bei der Auswertung dieser Protokolldatei unterstützt. Der Tutor kann mit diesem Werkzeug Protokolldateien einlesen und dann über eine graphische Benutzungsoberfläche Anfragen an die eingelesenen Protokolldateien stellen. Das Werkzeug kann die Ergebnisse dieser Anfragen als Diagramm in Form von Tabellen, Balken- oder Liniendiagrammen oder Ganttcharts anschaulich darstellen. Die Art der Darstellung ist vom Benutzer wählbar. Die Ergebnisse sollen als Diagramme gedruckt werden oder zur Weiterverarbeitung in anderen Programme wie beispielsweise MS Excel gespeichert werden können. Eine Anfrage und die zugehörige Art der Darstellung bilden ein Schema, mit dem Protokolldateien ausgewertet werden können. Schemata sollen in Dateien gespeichert werden können, so dass der Tutor sie wieder in das Werkzeug laden kann.

Das Werkzeug trägt den Namen Sesamscore, score steht für Auswertung oder Spielstand.

Der Ablauf der Studienarbeit orientiert sich am Standard-Phasenmodell. Erste Tätigkeit ist die Erstellung eines Projektplans. Im Projektplan sind die einzelnen Arbeitspakete beschrieben. Meilenstein und Termin des Meilensteins sind im Projektplan festgelegt. Jeder Meilenstein ist durch eine Prüfung durch die Betreuerin mit anschließender Korrektur der Dokumente gekennzeichnet.

Die Arbeitspakete nach der Erstellung des Projektplans sind Spezifikation des Werkzeugs, Entwurf und Implementierung.

In einem Vortrag nach der Implementierungsphase wurde das Werkzeug und die bisher in der Studienarbeit erzielten Resultate im Rahmen eines Abteilungskolloquiums vorgestellt. Letzte Phase ist die Erstellung dieses Berichts.

Neben dem lauffähigen Werkzeug wird in der Studienarbeit besonderen Wert auf die Dokumente der frühen Phasen gelegt, also Spezifikation und Entwurf. Da das Werkzeug in das SESAM-Projekt eingebunden ist, ist die Wartbarkeit eine wichtige Qualitätsanforderung. Sie wird durch die Qualität der Dokumente der frühen Phasen unterstützt.

### **1.3 Übersicht über die weitere Arbeit**

In diesem Bericht werden das Vorgehen und die Resultate der Studienarbeit beschrieben. In Kapitel 2 wird ein kurzer Überblick über das SESAM-Projekt gegeben, da die Studienarbeit in dieses Projekt eingebunden ist. Die Grundlagen, die sich daraus für das Werkzeug ergeben, werden ausführlicher dargestellt.

Kapitel 3 beschreibt das Vorgehen während der Spezifikation und die Anforderungen an das Werkzeug. In Kapitel 4 folgt die Architektur des Werkzeugs und in Kapitel 5 ein Überblick über die Implementierung.

Das Vorgehen und der Projektverlauf wird in Kapitel 6 bewertet, Kapitel 7 enthält das Begriffslexikon. Das Literaturverzeichnis bildet Kapitel 8 und damit den Schluss dieses Berichts.

Spezifikation, Entwurf und Code des Werkzeugs sind separate Dokumente und nicht in diesem Bericht enthalten. Diese Dokumente werden im Projektordner bei der Betreuerin aufbewahrt.

## 2 Das SESAM-System

Die Qualifikation und Erfahrung, die für die Durchführung eines Softwareprojekts nötig sind, lassen sich nur schwer theoretisch vermitteln (Mandl-Striegnitz, 2000a). Die Bedeutung der Theorie für erfolgreiches Projektmanagement wird nur durch eigene Erfahrung klar, zusätzlich wird der Projektleiter durch theoretisches Wissen nur wenig auf Probleme und kritische Situationen bei der Durchführung eines Projekts vorbereitet. Erfahrung durch "Learning by doing" zu gewinnen, erweist sich jedoch als riskant und teuer.

Darum wird in der Abteilung Software Engineering an der Fakultät Informatik der Universität Stuttgart ein Ansatz verfolgt, die Ausbildung von Projektleitern durch die Simulation von Projekten am Rechner zu erweitern. Ähnlich wie zur Ausbildung von Piloten ein Flugsimulator eingesetzt wird, können mit diesem Simulator kritische Situationen und typische Probleme nachgebildet werden, die bei der Durchführung von Projekten auftreten. Der Projektleiter hat damit die Möglichkeit, Erfahrung bei der Leitung eines Projekts zu sammeln, ohne dass ein reales Projekt betroffen ist. Er soll mit dem Simulator lernen, welche Fehler dabei möglich sind und wie sie vermieden werden können. Er soll die Effekte erkennen, die in einem Projekt zusammenspielen können und die letztlich den Erfolg oder Misserfolg eines Projektes bestimmen.

### 2.1 Simulation von Softwareprojekten

Das SESAM-2-System ist ein Werkzeug zur Simulation von Softwareprojekten. Dabei werden alle Aspekte eines Softwareprojekts mit Ausnahme des Projektleiters, der die Simulation am Rechner durchführt, nachgebildet. Die Simulation basiert auf Modellen von Softwareprojekten. Ein Modell umfasst beliebige Objekte, Beziehungen und Eigenschaften des Projekts. Ein Objekt eines Simulationsmodells kann beispielsweise ein Mitarbeiter oder ein Dokument sein. Eine Beziehung wird zum Beispiel zwischen Mitarbeiter und Dokument gebildet, wenn der Mitarbeiter als Autor für dieses Dokument eingesetzt wird. Eigenschaften werden durch Attribute der Objekte oder Beziehungen beschrieben. Beispiele für solche Attribute sind etwa Kosten eines Mitarbeiters pro Tag, Größe eines Dokuments oder die Intensität, mit der ein Mitarbeiter an einem Dokument arbeitet.

Nach Schneider (1994) lässt sich das Simulationsmodell in drei Teilmodelle untergliedern:

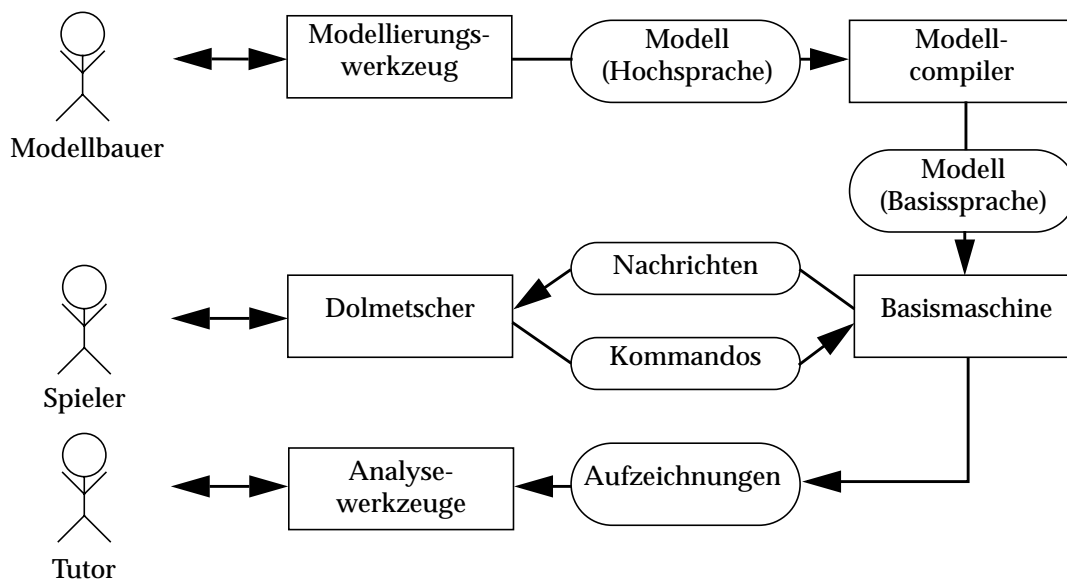
- Das Schemamodell definiert Typen für Objekte, Beziehungen und ihre Attribute. Diese werden bei der Ausführung zu Objekten und Beziehungen instanziiert und beschreiben damit die abstrakte Welt, in der das Projekt abläuft.
- Der Zustand des Projekts wird durch das Situationsmodell repräsentiert. Das Situationsmodell kann als eine Instanzierung des Schemamodells aufgefasst werden und bildet als Graph mit den Objekten als Knoten und den Beziehungen als Kanten die interne Datenstruktur des Simulators.

Ein besonderes Situationsmodell ist die Startsituation, die den Anfangszustand des Projekts festlegt.

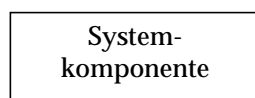
- Das Regelmodell beschreibt das dynamische Verhalten des Projekts und verändert während der Simulation den durch den Graph repräsentierten Zustand. Eine Regel besteht aus einem Bedingungsteil und einem Aktionsteil. Der Bedingungsteil beschreibt einen bestimmten Zustand des Situationsmodells als Teilgraph des Modells. Sind zusätzlich mögliche Bedingungen für Attribute erfüllt, wird der Aktionsteil der Regel ausgeführt, der den Graph des aktuellen Zustands verändert.

Wie in Abbildung 1 dargestellt, wird ein Modellierungswerkzeug benutzt, um ein Simulationsmodell zu erstellen. Der Modellbauer benutzt das Werkzeug, um ein Modell in der Hochsprache zu entwickeln (Schneider, 1996). Diese Sprache bietet mehr Komfort als die Basissprache und besitzt daher viele Sprachkonstrukte. Das Modell in der Hochsprache wird dann durch einen Modellcompiler in die Basissprache (Reißing, 1996) übersetzt. Die Basissprache wird zur Modellausführung, also zur eigentlichen Projektsimulation eingesetzt.

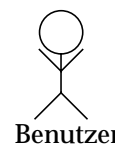
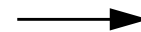
Den Beispielen in den folgenden Abschnitten liegt das QS-Modell zugrunde (Drappa, 2000), das vor allem Effekte im Zusammenhang mit Maßnahmen zur Qualitätssicherung simuliert.



Legende:



Datenfluss



**Abbildung 1: Überblick über das SESAM-System (aus Reißing, 1996)**

Ein Spieler in der Rolle des Projektleiters führt das Projekt durch, in dem er Kommandos über die Oberfläche von SESAM-2 eingibt. Das System gibt Rückmeldungen in Form von Nachrichten an den Spieler zurück. Beispielsweise kann der Spieler im QS-Modell einen Entwickler anweisen, die Spezifikation zu erstellen. Das SESAM-System gibt zurück, dass der Entwickler mit dieser Tätigkeit begonnen hat und nach einiger Zeit, dass der Entwickler fertig mit der Spezifikation ist. Die Kommandos und Nachrichten werden im Schemamodell vom Modellbauer definiert.

Um die Eingabe komfortabel zu gestalten, ist es möglich, Anweisungen an SESAM in beschränkt natürlichsprachlicher Form einzugeben. Diese werden dann über einen Dolmetscher in die Sprache der Basismaschine (siehe Abbildung 1) umgesetzt. So kann der Benutzer zum Beispiel 'Let Richard specify' eingeben, die Übersetzung für die Basismaschine ist dann in diesem Fall das Kommando `Lasse_spezifizieren (Richard)`.

Nachrichten des Systems werden ebenfalls durch den Dolmetscher übersetzt. Dazu werden Nachrichten der Basismaschine in natürliche Sprache umgewandelt. Um das Spiel interessanter zu gestalten, gibt es verschiedene Rückmeldungen des Systems mit gleicher Semantik. Diese werden von der Basismaschine ausgewählt und der Dolmetscher gibt den entsprechenden Text aus. Eine Nachricht als Antwort auf das Kommando `Lasse_spezifizieren (Richard)` kann beispielsweise die Meldung sein, dass Richard mit der Spezifikation angefangen hat. Diese Meldung wird vom System als Nachricht `Begonnen_zu_spezifizieren_1(Richard)` zurückgegeben, der Suffix "1" bestimmt den Text, der an den Spieler ausgegeben wird, beispielsweise dass sich Richard freut, an der Spezifikation zu arbeiten.

Damit ein am Simulator durchgeführtes Projekt nach Beenden der Simulation genauer untersucht werden kann, können zwei Protokolldateien während der Simulation vom System mitgeführt werden. Möchte der Tutor einer Schulung die am Simulator durchgeführten Projekte der Teilnehmer auswerten, kann er mit Hilfe von Auswertungswerkzeugen die benötigten Informationen aus diesen Aufzeichnungen gewinnen.

## 2.2 Schulungen mit SESAM-2

In einem Experiment von Notter (1999) hat sich gezeigt, dass die Spieler Erfahrungen mit simulierten Softwareprojekten eigenständig nicht umsetzen können. Die gleichen Fehler werden bei folgenden Spielen wiederholt, Verbesserungen im Projektverlauf werden nicht alleine durch Spiele am Simulator erzielt.

Darum sind die Spiele am Simulator in ein Schulungskonzept eingebunden worden. Dieses Schulungskonzept wird in Mandl-Striegnitz (2000a) beschrieben. Das Schulungskonzept besteht aus einer Einführungsveranstaltung, einer ersten Spielrunde mit dem Simulator, einer Analyserunde, einem Projektmanagement-Seminar und einer zweiten Spielrunde. Die Spieler einer in dieses Schulungskonzept eingebundenen Simulation werden sich ihrer Schwächen und Fehler im Vorgehen bewusst. Sie können die dabei

gewonnene Erfahrung in einem folgenden Spiel am Simulator umsetzen. Dieser Lerneffekt wurde in einem Experiment nachgewiesen (Mandl-Striegnitz, 2000a).

Damit dieser Lerneffekt erzielt wird, werden in der Analyserunde Stärken und Schwächen bei der Durchführung der Spiele von Tutor aufgezeigt und mit den Spielern diskutiert. Dazu werden die Ergebnisse der Spiele gegenübergestellt und unterschiedliche Resultate aus dem jeweils gewählten Vorgehen erklärt.

### **2.3 Analyse der Spielverläufe**

Damit der Tutor während der Analyserunde den Teilnehmern einer Schulung und damit den Spielern die erforderlichen Rückmeldungen geben kann, analysiert er die Spielverläufe der ersten Spielrunde. Der Tutor wertet die erzielten Resultate des Projekts aus, also etwa Eigenschaften der Spezifikation wie im Dokument enthaltenen Fehler. Durch Auswertung des Vorgehens können die unterschiedlichen Resultate der verschiedenen Spiele erklärt werden.

Im Rahmen eines Studienprojekts entstand bereits das Werkzeug Sesamalyzer (Abteilung SE, 1999a). Mit diesem Werkzeug kann der Zustand eines simulierten Projekts untersucht werden. Dazu werden vom Werkzeug Protokolldateien von Spielen geparkt, in der die Startsituation des Modells und alle Änderungen am Situationsmodell während des Spiels enthalten sind. Mit dem Sesamalyzer lassen sich dann Werte des Situationsmodells zu jedem beliebigen Zeitpunkt und über einen beliebigen Zeitraum graphisch darstellen. Beispielsweise kann der Verlauf der in der Spezifikation enthaltenen Fehler über den Erstellungszeitraum angezeigt werden. Zusätzlich können Berechnungen mit diesen Werten durchgeführt werden. Die Ergebnisse der Berechnungen werden ebenfalls graphisch dargestellt.

Um aber das Projekt aus der Sicht des Vorgehens des Spielers zu untersuchen, möchte der Tutor wissen, wodurch diese Änderungen an der Situation verursacht wurden. Was hat der Spieler unternommen? Wie ist er dabei vorgegangen? Welche Mitarbeiter wurden für welche Tätigkeiten eingesetzt? Mit Hilfe dieser Informationen können die erzielten Resultate in der Analyserunde erklärt werden.

Diese Informationen zum Vorgehen und die interessanten Ergebnisse können aus den Kommandos, das heißt den Aktionen des Spielers und den Nachrichten, also den Reaktionen des Systems auf die Aktionen des Spielers, abgeleitet werden. Die Kommandos des Spielers und die Nachrichten mit den dazugehörigen Parametern sowie die jeweils aktuelle Projektzeit werden in einer zweiten Protokolldatei mitgeführt.

Der Tutor kann in diese Protokolldatei vom Simulator zusätzliche Informationen schreiben lassen. Dazu gibt er Kommandos ein, die dem Spieler nicht zugänglich sind. Als Antwort auf diese Kommandos werden Tutornachrichten ausgegeben, deren Parameter dann beispielsweise Werte bestimmter Attribute des Modells enthalten. Diese Tutorkommandos und -nachrichten lassen sich im Modell beliebig definieren.

Die Protokolldatei wurde bisher von einer Tutorin, die das Modell sehr gut kennt, von Hand ausgewertet. Dazu wurden Betriebssystemkommandos wie 'grep' verwendet. Um beispielsweise herauszufinden, welcher Mitarbeiter an welchem Dokument beteiligt war, muss die gesamte Datei nach Kommandos durchsucht werden, die einen Mitarbeiter an einem Dokument arbeiten lassen. Die Erstellung der für die Analyserunde nötigen Auswertungen ist so sehr aufwendig, vor allem wenn mehrere Protokolldateien ausgewertet werden müssen.

Darum wurde in dieser Studienarbeit das Werkzeug Sesamscore realisiert, mit dem die Auswertung dieser Protokolldatei weitgehend automatisiert werden kann. Es werden damit Protokolldateien geparkt und vom Tutor vorgegebene Auswertungen durchgeführt. Die Ergebnisse können dann graphisch in Form von Diagrammen dargestellt und für die Analyserunde verwendet werden.

Das Werkzeug ist eine Ergänzung zum schon vorhandenen Sesamalyzer, keines der beiden Werkzeuge könnte das andere ersetzen. Es können zwar durch Tutornachrichten beliebige Attribute aus dem Situationsmodell in die Protokolldatei geschrieben und dann mit Sesamscore ausgewertet werden, um aber einen wirklichen Verlauf der Attribute darzustellen, müssten diese Attribute zu jedem Zeitschritt in die Protokolldatei ausgegeben werden. Zusätzlich bietet der Sesamalyzer auch die Möglichkeit, beliebige Formeln zur Berechnung aus Attributwerten anzugeben und diese Ergebnisse darzustellen. Mit dem Sesamalyzer können aber qualitative Daten über den Projektverlauf nicht ausgewertet werden.

### **3 Anforderungen an das Auswertungswerkzeug**

In diesem Kapitel wird die Funktionalität des Auswertungswerkzeugs beschrieben. Da die Funktionalität in Form von funktionalen Anforderungen in der Spezifikation festgelegt ist, wird zuerst ein Überblick über dieses Dokument gegeben. Die Spezifikation beschreibt die wesentlichen Anforderungen an die Software und ihre Schnittstellen.

In diesem Kapitel werden im ersten Abschnitt die Eigenschaften, die das Dokument erfüllen soll und das Vorgehen bei der Erstellung beschrieben. In Abschnitt 3.2 wird kurz der Inhalt des Dokuments beschrieben. Abschnitt 3.3 gibt einen Überblick über die funktionalen Anforderungen und Anforderungen an die Schnittstellen und erklärt an Beispielen die Funktionen des Werkzeugs. Qualitätsanforderungen werden kurz in Abschnitt 3.4 dargestellt.

#### **3.1 Eigenschaften und Vorgehen**

Die Spezifikation ist die Grundlage für alle nachfolgenden Dokumente wie Entwurf, die Implementierung, das Benutzerhandbuch und die Spezifikation der Testfälle. Daher soll der Inhalt der Spezifikation zutreffend sein, also die Vorstellung des Kunden richtig widerspiegeln. Die Anforderungen sollen vollständig und widerspruchsfrei enthalten und unabhängig von der Realisierung beschrieben sein (Ludewig, 1998).

Für das Werkzeug Sesamscore sind die Anforderungen von der Betreuerin klar vorgegeben, da sie den Ist-Zustand aus eigener Erfahrung kennt. Diese Anforderungen sind in Mandl-Striegnitz (2000b und 2000c) beschrieben und wurden während der Spezifikationsphase zusätzlich mündlich oder als Skizze genauer beschrieben. Durch Befragung und Diskussion konnten diese dann weiter geklärt und präzisiert werden.

Während der gesamten Spezifikationsphase wurde das Begriffslexikon erstellt. Im Begriffslexikon sind alle wichtigen Objekte und Begriffe des Werkzeugs definiert.

Im Mittelpunkt der Spezifikationsphase stand die Funktionalität des Werkzeugs. Die Lösung und damit die Struktur und die Algorithmen werden erst im Entwurf behandelt.

Die Spezifikation soll die wesentlichen Anforderungen leicht verständlich und präzise darstellen, damit beispielsweise für eine Wartung klar ist, was das Werkzeug leisten soll. Im Dokument werden daher alle Anforderungen natürlichsprachlich beschrieben und teilweise zusätzlich durch Skizzen verdeutlicht. Damit können sie ohne Kenntnis einer bestimmten Notation verstanden werden.

