

Modellbasiertes Hilfesystem für AutoFocus II

Sabine Wildgruber

02. Dezember 2002

Inhaltsverzeichnis

1	Einführung	5
1.1	AutoFOCUS2/Quest	5
1.2	Aufgabenstellung	5
1.2.1	Benutzerschnittstelle	6
1.2.2	Modellbasierung	6
1.2.3	Wunschkriterien	7
1.3	Projektdurchführung	7
2	Benutzerunterstützungssystemen	8
2.1	Rechnergestützte Systeme: Benutzer-Handbücher	8
2.2	Rechnergestützte Benutzerunterstützungssysteme	8
2.2.1	Vor- und Nachteile gegenüber Handbüchern	8
2.2.2	Klassifizierungsmerkmale	10
3	Analyse des bisherigen Hilfesystems	13
3.1	Leistungsumfang	13
3.1.1	Online-Tutor-System "Nelli"	13
3.1.2	Online-Hilfesystem (statisch, uniform, passiv)	14
4	Konzeption des Hilfesystems	16
4.1	Überblick der Anforderungen	16
4.2	Umfang des Hilfesystems	17
4.2.1	"Overview" (Allgemeine Hilfe)	19
4.2.2	"for this Screen" (Hilfe zu Browser/Editor)	20
4.2.3	"What is .." Direkthilfe	20
4.2.4	Training (Nelli)	20
4.2.5	Aktive dynamische Hilfe für Funktionsbuttons (Tooltips)	21
4.2.6	"About" (Information)	21
4.3	Zugang zum Hilfesystem	21
4.3.1	Zugang über den Menüpunkt "Hilfe"	22

- 4.3.2 Zugang über die Funktionstaste “F1“ 22
- 4.3.3 Optional: Zugang über Tooltips zur Hilfe 22
- 4.3.4 Sonstige Zugangsoptionen 23

5 Design 24

- 5.1 Überblick 24
- 5.2 Randbedingungen 25
 - 5.2.1 Konzeptberücksichtigung 25
 - 5.2.2 Editoren und Browser 25
 - 5.2.3 Erstellung eines Helpsets 26
- 5.3 Design-Basis-Entscheidungen 26
 - 5.3.1 Einsatz von Hypertextsystemen 26
 - 5.3.2 Festlegung der Granularität des Systems 26
- 5.4 Einsatz der Hilfetechnologie JavaHelp 27
- 5.5 Realisierung der kontextsensitiven Hilfe 28
 - 5.5.1 Feststellung der Art der Kontextsensitivität (Zugangsarten zum Hilfesystem) 30
 - 5.5.2 Feststellung des Kontexts 30
 - 5.5.3 Zuordnung der Hilfeinformation zum jeweiligen Kontext 32
- 5.6 Identifikation von Objekten der Modellierungstechnologie 34
- 5.7 Funktionsweise des Systems JavaHelp 35
 - 5.7.1 Das HelpSet und die Organisation 35
 - 5.7.2 Grundlagen XML 37
 - 5.7.3 Funktionalitäten im Hilfefenster 38
- 5.8 Standard-HelpSet: Die Hilfeinformationsverwaltung in AutoFOCUS 39
 - 5.8.1 Verzeichnisstruktur 39
 - 5.8.2 HelpSet- und Navigationsdateien 39
 - 5.8.3 Erweiterungen des Standard-HelpSets 40
 - 5.8.4 Erstellen von Hilfeinformationsdateien 40
- 5.9 Das Hilfesystem aus Systemsicht 41

<i>INHALTSVERZEICHNIS</i>	4
6 Implementierung des Prototyps	42
6.1 Programm-Bestandteile des Hilfesystems	42
6.1.1 Klassen	42
6.1.2 Aufrufende Klassen	44
6.2 Aufbau des Menüs "Hilfe" in Editor/Browser	44
6.3 Realisierung der "What is ..."-Hilfe (Direkthilfe)	45
6.3.1 Menüleisten und Funktionsbuttons	46
6.3.2 Browser: Objekte in Navigationsstruktur	47
6.3.3 Editor: Objekte in Editorfläche	50
6.3.4 Menüleiste	52
6.3.5 Toolbar	52
6.4 weitere Implementierungen	52
6.4.1 Realisierung der "for this Screen ..."-Hilfe (Editor/Browser-Hilfe)	52
6.4.2 Realisierung des "Overview"s	52
6.5 Erweiterungen am Hilfesystem	53
7 Zusammenfassung und Ausblick	55
7.1 Zusammenfassung	55
7.2 Ausblick	56
7.2.1 Aktives Hilfesystem	56
7.2.2 Prozessunterstützung	57
7.2.3 Inspektion von Objekten	58
7.2.4 Wartung und Erweiterung	58
7.2.5 Automatische Generierung	59
7.2.6 Hilfesystem für Entwickler/innen	59
A Pflichtenheft	61
B Programmcode	66
C Literaturverzeichnis	79

1 Einführung

1.1 AutoFOCUS2/Quest

AutoFOCUS2/Quest, im folgenden AutoFOCUS genannt, ist ein Werkzeugprototyp, der auf formalen Methoden der Systementwicklung basiert, und als Evaluationsmittel für weitere Werkzeugkonzepte zur Spezifikation und Entwicklung verteilter Systeme dienen soll. Mit seiner Ausrichtung auf grafische Darstellungsmittel soll AutoFOCUS ein möglichst intuitives Arbeiten ermöglichen. Dem Anwender stehen dabei als Arbeitsmittel ein Projektbrowser zur Verwaltung der Spezifikationsdokumente verschiedener Projekte sowie verschiedene Editoren für die einzelnen Spezifikationsdokumente zur Verfügung.

1.2 Aufgabenstellung

Im Rahmen des Systementwicklungsprojekts ist ein Hilfesystem für das Modellierungswerkzeug AutoFOCUS zu konzipieren und in Teilen zu realisieren. Die Konzipierung des Hilfesystems bezieht sich hierbei in erster Linie auf die Definition der Benutzerschnittstelle, d.h. einen geeigneten Benutzerführungsmechanismus für ein modellbasiertes Hilfesystem.

Es ist Gegenstand der Aufgabe sowohl auf das von Quest eingesetzte Metamodell einzugehen, indem zu für den Benutzer sichtbaren Elementen Hilfe angeboten wird, als auch die Integration verschiedener Zugriffsmechanismen auf das Hilfesystem, wie auch die Erweiterung des Hilfesystems in Form von kontextsensitiver Hilfe vorzunehmen. Kontextsensitive Hilfe kann hierbei in verschiedenen Granularitätsstufen verstanden werden. Fokus dieses Projekts ist die Realisierung der Kontextsensitivität geknüpft an Objekte der graphischen Benutzeroberfläche wie Editor oder Fenster (geringe Granularität der kontextsensitiven Hilfe), Eingabeframes oder Funktionsbuttons.

Die für ein Hilfesystem notwendigen Rahmenbedingungen zur Dokumentenorganisation, -struktur sowie -erstellung wird in der Konzeption des Systems berücksichtigt.

Die Aufgabenstellung kann grob skizziert dem im Anhang befindliche Pflichtenheft in Kapitel "Pflichtenheft" aus dem Anhang entnommen werden. Eine detailliertere Beschreibung der Aufgaben und der Realisierungsmöglichkeiten erfolgt in den nachfolgenden Kapiteln.

Im Folgenden wird auf die Aufgabenstellung näher eingegangen. Die Bewältigung der Aufgaben wird im Anschluss im unteren Abschnitt 1.3 dieses Kapitels

beschrieben.

1.2.1 Benutzerschnittstelle

Die Benutzerschnittstelle für das Hilfesystems ist für die Unterstützung des Benutzers zur Bedienung des Hilfesystems zuständig. Innerhalb der Schnittstelle ist der Benutzerführungsmechanismus definiert.

Der Benutzer soll durch die zu definierende Schnittstelle in die Lage versetzt werden, selbständig seine Fragen zur Bedienung der Anwendung AutoOCUS beantworten können. Die Bedienung der Anwendung bedeutet in diesem Zusammenhang nicht die korrekte und erfolgreiche Modellierung eines Systems, sondern den Umgang beziehungsweise die Bedienung des Werkzeug zur Durchführung einer Modellierung.