Die Bedienung über eine graphische Benutzungsoberfläche ist ein wichtiger Aspekt des Werkzeugs. Die Oberfläche soll leicht zu bedienen sein. Da sich eine Oberfläche aber nur schwer sprachlich präzise beschreiben lässt, wurde während der Spezifikation ein Prototyp erstellt. Dieser Prototyp wurde in Tcl/Tk implementiert und ist ein Prototyp im Sinne eines "Mock-ups", also ohne eigentliche Funktionalität. Anhand des Prototyps

konnten dann die Anforderungen an die Oberfläche mit der Betreuerin weiter geklärt werden. Die Erstellung und nachfolgende Änderungen erfordern weniger Aufwand, als für Skizzen nötig wäre, die von Hand mit FrameMaker erstellt sind.

Das gesamte Dokument wurde mit FrameMaker erstellt, lässt sich also leicht nachführen und verwalten. Das Dokument umfasst 129 Seiten und ist in 12 Kapitel untergliedert.

Um zu verhindern, dass fehlerhafte Anforderungen in den Entwurf übernommen werden oder Anforderungen fehlen, wurde das Dokument nach der Erstellung von der Betreuerin geprüft und vor Beginn des Entwurfs korrigiert.

### **3.2 Überblick über die Spezifikation**

Die Gliederung der Spezifikation ist nach verschiedenen Aspekten geordnet und orientiert sich an der Spezifikation des Werkzeugs SesamIDE (Abteilung SE, 1999b).

Im Kapitel Produktüberblick ist neben einem Überblick über das SESAM-System der Ist- und Soll-Zustand vor beziehungsweise nach Einsatz des Werkzeugs festgehalten. In diesem Kapitel werden auch mögliche Innovationsfolgen durch den Einsatz des Werkzeugs aufgezeigt. Eine Beschreibung der möglichen Benutzer und Voraussetzungen für die Bedienung des Werkzeugs wurden mit der Betreuerin geklärt und in einem eigenen Kapitel festgelegt.

Konzepte zur Erstellung der Spezifikation werden in einem eigenen Kapitel erläutert. Dazu gehört auch, dass als Vorlage für den Prototyp das Benutzerhandbuch für das Werkzeug Sesamalyzer (Abteilung SE, 1999a) verwendet wurde.

Das Kapitel "Produktübersicht" enthält eine Beschreibung der Referenzumgebung und der Entwicklungsumgebung für das Werkzeug. In diesem Kapitel wird zusätzlich ein Überblick über die funktionalen Anforderungen gegeben.

Funktionale Anforderungen sind in einem eigenen Kapitel beschrieben. Die Gliederung des Kapitels folgt der Struktur der Aufgabenstellung. Für jede Anforderung sind in Anlehnung an die Use-Case-Notation (Jacobson, 1995) Vor- und Nachbedingungen, Fehlerbehandlung und Überprüfungen sowie Erweiterungen als Verweis in einem eigenen Absatz beschrieben. Um die Quantität von im Werkzeug bearbeiteten Daten zu definieren, wurde ein Mengengerüst gebildet. In diesem Mengengerüst sind für alle Daten das Minimum, Durchschnitt und größtmöglicher Wert festgehalten.

Die Schnittstelle zur Protokolldatei wird im Kapitel "Schnittstellen" beschrieben und durch einen Auszug ergänzt. Die Schnittstelle zum Benutzer soll als graphische Benutzungsoberfläche implementiert werden. In diesem Kapitel wird die Benutzungsoberfläche an Hand von Screenshots eines Prototyps beschrieben. Durch Verweise von den funktionalen Anforderungen auf die einzelnen Screenshots sind die Screenshots den entsprechenden Anforderungen zugeordnet.

Da schon in der Aufgabenstellung mögliche Erweiterungen beschrieben sind, enthält das nächste Kapitel die Anforderungen mit diesen Erweiterungen. Verweise zwischen funktionalen Anforderungen und den zugehörigen Erweiterungen erleichtern es, Zusammenhänge zwischen den Kapiteln zu erfassen.

In der Aufgabenstellung sind nicht nur Erweiterungen beschrieben, sondern auch Vorgaben an die Implementierung. So wurde die Struktur des Syntaxbaums bereits festgelegt. Während die Anforderungen geklärt wurden, sind einige Implementierungsvorgaben ebenfalls diskutiert worden. Diese Vorgaben sind in einem eigenen Kapitel zusammengefasst.

Anforderungen an die Qualität des Produkts wurden ebenfalls während der Spezifikationsphase definiert.

Den Abschluss des Dokuments bildet das Begriffslexikon und als letztes Kapitel das Literaturverzeichnis.

### **3.3 Funktionalität des Auswertungswerkzeugs**

In den nächsten Unterkapiteln werden die funktionalen Anforderungen und die Anforderungen an die Schnittstellen zu den Dateien und zum Benutzer anhand von Beispielen beschrieben.

Falls in der Spezifikation festgelegte Anforderungen in der Implementierung nicht umgesetzt wurden, sind sie in den nächsten Abschnitten trotzdem beschrieben. Eine Übersicht, welche Anforderungen nicht umgesetzt wurden, ist in diesem Bericht im Kapitel 5.3 "Erweiterungen" dargestellt.

Allen Beispielen liegt das QS-Modell zugrunde (Drappa, 2000). Die Spielverläufe in Form von Protokolldateien, die für die Beispiele und Abbildungen mit dem Werkzeug ausgewertet wurden, stammen aus einem Experiment mit SESAM (Mandl-Striegnitz, 2000a).

#### **3.3.1 Einleitung**

Damit der Tutor die Spielverläufe auswerten kann, untersucht er die Kommandos des Spielers und die Nachrichten des Simulators. Nachrichten und Kommandos haben denselben Aufbau und bestehen, wie Abbildung 2 zeigt, jeweils aus dem Namen und den Parametern der Nachricht oder des Kommandos.

```
LASSE_SPEZIFIZIEREN(Richard)
REWEWSITZUNG_FINDET_STATT_2 (Specification,30,17,0.69652)
```

The diagram illustrates the structure of a command and a message. The command `LASSE_SPEZIFIZIEREN(Richard)` has the parameter `Richard`. The message `REWEWSITZUNG_FINDET_STATT_2 (Specification,30,17,0.69652)` has parameters `Specification`, `30`, `17`, and `0.69652`. Arrows point from the label "Name" to the command name and from the label "Parameter" to the parameters of both the command and the message.

**Abbildung 2: Nachrichten und Kommandos**

Der Name und die Parameter werden durch das Modell definiert. Im QS-Modell wird beispielsweise durch das Kommando “LASSE\_SPEZIFIZIEREN” der Mitarbeiter, der als Parameter angegeben ist, zur Spezifikation eingesetzt. Veranlasst der Spieler eine Prüfung der Spezifikation durch ein Review, gibt das SESAM-System nach jeder Reviewsitzung eine Meldung über die erzielten Ergebnisse dieser Sitzung aus. Diese Meldung wird als Nachricht “REWEWSITZUNG\_FINDET\_STATT” mit unterschiedlichem Suffix an den Dolmetscher gegeben. Die Bedeutung des Suffix ist in Abschnitt 2.1 erklärt. In den Parametern steht der Prüfling, die Anzahl der geprüften Seiten, entdeckte Fehler und entdeckte Verluste in Function Points. Ein anderes Modell könnte aber auch für jedes geprüfte Dokument eine eigene Nachricht definieren, etwa “SPEZREVIEW\_FINDET\_STATT” oder andere Parameter ausgeben.

Um beispielsweise den Erfolg der Prüfung der Spezifikation zu beurteilen, kann der Tutor untersuchen, wieviele Fehler in der Spezifikation insgesamt durch ein Review entdeckt wurden. Dazu untersucht er alle Nachrichten von Reviewsitzungen des Dokuments Spezifikation, also alle Nachrichten mit dem Namen “REWEWSITZUNG\_FINDET\_STATT” und einem beliebigen Suffix, deren erster Parameter “Specification” ist. Aus diesen Nachrichten summiert er die Anzahl der gefundenen Fehler, die im dritten Parameter stehen.

Möchte der Tutor die Prüfung anderer Dokumente wie Systementwurf und Modulspezifikation ebenfalls beurteilen, sucht er nach den Nachrichten für diese Dokumente und zählt die Anzahl der gefundenen Fehler getrennt für jedes Dokument zusammen.

Untersucht der Tutor einen ganz anderen Aspekt, etwa welcher Entwickler welches Dokument geschrieben hat, kann er ähnlich vorgehen. Er untersucht dann die Kommandos des Spielers, mit denen ein Entwickler für ein Dokument eingesetzt wird und notiert sich beispielsweise das Dokument und den Entwickler.

Mit Sesamscore werden Spielverläufe durch ein ähnliches Vorgehen ausgewertet:

- Durch Bedingungen werden bestimmte Nachrichten oder Kommandos mit bestimmten Parameterwerten ausgewählt. Im Beispiel ist das die Bedingung, dass nur Nachrichten mit dem Namen “REWEWSITZUNG\_FINDET\_STATT” und beliebigem Suffix und dem ersten Parameter “Specification” betrachtet werden.

- Die für den Tutor interessantesten Daten wählt der Tutor durch eine Projektion. Im Beispiel wird der Parameter mit der Anzahl der gefundenen Fehler betrachtet.
- Die gewählten Daten werden mit Optionen und Operationen weiterbearbeitet, beispielsweise wird die Anzahl der gefundenen Fehler zusammengezählt.

Bedingungen, Projektion, Optionen und Operationen bilden im Werkzeug eine Anfrage. Zusätzlich zu der Anfrage kann der Tutor ein Layout eingeben, um zu bestimmen, wie die Ergebnisse dargestellt werden sollen. Eine Anfrage und das zugehörige Layout bilden ein Auswertungsschema. Das Auswertungsschema beschreibt, wie Spielverläufe ausgewertet werden sollen. Da alle Teile eines Schemas vom Tutor relativ frei gewählt werden können, können damit unterschiedliche Aspekte eines Projektverlaufs untersucht und dargestellt werden.

Das Schema wird vom Werkzeug benutzt, um Spielverläufe auszuwerten. Zuerst wird die Anfrage durchgeführt, dabei werden die Bedingungen und die Projektionen bearbeitet und diese Ergebnisse in einer internen Ergebnistabelle gespeichert. Auf diese werden dann Operationen und Optionen angewendet. Die Daten in der Ergebnistabelle werden dem Layout entsprechend in eine Diagrammtabelle geschrieben. Anschließend stellt das Werkzeug die Ergebnisse dar. In einer Voransicht wird zuerst die Diagrammtabelle gezeigt. Danach wird das Diagramm graphisch dargestellt.

Sesamscore unterscheidet zwei verschiedenen Schemaarten. Um beliebige Daten des Projekts darzustellen, kann der Tutor allgemeine Schemata benutzen. Deren Ergebnisse können als Tabellen, Linien- oder Balkendiagramm dargestellt werden. Um Abläufe anschaulich als Gantt-Diagramm darzustellen, benutzt der Tutor Ganttschemata. Diese beiden unterschiedlichen Arten von Schemata unterscheiden sich hauptsächlich durch Optionen und Operationen und im Layout. Unterschiede und Gemeinsamkeiten der Schemata werden in den beiden Abschnitten 3.3.5 und 3.3.6 näher erläutert.

Da Nachrichten, Kommandos und ihre Parameter vom Simulationsmodell abhängen, wird das Werkzeug durch Informationen über das Modell angepasst. Der Tutor muss Kommandos, Nachrichten und ihre Parameter kennen, um Anfragen einzugeben. Die Verbindung zum Modell unterstützt ihn dabei.

### **3.3.2 Das Hauptfenster**

Nachdem der Tutor das Werkzeug gestartet hat, wird das Hauptfenster des Werkzeugs am Bildschirm angezeigt (Abbildung 3).



**Abbildung 3: Hauptfenster**

Zuerst muss der Tutor die Nachrichten und Kommandos des Simulationsmodells einlesen lassen. Über den Button “Model” kann der Tutor eine Datei mit Informationen über das Modell auswählen. Diese Verbindung zum Modell wird im nächsten Abschnitt näher beschrieben.

Erst danach kann der Tutor über den Button “Analysis” Schemata laden, eingeben oder verändern. Über den Button “Games” kann er dann Protokolldateien mit Spielverläufen laden. Diese können mit den Schemata ausgewertet werden. Die Anforderungen zum Laden von Protokolldateien werden in Abschnitt 3.3.4 dargestellt, Auswertungen und ihre graphische Darstellung in den Abschnitten 3.3.5 “Allgemeine Auswertungsschemata” und 3.3.6 “Auswertungsschemata für Gantt diagramme”.

### 3.3.3 Verbindung zum Modell von Sesam

Damit das Werkzeug Sesamscore für beliebige Modelle eingesetzt werden kann, wird zuerst eine Verbindung zu einem Modell hergestellt. Alle Spiele, die nach der Wahl des Modells ausgewertet werden sollen, gehören zu diesem Modell. Die nötige Information über das Modell wird in einer Informationsdatei gespeichert, die der Benutzer wählt.

In dieser Datei stehen alle Namen von Nachrichten und Kommandos, die im Modell der zu analysierenden Spiele definiert sind. Alle Namen werden in einer Tabelle im Werkzeug gespeichert. Die Tabelle kann zu einer vollständigen Symboltabelle erweitert werden, indem zusätzlich zu den Namen der Kommandos und Nachrichten Typinformationen über die Parameter der Kommandos und Nachrichten gespeichert werden. Diese Erweiterung ist in der Spezifikation beschrieben und wird im Entwurf berücksichtigt.

Die Namen der Nachrichten und Kommandos werden beim Laden der Protokolldateien verwendet und zur Benutzerführung, damit der Tutor Kommandos und Nachrichten bequem über eine Auswahl eingeben kann.

Die Informationsdatei soll vom Modellbauer erstellt werden. Er kann diese Datei beispielsweise über ein einfaches Shellskript aus den Dateien, die das Modell definieren, erzeugen.

### 3.3.4 Protokolldateien

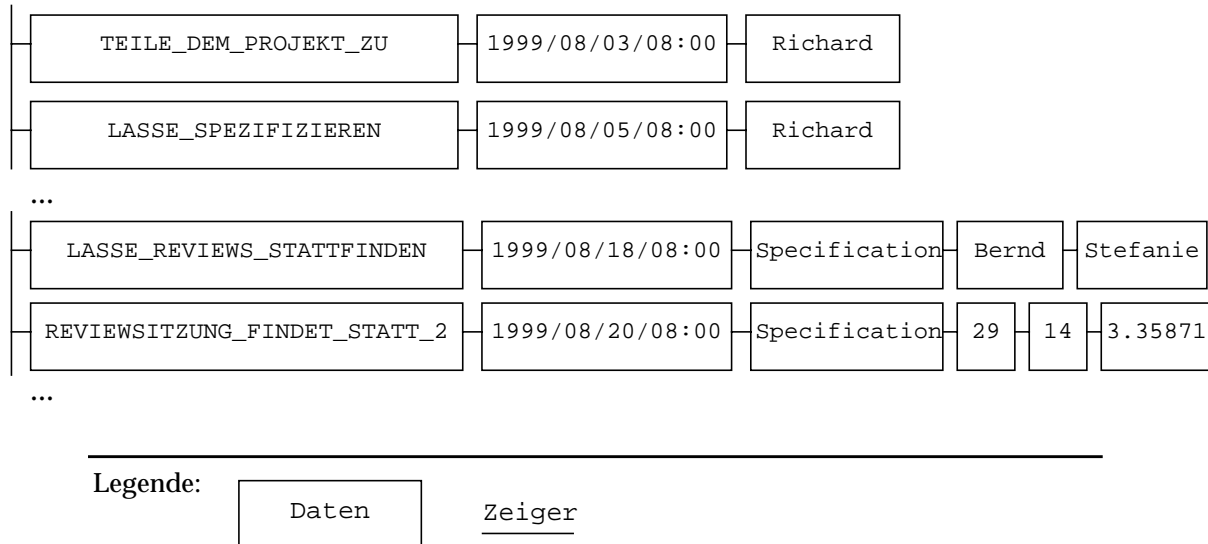
Um die Spielverläufe zu analysieren, lässt der Tutor die Projektsimulationen, die von den Teilnehmern der Schulung durchgeführt wurden, durch den Simulator nachspielen. Dabei entsteht für jedes Spiel eine Protokolldatei mit dem Suffix `.ist`, in der alle Aktionen des Spielers und die Reaktionen des Simulators stehen. Wie die folgenden Auszüge aus der Datei zeigen, ist die Datei einfach strukturiert. Es steht in jeder nichtleeren Zeile entweder das Datum, ein Kommando oder eine Nachricht mit den Parametern in Klammern.

```
1999/08/03/08:00
TEILE_DEM_PROJEKT_ZU(Richard)
ENTWICKLER_WURDE_EINGESTELLT_1 (Richard)
...
1999/08/05/08:00
LASSE_SPEZIFIZIEREN(Richard)
1999/08/05/08:00
PRINT_DATE (1999/08/05/08:00)
BEGONNEN_ZU_SPEZIFIZIEREN_4 (Richard)
...
1999/08/13/08:00
AUFGEHOERT_ZU_SPEZIFIZIEREN_2 (Richard)
...
1999/08/18/08:00
LASSE_REVIEWS_STATTFINDEN(Specification, Bernd, Stefanie)
1999/08/18/08:00
PRINT_DATE (1999/08/18/08:00)
BEGONNEN_DOKUMENT_ZU_BEGUTACHTEN_2 (Bernd,Specification)
BEGONNEN_DOKUMENT_ZU_BEGUTACHTEN_3 (Stefanie,Specification)
...
1999/08/20/08:00
REVIEWSITZUNG_FINDET_STATT_2 (Specification,29,14,3.35871)
```

#### Abbildung 4: Auszug aus einer Protokolldatei

Der Tutor kann diese Protokolldateien in das Werkzeug laden. Jede Protokolldatei wird in einem Syntaxbaum gespeichert. Dabei werden nur im Modell enthaltene Kommandos und Nachrichten berücksichtigt. Sind in der Protokolldatei andere Nachrichten oder Kommandos als im Modell enthalten, erhält der Benutzer eine Warnung.

Die Struktur des Syntaxbaums ist durch die Aufgabenstellung vorgegeben und wird als Implementierungsvorgabe in der Spezifikation beschrieben. In Abbildung 5 ist eine Skizze eines Syntaxbaums mit einem Teil der Kommandos und Nachrichten aus der in Abbildung 4 angegebenen Protokolldatei dargestellt.



**Abbildung 5: Skizze eines Syntaxbaums**

Ein Syntaxbaum besteht aus einer Liste mit einem Eintrag pro Kommando oder Nachricht der Protokolldatei. Dieser Eintrag enthält den Namen der Nachricht oder des Kommandos, das Projektdatum, an dem die Aktion vom Benutzer oder die Rückmeldung vom System gegeben wurde und die Parameter des Kommandos oder der Nachricht. Da keine Information über Parameter oder ihre Typen vorhanden ist, werden alle Daten im Syntaxbaum als Datentyp String gespeichert.

### 3.3.5 Allgemeine Auswertungsschemata

Mit allgemeinen Auswertungsschemata kann der Tutor beliebige Daten aus den Spielverläufen gewinnen. Die Ergebnisse können als Tabelle, Balken- oder Liniendiagramm dargestellt werden.

#### 3.3.5.1 Eingabe von allgemeinen Schemata

Wie schon in der Einleitung in Abschnitt 3.3.1 kurz erklärt, werden Auswertungen als Schema definiert. Syntaxbäume werden mit einem Schema ausgewertet und die Ergebnisse dieser Auswertung als Diagramm dargestellt. In diesem Abschnitt wird beschrieben, wie der Tutor ein allgemeines Schema eingeben kann. Als Beispiel wird die in der Einleitung beschriebene Auswertung für die Anzahl gefundener Fehler in Reviews verwendet und erweitert.

## Bedingungen

Als ersten Schritt wählt der Tutor Einträge mit den benötigten Daten aus dem Syntaxbaum aus. Dazu gibt der Tutor in Form von Bedingungen an, welche Einträge und damit welche Kommandos und Nachrichten untersucht werden sollen. Abbildung 6 zeigt das Fenster des Werkzeugs für diese Eingabe. Der Benutzer kann Bedingungen für den Namen des Kommandos, das Datum und Parameter angeben. Da keine Information über die Parameter im Werkzeug verfügbar ist, werden die Parameter nummeriert.

Condition 1				
0 (Name)	1 (Alias)	2 (Date)	3 (Par 1)	4 (Par 2)
REVIEWSITZUNG_FINDET_S			Specificatio	
choose ...	choose ...	choose ...	more	more

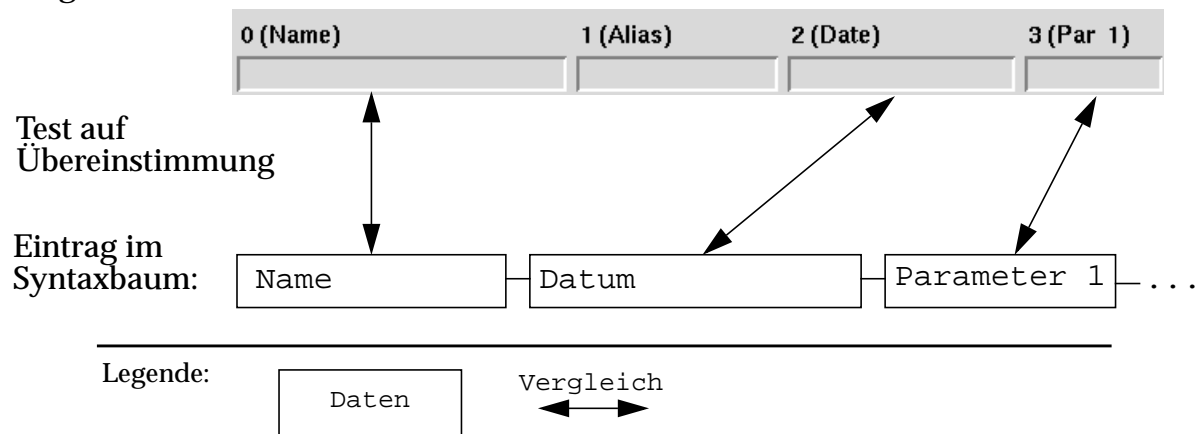
Condition 2				
0 (Name)	1 (Alias)	2 (Date)	3 (Par 1)	4 (Par 2)
choose ...	choose ...	choose ...	more	more

**Abbildung 6: Eingabefenster für Bedingungen**

Es werden dann die Einträge im Syntaxbaum gesucht, die diese Bedingungen erfüllen. Die Skizze in Abbildung 7 soll den Zusammenhang zwischen der Eingabe und den Einträgen im Syntaxbaum verdeutlichen. Jeder Eintrag im Syntaxbaum wird mit diesen Bedingungen untersucht. Auf die Bedeutung der Aliasnamen wird später eingegangen.

Mit den Pfeilen am rechten und unteren Rand des Eingabefensters können weitere Bedingungen eingegeben werden.

Eingabe des Tutors:



**Abbildung 7: Bedingungen und Syntaxbaum**

Um beispielsweise alle Meldungen über Reviewsitzungen mit der Spezifikation als Prüfung zu untersuchen, wird für den Namen "REVIEWSITZUNG\_FINDET\_STATT\*" angegeben und für den ersten Parameter "Specification".

Für die Prüfung der Einträge im Syntaxbaum werden die Teile einer Bedingung für Name, Datum oder die Parameter mit "UND" verknüpft. Der Eintrag im Syntaxbaum muss dann als Name "REVIEWSITZUNG\_FINDET\_STATT\*" haben und im ersten Parameter "Specification". Leere Angaben wie das Datum im Beispiel werden ignoriert.

Mehrere Eingaben für einen Teil einer Bedingung werden mit "ODER" verknüpft. Gibt der Tutor im Beispiel für den ersten Parameter "Specification", "Systemdesign" und "Modulspecification" an, muss der erste Parameter eine der drei Angaben erfüllen. Ebenso werden die Namen mit "ODER" verknüpft, wenn der Tutor in einer Bedingung mehrere Namen angibt.

Um dem Tutor die Eingabe zu erleichtern, kann das Zeichen "\*" als Wildcard verwendet werden. Damit lassen sich Namen von Kommandos oder Nachrichten abkürzen. So kann etwa der Suffix von Nachrichten, der dem Dolmetscher abwechslungsreichere Ausgaben erlaubt, durch den Wildcard "\*" ersetzt werden. Im QS-Modell werden die Nachrichten aus Reviewsitzungen mit unterschiedlichem Suffix ausgegeben, etwa "REVIEWSITZUNG\_FINDET\_STATT\_1". Der Tutor muss dann nicht alle möglichen Nachrichten angeben, sondern kann dies verkürzt durch "REVIEWSITZUNG\_FINDET\_STATT\*" erreichen oder den Namen weiter abkürzen.

Gibt der Tutor Namen von Kommandos oder Nachrichten ein, die nicht im Simulationsmodell enthalten sind, erhält er vom Werkzeug eine Warnung.

Kommando- und Nachrichtennamen können durch einen Aliasnamen ersetzt werden. Aliasersetzung kann etwa verwendet werden, um Nachrichten des Systems mit selber

Semantik, aber unterschiedlichen Namen zusammenzufassen. Aliasnamen werden auch in Gantt-Schemata eingesetzt, in Abschnitt 3.3.6 ab Seite 27 werden sie näher erklärt.

Um die Eingabe der Kommando- und Nachrichtennamen komfortabler zu gestalten, kann über Buttons unter den Eingabefeldern ein Fenster zur Auswahl von Kommandos und Nachrichten geöffnet werden. Mit diesem Fenster kann der Tutor aus allen Namen der Kommandos und Nachrichten des Simulationsmodells die gewünschten Namen wählen.

Aliasnamen können ebenfalls über ein eigenes Fenster angegeben werden, in dem sich Aliasnamen und die Zuordnung von Aliasnamen zu bestimmten Kommando- oder Nachrichtennamen angeben lassen. Kommando- und Nachrichtennamen kann der Tutor mit dem Fenster in Abbildung 8 auswählen, Aliase lassen sich über das Fenster aus Abbildung 9 wählen. Im Fenster in dieser Abbildung werden die verschiedenen Nachrichten, mit denen ein Mitarbeiter meldet, dass er an einem Dokument arbeitet, durch den Aliasnamen "Spezifikation" zusammengefasst.

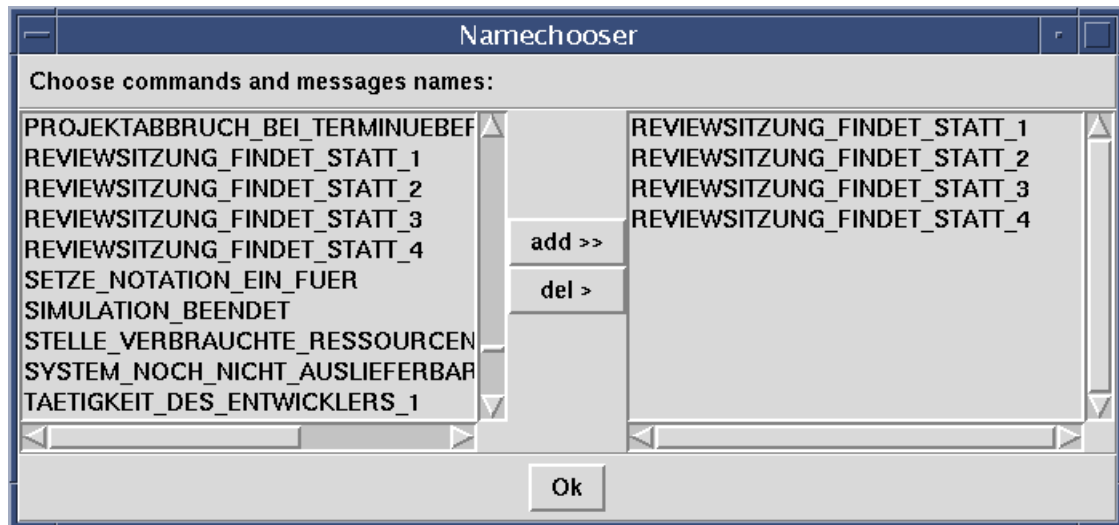


Abbildung 8: Auswahl von Namen



**Abbildung 9: Auswahl von Aliasnamen**

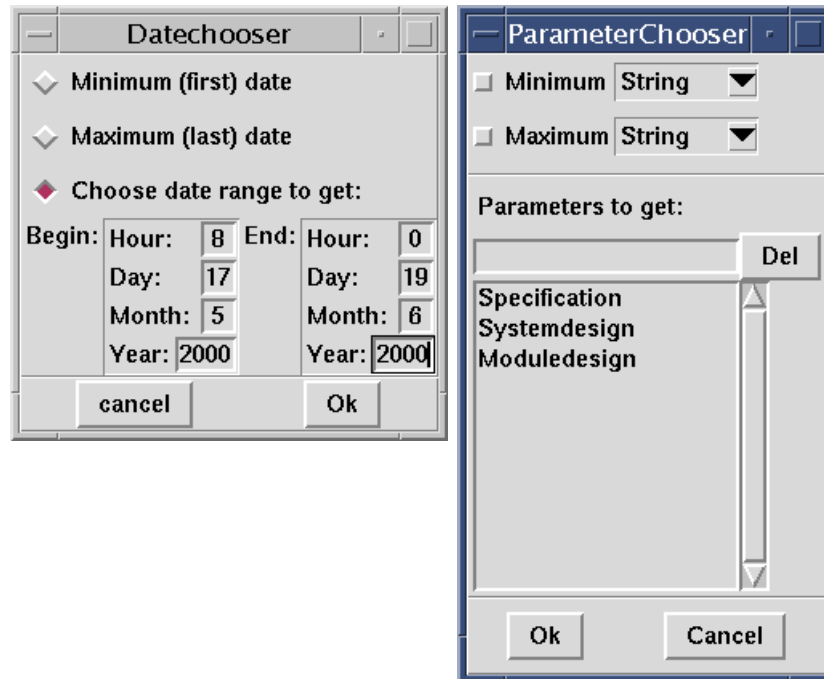
Bedingungen für das Datum oder die Parameter kann der Tutor entweder direkt über das jeweilige Eingabefeld eingeben oder über separate Fenster, die mit den Buttons unter dem Eingabefeld geöffnet werden.

Für das Datum lassen sich über dieses Fenster ein Zeitraum, das Minimum oder das Maximum des Datums wählen, an dem der Spieler das Kommando gegeben oder das System die Nachricht zurückgegeben hat (Abbildung 10). Damit lässt sich also der Zeitraum der betrachteten Kommandos und Nachrichten einschränken. Minimum und Maximum ergeben das erste beziehungsweise letzte Vorkommen eines Kommandos oder einer Nachricht im Spielverlauf.

Über das Fenster zur Eingabe von Bedingungen für Parameter lassen sich mehrere Werte, der kleinste oder der größte Wert eines Parameters wählen. Wird etwa vom Tutor das Minimum ausgewählt, wird der Eintrag des Syntaxbaums weiterverarbeitet, der den kleinsten Wert in diesem Parameter hat und alle anderen Teile der Bedingung erfüllt. Werden sowohl Minimum, Maximum und weitere Werte angegeben, werden diese mit "ODER" verknüpft. In Abbildung 10 sind die Dokumente "Specification", "Systemdesign" und "Moduledesign" gezeigt.

Da Sesamscore keine Typinformation über die Parameter hat, bei der Auswertung aber der Typ bei Minimum- oder Maximumbedingungen berücksichtigt werden muss, gibt der Tutor den Datentyp an. Mögliche Datentypen sind String (Default), Integer, Real, Date oder Boolean. Die Datentypen von Sesamscore sind damit die gleichen wie die Basistypen der Basissprache von SESAM-2 (Reißing, 1996). Datentypen können über eine Auswahl angegeben werden. Passen der angegebene Typ und der Typ des Parameters nicht zusammen, kann dieser Fehler nur zur Laufzeit bei der Auswertung entdeckt

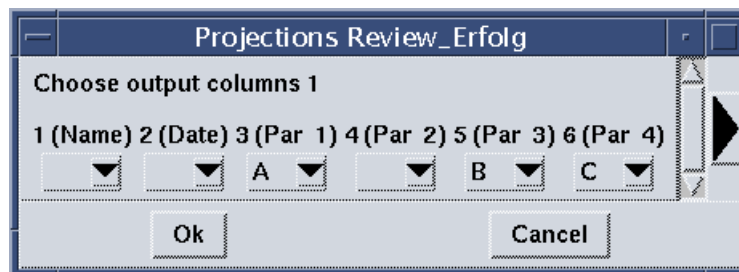
werden und nicht bereits, wenn das Schema eingegeben wird. Diese Typfehler werden abgefangen, der Benutzer erhält eine Meldung. Die Auswertung wird abgebrochen.



**Abbildung 10: Wahl von Datum und Parametern**

### Projektionen

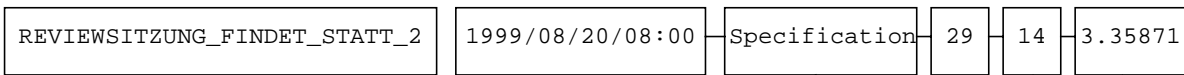
Im nächsten Schritt werden durch den Benutzer die für die Auswertung benötigten Daten ausgewählt. Dazu werden Kommando- oder Nachrichtenname oder der Aliasname, das Datum oder die Parameter in Spalten einer internen Ergebnistabelle projiziert. Abbildung 11 zeigt das Fenster für die Eingabe, Abbildung 12 eine Skizze für die Funktion.



**Abbildung 11: Eingabe von Projektionen**

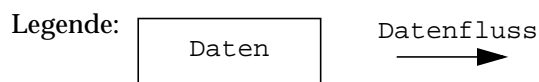
In der Beispielanfrage werden die Dokumente in die Spalte "A" eingetragen, die Anzahl der entdeckten Fehler in die Spalte "B" und die entdeckten Verluste der Function Points in die Spalte "C".

Eintrag im Syntaxbaum:



Projektion:

Spalten der  
Ergebnistabelle:



**Abbildung 12: Skizze für die Projektion**

Die folgende Abbildung zeigt einen Ausschnitt aus der internen Ergebnistabelle, in der einige Daten aus Reviewsitzungen eingetragen sind.

A	B	C
Specification	14	3.35871
Specification	13	0.0
Specification	2	1.53941
Systemdesign	1	0.139946
Systemdesign	7	1.74355
Moduledesign	11	0.821707
Moduledesign	10	1.05708
...		

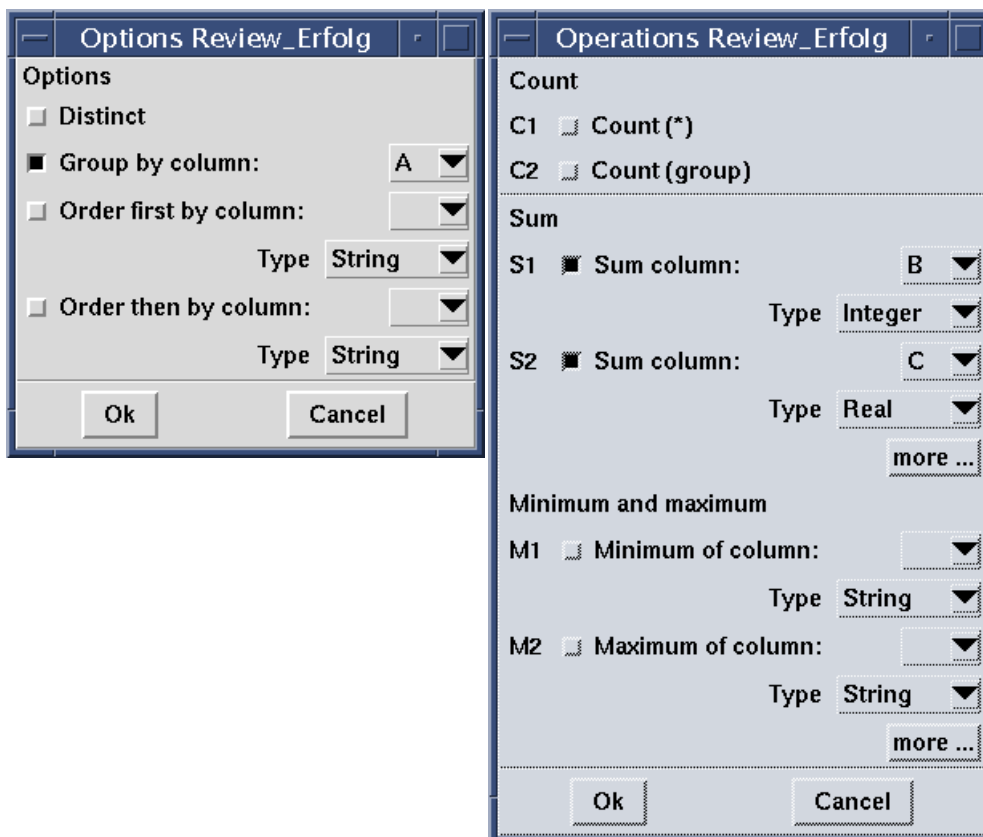
**Abbildung 13: Beispiel für eine Ergebnistabelle**

### Optionen und Operationen

Sesamscore bietet dem Tutor verschiedene Optionen und Operationen. Mit diesen kann der Tutor die durch die Projektionen ausgewählten Werte weiter bearbeiten lassen. Eingabefenster für Optionen und Operationen sind in Abbildung 14 dargestellt. Diese Fenster sind mit den Optionen und Operationen für die Beispielanfrage ausgefüllt.

Über die Option “Distinct” lassen sich Duplikate eliminieren. Es können damit mehrfach vorkommende gleiche Zeilen in der Ergebnistabelle zu einer Zeile zusammengefasst werden. Hat der Tutor etwa eine Anfrage nach Dokumenten und ihren Autoren gestellt, steht ein Autor, der mehrfach an einem Dokument geschrieben hat, auch mehrfach in der Ergebnistabelle. Mit “Distinct” können diese Duplikate gelöscht werden.

Es kann eine Gruppierung der Ergebnistabelle durchgeführt werden. Damit wird die Ergebnistabelle in einzelne Gruppen unterteilt, die in einer Spalte den gleichen Wert haben. Abbildung 15 zeigt verkürzt die nach Dokumenten gruppierte Ergebnistabelle aus dem Beispiel. Operationen, die im nächsten Abschnitt beschrieben werden, werden dann für jede Gruppe einzeln durchgeführt.



**Abbildung 14: Eingabe von Optionen und Operationen**

Die Ergebnistabelle kann nach den Werten in einer vom Tutor angegebenen Spalte sortiert werden. Da das Werkzeug keine Information über die Datentypen in dieser Tabelle hat, muss der Tutor wieder den Datentyp angeben. Soll die Spalte “A” im Beispiel sortiert werden, würde der Tutor den Datentyp “String”, für die Spalte “C” den Datentyp “Real” angeben.

A	B	C	
Specification	14	3.35871	] Gruppe "Specification"
Specification	13	0.0	
Specification	2	1.53941	
Systemdesign	1	0.139946	] Gruppe "Systemdesign"
Systemdesign	7	1.74355	
Moduledesign	11	0.821707	] Gruppe "Moduledesign"
Moduledesign	10	1.05708	
...			

**Abbildung 15: Beispiel für eine Gruppierung**

Zusätzlich können Operationen angegeben werden. Das Ergebnis dieser Operationen wird durch einen eigenen, fest vorgegebenen Namen bezeichnet, etwa "S1" für die Summe der Werte einer bestimmten Spalte. Auf die Ergebnisse der Operationen kann der Tutor für die Darstellung genauso zugreifen wie auf die Spalten der Tabelle.

- Der Benutzer kann die Anzahl aller Einträge der Ergebnistabelle wählen, die Einträge werden dann unabhängig von der Gruppierung gezählt. Im Beispiel wäre das die Anzahl aller Reviewsitzungen.
- Die Anzahl der Einträge pro Gruppe ist ebenfalls möglich, hier also die Anzahl der Reviewsitzungen pro Dokument.
- Werte in einzelnen Spalten können pro Gruppe zusammengezählt werden. Wurde keine Gruppierung durchgeführt, dann werden alle Werte zusammengezählt. Für das Beispiel kann also die gesamte Anzahl der gefundenen Fehler und die gesamte Anzahl der entdeckten Verluste von Function Points für jedes Dokument einzeln berechnet werden.
- Es kann für jede Spalte nach dem kleinsten oder größten Wert gesucht werden. Wieder wird diese Operation für jede Gruppe einzeln angewendet, wenn die Option Gruppierung gewählt wurde.

Auch bei Operationen muss der Datentyp angegeben werden, damit die dem Typ entsprechende Operation angewendet werden kann. Wie bei der Auswertung der Bedingungen kann hier ein Laufzeitfehler auftreten, wenn der Benutzer einen falschen Datentyp wählt. Die Auswertung wird dann mit einer entsprechenden Meldung an den Benutzer abgebrochen.

## Layout

Das Layout legt fest, wie die durch die Anfrage gewonnenen Daten dargestellt werden. Dabei kann der Tutor auswählen, ob die Ergebnisse als Tabelle, als Balken- oder als Liniendiagramm dargestellt werden. Mit dem Layout bestimmt der Tutor, wie die einzelnen Spalten der Ergebnistabelle den verschiedenen Elementen der Diagramme zugewiesen werden. Die Elemente einer Tabelle sind beispielsweise Zeilen und Spalten, Elemente eines Balkendiagramms die beiden Achsen und die Balken und ihre Anordnung. Die Werte in diesen Spalten werden dann vom Werkzeug in eine Diagrammtabelle geschrieben, die die Struktur der Darstellung widerspiegelt.