Eine angemessene Unterstützung durch das Hilfesystem berücksichtigt hierbei vor allem die folgenden Punkte: vgl.[BAL98]

- Im aktuellen Kontext wählbare Objekte, Funktionen bzw. Kommandos, sowie die Optionen zu Objekten und Funktionen.
- (Bedeutung der Funktionstasten, insbesondere von mehrfach belegten Funktionstasten und Mausknöpfen)

Tooltips Eine weitere Form der Benutzerunterstützung ist die Unterlegung von Funktionssymbolen innerhalb der Anwendung von kurzen Hilfstexten, sog. "Tooltips", die in dem Moment erscheinen, wenn der Mauszeiger z.B. das Funktionssymbol mit ihrer Spitze berührt.

1.2.2 Modellbasierung

Es wird angestrebt, das Hilfesystem basierend auf einem zu entwickelnden Datenmodell zu konzipieren. Prototypisch erfolgt eine Sicht auf das Datenmodell des Hilfesystems durch Einsatz der Softwarelösung JavaHelp.

Die bei der Konzipierung des modellbasierten Hilfesystems hilfreichen und vor allem notwendigen Anforderungen an ein Editorenframework sind gegebenenfalls geeignet zu definieren oder festzulegen. Einige Ausführungen finden sich dazu im Kapitel 4. Im Kapitel 5 finden sich die Anforderungen zum Teil als Randbedingungen für die Realisierung in der prototypischen Realisierung wieder.

1.2.3 Wunschkriterien

Die gemäß Pflichtenheft im Anhang A aufgezeigten Wunschkriterien sind nicht Gegenstand der Aufgabenstellung. Laut Balzert (vgl.[BAL98]) sollten diese jedoch bei der Optimierung des Systems mit höherer Priorität realisiert werden, da sie als essentielle Bestandteile einer professionellen Anwendung gelten und somit durch den Benutzer erwartet werden.

Zu den Wunschkriterien gehören unter anderem:

- Unterstützung bei Eingabedialogen
- Spezifische Fehlermeldungen und Fehlererklärungen, ggf. mit Angabe einer Methode zur Fehlerkorrektur
- Erläuterungen zu Ergebnissen von Funktionsausführungen

1.3 Projektdurchführung

Parallel zur Erstellung der Konzeption des modellbasierten Hilfesystems für Auto-FOCUS erfolgt eine prototypische Implementierung, anhand derer Teile der Konzeption erläutert werden sollen. Die Implementierung erfolgt mit Hilfe der auf dem Markt frei erhältlichen Softwarelösung JavaHelp (vgl. [JAV02]).

2 Benutzerunterstützungssystemen

2.1 Rechnerungestützte Systeme: Benutzer-Handbücher

Zum Erlernen und Verstehen der Benutzeroberfläche und der Funktionalität der Anwendung sind Handbücher hilfreich. Sie dienen als Nachschlagewerk. Es gibt diverse Vor- und Nachteile für Handbücher, von denen im Folgenden einige aufgezählt werden.

Vorteile	Nachteile
Sind universell geschrieben	berücksichtigen weder die persönliche Benutzersituation noch die individuelle Anwendungssituation
Nutzer kann unmittelbar an seine Lesegewohnheiten und seine geübten Methoden des Lernens anknüpfen	Oft sehr umfangreich, schwer verständlich, nicht oder schlecht übersetzt und abstrakt formuliert
Schnelles, wenig aufwändiges Blättern möglich (zur bekannten Seite zurückkehren, Überblick verschaffen)	Entsprechen häufig nicht der aktuellen Programmversion oder sind nicht verfügbar
Kann aufgeschlagen liegen bleiben, Lesezeichen können in Buch gelegt werden, Notizen machen ist möglich, Textstellen markieren	Man muss oft viel lesen, bevor man den Funktionsumfang abschätzen, auf die eigene Situation anwenden und praktisch mit dem Software-System arbeiten kann.
Ohne technische Hilfsmittel nutzbar	Beim Auftreten von Fragen sind die Antworten meist sehr allgemein, schwer zu finden und unbefriedigend.

Handbücher sind bei der Lieferung eines professionellen Softwaresystems eingeschlossen und können in Trainings-Handbuch, Benutzer-Leitfaden, Referenz-Handbuch sowie Referenzkarte klassifiziert werden. Der Abbildung 1 können Sie die Klassifizierungen entnehmen.

2.2 Rechnergestützte Benutzerunterstützungssysteme

2.2.1 Vor- und Nachteile gegenüber Handbüchern

Um die Nachteile von Handbüchern zu vermeiden, werden rechnergestützte Benutzer-Unterstützungssysteme eingesetzt. Die Systeme lassen sich in die verschiedenen Kategorien einteilen, wie in Abbildung 2 zu erkennen:

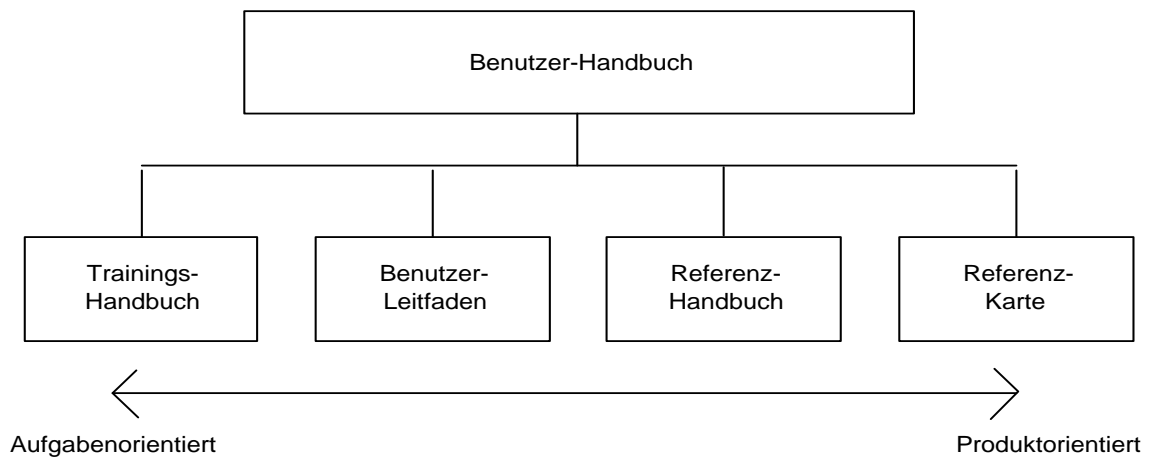


Abbildung 1: Klassifizierungen von Handbüchern

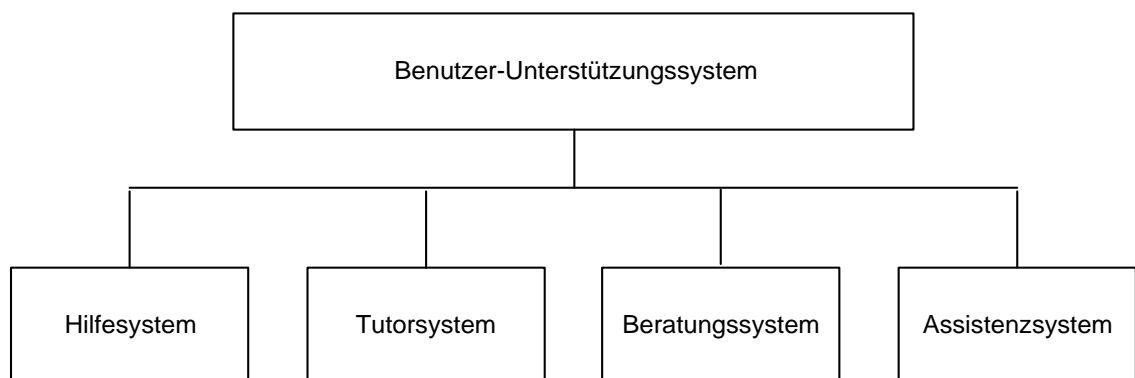


Abbildung 2: Klassifizierungen von rechnergestützten Benutzerunterstützungssystemen

Computergestützte Systeme besitzen gegenüber Handbüchern folgende Vor- und Nachteile:

Vorteile	Nachteile
Interaktive und multimediale Unterweisung ist möglich	Text wird auf Bildschirm meistens langsamer gelesen, als auf gedrucktem Material.
Leicht und schnell zu aktualisieren	Lesen auf Bildschirm wirkt ermüdend
Potentiell kann schneller zugegriffen werden und schneller nachgeschlagen werden Blättern auf Bildschirm ist langsamer beschädigt werden	Kleine Bildschirme können keine ganzen Seiten anzeigen heightKönnen nicht verloren gehen,
Ein gutes System erlaubt dem Benutzer, den Detaillierungsgrad der Information zu steuern	Navigieren erfordert zusätzlichen Lernaufwand
	Teil des Bildschirms wird für die Anzeige von Information benötigt

2.2.2 Klassifizierungsmerkmale

Rechnergestützte Systeme können anhand verschiedener Merkmale klassifiziert werden. Im Folgenden werden die sechs Hauptmerkmale von Hilfesystemen näher erläutert.

passive und aktive Hilfe Bei dem zu realisierenden Hilfesystem wird ein passives System konzipiert, welches sich "passiv" dem Benutzer gegenüber verhält, solange der Benutzer keine Hilfe anfordert.