Für das Beispiel soll eine Tabelle für mehrere Spiele entstehen, in der die Summe der im Review gefundenen Fehler und der gefundenen Function Points für jedes Dokument dargestellt werden. Das Layout für dieses Diagramm ist in Abbildung 16 dargestellt.

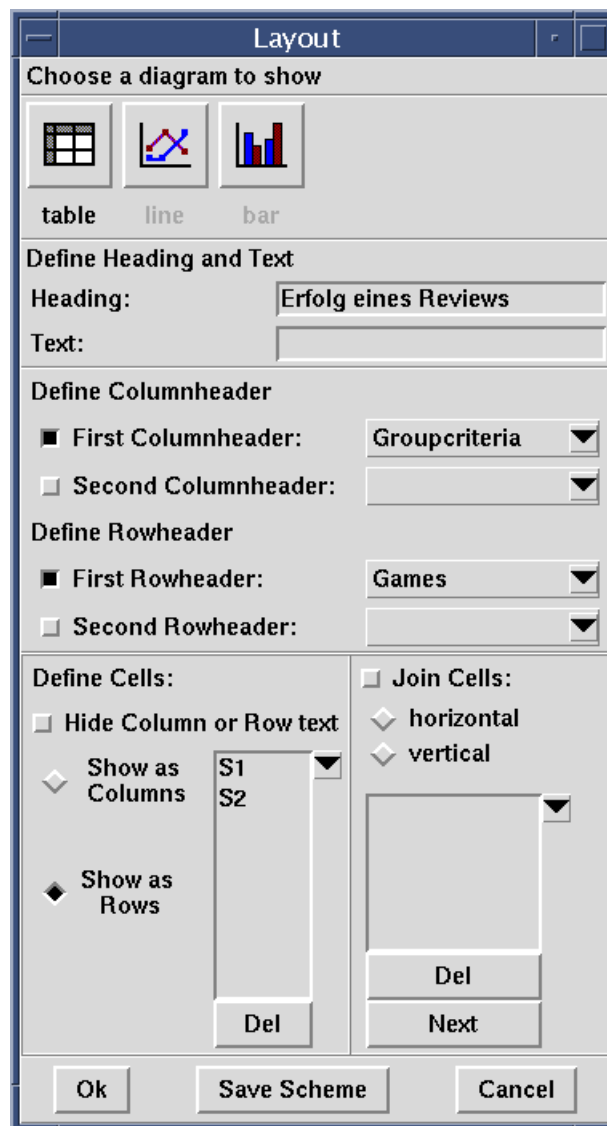


Abbildung 16: Layout einer Tabelle

Im Layout einer Tabelle werden die Spalten der Ergebnistabelle organisiert. Für das Beispiel sollen die Werte, nach denen gruppiert wurde, als Kopf für die Spalten verwendet werden. Damit werden die Spalten der Tabelle nach den Dokumenten angeordnet. Die Zeilen werden durch die Spiele organisiert, die der Tutor mit dem Schema auswertet.

Die Werte der Spalten, die in der Tabelle dargestellt werden, können im unteren Teil des Fensters angegeben werden. Die einzelnen Spalten der Ergebnistabelle können in der Tabelle nicht nur als Spalten, sondern auch als Zeilen angeordnet werden. Die Tabelle mit den in Abbildung 16 gezeigten Angaben ist in Abbildung 17 auf Seite 26 abgebildet. Damit Werte aus verschiedenen Spalten der Ergebnistabelle zusammengefasst werden können, kann der Tutor diese Spalten im unteren rechten Teil des Fensters wählen.

### **3.3.5.2 Starten der Auswertung**

Das Auswertungsschema legt nur die Vorschrift fest, wie die Spielverläufe ausgewertet werden sollen. Mit einem Schema kann der Tutor einen oder mehrere Spielverläufe auswerten, indem er einen oder mehrere Syntaxbäume auswählt. Das Werkzeug erstellt dann eine Auswertung und zeigt die Ergebnisse als Voransicht oder Diagramm. Für jede Auswertung wird zuerst eine Voransicht gezeigt und dann das Diagramm. Möchte der Tutor die Ergebnisse aus jedem Spiel in einem eigenen Diagramm darstellen, startet er für jeden Syntaxbaum eine eigene Auswertung mit demselben Schema.

Bei der Auswertung wird durch die Bedingungen und Projektionen aus einem oder mehreren Syntaxbäumen die Ergebnistabelle erstellt. Diese wird mit den Optionen und Operationen weiter bearbeitet. Durch das Layout wird aus der Ergebnistabelle die Diagrammtabelle erstellt, diese kann als Voransicht oder Diagramm dargestellt werden.

### **3.3.5.3 Darstellung der Ergebnisse**

Nachdem die vom Tutor angegebenen Syntaxbäume mit einem Schema ausgewertet sind, werden die Ergebnisse zuerst in einer Voransicht dargestellt. Diese Voransicht zeigt die Diagrammtabelle. In dieser Voransicht kann der Tutor die Darstellung der Ergebnisse verändern.

Der Tutor kann die Reihenfolge von Zeilen und Spalten von Tabellen oder den Balken eines Balkendiagramms ändern, also die Sortierung anpassen. Sollen beispielsweise Tätigkeiten eines Projekts dem Ablauf des Standard-Phasenmodells folgend dargestellt werden, kann diese Reihenfolge nicht immer durch die Anfrage gesichert werden. Eine Sortierung in alphabetischer Reihenfolge entspricht nicht diesem Ablauf. Diese Reihenfolge kann in der Voransicht korrigiert werden.

Alle Diagramme können durch den Tutor in der Voransicht gedreht werden.

Die Beschriftungen und Werte, die bei Balken- und Liniendiagrammen an den Diagrammachsen dargestellt werden, können durch einen Diagrammalias ersetzt werden.

Die Voransicht von Balken- und Liniendiagrammen bietet zudem die Möglichkeit, eine Legende für die dargestellten Balken oder Linien anzugeben.

Die Voransicht für eine Tabelle erlaubt dem Tutor, alle Werte in der Tabelle durch Diagrammmaliase zu ersetzen, um zu einer kompakteren Darstellung zu gelangen oder die Tabelle anschaulicher zu machen. Abbildung 17 zeigt eine Tabelle, in der die Spaltennamen S1 und S2 mit den Summen durch die Diagrammmaliase "Fehler" beziehungsweise "AFP" ersetzt wurden.

Diagrammmaliase und die Sortierung können von anderen Diagrammen übernommen werden. Will der Tutor etwa für jedes Spiel ein eigenes Diagramm erzeugen, muss er nicht für jedes Diagramm die Reihenfolge der Darstellung neu ändern.

Der Tutor kann nach der Voransicht das Diagramm erstellen lassen. Ein Wechsel zwischen Voransicht und Diagramm ist jederzeit möglich, so dass die Darstellung im Diagramm wenn nötig korrigiert werden kann.

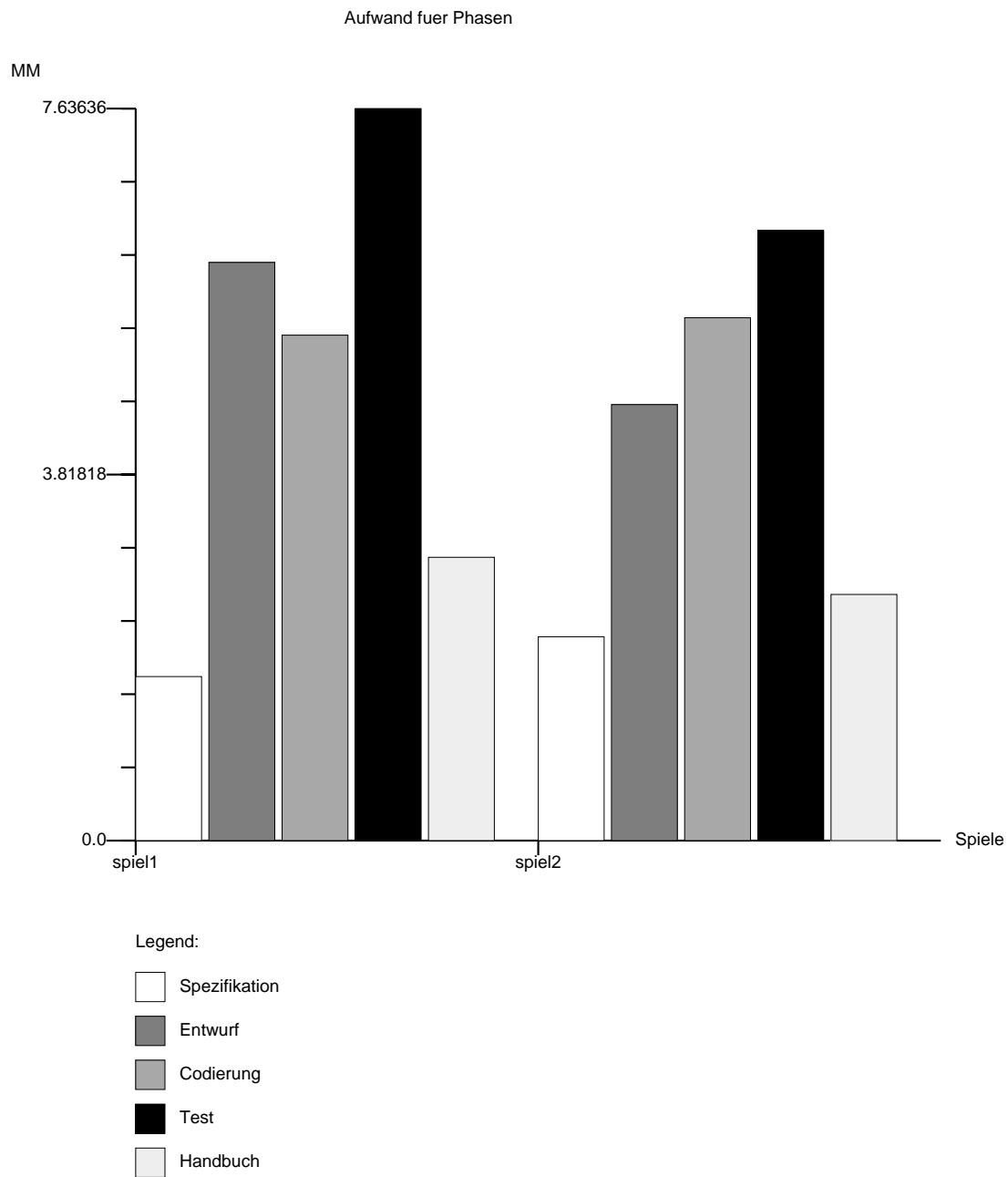
In Abbildung 17 wird die mit dem Beispielschema aus Abschnitt 3.3.5.1 erzeugte Tabelle für zwei Spiele abgebildet.

Erfolg eines Reviews

		Specification	Systemdesign	Moduledesign
spiel1	Fehler	81	57	51
	AFP	6.1409	5.8815	4.9545
spiel2	Fehler	94	33	49
	AFP	7.5413	3.7561	4.9475

**Abbildung 17: Tabellendiagramm**

Die Darstellung von Auswertungsergebnissen als Balkendiagramm wird in Abbildung 18 gezeigt. Die Anfrage wertet Nachrichten aus mehreren Spielen aus, die den Aufwand für die einzelnen Phasen Spezifikation, Entwurf, Codierung, Test und Handbuchenstellung als Parameter enthalten. Damit kann der Tutor einen Überblick über die Projekte gewinnen und feststellen, für welche Phasen eher zuviel oder zuwenig Aufwand investiert wurde.



**Abbildung 18: Aufwand für die Phasen**

### 3.3.6 Auswertungsschemata für Gantt diagramme

Mit dem Werkzeug Sesamscore sind neben allgemeinen Schemata auch Schemata für Gantt diagramme möglich. Diese erlauben dem Tutor, den Projektlauf anschaulich und übersichtlich darzustellen. In einem Gantt diagramm werden Zeiträume als Balken über einer Zeitachse dargestellt. Die Balken lassen sich im Diagramm nach Tätigkeiten oder Mitarbeitern ordnen.

Die für ein Ganttchart benötigten Informationen sind daher Tätigkeit, eingesetzte Mitarbeiter, der Beginn und das Ende dieser Tätigkeit. Diese Daten können aus den Nachrichten vom Simulator an den Dolmetscher entnommen werden. Dabei werden zwei Arten von Nachrichten unterschieden:

- Es kann Nachrichten geben, in denen unter anderem das Datum für Beginn und Ende einer Phase als Parameter ausgegeben werden. Im QS-Modell sind Nachrichten mit diesen Informationen als Tutornachrichten realisiert. Diese Nachrichten kann der Tutor ausgeben lassen, wenn er das Projekt vom Simulator nachspielen lässt. Er gibt dafür Tutorkommandos ein. Tutorkommandos kann der Spieler nicht absetzen, da mit den Tutornachrichten genaue Informationen über das Projekt ausgegeben werden können.
- Beginn und Ende einer Phase oder Tätigkeit eines Mitarbeiters sind durch unterschiedliche Nachrichten gekennzeichnet. Im QS-Modell meldet etwa ein Mitarbeiter, dass er angefangen hat zu spezifizieren und später, dass er damit aufgehört hat.

Die folgende Abbildung zeigt diese beiden Nachrichtenarten als Auszug aus einer Protokolldatei:

```

1999/08/05/08:00
PRINT_DATE (1999/08/05/08:00)
BEGONNEN_ZU_SPEZIFIZIEREN_4 (Richard)
...
1999/08/13/08:00
AUFGEHOERT_ZU_SPEZIFIZIEREN_2 (Richard)
...
2000/03/01/07:00
GIB_RESULTATE_SPEZIFIKATION
ZUSTAND_VON_SPEZIFIKATION (134.822,1999/08/04/08:00,1999/08/19/08:00,12)

```

Beginn und Ende einer Tätigkeit

Tutorkommando und folgende Nachricht mit Datum für Beginn und Ende

**Abbildung 19: Beispiel der Nachrichten für ein Ganttchart**

Mit Sesamscore kann der Benutzer ein Ganttogramm aus diesen verschiedenen Nachrichten erstellen. Damit sind folgende Auswertungen möglich:

- Der Tutor kann die Tutornachrichten untersuchen und Anfangs- und Endedatum aus den Parametern für das Ganttogramm verwenden.
- Er kann Anfangs- und Endedatum aus paarweise zusammengehörigen Nachrichten, jeweils für den Beginn und das Ende einer Tätigkeit verwenden.
- Zusätzlich kann der Tutor diese beiden Ansätze kombinieren und damit ein Ganttogramm erzeugen, in dem die Dauer einer Phase und die verschiedenen Tätigkeiten der Mitarbeiter dieser Phase dargestellt werden. Mit diesem Diagramm können

Unterbrechungen, Zeitverzögerungen und unbeschäftigte Mitarbeiter aufgezeigt werden.

Diese drei Möglichkeiten kann der Tutor mit Gantt-Schemata realisieren. Bedingungen und Projektionen einer Anfrage sind so allgemein, dass der Unterschied zu allgemeinen Schemata bei diesen Teilen einer Anfrage nur in der Darstellung des Eingabefensters liegt. Wie Abbildung 20 zeigt, wird das Fenster untergliedert. Die Bedingungen werden vom Werkzeug benannt. Operationen und Optionen werden für Gantt-Schemata nicht benötigt.

Als Beispiel wird in den nächsten beiden Abschnitten ein Schema gezeigt, das die beiden Ansätze kombiniert. Abschnitt 3.3.6.1 zeigt die Eingabe eines Schemas für ein Gantt-Diagramm, Abschnitt 3.3.6.2 stellt das Ergebnis einer Auswertung mit diesem Schema vor.

### **3.3.6.1 Eingabe von Gantt-Schemata**

Der Tutor wählt zuerst die Variante des Schemas. Die Variante bestimmt, welche der drei möglichen Schemaarten verwendet werden soll. Variante 1 untersucht nur Nachrichten, die Beginn und Ende als Parameter enthalten. Paarweise zusammenhängende Nachrichten können mit Variante 2 untersucht werden. Mit Variante 3 wird die Kombination der beiden Nachrichtenarten realisiert. Wie bei einem allgemeinen Schema, gibt der Tutor zuerst Bedingungen und danach die zugehörigen Projektionen ein.

#### **Bedingungen**

In Abbildung 20 ist das Eingabefenster für die Bedingungen eines Gantt-Schemas mit der Variante 3 dargestellt. Das Eingabefenster ist zweigeteilt, im oberen Teil gibt der Tutor Bedingungen für Nachrichten ein, die Beginn- und Endedatum einer Tätigkeit als Parameter enthalten. Im unteren Teil gibt der Tutor die Bedingungen für die Nachrichten ein, die den Beginn oder das Ende einer Tätigkeit kennzeichnen.

Für Variante 1 wird dem Tutor bei der Eingabe von Bedingungen nur der obere Teil zur Verfügung gestellt, bei Variante 2 der untere Teil.

The screenshot shows a dialog box titled "Conditions Gantt\_V3" with the following structure:

- Condition 1:**
  - 0 (Name): ZUSTAND\_VON\_SPEZIFIKAT
  - 1 (Alias): Spezifikation
  - 2 (Date): MAX
  - 3 (Par 1):
- Condition 2:**
  - 0 (Name): ZUSTAND\_VON\_ENTWURF
  - 1 (Alias): Entwurf
  - 2 (Date): MAX
  - 3 (Par 1):
- Condition Begin1:**
  - 0 (Name): BEGONNEN\_ZU\_SPEZIFIZIER
  - 1 (Alias): Spezifikation
  - 2 (Date):
  - 3 (Par 1):
- Condition End1:**
  - 0 (Name): AUFGEHOERT\_ZU\_SPEZIFIZ
  - 1 (Alias): Spezifikation
  - 2 (Date):
  - 3 (Par 1):
- Condition Begin2:**
  - 0 (Name): BEGONNEN\_SYSTEM\_ZU\_EI
  - 1 (Alias): Entwurf
  - 2 (Date):
  - 3 (Par 1):
- Condition End2:**
  - 0 (Name): AUFGEHOERT\_SYSTEM\_ZU
  - 1 (Alias): Entwurf
  - 2 (Date):
  - 3 (Par 1):

Annotations on the right side of the dialog box:

- A bracket on the right side of the first two conditions (Condition 1 and Condition 2) is labeled "Nachrichten mit Beginn und Ende als Parameter".
- A bracket on the right side of the last four conditions (Condition Begin1, Condition End1, Condition Begin2, and Condition End2) is labeled "Nachrichten, die Beginn oder Ende einer Tätigkeit kennzeichnen".

At the bottom of the dialog box are "Ok" and "Cancel" buttons.

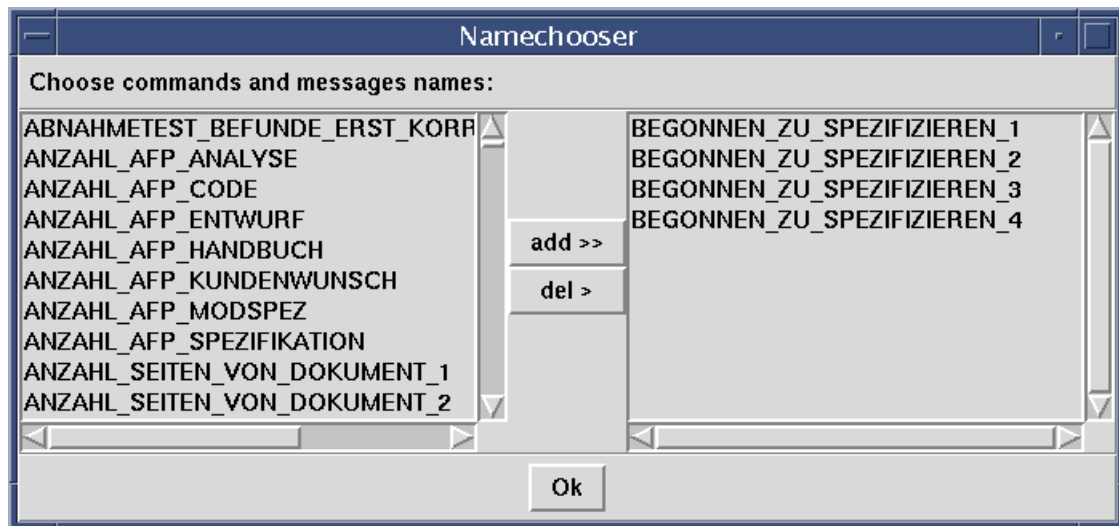
**Abbildung 20: Bedingungen für ein Gantt-schema**

Als Bedingung für die Nachrichten der Phase Spezifikation ist in diesem Schema die Nachricht "ZUSTAND\_VON\_SPEZIFIKATION" angegeben. Für das Datum wird "MAX" verwendet, damit die letzte Tutornachricht mit den Werten für die Spezifikation untersucht wird. Hat der Tutor nicht nur am Ende eines Projekts, sondern schon während der Spezifikationsphase Tutornachrichten ausgeben lassen, enthalten diese möglicherweise nicht den Zeitraum, der schließlich tatsächlich benötigt wurde.

Der Name der Nachricht wird durch einen Aliasnamen ersetzt. Damit wird die Tätigkeit bezeichnet und nicht mehr ein Name einer Nachricht. Im Beispiel wird die Nachricht

“ZUSTAND\_VON\_SPEZIFIKATION” durch die Tätigkeit “Spezifikation” ersetzt. Die Bedingung für die Tutornachrichten der Entwurfsphase sind genauso aufgebaut.

Beginn und Ende von Tätigkeiten, beispielsweise für die Spezifikation, werden durch zwei Bedingungen angegeben. In der ersten Bedingung werden die Nachrichten für den Beginn einer Tätigkeit angegeben, Abbildung 21 zeigt das Fenster mit der Auswahl der Nachrichtennamen. In der zweiten Bedingung sind dann die entsprechenden Nachrichten für das Ende der Tätigkeit angegeben. Diese Nachrichtennamen werden durch einen Aliasnamen ersetzt. Damit wird definiert, welche Beginn- und Endenachrichten zusammengehören. Alle Nachrichten, die den Beginn einer Tätigkeit kennzeichnen, müssen denselben Aliasnamen bekommen wie Nachrichten, die das Ende einer Tätigkeit kennzeichnen. In Variante 3 wird mit dem Alias ebenfalls zugeordnet, welche Tutornachricht zu der Tätigkeit gehört.

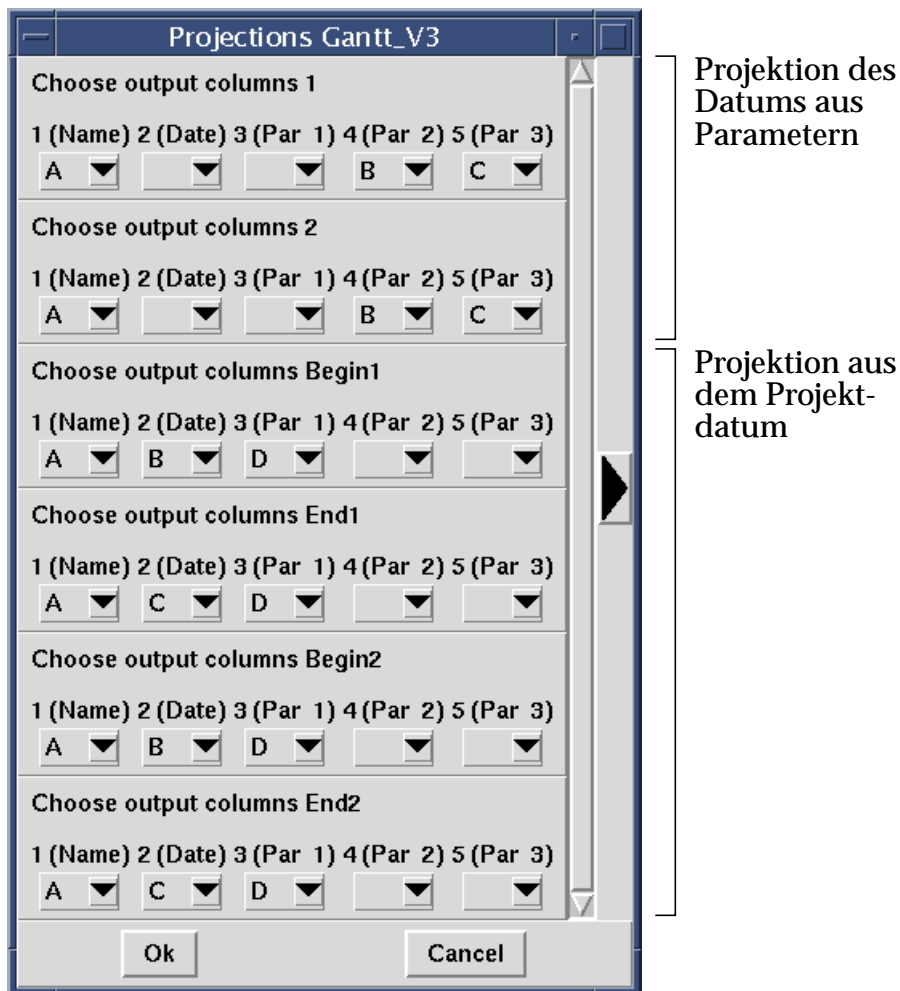


**Abbildung 21: Nachrichten für den Beginn einer Tätigkeit**

Wie bei einem allgemeinen Schema auch, werden die durch die Bedingungen gewählten Einträge des Syntaxbaums mit Projektionen weiterverarbeitet.

### **Projektion**

Die relevanten Daten sind für ein Gantt-Diagramm die Phase oder Tätigkeit, das Datum für den Beginn oder das Ende dieser Tätigkeit und der Mitarbeiter. Diese Daten wählt der Tutor mit der Projektion. Abbildung 22 zeigt die Projektion für die Bedingungen aus Abbildung 20.



**Abbildung 22: Projektion für ein Gantt-Diagramm**

Das Datum für den Beginn und das Ende wird bei den Tutornachrichten aus den Parametern projiziert. Im QS-Modell steht zum Beispiel das Datum für den Beginn als zweiter Parameter, das Datum für das Ende als dritter Parameter in der Tutornachricht "ZUSTAND\_VON\_SPEZIFIKATION".

Der Tutor erhält in diesem Beispiel das Datum für den Beginn und das Ende einer Tätigkeit eines Mitarbeiters, indem er das Projektdatum der Nachrichten "BEGONNEN\_ZU\_SPEZIFIZIEREN\_1" bis "BEGONNEN\_ZU\_SPEZIFIZIEREN\_4" in eine Spalte, "AUFGEHOERT\_ZU\_SPEZIFIZIEREN\_1" bis "AUFGEHOERT\_ZU\_SPEZIFIZIEREN\_4" in eine andere Spalte schreibt. Abbildung 23 zeigt eine mögliche Ergebnistabelle, die durch diese Projektion entsteht.

A	B	C	D
Spezifikation	1999/09/01	1999/09/20	
Spezifikation	1999/09/01		Diana
Spezifikation	1999/09/03		Thomas
Spezifikation		1999/09/15	Diana
Spezifikation		1999/09/20	Thomas
...			
Entwurf	1999/09/22	1999/10/12	
Entwurf	1999/09/22		Diana
Entwurf	1999/09/23		Thomas
...			

Tutornachricht

Beginnnachrichten  
je Mitarbeiter

Endenachricht  
je Mitarbeiteren

**Abbildung 23: Beispiel für eine Ergebnistabelle**

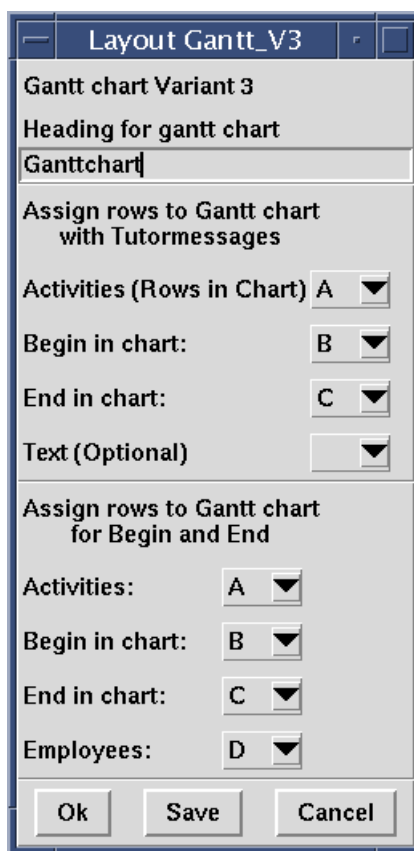
Das Datum ist in der Abbildung verkürzt dargestellt.

Damit das Werkzeug diese Daten als Gantt-Diagramm darstellen kann, gibt der Tutor das Layout ein.

### Layout

Im Layout gibt der Tutor an, in welcher Spalte der Ergebnistabelle Beginn- und Enddatum, Tätigkeit oder Mitarbeiter stehen. Die Diagramme für Variante 1 und Variante 3 sind immer nach Tätigkeiten oder Phase organisiert. Um die einzelnen Tätigkeiten nach Mitarbeitern aufzugliedern, wird diese Spalte zusätzlich angegeben. Abbildung 24 zeigt das Layout für das Schema, das in den vorherigen Abschnitten gezeigt ist.

Ein Diagramm, das entsteht, wenn ein Schema mit der Variante 2 ausgewertet wird, kann nicht nur nach der Tätigkeit, sondern auch nach Mitarbeitern organisiert werden.



**Abbildung 24: Layout für ein Gantt diagramm**

Wie in der Projektion angegeben, stehen Phasen oder Tätigkeiten in der Spalte "A" der Ergebnistabelle, das Beginndatum in der Spalte "B" und das Endedatum in der Spalte "C". Für die Nachrichten, die Beginn und Ende kennzeichnen, werden die Balken zusätzlich nach Mitarbeitern getrennt gezeichnet. Pro Mitarbeiter wird eine eigene Zeile im Gantt diagramm erstellt.

### **Starten der Auswertung**

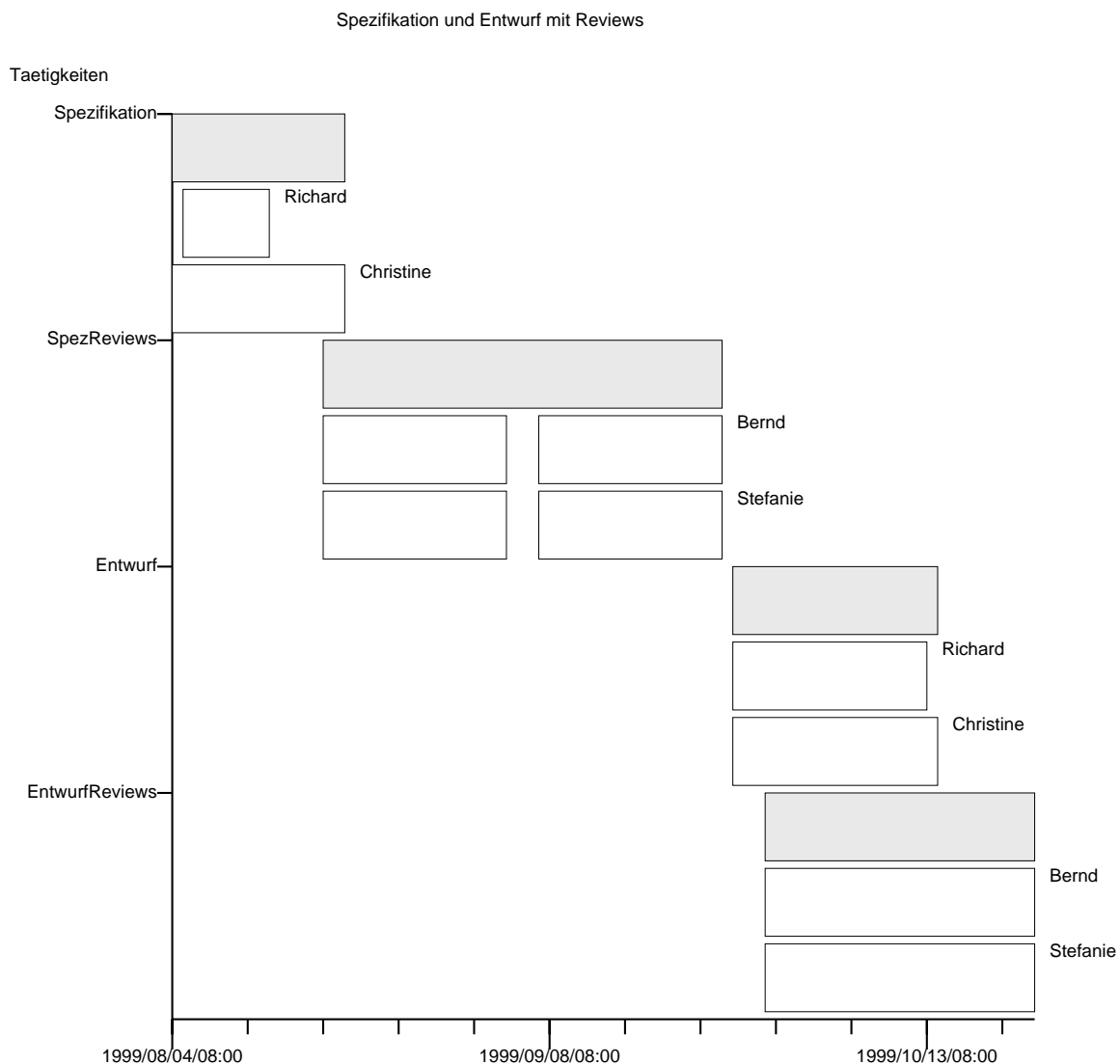
Wie bei allgemeinen Schemata auch, kann der Tutor die Auswertung starten, wenn Bedingungen, Projektionen und das Layout des Schemas definiert ist. Ein Gantt diagramm kann nur einen Spielverlauf darstellen.

Werden im Layout Spalten für Beginn- und Endedatum angegeben, die kein Datum enthalten, wird dies erst bei der Auswertung mit dem Schema festgestellt, da das Werkzeug keine Typinformation über die Parameter hat. Der Tutor erhält eine Meldung und die Auswertung wird abgebrochen.

### 3.3.6.2 Gantt diagramme

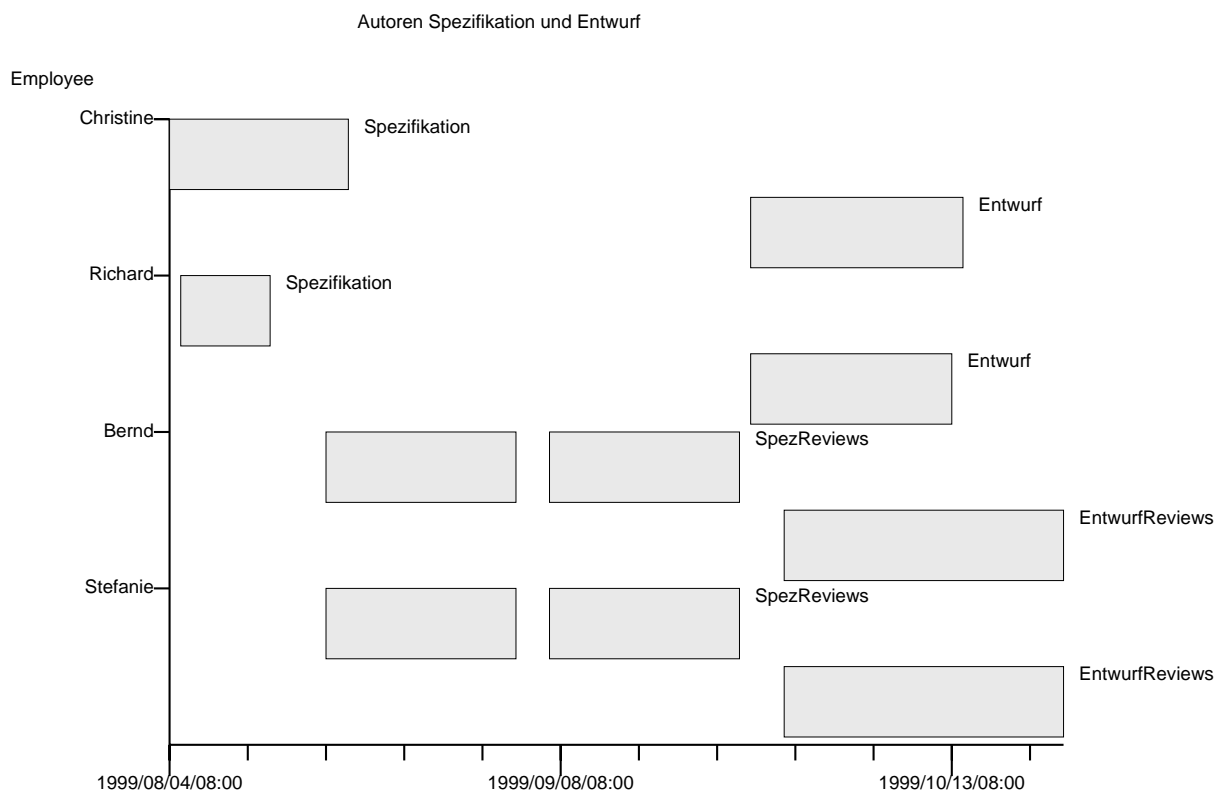
Nachdem ein Spielverlauf mit einem Gantt-schema ausgewertet wurde, werden die Ergebnisse zuerst tabellarisch in einer Voransicht angezeigt. Diese Tabelle enthält für jede Zeile im Gantt-diagramm die Daten der Balken. Der Tutor kann wie bei der Voransicht von allgemeinen Auswertungen die Reihenfolge der Zeilen umsortieren und für die Beschriftung Diagrammaliase angeben. Von der Voransicht aus kann der Tutor das Diagramm anzeigen lassen. Ein Wechsel zwischen Voransicht und Diagramm ist jederzeit möglich.

Die folgende Abbildung zeigt ein Gantt-diagramm der Variante 3. Das Schema, das im vorigen Abschnitt als Beispiel gezeigt wird, ist für diese Auswertung um Aspekte zu den Reviews in den einzelnen Phasen erweitert. Wie das Diagramm zeigt, wurden im Spiel zwei Reviews für die Spezifikation durchgeführt.



**Abbildung 25: Gantt-diagramm Spezifikation und Entwurf**

Ein Gantt-Diagramm der Variante 2, das nicht nach Tätigkeiten, sondern nach Mitarbeitern organisiert ist, wird in der folgenden Abbildung gezeigt. Dabei wurde ein Schema verwendet, das dieselben Bedingungen und Projektionen für Beginn- und Endnachrichten enthält wie das Beispielschema im vorherigen Abschnitt. Reviews werden ebenfalls dargestellt.



**Abbildung 26: Gantt-Diagramm der Mitarbeiter**

### 3.3.7 Zusätzliche Funktionalität

Das Werkzeug Sesamscore bietet dem Tutor noch weitere Funktionen zur Verwaltung der Diagramme und Auswertungsschemata. Diese werden in den nächsten Abschnitten kurz erläutert.

#### Verwaltung der Auswertungsschemata

Mit Sesamscore können mehrere Auswertungsschemata bearbeitet werden. Ein Schema wird vom Tutor über seinen Namen ausgewählt, um das Schema zu ändern, zu löschen oder um eine Auswertung zu starten. Wird ein Schema neu eingegeben, wird es vom System mit einem Defaultnamen versehen.

Damit eingegebene Schemata nicht verloren sind, wenn das Werkzeug beendet wird, kann der Tutor ein Schema in einer Datei speichern und später wieder laden. Wie beispielsweise bei Textverarbeitungsprogrammen üblich, kann beim Speichern des Sche-

mas ein neuer Name angegeben werden. Ändert der Tutor das Schema, kann er entweder die Datei mit dem neuen Schema überschreiben oder das Schema unter einem neuen Namen speichern.

Da das Werkzeug Schemata laden kann, können auch vorgefertigte Auswertungsschemata mit dem Werkzeug geliefert werden. Übliche Auswertungen von Spielverläufen müssen so von Tutoren nicht neu eingegeben werden.

Wird ein Schema eingegeben, verhindert das Werkzeug, dass der Tutor fehlerhafte Eingaben macht. Es lassen sich beispielsweise für Optionen und Operationen oder für das Layout nur die in der Projektion definierten Spalten angeben. Ändert der Tutor ein Auswertungsschema, wird vom Werkzeug überprüft, ob durch die Änderung solche Fehler im Schema entstehen. Der Tutor erhält dann eine Warnung mit einem entsprechenden Hinweis. Löscht der Tutor etwa eine Spalte in der Projektion, wird überprüft, ob Optionen, Operationen und das Layout auf diese Spalte zugreifen.

### **Diagramme und Voransicht**

Jedes Diagramm und seine Voransicht sind genau einem Schema zugeordnet. Wird das Schema eines bereits erstellten Diagramms geändert, so kann der Tutor das Diagramm an das geänderte Schema anpassen lassen und somit aktualisieren. Dabei werden dann dieselben Spielverläufe erneut ausgewertet. Mit diesen Ergebnissen wird das Diagramm neu erstellt.