(Im Gegensatz zu einem **passiven System** könnte auch ein "**aktives**" **Hilfesystem** konzipiert werden, welches dem Benutzer seine Hilfe "aktiv" anbietet, indem es das Verhalten des Benutzers im Laufe der Benutzung der Anwendung beobachtet und anschließend analysiert.)

Das Hilfesystem erwartet bei einem passiven System demnach, dass der Benutzer von sich aus aktiv wird und Hilfe anfordert, z.B. durch Drücken einer "Hilfeg-Taste" (F1, help) oder Menüoptionen. Das Hilfesystem verhält sich selbst passiv. Der Benutzer kann seine Anfrage auf unterschiedliche Weise stellen:

- Direkter Zugriff, z.B. durch Eingabe des Kommandonamens

- Spezifikation durch Eingabe von Schlüsselwörtern
- Navigation durch Informationsnetze, z.B. Hypertextsysteme
- Anfragen in natürlicher Sprache

Beispiele:

Anfragen in natürlicher Sprache können auf Was, Warum, Wie-Fragen beschränkt werden, die dem Benutzer aber bereits eine Vielzahl von Möglichkeiten zur Hilfe anbieten.

Ein Benutzer ist immer dann in der Lage passiv Hilfe anzufordern, wenn er selbst Probleme sieht, bei denen er Unterstützung benötigt, oder wenn er nach bestimmten Informationen, Funktionen oder Kommandos sucht.

Probleme: vgl. [BAL98]

- Ca. 40/100 der vom System angebotenen Funktionalitäten werden vom Benutzer überhaupt benutzt.*
- Oft sind Konzepte realisiert, die der Benutzer nicht vermutet.*
- Der Benutzer sucht vergeblich nach Funktionalitäten, die jedoch nicht im System realisiert sind.*

statische und dynamische Hilfe Das Hilfesystem sollte sowohl über statische Hilfe als auch über dynamische Hilfe verfügen. **Statische Hilfe** bedeutet, dass die angebotene Hilfe Informationen beinhaltet, die unabhängig vom aktuellen Kontext des Benutzers gegeben wird.

Unter **dynamischer Hilfe** wird im Gegensatz dazu Information verstanden, die dem Benutzer zur Verfügung gestellt wird, unter Berücksichtigung des aktuellen Kontexts. Es wird im Zusammenhang mit dynamischer Hilfe auch von "kontextsensitiver Hilfe" gesprochen.

Beispiele:

- *Wird aus dem "Projekt-Browser", einem speziellen Fenster der Anwendung, heraus durch den Benutzer Hilfe angefordert, so kann es sich hierbei entweder um einen Überblick in Form eines Gesamtinhaltsverzeichnisses zum Hilfesystem handeln (statische Hilfe) oder um Hilfeinformation zum speziellen Ausschnitt des Systems, wobei die Information sich in diesem Beispiel auf den "Projekt-Browser" beziehen (dynamische Hilfe).*
- *Ein Benutzer bekommt bei unterschiedlichen Eingabefeldern z.B. nur die Meldung "unzulässige Eingabe" angezeigt. (statische Hilfe)*

- *Ein Benutzer bekommt bei unterschiedlichen Eingabefeldern angezeigt, worin der Fehler im konkreten Fall besteht, z.B. "Wert zu klein - wählen Sie einen Wert zwischen 20 und 100" oder noch besser, wenn die Eingabe des Benutzer zur Erklärung benutzt wird, z.B. "Wert 0,005 ist zu klein, wählen Sie einen Wert zwischen 20 und 100." z.B. nur die Meldung "unzulässige Eingabe" angezeigt.*
- *Editor-/Browsereigenschaften werden durch unterschiedliche Antworten wiedergegeben. Bei unterschiedlichen Eingabefeldern unterschiedliche Antworten/Erklärungen. (dynamische Hilfe)*
- *Funktions-Buttons werden mit Erklärungen versehen, wenn man mit der Maus auf den Button kommt. (dynamische Hilfe)*

uniforme und individuelle Hilfe Während uniforme Hilfe nicht auf die Bedürfnisse einzelner Benutzergruppen eingeht, wird dieses durch individuelle Hilfe bis zu einem vorab zu definierenden Grad realisiert. Inhalt dieses Projekts ist die Realisierung eines zweistufigen und damit individuellen Hilfesystems, welches die Benutzergruppen Anfänger und Geübte berücksichtigt.

Hilfesysteme *Beispiel: "Ein einfacher Fall eines Hilfesystems"*
Nach Drücken der Hilfe-Taste erscheint das entsprechende Kapitel aus dem Benutzerhandbuch auf dem Bildschirm.

Weitergehende Hilfesysteme unterstützen jeden Benutzer individuell und bieten zum Teil auch Hilfe von sich aus an. Hierzu gehören auch Tutorssysteme, Beratungssysteme sowie Assistenzsysteme.

3 Analyse des bisherigen Hilfesystems

3.1 Leistungsumfang

Das bisherige Hilfesystem in AutoFOCUS umfasst das in die Anwendung eingebundene Online-Hilfesystem und das selbstständig implementierte Online-Tutor-System "Nelli", welches über einen Browser zur Verfügung gestellt wird.

Die beiden Arten der Benutzerunterstützungssysteme sind unabhängig von einander nutzbar.

3.1.1 Online-Tutor-System "Nelli"

Der Umfang des Online-Tutor-Systems Nelli beschreibt alle Funktionalitäten, die mit AutoFOCUS zur Verfügung stehen. Der Benutzer kann durch das Tutor-System sowohl eine produktbezogene Einführung in die in AutoFOCUS enthaltenen Funktionalitäten bekommen, als auch eine aufgabenbezogene Einführung in die Modellierung (komplexer) Systeme. Das Tutor-System bietet dem Benutzer hierfür ein durchgängiges Beispiel indem eine Ampelsteuerung modelliert werden soll.

Vorteile	Nachteile
Das Tutor-System ist unabhängig vom Produkt in einem HTML-Viewer (Browser) lesbar.	Man benötigt einen HTML-Viewer (Browser), um Tutor-System lesen und nutzen zu können.
Unabhängigkeit vom Hilfesystem. Ein Neueinsteiger kann dann zuerst das Tutor-System verwenden und später mit dem Hilfe-System in AutoFOCUS arbeiten.	Das Tutor-System ist nicht in das Hilfesystem in AutoFOCUS integriert. Falls jemand zuerst mit dem Tutor-System gearbeitet hat, muss er sich im Hilfe-System neu orientieren.
Ausführliche Erläuterungen in Fließtext untermalt durch Produkt-Schnappschüsse (Bilder) helfen einem Neueinsteiger das Produkt AutoFOCUS zu verstehen	Bei einem höheren Kenntnisstand des Nutzers zum Produkt AutoFOCUS, sind ausführliche Erläuterungen "zu viel".
	Das Tutor-System ist nicht interaktiv nutzbar: * Kein Hinzufügen von Anmerkungen möglich. * Setzen von Lesezeichen wird nicht unterstützt.
	Struktur des Tutor-Systems muss selbständig durch den Benutzer erkannt werden, bevor man effizient damit arbeiten kann.

3.1.2 Online-Hilfesystem (statisch, uniform, passiv)

Die Version 0.6.0 von AutoFOCUS beinhaltet ein passives Online-Hilfesystem. Im Online-Hilfesystem sind alle Funktionalitäten des Produktes beschrieben, allerdings

Vorteile	Nachteile
Bei Hilfeanforderungen während der Arbeit mit AutoFOCUS werden keine Handbücher oder HTML-Viewer (Browser)	Contents und Browser beinhalten die gleichen Hilfe-Dokumente.
Beim Aufruf des Online-Hilfesystem wird ein neues Fenster geöffnet, welches das Hilfesystem enthält.	
Der Benutzer hat ein Inhaltsverzeichnis, indem alle relevanten Themengebiete aufgelistet sind.	Der Benutzer hat keine Möglichkeit per Stichworteingabe, das für ihn relevante Themengebiet zu ermitteln.
Das Hilfesystem beschreibt textuell alle produktbezogenen Funktionalitäten	Der Benutzer bekommt keine Unterstützung durch Symbole o.ä., welche ihm schneller das Thema zeigen könnten, das er gerade sucht.
	Das Hilfesystem ist nicht interaktiv nutzbar: * Kein Hinzufügen von Anmerkungen möglich * Setzen von Lesezeichen wird nicht unterstützt.
Das Hilfesystem beinhaltet ein ausführliches themenbezogenes Hilfesystem.	Fragen in natürlicher Sprache, in denen nach Stichworten gefiltert wird, sind nicht möglich.
	Das Hilfesystem unterstützt keine stichwortartige Suchfunktion.
Die Verfügbarkeit vom Hilfesystem wird sichtbar in der Menüleiste im jeweiligen AutoFOCUS-Editor angeboten.	Es gibt ausschließlich eine Zugriffsmöglichkeit durch Klicken der Hilfe in der Menüleiste im AutoFOCUS-Editor.
Es gibt die Möglichkeit in speziellen Editoren (EET, STD, etc.) spezielle Hilfe zu diesen Editoren zu bekommen.	In der Hilfe für spezielle Editoren, wird das universelle und statische Hilfesystem für das Gesamtsystem AutoFOCUS angeboten. Hierbei handelt es sich um eine Irreführung des Benutzers.

4 Konzeption des Hilfesystems

Im Folgenden wird das theoretische Konzept für das modellbasierte Hilfesystem erstellt.

Das Konzept ist in Teilen in Form einer prototypischen Implementierung realisiert und wird im anschließenden Kapitel 6 "Implementierung des Prototyps" näher erläutert.

Die Konzeption des Systems unterliegt verschiedenen Anforderungen, die erreicht werden sollen, um das Ziel der Entwicklung eines professionellen Benutzerunterstützungssystems realisieren zu können. Der nachfolgende Abschnitt 4.1 gibt einen Überblick über diese Anforderungen.

4.1 Überblick der Anforderungen

– **Konsistenz über das gesamte System**

Es ist wichtig, dass das Hilfesystem vollständig in AutoFOCUS integriert ist. Das bedeutet insbesondere, dass dem Benutzer zu jedem Zeitpunkt bei der Arbeit mit AutoFOCUS eine Zugangsmöglichkeit zum Hilfesystem angeboten wird.

– **Intuitive Benutzerführung**

Benutzer von Anwendungssoftware sind einen gewissen Standard an Hilfe gewohnt, der auch in AutoFOCUS wieder zu finden sein sollte. Hierzu gehört beispielsweise ein Menüpunkt "Hilfe", ein gleichnamiger Button oder das (berühmte) "?", welche auch als "What is ..."- oder "Direkthilfe" bekannt ist. Eindeutige Bezeichnungen und Erklärungen sind ebenfalls wichtig, damit der Benutzer das Hilfesystem anerkennt und einsetzt.

– **Site-Map**

Dem Benutzer sollte eine Übersicht zur Verfügung gestellt werden, aus der er entnehmen kann, was AutoFOCUS seinem Benutzer zur Unterstützung anbietet.

– **Berücksichtigung der Informationsbedürfnisse der Benutzer (universell - individuell)**

Grundsätzlich sollte das Hilfesystem individuell für Benutzergruppen gehalten werden, indem beispielsweise zwei verschiedene Detaillierungsmöglichkeiten angeboten werden. Der erfahrene Benutzer kann hier auf Kurzinformationen zurückgreifen und ein ungeübter Benutzer hat zusätzlich die Möglichkeit, Detailinformationen anzufordern.

– **Anforderung seitens des Benutzers
(passiv - aktiv)**

Das Hilfesystem in AutoFOCUS kann durch den Benutzer aktiviert werden ist aber von sich aus im Allgemeinen nicht aktiv. (Eine Ausnahme stellt die aktive, dynamische Hilfe für die Menüauswahlpunkte oder Funktionsbuttons dar, die so genannten Tooltips“.)

– **Anpassung der Hilfeinformation an den aktuellen Kontext
(dynamisch - statisch)**

Dynamische (kontextsensitive) Hilfeinformation kann der Benutzer zu den verschiedenen Editoren/Browser , wie auch zu deren Elementen bekommen. Der Kontext bezieht sich dann entweder als Überblick zu den verschiedenen Editoren/Browsern oder speziell auf deren angebotenen Funktionsbuttons, Menüleisteneinträgen oder Inhalten.

Der Benutzer hat zusätzlich die Möglichkeit aus der angebotenen Hilfe heraus zu beliebigen anderen Hilfethemen zu gehen. Da diese Hilfeinformation dann gegebenenfalls nicht mehr kontextsensitiv ist, spricht man von statischer Hilfe.

Der Zugang zur Übersicht der angebotenen Themen im Hilfesystem, wird aus allen Editoren/Browsern angeboten.

4.2 Umfang des Hilfesystems

Das Hilfesystem besteht aus verschiedenen Komponenten, wie Direkthilfe, allgemeiner Hilfe, Editor/Browser- bezogener Hilfe, dem Online-Tutorsystem oder der Information. Während die Grundfunktion eines Hilfesystems die ist, dem Benutzer Information für einen komfortablen Umgang mit AutoFOCUS zur Verfügung zu stellen, ist lediglich die Komponente “Information“ unabhängig von der Arbeit mit AutoFOCUS einzuordnen.

Das Hilfesystem für AutoFOCUS verfolgt das Ziel die im vorherigen Abschnitt genannten Anforderungen zu erfüllen und dem Benutzer somit ein geschlossenes System in AutoFOCUS zur Unterstützung anzubieten.

Prototypische Realisierung Der Umfang des Hilfesystems wird prototypisch in Teilen in diesem Projekt mit Hilfe der Softwarelösung JavaHelp realisiert und in der Programmiersprache Java implementiert.

Durch JavaHelp bekommt der Benutzer die Möglichkeit ein funktionsfähiges Hilfesystem integriert in AutoFOCUS zu erleben, welches dem Standard einer professionellen Anwendungssoftware entspricht.

Dieses ist beispielsweise daran erkennbar, dass Hilfeinformationen zur Anwendung grundsätzlich mit Hilfe eines *“Hilfefensters“* zur Verfügung gestellt werden. Das Hilfefenster ist der Einstieg in die Hilfs informationstexte.