Wird ein Schema unter einem neuen Namen gespeichert, kann der Tutor die zum Schema gehörenden Diagramme entweder dem Schema mit dem bisherigen Namen zugeordnet lassen oder dem neuen Schema zuweisen. Die Diagramme, die dem neuen Schema zugeordnet werden, werden dann vom Werkzeug aktualisiert und als Voransicht angezeigt.

Diagramme können auf zwei Arten in einer Datei gespeichert werden. Der Benutzer kann das Diagramm als Postscript-Datei speichern, um das Diagramm zu drucken. Das Diagramm kann dann auch mit einem Textverarbeitungsprogramm in ein beliebiges Dokument, beispielsweise Folien, die der Tutor in der Analyserunde zeigen möchte, eingebunden werden kann. Die andere Form der Speicherung ist die Möglichkeit, die Daten des Diagramms als strukturierten Text in eine Datei zu schreiben. Dabei werden die Daten aller Diagramme in Tabellenform gespeichert. Das Trennzeichen zwischen den Daten ist vom Tutor frei wählbar. Der so strukturierte Text kann mit anderen Programmen wie etwa MS Excel weiterverarbeitet werden.

## **3.4 Anforderungen an die Qualität**

Die Qualitätsanforderungen an das Produkt sind in einem eigenen Kapitel der Spezifikation definiert. Die wichtigsten Anforderungen an die Produktqualität entstehen aus dem Einsatz des Werkzeugs im Rahmen des SESAM-Projekts und den schon in der Aufgabenstellung vorgegebenen zusätzlichen Erweiterungen.

Darum wird neben der Brauchbarkeit auch auf die Wartbarkeit besonderen Wert gelegt. Zur Wartbarkeit zählt die Prüfbarkeit, Änderbarkeit und Portabilität (Ludewig, 1998). Prüfbarkeit entsteht durch eine möglichst vollständige Spezifikation, die genau alle Anforderungen an das Werkzeug wiedergibt und eine hohe Lokalität und Testbarkeit der Software.

Die Änderbarkeit wird durch eine sinnvolle Struktur erreicht. Im Entwurf wurde das System in überschaubare Einheiten mit möglichst loser Kopplung und hohem Zusammenhalt gegliedert und die Lösungsstruktur festgelegt. Kapitel 4 erläutert den Entwurf. Simplizität, Knappheit und Lesbarkeit sind ebenfalls wichtige Aspekte der Änderbarkeit. In der Spezifikation wurde darum die sehr gute Dokumentation des Codes verlangt.

Portabilität wird durch die verwendeten Programmiersprachen erreicht, die für verschiedene Systeme erhältlich sind. Das Werkzeug greift nur über die Bibliotheken dieser Sprachen auf das Betriebssystem und damit auf die Hardware zu. Die Verwendung der Programmiersprachen Tcl/Tk für die Oberfläche und Ada 95 für die Verarbeitung der Daten ist in diesem Kapitel vorgeschrieben, ebenfalls dass die Dokumentation mit Ausnahme des Quellcodes mit FrameMaker erstellt werden soll.

Leistungsanforderungen sind in Form von geschätzten Antwortzeiten definiert und sollen vom Werkzeug eingehalten werden.

## 4 Architektur des Auswertungswerkzeugs

Dieses Kapitel beschreibt den Entwurf und das Vorgehen während der Entwurfsphase. Abschnitt 4.1 beschreibt die erwünschten Eigenschaften, der darauf folgende Abschnitt das Vorgehen. In Abschnitt 4.3 werden die Ergebnisse vorgestellt.

### 4.1 Eigenschaften

Die Aufgabe des Entwurfs ist nach Ludewig (1998), die Lösungsstruktur festzulegen und das System in überschaubare Einheiten zu gliedern.

Die Gliederung des Werkzeugs ist in der Aufgabenstellung grob vorgegeben. Die Oberfläche soll in der Skriptsprache Tcl/Tk implementiert werden und Syntaxbäume und Zugriffe darauf in Ada 95. Die weitere Gliederung des Systems orientiert sich an der Struktur der Spezifikation, in der die funktionalen Anforderungen nach den verschiedenen Hauptfunktionen des Werkzeugs gegliedert sind. Damit wird das System in Komponenten unterteilt, die die erzeugten oder eingelesenen Daten wie Schemata, Syntaxbäume und die Resultate von Auswertungen enthalten oder Funktionen wie Laden von Dateien oder Auswerten der Syntaxbäume realisieren.

Diese Komponenten lassen sich in ein Schichtenmodell einordnen, in dem jede Komponente nur auf die Komponenten in derselben Schicht oder in darunterliegenden Schichten zugreift.

Diese Gliederung soll zu einer losen Kopplung zwischen den einzelnen Komponenten und einem engen Zusammenhalt innerhalb der Komponenten führen. Kopplung beschreibt dabei die Komplexität und Breite der Schnittstelle, Zusammenhalt die Verwandtschaft zwischen den Teilen einer Komponente. Da die Schnittstelle zwischen Tcl und Ada jedoch mit einem einzigen Modul realisiert werden muss, bietet dieses Modul eine eher breite und komplexe Schnittstelle an.

Im Feinentwurf sind die Komponenten in einzelne Module zerlegt. Jede Komponente besteht aus mehreren Modulen. Auch hier soll wieder lose Kopplung und hoher Zusammenhalt innerhalb der Module und Komponenten erreicht werden. Alle Daten sind als abstrakter Datentyp oder in wenigen Fällen als abstraktes Datenobjekt entworfen. Bei der Eingabe von Auswertungsschemata wurde beispielsweise für jedes Fenster ein abstrakter Datentyp entworfen, der die Eingaben enthält und Zugriffsoperationen darauf realisiert. Dadurch wurde eine hohe Lokalität erreicht.

Der Entwurf sollte auf Zuwachs ausgelegt sein, schon in der Aufgabenstellung ist festgelegt, dass Erweiterungen nicht nur in der Spezifikation, sondern auch im Entwurf berücksichtigt werden sollen.

Das gesamte Dokument wurde wie die Spezifikation mit FrameMaker erstellt. Es umfasst 59 Seiten und besteht aus 12 Kapiteln.

## 4.2 Vorgehen

Die Vorgehensrichtung bei der Entwicklung kann von verschiedenen Ansätzen ausgehen. Bei einer schrittweisen Verfeinerung wird Top-down vorgegangen, die Aufgabe also immer in kleinere Einheiten zerlegt. Wird von der Benutzerschnittstelle ausgegangen, wird die Richtung als Outside-in bezeichnet. Bottom-up bedeutet, dass kleine Einheiten wie Befehle solange kombiniert werden, bis die Lösung entsteht. Wird von den zentralen Algorithmen und Datenstrukturen ausgegangen, wird dieses Vorgehen als Inside-out bezeichnet (Ludewig, 1998).

Für den Grobentwurf wurde sowohl von der Zerlegung der Aufgabe wie in Abschnitt 4.1 beschrieben als auch von der durch den Prototyp spezifizierten Oberfläche ausgegangen. Damit wurden die Ansätze "Top-down" und "Outside-in" kombiniert.

Der Feinentwurf entstand hauptsächlich aus der entgegengesetzten Richtung. Für die im Grobentwurf beschriebenen Komponenten wurden zuerst die Datenstrukturen festgelegt, aus denen dann abstrakte Datentypen oder Datenobjekte gebildet wurden. Grundlage für die Daten waren die im Begriffslexikon der Spezifikation definierten Objekte. Diese sind mit der in Booch (1994) beschriebenen Notation im Entwurf dargestellt. Diese Notation verfolgt einen objektorientierten Ansatz, Ada95 ist um die entsprechenden Konstrukte erweitert. Merkmale der Objektorientierung werden beim Datenentwurf jedoch nur verwendet, um die Daten der Auswertungsschemata zu modellieren. Hier ist eine Vererbungshierarchie möglich und diese erlaubt, beispielsweise die Daten für das Layout der verschiedenen Diagramme in einer Datenstruktur der Schemata zu halten. Die anderen Daten wurden ohne objektorientierte Merkmale modelliert, lassen sich aber ebenfalls in der Notation von Booch darstellen.

Funktionen, die mit Daten arbeiten, die nach dem objektorientierten Ansatz modelliert sind, werden in eigenen Modulen untergebracht. Diese Module sind einer Komponente zugeordnet, der Komponente "Schemadaten" in Abbildung 27. Ausserhalb dieser Komponente muss nur bei der Auswertung von Syntaxbäumen mit einem Schema die Vererbungsstruktur der Schemadaten berücksichtigt werden.

Der Entwurf wurde wie die Spezifikation von der Betreuerin im Rahmen eines Meilensteins geprüft, bevor mit der nächsten Phase, der Implementierung, begonnen wurde.

## 4.3 Ergebnis des Entwurfs

Eine Skizze des Grobentwurfs ist in Abbildung 27 auf Seite 43 gezeigt. Jede Komponente und damit jedes Modul lässt sich einer bestimmten Schicht zuordnen. Die Bedeutung der Schichten und ihrer Komponenten wird im folgenden Abschnitt kurz erklärt:

- Schicht 4 enthält das Hauptprogramm, von dem aus der Benutzer die einzelnen Hauptfunktionen des Werkzeugs aufrufen kann.

- In der Schicht 3 werden diese Hauptfunktionen realisiert. Diese Schicht wird durch die Schnittstelle zwischen den Programmiersprachen Tcl/Tk und Ada 95 geteilt. Oberhalb dieser Schnittstelle liegen Prozeduren und Funktionen, die die Steuerung des Werkzeugs durch den Benutzer realisieren, die Komponenten unterhalb dieser Schnittstelle dienen der Verarbeitung der Daten aus Schicht 2.

Die Schicht enthält die Komponente für die Auswahl von Dateien, um etwa Protokolldateien, die Informationsdatei oder Schemata aus der Verzeichnisstruktur des Dateisystems auszuwählen. Die Datei wird dann geladen. Die Funktionalität zum Laden einer Informationsdatei wird in einer eigenen Komponente in Ada implementiert. Ebenfalls wird durch eine Komponente die Funktion zum Laden der Protokolldateien realisiert.

Die Komponente, um Schemata zu bearbeiten, stellt die Fenster zur Eingabe bereit. In dieser Komponente sind zusätzlich die Module enthalten, mit denen Schemata geladen und gespeichert werden können und ein Modul, um Auswertungen zu starten.

Die Funktionen und Prozeduren, mit denen Syntaxbäume durch Schemata ausgewertet werden können, sind in der Komponente "Auswerten der Schemata" untergebracht.

Diagramme und die Voransicht werden durch Module in der Komponente "Diagramme darstellen und bearbeiten" realisiert.

- Alle Daten, die mit Sesamscore verarbeitet und dargestellt werden, sind durch Komponenten in Schicht 2 realisiert. Eine Unterteilung in abstrakte Datenobjekte, die die Daten verwalten und abstrakte Datentypen für diese Daten ist zwar möglich, wird aber nicht im Grobentwurf dargestellt.

Die Verwaltung der Symboltabelle wird in der Komponente "Modelldaten" realisiert. Diese Komponente stellt Funktionen zur Überprüfung, ob ein Kommando- oder Nachrichtenname im Modell enthalten ist, bereit.

Protokolldateien werden durch Module in einer eigenen Komponente verwaltet und können zur Auswertung durch Funktionen aus dieser Komponente gelesen werden.

Alle Schemadaten werden von einer Komponente verwaltet, die Funktionen und Prozeduren zum Lesen und Schreiben der einzelnen Teile eines Schemas enthält. Es wird entweder von der Oberfläche aus auf diese Komponente zugegriffen, um die Eingaben und die Anzeige der Schemata zu realisieren oder zum Lesen von den Modulen der Komponente "Auswerten der Schemata".

Die Daten für die Ergebnistabelle werden ebenfalls in einer Komponente realisiert und verwaltet. Diese Komponente enthält Module für die Ergebnistabelle und um die Daten aus der Ergebnistabelle zu lesen.

In der Diagrammtabelle werden dann die Ergebnisse gespeichert. Module aus der Komponente für die Diagramme greifen auf die Diagrammtabelle zu, um sie mit den Daten aus der Ergebnistabelle zu füllen. Auf die Diagrammtabelle wird von diesen Modulen zugegriffen, um die Daten auszulesen, damit sie als Diagramm oder in der Voransicht dargestellt werden können. Die Daten werden in der Diagrammtabelle verändert, um die Funktionalität der Voransicht zu realisieren.

- Schicht 1 enthält Bibliotheken. Die generische Liste wird verwendet, um alle Datentypen für Daten, die im Werkzeug als Liste gespeichert werden sollen, zu realisieren.

Konstanten und atomare Typen des Werkzeugs werden in einer Komponente realisiert.

Da alle Daten im Syntaxbaum und in der Ergebnistabelle als Strings gespeichert werden, verarbeiten mehrere Module diesen Datentyp. Diese Module können auf die Bibliothek mit häufig gebrauchten Funktionen zurückgreifen.

Da auf diese Komponenten von vielen anderen Komponenten zugegriffen wird, sind in Abbildung 27 keine Verbindungspfeile für den Zugriff eingezeichnet.

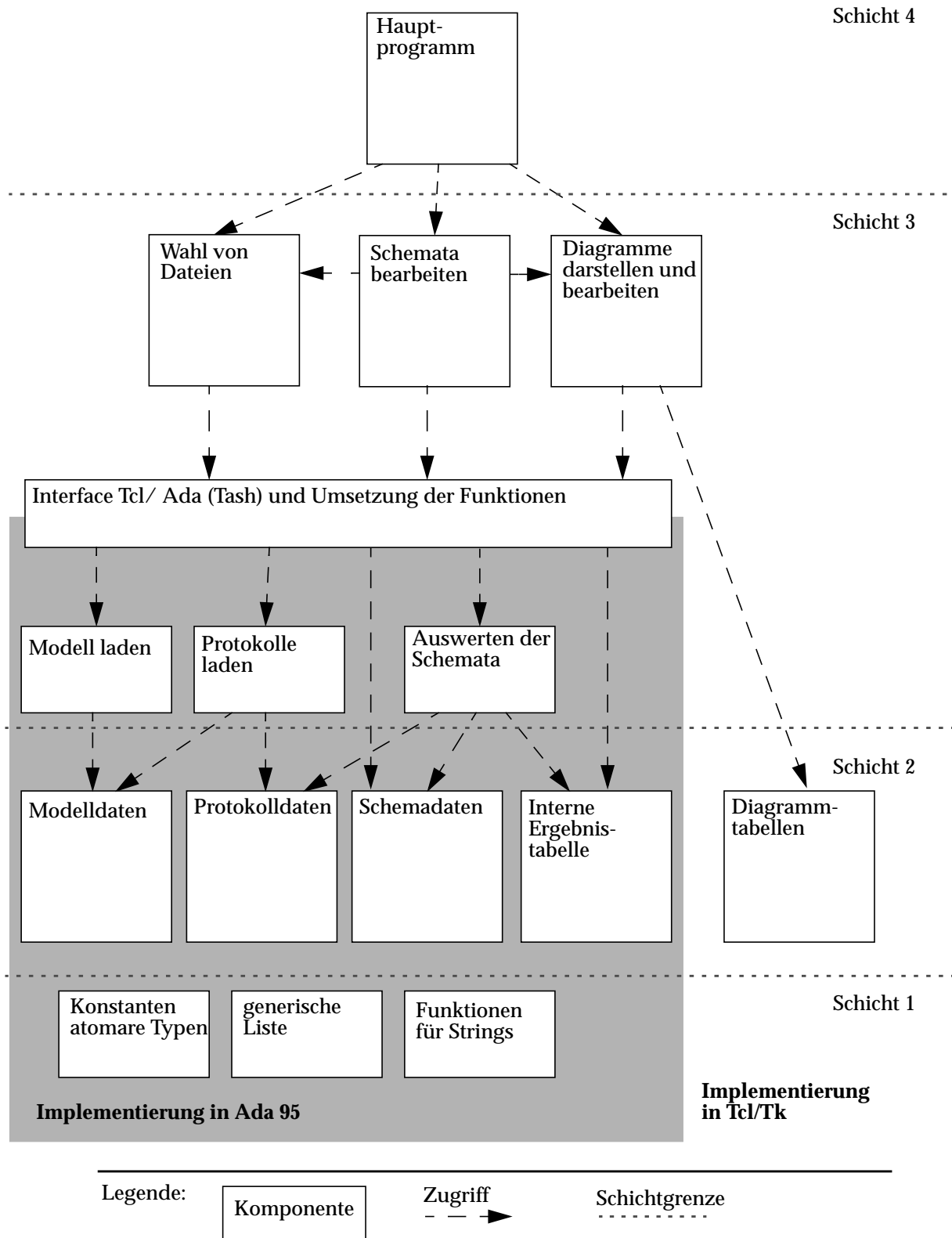
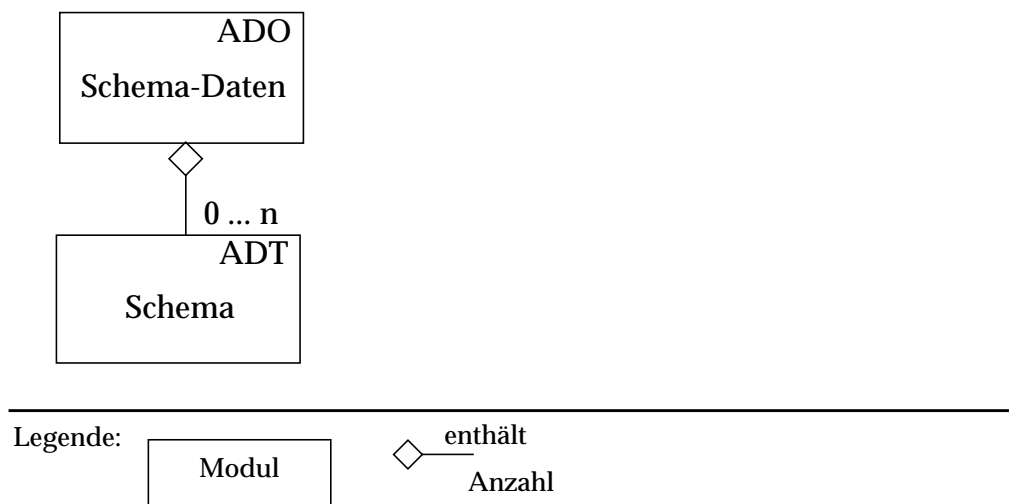


Abbildung 27: Grobentwurf

Im Feinentwurf werden die Komponenten in Module unterteilt. Jedes Modul realisiert entweder einen abstrakten Datentyp, ein abstraktes Datenobjekt oder eine Funktion des Systems. Alle Module sind im Dokument als Klassendiagramme dargestellt. Die Prozeduren, die Zugriffsoperationen auf die Daten realisieren, sind natürlichsprachlich beschrieben. Der Name der Prozedur oder Funktion und ihre Parameter werden aufgeführt.

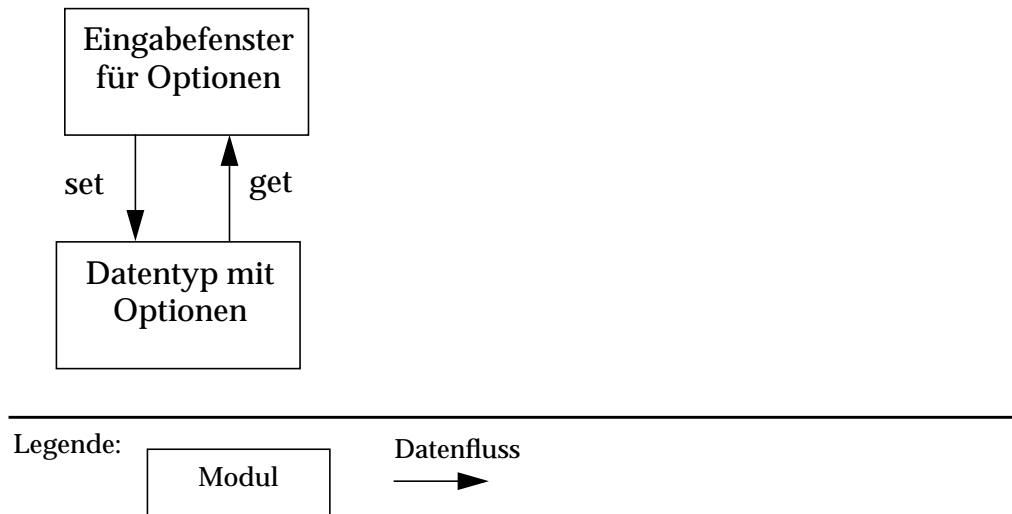
Die Daten sind nach Baumstrukturen organisiert, es wurde jeweils ein abstraktes Datenobjekt für die Symboltabelle mit den im Simulationsmodell vorhandenen Namen der Nachrichten und Kommandos, Syntaxbäume mit den Daten aus den Protokolldateien, Auswertungsschemata und die Ergebnistabellen der Auswertungen entworfen. Die Datenobjekte für Schemata, Syntaxbäume und Ergebnistabellen enthalten mehrere Schemata, Syntaxbäume beziehungsweise Ergebnistabellen. Abbildung 28 zeigt diese Struktur am Beispiel der Schemata.