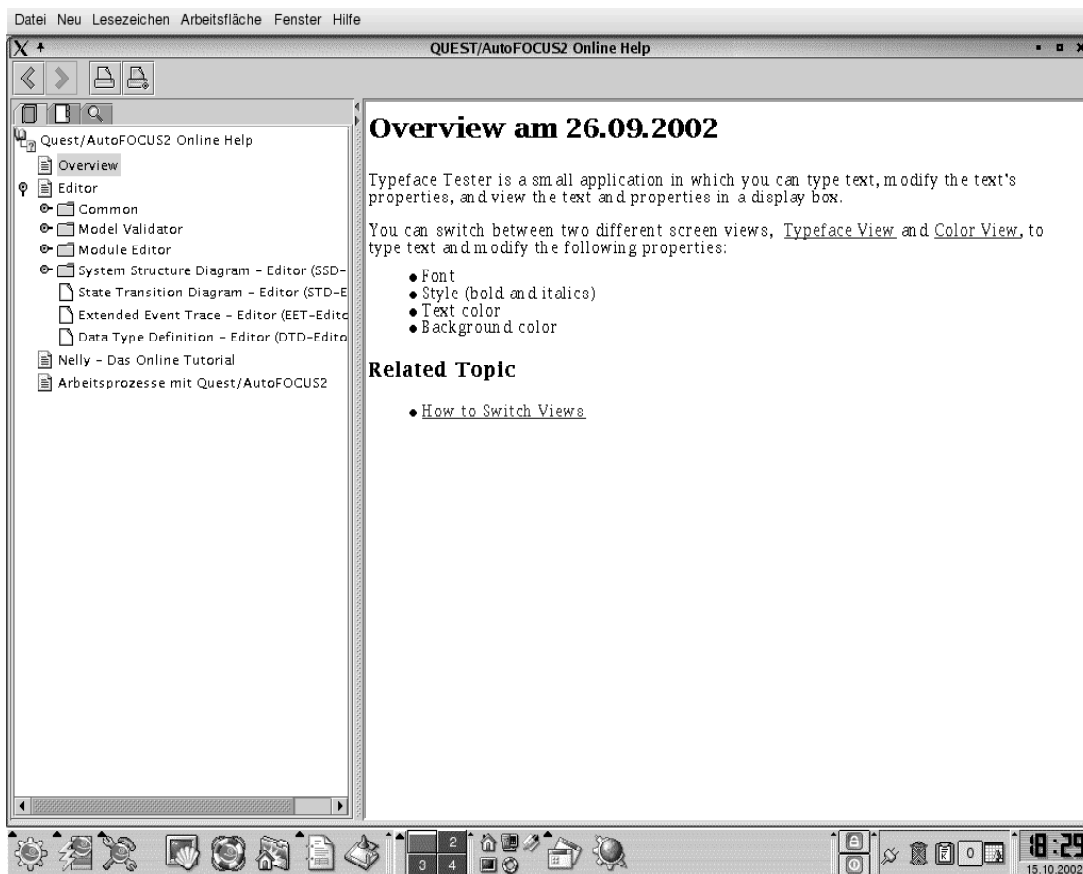


Abbildung 3: Das Hilfeinformationsfenster

Der obigen Abbildung 7 kann entnommen werden, dass das Hilfefenster unter anderem aus einem vertikal geteilten Sichtfenster besteht, in dem auf der linken Seite beim Start ein Themenüberblick gegeben wird und auf der rechten Seite der Hilfeinformationstext zu dem aktivierten Hilfethema. Weitere Informationen zum Hilfefenster folgen im nachfolgenden Abschnitt. Der Zugang zum Hilfefenster und damit auch zum Hilfesystem kann grundsätzlich auf verschiedene Weisen erfolgen, die im nachfolgenden Abschnitt

4.3 beschrieben werden.

Im realisierten Prototyp erfolgt der Zugang grundsätzlich über die Menübar.

Anzeigen von Hilfeinformation: Das Hilfefenster

Drei Sichten im Hilfefenster Die Anordnung des Hilfefensters beinhaltet die Verwaltung dreier Registerkarten, wobei diese verschiedenen Sichten auf die im Hilfesystem enthaltene Information ermöglichen. Die verschiedenen Sichten auf Hilfeinformationen werden im Folgenden kurz vorgestellt.

Themenüberblick (Table of Contents) Der Themenüberblick entspricht einem Inhaltsverzeichnis, durch das der Benutzer einen groben Überblick über die im Hilfesystem von AutoFOCUS angebotenen Themen bekommt. Durch Klicken verschiedener Hauptthemen auf der linken Seite des Hilfefensters verändert sich die auf der rechten Seite angezeigte Hilfeinformation.

Volltextsuche (Word Search) Mit Hilfe der Volltextsuche hat der Benutzer die Möglichkeit nach speziellen Schlagworten im vorhandenen Hilfesystem Themenbereiche angezeigt zu bekommen. Der Benutzer kann sich nach Anzeige der Themengebiete das für ihn relevante Themengebiet anzeigen lassen und seine Hilfeinformationsbedürfnisse an dieser Stelle befriedigen.

Indexierung (Index) Die Indexierung im Hilfesystem bietet dem Benutzer die Möglichkeit nach bekannten Schlagwörtern zu suchen und auf diese Weise Synonyme zu seinen Worten zu finden, die im Hilfesystem verwendet wurden, für die weitere Arbeit mit AutoFOCUS zu verwenden.

4.2.1 “Overview“ (Allgemeine Hilfe)

Den Menüpunkt “Hilfe → Overview“ ermöglicht dem Benutzer einen Überblick über die im Hilfesystem angebotenen Informationen zu bekommen. Über den Menüpunkt öffnet sich das in der Abbildung 7 dargestellte Hilfefenster und unterstützt den Benutzer beim Umgang mit der Hilfe in AutoFOCUS. Während auf der rechten Seite des Hilfefensters einleitende Worte

geschrieben stehen sollten, befindet sich auf der linken Seite ein Überblick in Form einer Inhaltsübersicht mit der Möglichkeit des Navigierens in den angebotenen Informationen.

4.2.2 “for this Screen“ (Hilfe zu Browser/Editor)

Den Menüpunkt “Hilfe -> for this Screen“ ermöglicht dem Benutzer einen Überblick über die im gerade geöffneten Editor angebotenen Informationen zu bekommen. Über den Menüpunkt öffnet sich das jeweilige Kapitel zu dem geöffneten Editor/Browser im Hilfefenster und unterstützt den Benutzer beim Umgang mit dem Editor/Browser. Hierzu werden beispielsweise im Editor angebotene Funktionen und der Umgang mit dem Editor beschrieben.

Des Weiteren kann der Benutzer jederzeit auch auf andere Kapitel im angebotenen Inhaltsverzeichnis navigieren.

4.2.3 “What is ..“ Direkthilfe

Die Direkthilfe erhält der Benutzer indem er in der Menüleiste unter dem Menüpunkt “Hilfe -> What is..“ das Häkchen aktiviert und mit der Maus an der Stelle in der Anwendung klickt, zu der er Hilfe benötigt. Man sollte die Direkthilfe zusätzlich in Form eines Mauszeigers mit Fragezeichen für den Benutzer realisieren. In der prototypischen Realisierung ist dieses für den Browser nicht realisiert, sondern lediglich für den exemplarischen Editor. Das Hilfefenster wird auf den Klick des beschriebenen Menübuttons geöffnet und zeigt dem Benutzer die Hilfeinformation zu der gewünschten Stelle in der Anwendung.

Nach Anzeigen der Hilfeinformation ist die Direkthilfe wieder deaktiviert und der Benutzer kann unbekümmert entweder im Hilfefenster weiternavigieren und weitere Informationen erfragen oder das Hilfefenster schließen, um in der Anwendung von AutoFOCUS fortzuführen.

Die Direkthilfe ist zu demonstrativen Zwecken bereits teilweise in der prototypischen Realisierung implementiert.

4.2.4 Training (Nelli)

Das Trainingshandbuch Nelli sollte direkt über das Menü “Hilfe -> Training (Nelli)“ aus der laufenden Anwendung aufgerufen werden können. Die derzeitige prototypische Implementierung berücksichtigt Nelli noch nicht.

4.2.5 Aktive dynamische Hilfe für Funktionsbuttons (Tooltips)

Zu einem vollständigen Hilfesystem sollte immer darauf geachtet werden, dass Entwickler neuer Funktionalitäten, die in Form von Toolbuttons oder Menüitems zugänglich sind, auch mit den so genannten Tooltips belegt sind. Tooltips dienen dem Benutzer in der korrekten Bedienung der Anwendungssoftware als Hilfestellung, ohne das Hilfefenster öffnen zu müssen. Reichen dem Benutzer diese Information nicht aus, so kann der Benutzer selbstverständlich über den Menüpunkt "Hilfe -> ..." weitere Hilfeinformation anfordern.

4.2.6 "About" (Information)

Der Menüpunkt "Hilfe -> About" ist ebenfalls teil des Hilfesystem, über das der Benutzer die Möglichkeit bekommt Informationen über Versionsbezeichnung, Verantwortlichkeiten zur Software sowie Serviceansprechpartner zu beziehen. Die Funktion "About" entspricht somit derer in anderen benutzerorientierten Softwareanwendungen.

Die "About" stellt keinen direkten Beitrag zur Bewältigung der Arbeit mit der Anwendung dar. Sie wird nicht durch das Hilfefenster angezeigt, sondern wird separat als Information zur Verfügung gestellt.

4.3 Zugang zum Hilfesystem

Der Zugang zum Hilfesystem muss durch den Benutzer vorgenommen werden. Dies bedeutet, dass der Benutzer entweder per Menüpunkt "Hilfe -> Overview" zum Hilfefenster gelangen kann oder per Menüpunkt "Hilfe -> for this Screen" oder aber über den Menüpunkt "Hilfe -> What is ..".

Grundsätzlich sind auch Zugangsmöglichkeiten per Funktionstaste F1, Button oder rechte Maustaste vorstellbar, jedoch in diesem Prototyp nicht realisiert.

Grundsätzlich sollte der Zugang zum Hilfesystem von AutoFOCUS aus Anwendersicht über verschiedene Wege erfolgen können. Die direkten Wege zur vorab definierten AutoFOCUS-Hilfe sind in den nachfolgenden Abschnitten dieses Kapitels dargestellt. In den Abschnitten 4.3.1 und 4.3.2 handelt es sich um eine passive Hilfe. Die in Abschnitt 4.3.3 erwähnte Art des Zugangs zur AutoFOCUS-Hilfe ist aktiv, da sie von sich aus aktiv wird und nicht durch zusätzliche Aktion des Anwenders.

Grundsätzlich ist bei der Tooltips-Hilfe auch eine Zugangsmöglichkeit zum

eigentlichen Hilfesystem der Anwendung denkbar. Diese Option ist allerdings eher unüblich in Anwendungssoftware und wird daher im Rahmen dieses Projektes nicht weiter verfolgt.

4.3.1 Zugang über den Menüpunkt “Hilfe“

Um auf die Online-Hilfe in Form der Registerkarten zuzugreifen, wählt der Benutzer in der grafischen Benutzeroberfläche der AutoFOCUS-Anwendung den Menüpunkt Hilfe aus.

Jeder der verwendeten sichtbezogenen Editoren (SSD, EET, ...) sowie der Projektbrowser in AutoFOCUS/Quest enthält diesen Menüpunkt.

Durch die Auswahl des Menüpunkts wird dem Anwender eine Auswahlliste verschiedener Hilfearten angeboten. Hierzu gehören neben dem Zugang zum “Overview“¹ beispielsweise auch der Zugang

- zur Direkthilfe oder “What is ..“, die Hilfeinformationen zu Feldern in Editoren oder dem Projekt Browser gibt,
- zum Tutorsystem “Nelli“, zur “Info“, das heißt der Information über Version, Herausgeber und Ansprechpartner der Anwendungssoftware.

4.3.2 Zugang über die Funktionstaste “F1“

Die Funktionstaste F1 ist in Softwareprodukten eine gängige Methode das Hilfesystem des Produkts zu aktivieren. Betätigt der Benutzer die Funktionstaste F1, so sollte das Hilfesystem in Form von Registerkarten aktiviert werden, indem ein Hilfefenster geöffnet wird.

4.3.3 Optional: Zugang über Tooltips zur Hilfe

Durch Platzierung des Mauszeigers auf einem beliebigen Funktionsbutton der Anwendung, wird ein Kurztext zur Funktionalität des betroffenen Funktionsbuttons angezeigt, ein so genannter “Tooltip“. Tooltips sind eine Art der Hilfe, die dem Benutzer eine Orientierung über die Funktionsweise von Buttons auf der Benutzeroberfläche geben.

Benutzerunterstützung durch Tooltips geschieht aktiv durch das Anwendungssystem und ist dynamisch an die Position des Mauszeigers gebunden

¹standardmäßig in SW-Anwendungen angebotene Hilfe in Form von Registerkarten, die eine Inhaltsübersicht, eine Indexierung sowie eine Volltextsuche anbietet.

und somit an die Funktionalität des betroffenen Buttons.

Ist die Hilfe in Form des Kurztextes nicht ausreichend für den Benutzer, so hat er die Möglichkeit weitere (detailliertere) Hilfe anzufordern, indem der "Mehr Hilfe-Button" auf dem jeweiligen Tooltip betätigt wird.

Anmerkung:

Da in den gängigen Softwareprodukten diese Art der Tooltip- Erweiterungen bisher nicht realisiert ist, kann davon ausgegangen werden, dass es sich bei dieser Option um eine Funktionalität handelt, die entweder vom Anwender nicht erkannt wird, nicht akzeptiert wird oder sogar zu Verwirrungen bei dem Anwender führen kann.

Weitere Informationen hierzu können unter Aktive, dynamische Hilfe für die Funktionsbuttons nachgelesen. Im Falle der späteren Realisierung oder der Information kann die Idee im Abschnitt nachgelesen werden.

4.3.4 Sonstige Zugangsoptionen

Der Zugang zum Hilfesystem in Form von Registerkarten (der so genannten Standardhilfe) von AutoFOCUS sollte zusätzlich zu den bereits dargestellten Wegen auch von der Direkthilfe, dem Tutorssystem "Nelli", der Informationsfunktion oder jedem Eingabeframe innerhalb eines Editors in AutoFOCUS aufgerufen werden können. Die hierfür vorgesehenen Verweise auf die Hilfe sind erkennbar durch beispielsweise einen mit dem Schriftzug "weitere Hilfe" (o.ä.) gekennzeichneten Button.

5 Design

Ziel dieses Kapitels ist es, aus den Anforderungen des Pflichtenhefts (vgl. Anhang A) sowie der zugrunde liegenden im vorigen Kapitel beschriebenen Konzeption an das Hilfesystem, eine software-technische Lösung, im Sinne einer Software-Architektur oder so genanntes “Design“ zu entwickeln.

Zu Beginn werden die gegebenen Randbedingungen für die Architektur aufgezeigt.

Die jeweilig getroffenen Designentscheidungen sind die Grundlage für das weitere Vorgehen im Projekt.

Die Ausführungen in diesem Kapitel stellen eine Diskussion möglicher Optionen für die Realisierung der Anforderungen aus dem vorhergehenden Abschnitt “Konzeption“ dar, sowie die resultierenden jeweiligen Designentscheidungen.

Im Kapitel 6 werden im Anschluss die Designentscheidungen teilweise prototypisch realisiert, so dass die Funktionsfähigkeit des Hilfesystems nachgewiesen werden kann.

5.1 Überblick

Wie bereits im Vorfeld beschrieben, gilt es den Benutzer beim Umgang mit AutoFOCUS umfangreich zu unterstützen. Um die Hilfeinformation unabhängig von einer möglichen eingesetzten Hilfe-Technologie in AutoFOCUS aufbauen, verwalten und erweitern zu können, ist es gefordert, eine Trennung zwischen Anwendung (Hilfeprogramm) und Daten (z.B. Hilfeinformationstexte) vorzunehmen.

Das Hilfesystem interagiert mit der gesamten AutoFOCUS-Anwendung über definierte Schnittstellen und bildet in der Gesamtarchitektur eine Einheit bzw. ein Modul.

Der Aufbau und die Pflege des Hilfesystems erfordert die strukturelle Trennung der Inhalte. Dazu wird ein Modell entwickelt, das diese Trennung widerspiegelt. (Dies ist vergleichbar mit dem Datenmodell in AutoFOCUS, dass die Elemente der Beschreibungstechnik enthält.)

Das prototypisch realisierte Hilfesystem basiert auf der Hilfe-Technologie “JavaHelp“. Die Entscheidungskriterien für JavaHelp werden in Kapitel 5.4 erläutert.

Das Design des Hilfesystems wird im Anschluss mittels JavaHelp beschrieben, welches einen Grossteil der zu realisierenden Hilfsfunktionen als freie Softwarelösung zur Verfügung stellt.

Um Hilfeinformationen dem Benutzer unter Einsatz von JavaHelp zu Verfügung zu stellen, ist ein sog. "HelpSet" für AutoFOCUS zu entwickeln. Auf dieses Thema wird in dem Kapitel 5.8 näher eingegangen.

Möglichkeiten für Erweiterungen des zu erstellenden notwendigen Help-Sets werden kurz in Kapitel 5.8.3 beschrieben. Hiermit wird vor allem das Ziel eines individuellen Hilfesystems verfolgt.

5.2 Randbedingungen

Bevor auf die näheren Designentscheidungen zu den Realisierungen einiger Funktionen des Hilfesystems eingegangen wird, soll im Folgenden kurz beschrieben werden, welche Randbedingungen beachtet werden müssen, die das Design des Systems mitbestimmen.

5.2.1 Konzeptberücksichtigung

Sämtliche Anforderungen des konzeptuellen Entwurfs des vorhergehenden Kapitels müssen als Bedingungen für den Entwurf des Hilfesystems berücksichtigt werden.

Um beispielsweise dem Benutzer eine gewisse Konsistenz des Zugangs zu ermöglichen, ist es daher nötig, die Menüleiste sämtlicher in AutoFOCUS verwendeter Editoren oder Browser mit derselben Bezeichnung für Hilfe auszustatten und natürlich demselben Umfang.

5.2.2 Editoren und Browser

Durch das Hilfesystem sollen sowohl Editor-/Browser-spezifische Eigenschaften für den Benutzer im Umgang mit der Anwendungssoftware AutoFOCUS erklärt werden, wie auch system- bzw. modellierungsspezifische Eigenschaften. Es müssen also durch das Hilfesystem sowohl Editor/Browser-Information als auch Modellinformation zur Verfügung gestellt werden. Während Modellinformationen zu einem späteren Zeitpunkt diskutiert werden, geht es in diesem Abschnitt um die Gewährleistung der Vollständigkeit der Editor/Browser-Information im Hilfesystem.

Da der Entwickler eines Editor/Browser-Konzepts am meisten geeignet erscheint, Informationen hierzu anzugeben, sollte die Entwicklung des Hilfesystems an dieser Stelle parallel stattfinden.