**Abbildung 28: Struktur der Daten**

Das Modul mit den Schemadaten enthält alle im Werkzeug verfügbaren Schemata. Auf diese kann dann über die Schnittstelle des abstrakten Datenobjekts mit dem Namen des Schemas zugegriffen werden.

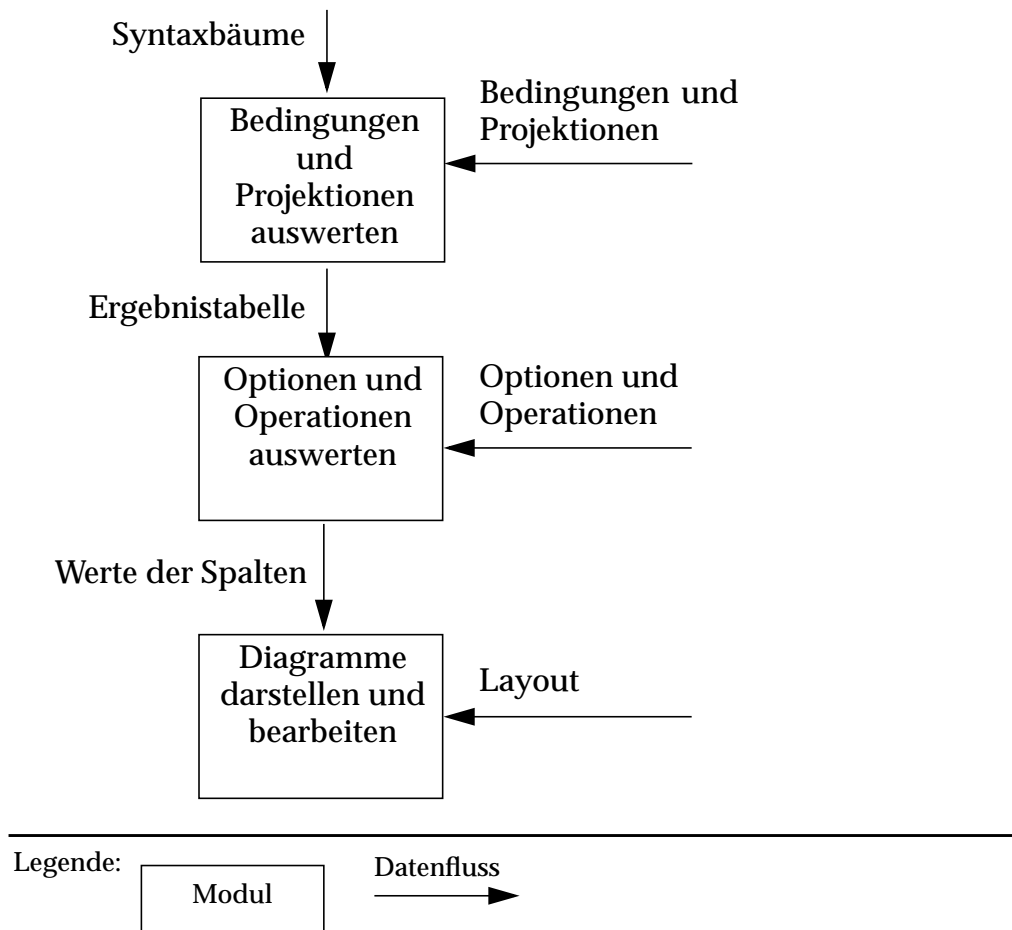
Wie schon in Abschnitt 4.1 kurz beschrieben, ist für jedes Eingabefenster bei der Eingabe von Schemata ein abstrakter Datentyp entworfen worden. Dieser Datentyp wird bei der Eingabe oder beim Laden eines Schemas instanziiert. Die Schnittstelle des abstrakten Datentyps stellt dann für die Oberfläche die Operation 'set' und die Funktion 'get' bereit, um die Daten einzutragen beziehungsweise zu lesen. Abbildung 29 zeigt den Datenfluss von der Oberfläche zu dem Datentyp bei der Eingabe und Änderung von Optionen.



**Abbildung 29: Zugriff auf die Daten**

Der Datentyp für die Schemata enthält die Datentypen für die Eingaben der Bedingungen, Projektionen, Optionen, Operationen und das Layout.

Um die Auswertung der Syntaxbäume mit einem Schema zu realisieren, werden zuerst die Einträge der Syntaxbäume nach den Bedingungen durchsucht und dann in eine interne Ergebnistabelle projiziert. Die Ergebnistabelle wird mit den Optionen und Operationen bearbeitet. Anschließend werden die Daten aus dieser Ergebnistabelle von den Modulen zur Darstellung und Bearbeitung der Diagramme an Hand des Layouts gelesen und in der Diagrammtabelle gespeichert. Die Struktur der Daten in der Diagrammtabelle ist passend für die Diagramme aufgebaut. Abbildung 30 zeigt den Datenfluss bei der Auswertung der Syntaxbäume.



**Abbildung 30: Auswertung der Syntaxbäume**

Jede Bearbeitung der Ergebnistabelle durch eine bestimmte Option oder Operation ist als eigene Funktion über die Ergebnistabelle entworfen. Dadurch ist es möglich, weitere Optionen oder Operationen zu realisieren.

## 5 Implementierung

In diesem Kapitel wird ein Überblick über die Implementierungsphase gegeben. Dabei wird zuerst das Vorgehen beschrieben und dann die Ergebnisse und nicht implementierte Anforderungen zusammengefasst. Einige Metriken über den Code werden am Schluss dieses Kapitels gezeigt.

### 5.1 Vorgehen

Zuerst wurden die Module des Werkzeugs implementiert, die in Ada95 zu codieren waren. Damit wurde die Grundlage für das Werkzeug geschaffen. Dabei wurden jeweils die Module für die Daten mit den Modulen für die Verarbeitung der Daten zusammen implementiert. In Ada wird jedes Modul durch ein Paket repräsentiert. Dieses Paket kann einen abstrakten Datentyp, ein abstraktes Datenobjekt oder auch eine Sammlung von Funktionen und Prozeduren realisieren.

Module und ihre Funktion wurden während der Implementierung überprüft. Dazu wurden Wegwerftests durchgeführt. Bei einigen wichtige Funktionen wie beispielsweise Funktionen der generischen Liste, die sehr oft und an unterschiedlichsten Stellen verwendet wird, wurde versucht, Funktionsüberdeckung zu erreichen und Grenzwerte zu prüfen.

Nachdem diese Module implementiert waren, konnte die Verbindung zwischen dem Interpreter für Tcl/Tk und den Ada-Modulen mit der Schnittstelle Tash (Tcl Ada Shell) erstellt werden. Diese Verbindung ist durch eine dynamisch ladbare Bibliothek realisiert, die von jedem Interpreter für Tcl/Tk geladen werden kann. Soll also ein neuer Interpreter verwendet werden, muss der Ada-Teil nicht verändert werden, ebensowenig die in Tcl/Tk selbst implementierten Module.

Im Anschluss wurden die Module für die Oberfläche geschrieben. Jedes Modul ist in einer eigenen Datei implementiert und realisiert einen eigenen Namensraum. In diesem Namensraum werden die Daten, die im Fenster gezeigt oder eingegeben werden, gekapselt. Das Modul enthält die Funktionen und Prozeduren, die auf die Daten in der Kapsel zugreifen.

Funktionen und Prozeduren, die über die Schnittstelle Tash auf Daten oder Funktionen in Ada zugreifen, liegen ebenfalls in einem eigenen Namensraum und sind damit von anderen Funktionen und Prozeduren leicht unterscheidbar.

### 5.2 Ergebnis der Implementierung

Die grundlegende Funktionalität wurde implementiert. Mit dem entstandenen Werkzeug können also Auswertungen eingegeben, gespeichert und geladen werden. Die Informationsdatei mit Namen und Nachrichten des Modells kann vom Benutzer geladen werden und wird verwendet, um die Eingaben für Schemata zu prüfen und Proto-

kolldateien zu laden. Der Benutzer kann die Protokolldateien mit den Schemata auswerten.

Die Ergebnisse lassen sich bereits als Diagramme und Voransicht darstellen, alle verschiedenen Diagrammartentypen sind implementiert. Die Diagramme können als die in der Spezifikation beschriebenen Dateien gespeichert werden. Die Diagramme, die im Kapitel 3.3 dargestellt sind, sind durch Auswerten existierender Spielverläufe entstanden und sind als Postskript-Datei in dieses Dokument eingebunden.

Die im Entwurf dargestellten Module, die in Ada 95 implementiert werden sollen, sind mit Ausnahme eines Moduls vollständig entstanden. Das fehlende Modul überprüft, ob ein Auswertungsschema in sich konsistent ist, beispielsweise ob eine Spalte verwendet werden soll, die nicht durch das Schema definiert ist und gibt dann Warnungen an den Benutzer über die Schnittstelle Tash zur Oberfläche.

### 5.3 Erweiterungen

Leider wurden aus Zeitgründen nicht alle Anforderungen umgesetzt. In diesem Abschnitt wird ein Überblick über die nicht implementierten Anforderungen gegeben. Anforderungen, die erst nach der Spezifikationsphase als sinnvoll erkannt wurden, werden gezeigt.

- Es fehlen zwei Module für die Eingabe von Bedingungen. Ein Modul realisiert das Fenster, mit dem der Tutor komfortabel einen Zeitraum oder Minimum und Maximum für das Datum eingeben kann (Abbildung 10). Das andere fehlende Modul erlaubt die Eingabe von Aliasnamen und deren Zuordnung zu verschiedenen Kommandos und Nachrichten (Abbildung 9).
- Beim Speichern eines Schemas unter einem anderen Namen sollen dem Schema mit dem neuen Namen Diagramme des alten Schemas zugeordnet werden können. Diese Anforderung wurde nicht umgesetzt. Die dazu gehörende Funktion, um die Diagramme zu aktualisieren, fehlt ebenfalls.
- Die volle Funktionalität der Voransicht wurde nicht implementiert. Sie zeigt nur die Daten, die im Diagramm dargestellt werden sollen, als Text. Die Funktionen, wie in Abschnitt 3.3.5.3 "Darstellung der Ergebnisse" kurz umrissen, um die Reihenfolge in der Darstellung zu ändern, Diagrammaliasse zu vergeben oder das Diagramm zu drehen sind nicht implementiert.
- Es werden nicht alle Benutzungsfehler abgefangen. Fehlerhafte Eingaben sind teilweise möglich, die Warnungen an den Benutzer müssen noch implementiert werden. Prüfungen zur Konsistenzsicherung der Schemata sind nicht vollständig implementiert.
- Bei dem Tabellendiagramm fehlt sowohl bei der Eingabe des Layouts als auch im Modul, mit dem das Diagramm dargestellt wird, die Möglichkeit, mehrere Spalten zusammenzufassen. Es können beispielsweise die an einem Review beteiligten Entwickler, die in einer Nachricht als verschiedene Parameter ausgegeben werden, nicht in einer Spalte der Tabelle dargestellt werden.

- Der Benutzer kann Schemata, Spielverläufe in Form von Syntaxbäumen und Diagramme nicht löschen.

Es kommen zwei Anforderungen an das Werkzeug dazu, die erst gegen Ende der Arbeit als nötig erkannt wurden:

- In einem Gantt-Diagramm sollen nicht nur Zeiträume in Form von Balken dargestellt werden können, sondern auch Zeitpunkte als Rauten. Dadurch kann beispielsweise in einem Diagramm dargestellt werden, wann während der Erstellung eines Dokuments der Spieler den Fortschritt bei der Arbeit am Dokument kontrolliert hat.
- Sollen Parameter beispielsweise in einem Balkendiagramm dargestellt werden, die aus unterschiedlichen Kommandos oder Nachrichten stammen, kann das mit diesem Werkzeug noch nicht geleistet werden. Die darzustellenden Werte werden in der Voransicht in Tabellenform dargestellt. Eine Zeile in dieser Tabelle ist dann nicht vollständig belegt, die Werte sind über mehrere Zeilen verteilt. Darum soll es in der Voransicht zusätzlich möglich sein, Zeilen in der Form zusammenzufassen, dass eine nicht vollständig ausgefüllte Zeile durch die Werte aus anderen, vom Benutzer gewählten Zeilen ergänzt wird.

## 5.4 Metriken

In diesem Abschnitt werden einige einfache Metriken über den Code gezeigt. Die Größe wird durch LOC (Lines of Code) bestimmt. Es wird zwischen Kommentarzeilen, Leerzeilen und der Gesamtzahl der Zeilen unterschieden.

Kommentarzeilen sind in Ada 95 alle Zeilen, die das Kommentarzeichen "--" enthalten und vor diesem Zeichen nur Leerzeichen. Leerzeilen sind definiert als Zeilen, die keine oder beliebig viele Leerzeichen enthalten und sonst nichts. Für die in Tcl/Tk implementierten Module sind Kommentarzeilen durch Zeilen definiert, die das Kommentarzeichen "#" und vor diesem Zeichen nur Leerzeichen enthalten.

Ada		LOC
	Zeilen insgesamt	17402
	Kommentarzeilen	7272
	Leerzeilen	1714
Tcl/Tk		
	Zeilen insgesamt	12692
	Kommentarzeilen	3669
	Leerzeilen	1860

Abbildung 31: Gesamtgröße in Zeilen Code

Module sind in Ada durch Pakete implementiert. Jedes Paket besteht aus einer Spezifikationsdatei mit den Prozedur- und Funktionsköpfen und damit der Schnittstelle, die andere Module benutzen können und der Definition von privaten Typen. Die Funktionen und Prozeduren werden in einer Implementierungsdatei realisiert. Für die Anzahl der Module wurden die Spezifikationsdateien gezählt.

Jedes Module in Tcl/Tk wird durch eine Datei realisiert, diese Dateien wurden für die Metrik gezählt.

Grundlage für die durchschnittliche Größe der Module ist die Gesamtzahl der Zeilen einschließlich Kommentar- und Leerzeilen. Diese wird durch die Anzahl der Module geteilt und das Ergebnis auf ganze Zahlen gerundet.

Module	Ada	Tcl
Anzahl Module	54	33
Durchschnittliche Größe	322 LOC	385 LOC

**Abbildung 32: Größe der Module**

Für die Metriken über die Funktionen und Prozeduren wurden für den Ada-Code alle Funktionen und Prozeduren im Implementierungsteil der Pakete betrachtet. Im Spezifikationsteil werden die Funktions- und Prozedurköpfe, die die Schnittstelle zu anderen Modulen bilden, nur wiederholt.

In Tcl/Tk gibt es syntaktisch keine Unterscheidung zwischen Prozeduren und Funktionen, bei der Erhebung der Metrik wurde deshalb nicht zwischen Prozeduren und Funktionen unterschieden. Grundlage für die durchschnittliche Größe ist die Gesamtzahl der Zeilen einschließlich Kommentar- und Leerzeilen.

Funktionen und Prozeduren	Ada	Tcl
Anzahl Funktionen	344	-
Anzahl Prozeduren	92	293
Durchschnittliche Größe	40 LOC	43 LOC

**Abbildung 33: Größe von Funktionen und Prozeduren**

## 6 Bewertung und Rückblick

In diesem Kapitel wird ein Überblick über den Verlauf des Projektes gegeben und das Projekt kurz bewertet.

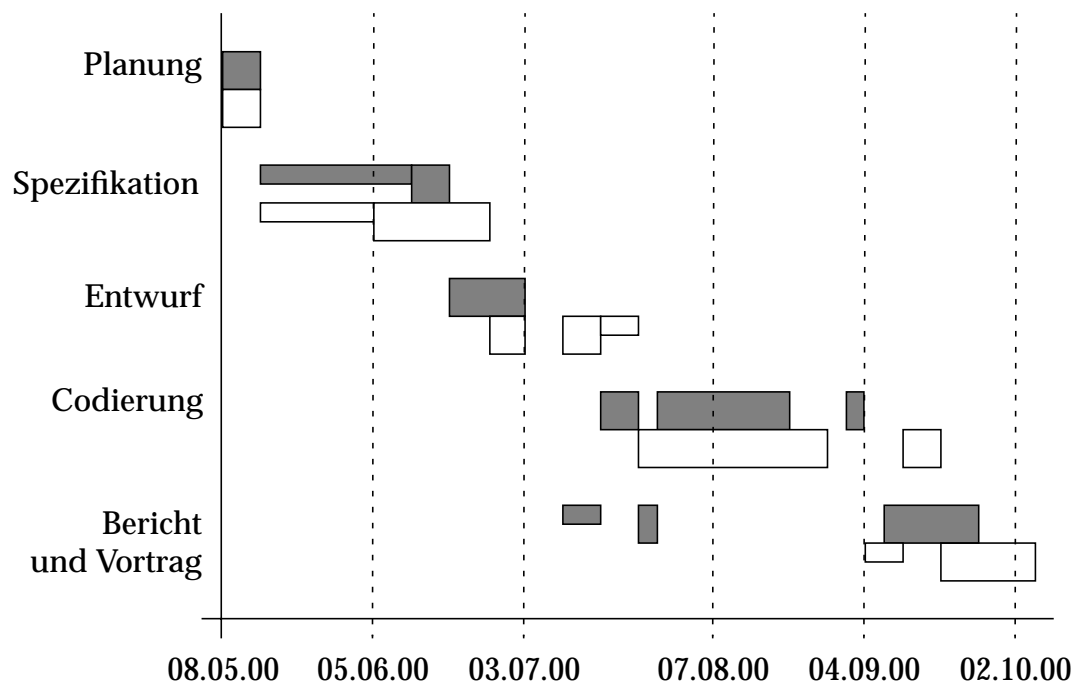
### 6.1 Projektverlauf

Als erste Phase wurde das Projekt geplant. Dabei entstand der Projektplan. Schon während der Planung wurden insgesamt fünf Wochen Puffer eingeplant. Der Puffer war vor allem in der Spezifikationsphase zur Einarbeitung in die Programmiersprachen Ada95 und Tcl/Tk gedacht, zur Korrektur der Dokumente nach Prüfungen und um Verzögerungen während der Implementierung aufzuholen. Der Plan musste während des Projekts mehrmals angepasst werden. Die Tabelle in Abbildung 34 zeigt die Termine der am Beginn des Projekts eingeplanten Meilensteine und die tatsächlich erreichten Termine. Meilensteine wurden erst erreicht, wenn die Korrektur des Dokuments abgeschlossen war.

Meilenstein	Termin Soll	Termin Ist
Projektplan	12.05.2000	12.05.2000
Spezifikation	16.06.2000	23.06.2000
Entwurf	30.06.2000	17.07.2000
Code, Werkzeug	30.08.2000	15.09.2000
Bericht	15.09.2000	05.10.2000

**Abbildung 34: Meilensteine**

Wie leicht zu sehen, hat sich der Ablauf um insgesamt drei Wochen verzögert. Abbildung 35 zeigt den Projektverlauf als Gantt-Diagramm.



Legende:  Soll (eine Woche)  Ist (eine Woche)  Woche mit halber Arbeitszeit

**Abbildung 35: Projektplan**

Die Gründe für die Verschiebung und mehrfache Änderung des Plans sind:

- Nach der Prüfung der Spezifikation durch die Betreuerin war eine umfangreiche Korrektur notwendig. Der Aufwand für die Korrektur wurde vom Bearbeiter bei der Planung unterschätzt.
- Beim Entwurf hat sich gezeigt, dass die Einarbeitung in die für mich neue Programmiersprache Ada 95 nicht ausgereicht hat. Darum wurde die Entwurfsphase unterbrochen, um eine gründlichere Einarbeitung zu ermöglichen.
- Trotzdem hat die Zeit für die Implementierung nicht ausgereicht. Wie in Abschnitt 5.3 beschrieben, konnte die volle Funktionalität nicht implementiert werden. Umfangreiche Tests müssen ebenfalls noch durchgeführt werden.

Die umfangreiche Spezifikationsphase war sicher notwendig, da das Werkzeug dem Benutzer meiner Meinung nach doch eher komplexe Funktionen zur Verfügung stellt. Um diese Funktionalität zu realisieren, müssen die Anforderungen geklärt und verstanden werden.

Während der Implementierung hat sich gezeigt, dass der Entwurf für die in Tcl/Tk realisierten Funktionen des Werkzeugs zu grob war. Dadurch hat sich vermutlich der Aufwand vom Entwurf zur Implementierung verlagert.

## **6.2 Ausblick**

Die fehlende Funktionalität wird im Rahmen einer Hiwi-Tätigkeit vervollständigt. Dabei können dann auch die Anforderungen, die erst nach Abschluss der Spezifikationsphase als nötig erkannt wurden, implementiert werden. Es werden dann ebenfalls Auswertungsschemata für die üblichen Auswertungen implementiert, die dann mit dem Werkzeug ausgeliefert werden können.