5.2.3 Erstellung eines Helpsets

Um Hilfe in geordneter Form und ausreichendem Umfang zur Verfügung stellen zu können, ist es notwendig, eine Menge von Hilfedateien mit Hilfeinformationstexten, Organisationsdateien etc. zu erstellen. Diese Menge der Hilfedateien wird im Folgenden mit "Helpset" bezeichnet und bedeutet einen wichtigen Teil des Hilfesystems. Die verwendeten Texte für die Hilfeinformation entstammen der AutoFOCUS I - Version und sind lediglich zu Vorführungszwecken eingesetzt worden. Die methodisch, didaktische Aufbereitung der Texte wurde in diesem Projekt nicht berücksichtigt.

5.3 Design-Basis-Entscheidungen

5.3.1 Einsatz von Hypertextsystemen

Laut Balzert (vgl. [BAL98]) werden heute vielfach HTML-basierte Hilfesysteme für den Zweck eines Hilfesystems eingesetzt. Darüber hinaus stellen moderne Betriebssysteme und/oder GUI-Systeme bereits Basisleistungen für ein Hilfesystem zur Verfügung.

Eben dieser Basisleistungen wird sich im Rahmen dieses Projektes mit Hilfe von JavaHelp bedient, worauf im nachfolgenden Kapitel 5.7 näher eingegangen wird. Innerhalb des Hilfe-Technologie JavaHelp wird darüber hinaus ebenfalls mit HTML-Dokumentenstrukturen gearbeitet, so dass es hierbei nicht zu Formatbrüchen kommt.

Designentscheidung Die für ein Hilfesystem notwendigen Informationstexte werden in HTML-Dateien abgelegt. Ein wichtiger Grund für diese Entscheidung ist neben der Unterstützung durch die Hilfe-Technologie JavaHelp, dass die in der bisherigen AutoFocus-Anwendung vorhandene Online-Hilfe wie auch das Online-Tutorial bereits im Html-Format vorliegen und somit ein großer Anteil der Texte übernommen werden kann.

5.3.2 Festlegung der Granularität des Systems

Zum Design des Hilfesystems für AutoFOCUS ist es notwendig, dass die User-Interface-Komponenten festgelegt werden, über die Hilfeinformation zur Verfügung gestellt werden soll.

Es kommen hierfür verschiedene Ebenen in Frage, die Berücksichtigung

im System finden können, da es sich um ein dynamisches (kontextsensitives) System handeln soll. Die Festlegung der Granularität des Systems wird zum derzeitigen Zeitpunkt der prototypischen Realisierung lediglich auf drei Stufen beschränkt. Die feinste Stufe der Granularität stellen hierbei Erklärungen zu einzelnen Feldern, Symbolen oder beispielsweise Buttons dar. Zwischen den Granularitätsstufen kann der Benutzer beliebig navigieren.

Designentscheidung Der Benutzer kann das Hilfesystem entweder

- als Gesamtübersicht sehen (geringe Granularität)
- als Übersicht für einen bestimmten Editor/Browser
- als Hilfestellung für einzelne Felder im Editor/Browser (z.B. Editorbuttons, Menüpunkte) oder auch Felder in den Editoren/Browsern, die als Repräsentant eines Elements des zugrunde liegenden Modells (z.B. Komponente im SSD-Editor) dienen.

5.4 Einsatz der Hilfetechnologie JavaHelp

Bereits in den in Kapitel 4 “Konzeption“ definierten Anforderungen zur Darstellung von Hilfeinformationen ist darauf hingewiesen, dass ein Grossteil durch Einsatz der freien Softwarelösung “JavaHelp“ realisiert werden kann. Im Rahmen dieses Projekts wird diese Hilfe-Technologie eingesetzt. Im Folgenden werden einige wichtige Entscheidungskriterien für den Einsatz von JavaHelp in diesem Projekt erläutert.

JavaHelp ...

- **... kann leicht als lose Kopplung (Modul) in die Anwendung AutoFOCUS integriert werden.**

Erläuterung:

Um den definierten Anforderungen nachzukommen, ist es notwendig, das Hilfesystem in die Anwendung AutoFOCUS zu integrieren. Im Gegensatz zu einem selbständigen Hilfesystem bedeutet die Integration insbesondere, dass das Hilfesystem an die Anwendung AutoFOCUS gebunden wird und nur bei Einsatz der Anwendung ausgeführt werden kann. (Allerdings handelt es sich um eine lose Kopplung, da JavaHelp auch problemlos gegen eine andere Technologie ausgetauscht werden kann.)

- **... unterstützt ein passives Hilfesystem.**

Erläuterung:

Obwohl JavaHelp auch die Möglichkeit eines aktiven Hilfesystems anbietet (eingebettetes System) ², kann JavaHelp auch als passives Hilfesystem eingesetzt werden, welches lediglich auf Wunsch des Benutzers erscheint. Da es sich gemäß den Anforderungen um ein passives Hilfesystem handeln soll, kommt die Realisierung eines eingebetteten Systems nicht in Frage.

- **... ermöglicht dem Benutzer kontextsensitive Anwendungsunterstützung**

Erläuterung:

Im Rahmen eines Hilfesystems wird gewährleistet, dass der Benutzer aus der AutoFOCUS Anwendung heraus das Hilfesystem aufrufen kann. Kontextsensitivität ermöglicht, dass die angezeigte Hilfeinformation sich auf den zuletzt aktivierten Kontext der Anwendung bezieht.

- **...bietet verschiedene Zugangsmöglichkeiten zum Hilfesystem an**

Erläuterung:

Der Aufruf kann hierbei beispielsweise aus einem Anwendungsfenster heraus (durch Betätigung des Hilfe-Buttons) oder auf der Ebene eines Feldes (durch Positionierung eines Fragezeichens) heraus aktiviert werden.

- **....bietet weitere Unterstützungen**

Weitere Möglichkeiten, die die Hilfe-Technologie JavaHelp bietet, können entweder in der Anwendungsspezifikation auf der Internetseite des Anbieters Sun Microsystems AG (vgl. [JAV02]) entnommen werden oder dem ebenfalls dort angebotenen User-Guide. API und Installationsanleitungen stehen ebenfalls zur Verfügung.

5.5 Realisierung der kontextsensitiven Hilfe

Um kontextsensitive Hilfe zu realisieren, ist es notwendig unter anderem über die Punkte

- Granularität
- Auslöse-, Kontroll- und Wartungsmechanismus

Entscheidungen zu treffen.

Während über **die Granularität** bereits eine Entscheidung unter dem Abschnitt 5.3.2 "Festlegung der Granularität des Systems" getroffen wurde

²Unter einem eingebetteten System wird in diesem Zusammenhang ein Hilfesystem verstanden, welches in die Benutzeroberfläche der eigentlichen Anwendung integriert wird und kontinuierlich für den Benutzer sichtbar ist.

und diese als Grundlage für weitere Entscheidungen verwendet wird, muss es nun noch zur Festlegung für den einzusetzenden **Mechanismus** des Hilfesystems kommen.

Kontextsensitivität bedeutet, dass sich der Benutzer bei Aufruf der Hilfe in einem beliebigen aber eindeutig zu identifizierenden Zustand der Anwendung befindet, zu dem er Hilfe auf einer von ihm ausgewählten Granularitätsstufe im Hilfesystem erhält. Ein fester Zustand der Anwendung könnte beispielsweise der Zeichenmodus in einem Editor sein, oder ein geöffnetes Dialogfenster, in dem der Benutzer seine Eingabe vornehmen muss, um fortfahren zu können.

Die folgende Abbildung zeigt die verschiedenen Zugriffsmöglichkeiten auf das Hilfesystem aus der Benutzersicht.