Das Werkzeug Sesamscore soll dann bei Schulungen tatsächlich eingesetzt werden, um Tutoren zu unterstützen.

## 7 Begriffslexikon

Das Begriffslexikon erklärt die für das Auswertungswerkzeug grundlegenden Begriffe. Die Begriffe werden alphabetisch sortiert angegeben. Synonyme sind ebenfalls angegeben, wo nötig, wird eine Abgrenzung angegeben. Begriffe aus dem SESAM-Begriffslexikon (Abteilung SE, 1998) sind durch (SESAM) gekennzeichnet.

### Anfrage

Bedingungen, Projektionen, Optionen und Operationen bilden eine Anfrage. Dabei gehört zu einer Bedingung genau eine Projektion. Ergebnisse der Anfragen werden in einer Ergebnistabelle intern gespeichert, bevor sie dargestellt werden.

### Alias

Es gibt zwei Arten von Aliasen im Werkzeug. Mit einem Alias kann der Benutzer einen Wert ersetzen. Aliasnamen werden vom Benutzer für Kommando- und Nachrichtennamen vergeben. Für die Darstellung der Auswertungen kann ein Diagrammalias verwendet werden.

### Aliasnamen

Durch einen Aliasnamen kann ein Kommando- oder Nachrichtename bei einer Auswertung ersetzt werden. Dadurch lassen sich Namen der Kommandos und Nachrichten zusammenfassen oder umbenennen. Aliasnamen werden bei einer Anfrage bei der Eingabe von Bedingungen vom Benutzer angegeben und den Kommando- oder Nachrichtennamen zugeordnet.

**allgemeine Auswertung, synonym andere Auswertung, universelle Auswertung**  
Auswertung als Balken-, Linien- oder Tabellendiagramm

### Arten von Auswertungen

Auswertungen, die eine ähnliche Struktur haben, werden zu einer Art von Auswertung zusammengefasst.

### Auswertung, synonym Auswertungsergebnisse

Eine Auswertung beschreibt einen Teil der Aspekte eines Spielverlaufs durch Untersuchen der Protokolldatei sowie die Darstellung dieser Aspekte. Die Aspekte und ihre Darstellung sind vom Nutzer des Werkzeugs wählbar, jedoch durch die Funktionalität des Werkzeugs begrenzt. Eine Auswertung ist die Anwendung eines Auswertungsschemas auf einen oder mehrere Syntaxbäume. Die Auswertung wird vom Werkzeug graphisch dargestellt, je nach Schema als Tabelle, Balken-, Linien- oder Gantt-Diagramm. Sie kann als Postscriptdatei oder als strukturierter Text gespeichert werden. Zur Auswertung gehört ebenfalls eine mögliche Umsortierung der dargestellten Werte und ein Umbenennen der Beschriftungen durch Diagrammalias. Auswertungen werden durch ihr Schema definiert.

**Auswertungsschema**

Ein Auswertungsschema beschreibt alle Aspekte, die zur Untersuchung von Syntaxbäumen vom Nutzer angegeben werden können. Dazu gehört eine Anfrage und das Layout, das die Darstellung der Ergebnisse beschreibt.

Abgrenzung

Auswertung: Anwendung und Ergebnis eines Auswertungsschemas

Art von Auswertung: Klassifizierung von Auswertungen.

**Balkendiagramm**

Graphische Darstellung einer Auswertung, deren Ergebnisse als Balken in einem Achsenkreuz dargestellt werden.

**Bedingung**

Bedingungen definieren, welche Einträge in einem Syntaxbaum weiterverarbeitet werden. Eine Bedingung besteht aus einzelnen Teilen für den Namen des Kommandos oder der Nachricht, das Datum und die einzelnen Parameter. Ein Eintrag, der diese Bedingung erfüllt, wird durch die zugehörige Projektion in eine interne Ergebnistabelle geschrieben.

**Benutzer, synonym Tutor, Anwender**

Der Tutor ist der Benutzer des Werkzeugs. Das Werkzeug wird für die Analyse von Spielen mit SESAM-2, die im Rahmen einer Schulung durchgeführt werden, vom Tutor der Schulung benutzt. Der Tutor wertet die Spielverläufe, die in der Protokoll-datei aufgezeichnet sind, aus, um den Spielern Rückmeldungen über ihr Vorgehen zu geben.

**Benutzungsoberfläche, synonym Benutzerschnittstelle**

Der Benutzer des Werkzeugs kommuniziert über eine graphische Fensteroberfläche mit dem System. Diese Oberfläche wird als Benutzungsoberfläche bezeichnet.

**Button**

Element der Oberfläche, das der Benutzer durch Anklicken mit der Maus aktivieren kann.

**Diagrammalias**

bezeichnet einen Alias für einen Wert in einem Diagramm. Er wird vom Tutor zur übersichtlicheren Darstellung eingesetzt. Der Benutzer kann durch die Diagrammalias Beschriftungen und (als Text dargestellte) Werte ersetzen oder abkürzen.

**Diagrammsicht, synonym Diagramm**

Ein Fenster, in dem ein Diagramm als Grafik gezeigt wird.

## **Diagrammtabelle**

Die Diagrammtabelle ist die Grundlage der Darstellung als Diagramm oder Vorschau. Alle Daten, die dargestellt werden, sind in der Diagrammtabelle. Die Organisation der Diagrammtabelle entspricht der Organisation der Darstellung.

### **Abgrenzung**

**Ergebnistabelle:** Die Ergebnistabelle wird nur intern verwendet. Aus der Ergebnistabelle entsteht durch Abfragen nach Werten die Diagrammtabelle.

## **Eintrag im Syntaxbaum, synonym Tupel, Eintrag**

Jedes Kommando oder Nachricht, das vom Parser in der Protokolldatei erkannt wurde, bekommt einen Eintrag im Syntaxbaum. Der Eintrag besteht aus dem Namen der Nachricht oder des Kommandos, dem Projektdatum und den verschiedenen Parametern.

### **Abgrenzung**

Eine Zeile in der Ergebnistabelle ist die Projektion eines bestimmten Eintrags im Syntaxbaum.

## **Eintrag in der Protokolldatei**

Ein Eintrag in der Protokolldatei ist ein Kommando, das der Spieler abgesetzt hat oder eine Nachricht, die das System an den Spieler weitergibt oder aber nur zu Protokollzwecken speichert (Tutornachrichten) und ihre Parameter.

### **Abgrenzung**

**Eintrag im Syntaxbaum:** Dieser repräsentiert ein Eintrag in der Protokolldatei im Werkzeug und wird vom System mit dem Datum, an dem die Nachricht gefeuert oder das Kommando abgesetzt wurde, ergänzt.

## **Ergebnistabelle**

Werden Auswertungen gestartet, also ein Schema instanziiert, dann werden die Anfragen auf den Syntaxbaum gestellt. Das Ergebnis dieser Anfragen ist die Ergebnistabelle. Sie besteht aus beliebig vielen Spalten, die mit Buchstaben benannt sind und beliebig vielen Zeilen. Die Ergebnistabelle wird nur zur Auswertung intern vom Werkzeug benutzt.

### **Abgrenzung**

**Auswertung:** Eine Auswertung entsteht durch Anwenden eines Schemas auf einen oder mehrere Syntaxbäume und Darstellen der Ergebnisse als Diagramm. Die Ergebnistabelle ist unabhängig von der Art der Darstellung, sie ist der Teil mit den eigentlichen Daten.

**Diagrammtabelle:** Die Diagrammtabelle liegt der Darstellung der Ergebnisse einer Auswertung zugrunde. Ihre Organisation ist entsprechend auf das Diagramm abgestimmt. Die Diagrammtabelle entsteht aus der Ergebnistabelle während der Auswertung.

**Ganttauswertung**

Auswertung, die als Gantt-Diagramm dargestellt wird.

**Informationsdatei**

Die Informationsdatei dient als Grundlage für Überprüfungen von Auswertungsschemata und Protokolldateien. Sie stellt den Bezug zum Modell her. In der Informationsdatei stehen alle Namen der Nachrichten und Kommandos, die durch das Modell definiert sind.

**Kommando**

Die Form einer Eingabe eines Spielers von SESAM-2, die nach dem Parsen der Eingabezeile durch den Dolmetscher an die Basismaschine des Systems SESAM-2 weitergegeben wird. Das Kommando wird mit allen Parameterwerten in das Protokollfile geschrieben.

**Kopfzeile**

Eine Kopfzeile ist ein Element eines Tabellendiagramms. Es sind drei Kopfzeilen möglich. Die ersten beiden Kopfzeilen enthalten Werte aus Spalten der Ergebnistabelle, die dritte Kopfzeile enthält Spaltennamen, die in der Tabelle dargestellt werden sollen. Durch die Zuordnung der Spalten zu den Kopfzeilen bestimmt der Benutzer die Anordnung in der Tabelle.

**Kopfspalte**

Eine Kopfspalte ist ein Element eines Tabellendiagramms. Pro Tabelle sind drei Kopfspalten möglich. Die drei Kopfspalten können wie die Kopfzeilen belegt werden.

**Layout, synonym Darstellung des Diagramms**

Das Layout beschreibt das Aussehen eines Diagramms und wird vom Benutzer festgelegt.

**Liniendiagramm**

Graphische Darstellung der Ergebnisse einer Auswertung. Die Ergebnisse werden durch Linien in einem Achsenkreuz dargestellt.

**Modell, synonym Simulationsmodell, SESAM-Modell**

Für die Simulation eines Softwareprojekts durch SESAM-2 wird das Verhalten der simulierten Objekte durch ein Modell bestimmt, in dem Beziehungen, Eigenschaften und Interaktion der Objekte definiert sind.

**Nachricht**

Eine Rückmeldung des Systems SESAM-2 an den Spieler, bevor der Dolmetscher die Nachricht übersetzt. Jede Nachricht wird mit allen Parameterwerten in das Protokollfile geschrieben.

**Operation**

Eine Operation ist eine Funktion über eine Spalten der Ergebnistabelle. Es ist Summenbildung und Suche nach Maximum oder Minimum in einer Spalte möglich.

**Option**

Eine Option bildet eine Funktion über eine Ergebnistabelle. Mögliche Optionen sind Distinct zur Duplikateliminierung, Gruppierung nach Werten einer Spalte und Sortierung nach bestimmten Spalten.

**Parameter**

Nachrichten oder Kommandos können mehrere Parameter haben. Ein Parameter enthält einen Wert aus dem Modell von SESAM.

**Parametername, synonym Bezeichner**

Der Name eines Parameters eines Kommandos oder einer Nachricht. Bezeichner werden im Modell vom Modellbauer definiert. Die Namen der Parameter werden nur für Erweiterungen verwendet, um die Konsistenz von Auswertungsschemata zum Modell zu sichern.

**Parser**

Im Werkzeug werden mehrere Parser verwendet. Ein Parser des Werkzeugs verarbeitet die Protokolldatei und füllt mit den in der Protokolldatei gespeicherten Nachrichten und Kommandos den Syntaxbaum. Ein anderer Parser wird benötigt, um die Informationsdatei zu verarbeiten.

**Position eines Parameters**

Ein Kommando oder eine Nachricht kann mehrere Parameter haben. Die Parameter werden immer in einer bestimmten, durch das Modell definierten Reihenfolge ausgegeben. Die Position bestimmt den Platz in dieser Reihenfolge.

**Projektion, synonym Selektion, Ausgabe**

Die Zuordnung der einzelnen Teile eines Eintrags im Syntaxbaum zu einer Spalte einer Ergebnistabelle. Es können Parameter, das Datum oder Namen der Kommandos und Nachrichten einer Spalte in der Ergebnistabelle zugeordnet werden. Diese Spalte wird vom Benutzer mit einem Buchstaben bezeichnet und zur Darstellung benutzt. Jede Projektion ist genau einer Bedingung zugeordnet.

**Protokolldatei, synonym .ist-Datei, Protokollfile**

Eine Datei, die bei einem Spiel mit SESAM-2 erzeugt werden kann. In diesem File werden Nachrichten des Systems SESAM-2 an den Nutzer, für den Tutor (Tutornachrichten) und die Kommandos des Spielers und des Tutors aufgeschrieben. Zusätzlich enthält die Datei Datumsangaben. Alle Einträge in der Datei werden in ihrer zeitlichen Reihenfolge geschrieben.

**Sesamalyzer**

Ein Werkzeug zur Analyse von Spielverläufen. Mit dem Werkzeug kann der Zustand von mit SESAM-2 simulierten Projekten untersucht werden. Es lassen sich beliebige Werte aus dem Situationsmodell und ihre Verläufe über die Projektzeit ausgeben.

**SESAM (SESAM)**

Abkürzung für "Software Engineering Simulation by Animated Models".

**SESAM-2 (SESAM)**

SESAM-System, das als Nachfolger des SESAM-1-Systems konzipiert wurde.

**SESAM-System (SESAM)**

Konkrete Implementierung der Konzepte von SESAM. Umfasst Aspekte der Modellerstellung und Modellanimation sowie der Spielauswertung. Besteht aus einer Menge von Werkzeugen.

**Sesamscore, synonym Werkzeug, Auswertungswerkzeug**

Das im Rahmen der Studienarbeit entstehende Werkzeug zur Auswertung von Spielverläufen, die mit dem Projektsimulator SESAM-2 entstanden sind.

**Screenshot**

Ein Screenshot ist die Abbildung eines Fensters am Bildschirm. Screenshots werden zur Darstellung der Oberfläche des Prototyps in der Spezifikation benutzt.

**Spiel (SESAM)**

Vorgang der Benutzung der Ausführungswerkzeuge durch einen Spieler zum Zwecke der (interaktiven) Ausführung eines SESAM-Modells.

**Spieler (SESAM)**

Benutzer des SESAM-Systems, der den Projektleiter des simulierten Projekts mimit. Er kommuniziert über das Ausführungswerkzeug mit einem SESAM-Modell. Dabei kann er Kommandos an das SESAM-Modell schicken und von ihm Nachrichten empfangen.

**Spielverlauf**

Mit dem Spielverlauf wird beschrieben, wie der Spieler als Projektleiter bei einem Spiel mit SESAM-2 vorgegangen ist, welche Dokumente entstanden sind und welche Ressourcen dabei verbraucht wurden.

**strukturierte Textdatei, synonym tabellarische Speicherung**

Diese Datei enthält die Ergebnisse einer Auswertung in der durch den Benutzer vorgegebenen und bei der Voransicht der Auswertung angegebenen Sortierung. Da der Benutzer Aliase für die Beschriftung angeben kann, sind diese auch in der Textdatei enthalten. Die strukturierte Textdatei kann verwendet werden, um Auswertungen anderen Werkzeugen zugänglich zu machen, denkbar sind Excel, aber auch Darstel-

lungen in Tex. Die Textdatei entspricht der Diagrammtabelle mit Werten, die durch Aliase ersetzt worden sind in der vom Benutzer angegebenen Sortierung.

### **Symboltabelle**

Eine Tabelle, die vom Auswertungswerkzeug intern angelegt wird. Sie enthält die Informationen aus der Informationsdatei. Auf diese Tabelle wird bei Prüfungen, die das Modell des Spiels betreffen, zugegriffen. Es existiert zu einem Zeitpunkt genau eine Symboltabelle.

### **Syntaxbaum**

Die interne Struktur, in der die Information aus der Protokolldatei vom Werkzeug gespeichert wird. Ein Syntaxbaum enthält für eine geladene Protokolldatei alle Nachrichten, Kommandos, Daten und Parameter. Jeder Syntaxbaum wird mit dem Namen der Protokolldatei, die der Syntaxbaum repräsentiert, versehen. Über diesen Namen kann auf den Syntaxbaum zugegriffen werden.

#### **Abgrenzung**

Protokolldatei: Der Syntaxbaum ist die Repräsentation des Werkzeuges für eine Protokolldatei und damit für einen Spielverlauf.

Ergebnistabelle: Der Syntaxbaum bildet die Basis für eine Ergebnistabelle, diese ist das Resultat von Anfragen auf den Syntaxbaum.

### **Tabellendiagramm**

Auswertung, deren Ergebnisse als Tabelle graphisch dargestellt werden.

### **Tutor (SESAM)**

Betreuer (Spielleiter) bei Spielen mit dem SESAM-System. Aufgabe des Tutors ist die Einführung des Spielers in das konkrete SESAM-Modell, die Vorbereitung geeigneter Startsituationen, die Betreuung der Spieler während des Spiels und die Auswertung und Besprechung der Spielergebnisse mit dem Spieler.

### **Tutornachricht, synonym Tutormessage**

Eine Nachricht in der Protokolldatei, die Informationen zu einer bestimmten Phase enthält. Die Tutornachricht wird vom Tutor beim Nachspielen eines Spiels durch ein Kommando erzeugt.

### **Typfehler**

Dieser Fehler tritt auf, wenn bei einer Auswertung der Benutzer den falschen Typ für einen Parameter im Auswertungsschema angegeben hat.

### **Voransicht, synonym Preview**

Eine Darstellung von Auswertungsergebnissen, bei der die Daten für die graphische Darstellung als Text dargestellt werden. Für Tabellen sind dies Kopfzeilen und -Spalten und die Werte der dargestellten Spalten der Ergebnistabelle, bei Linien-, Balken- und Gantt-Diagrammen die Beschriftung der Achsen, der Linien und Balken und die Werte der Balken. Änderungen an der Sortierung und die Eingabe von Aliasnamen

sind in der Voransicht möglich. Nach einer Änderung wird wieder die Voransicht dargestellt. Die Voransicht stellt die Diagrammtabelle dar.

**Warnung**

Eine Warnung an den Benutzer gibt eine kurze Nachricht in einem Fenster an, wenn ein Fehler aufgetreten ist. Der Benutzer muss mit 'OK' das Fenster schließen, damit er mit dem Werkzeug weiterarbeiten kann.

## 8 Literaturverzeichnis

Abteilung SE (1998): **Begriffslexikon für SESAM**. Internes Dokument, Abteilung Software Engineering, Universität Stuttgart.

Abteilung SE (1999a): **Benutzerhandbuch zum Auswertungstool SESAMALYZER für Projektsimulationen in SESAM**. Internes Dokument, Abteilung Software Engineering, Universität Stuttgart.

Abteilung SE (1999b): **Spezifikation für das Produkt SesamIDE. Studienprojekt I, WS 1999/2000**. Internes Dokument, Abteilung Software Engineering, Universität Stuttgart.

Booch, G. (1994): **Objektorientierte Analyse und Design**. Addison Wesley (Bonn)

Drappa, A. (2000): **Quantitative Modellierung von Softwareprojekten**. Dissertation, Shaker Verlag (Aachen).

Jacobson, I. (1995): **Object Oriented Software Engineering - A Use Case driven Approach**. Addison Wesley (Wokingham), ACM.

Ludewig, J. (1994): **SESAM: Software-Engineering-Simulation durch animierte Modelle**. Fakultätsbericht Nr. 05/1994, Fakultät Informatik, Universität Stuttgart.

Ludewig, J. (1998): **Software Engineering. Vorläufiges, unvollständiges Skript zur Vorlesung Software Engineering in der Fakultät Informatik an der Universität Stuttgart**. Skript, Abteilung Software Engineering, Universität Stuttgart.

Mandl-Striegnitz, P. (2000a) in: Dumke, R.; Lehner, F. (Hrsg.): **Software Metriken: Entwicklung, Werkzeuge und Anwendungsverfahren**. Deutscher Universitätsverlag (Wiesbaden).

Mandl-Striegnitz, P. (2000b): **Konzeption und Realisierung eines Auswertungswerkzeugs für SESAM-2**. Internes Dokument, Abteilung Software Engineering, Universität Stuttgart.

Mandl-Striegnitz, P. (2000c): **Konzeption und Realisierung eines Auswertungswerkzeugs für SESAM-2 - Ideen und Anforderungen**. Internes Dokument, Abteilung Software Engineering, Universität Stuttgart.

Notter, A. (1999): **Eine Untersuchung zur Wirksamkeit der Projektmanagement-Ausbildung am Simulator**. Diplomarbeit, Abteilung Software Engineering, Universität Stuttgart.

- Reißing, R. (1996): **Konzeption und Realisierung einer Basismaschine für SESAM-2.** Diplomarbeit, Abteilung Software Engineering, Universität Stuttgart.
- Schneider, B. (1996): **Entwicklung einer Modellbeschreibungssprache für SESAM-2.** Diplomarbeit, Abteilung Software Engineering, Universität Stuttgart.
- Schneider, K. (1994): **Ausführbare Modelle der Software-Entwicklung: Struktur und Realisierung eines Simulationssystems.** vdf Hochschulverlag (Zürich).

## **Erklärung**

Hiermit versichere ich, diese Arbeit selbständig verfasst  
und nur die angegebenen Quellen benutzt zu haben.

---