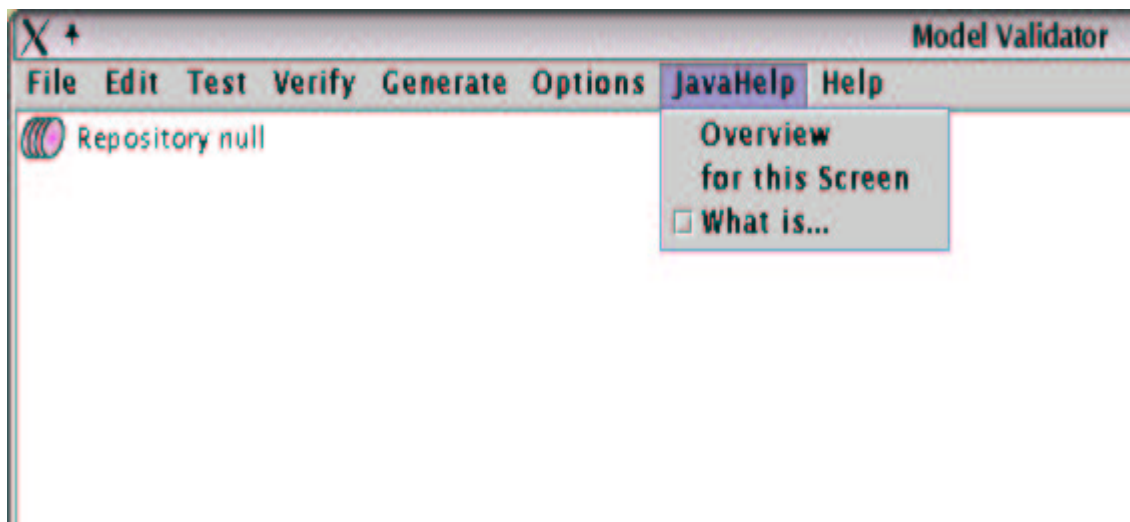


Abbildung 4: Zugriffsmöglichkeiten auf das Hilfesystem über die Menüleiste

Um die kontextsensitive Hilfe auf den drei festgelegten Granularitätsstufen zu Verwirklichen, soll im Folgenden der Mechanismus (die Zugangsart) festgelegt werden, der es ermöglicht, die seitens des Benutzers gewünschte Hilfeinformation korrekt anzuzeigen.

5.5.1 Feststellung der Art der Kontextsensitivität (Zugangsarten zum Hilfesystem)

Zur Feststellung, welche der drei aufgezeigten Granularitätsstufen der Benutzer zum Zeitpunkt des Hilfeaufrufs wünscht, gibt es für jede der drei Granularitätsstufen eine separate Zugangsfunktion. Wählt der Benutzer also in der Menüleiste und "HELP" den Menüpunkt

- **"Overview"**,
so ruft das Hilfesystem das Hilfefenster auf und lädt im Textfenster den einführenden Informationstext für den Überblick im Hilfesystem.
- **"for this Screen ..."**,
so ruft das Hilfesystem das Hilfefenster auf und lädt im Textfenster den Informationstext für den Überblick im Editor/Browser.
- **"What is ..."**,
so muss der Benutzer zunächst per Mausklick auf das gewünschte Objekt im Editor/Browser klicken und anschließend ruft das Hilfesystem das Hilfefenster auf und lädt im Textfenster den Informationstext für das ausgewählte Objekt.

5.5.2 Feststellung des Kontexts

Hat der Benutzer sich für eine der drei Granularitätsstufen entschieden, so muss jetzt das System erkennen, welche Hilfeinformation der Benutzer erwartet.

Auf der Ebene

- **"Overview"**
ist festzustellen, dass es sich um den allgemeinsten Kontext "AutoFOCUS" handelt und das hierzu gehörige Hilfeinformationssystem mit dem einleitenden Hilfethema "Overview" anzuzeigen. Die anzuzeigende Hilfe ist demnach unabhängig beispielsweise vom Editoren-/Browser-kontext.
- **"for this Screen ..."**
der Editoren/Browser-Ebene, muss festgestellt werden, um welchen Editor/Browser es sich handelt, aus dem das Hilfesystem aufgerufen wurde. Eine elegante Lösung wäre hierbei eine Art der Registrierung beim Hilfesystem seitens der Editoren/Browser, so das das Hilfesystem jederzeit "weiß", welche Editoren/Browser gerade aktiv sind und für diese beispielsweise jeweils intern mittels einer Tabelle verwaltet.

Für die prototypische Implementierung wurde die Realisierung derart gewählt, dass der exemplarische Browser sowie Editor jeweils eine eigene Instanz des Hilfesystems anlegen. Die Instanzen greifen dann allerdings auf dieselbe Datenquelle für die Hilfeinformationstexte zurück.

- **“What is ...“**

wird der Benutzer aufgefordert, genauer zu präzisieren, zu welchem Objekt auf dem Editor/Browser er gern Hilfe angezeigt bekommen möchte. Die Auswahl ist hier mittels des Mauszeigers vorzunehmen.

Der Benutzer kann demnach sowohl Hilfe

- zur Menüleiste und deren Unterpunkte anfordern wie auch
- zur Toolbar oder
- zu Objekten in der Modellierungstechnik der Anwendung.

Da es sich sowohl bei den Menüpunkten wie auch den Toolbar-Buttons um Komponenten handelt, die bereits vor der Generierung des Systems sowohl definiert sind als auch zur Laufzeit des Systems fest und vorgegeben sind, handelt es sich bei den Objekten in der Modellierungstechnologie von AutoFOCUS um Komponenten, die zwar vorab ebenfalls definiert sind, aber erst zur Laufzeit durch den Benutzer angelegt werden.

Der Umgang mit dem Kontext in Objekten der Modellierungstechnik wird im nachfolgenden Abschnitt 5.6 beschrieben.

Realisierung der Kontextsensitivität Obwohl Einzelheiten aus JavaHelp jetzt noch nicht diskutiert werden sollen, kann vorab gesagt werden, dass es mittels der Technologie von JavaHelp relativ leicht ist, Menüleisteneinträge oder Toolbarbuttons zu adressieren, da hierfür leichte Zuordnungen vorgenommen werden können zwischen beispielsweise Toolbarbutton A und dem dazugehörigen Hilfetext. Objekte in der Modellierungstechnologie der Anwendung AutoFOCUS, wie sie beispielsweise im Projektbrowser angezeigt werden, bedeuten hingegen schon eine Herausforderung, da die Schnittstellen von JavaHelp und dem Projektbrowser konzeptionell verschieden sind. Details zur Realisierung der “What is ...“-Hilfe werden ausführlich im anschließenden Kapitel 6 diskutiert.

5.5.3 Zuordnung der Hilfeinformation zum jeweiligen Kontext

Ist nun seitens des Hilfesystems eindeutig zuzuordnen, in welchem Kontext sich der Benutzer befindet, so soll die dazugehörige Hilfeinformation im Hilfefenster angezeigt werden. Es handelt sich hierbei um einen Suchvorgang in der Menge der Hilfetexte. Während es für die "Overview"-Hilfe lediglich eine festgelegte Datei gibt, die zum gegebenen Zeitpunkt aufgerufen wird, gibt es entsprechend der Anzahl der zur Verfügung stehenden Editoren/Browsern hierzu mehrere Hilfeinformationsdateien. Die richtige Datei muss genau spezifiziert werden, so dass die Zuordnung des Editors "A" auf die Hilfeinformationsdatei "A" gelingt. Noch umfangreicher wird die Zuordnung bei der "What is ..."-Hilfe (Direkthilfe), so dass ein Mechanismus zur Zuordnung entworfen werden muss, der im Folgenden kurz beschrieben wird.

Modellbasierung durch Verzeichnisstruktur Um Hilfeinformationstexte korrekt anzuzeigen, Texte möglichst ohne Redundanzen zur Verfügung zu stellen und dem Entwickler von Hilfeinformationstexten eine deutliche Gliederung anzubieten, in die er intuitiv korrekt seine Dateien einordnen kann, wurde eine Verzeichnisstruktur angefertigt, welche diesen Anforderungen entspricht. Durch die Verzeichnisstruktur wird das Hilfesystem "modellbasiert", denn die wesentlichen Informationen, auf die mittels der eingesetzten Hilfetechnologie zugegriffen wird, und die durch die Benutzerschnittstelle sichtbar gemacht werden, sind in der Verzeichnisstruktur verankert. Die Verzeichnisstruktur ist somit sowohl als Ablagestruktur für Hilfeinformationstexte, die vom Entwickler anzufertigen sind hilfreich, wie auch als Grundstein für das modellbasierte Hilfesystem.

Die Verzeichnisstruktur ist grob in der folgenden Abbildung 5 dargestellt.

Erkennbar ist, dass die Editoren, die ja die Sicht (i.S.v. "View") des Benutzers von AutoFOCUS bilden, Basis für die Struktur sind. Die Entscheidung für diese konzeptionelle Strukturierung ist sinnvoll, da das Hilfesystem aus einem Editor/Browser der korrespondierenden Anwendungssoftware AutoFOCUS heraus gestartet wird. Es wird eine Art "Vererbungsmechanismus" gebildet, welcher einen (SSuper"-)Editor beinhaltet, der allgemeine Informationen beinhaltet, die in allen Editoren/Browsern enthalten sind. Dieser (SSuper"-) Editor ist in der Abbildung mit "Common" bezeichnet. Hilfeinformationstexte, die hierin enthalten sind, könnten beispielsweise zu den Menüpunkten "File -> Exit" oder "Help" gehören, da diese Menüpunkte in jedem Editor/Browser vorkommen.

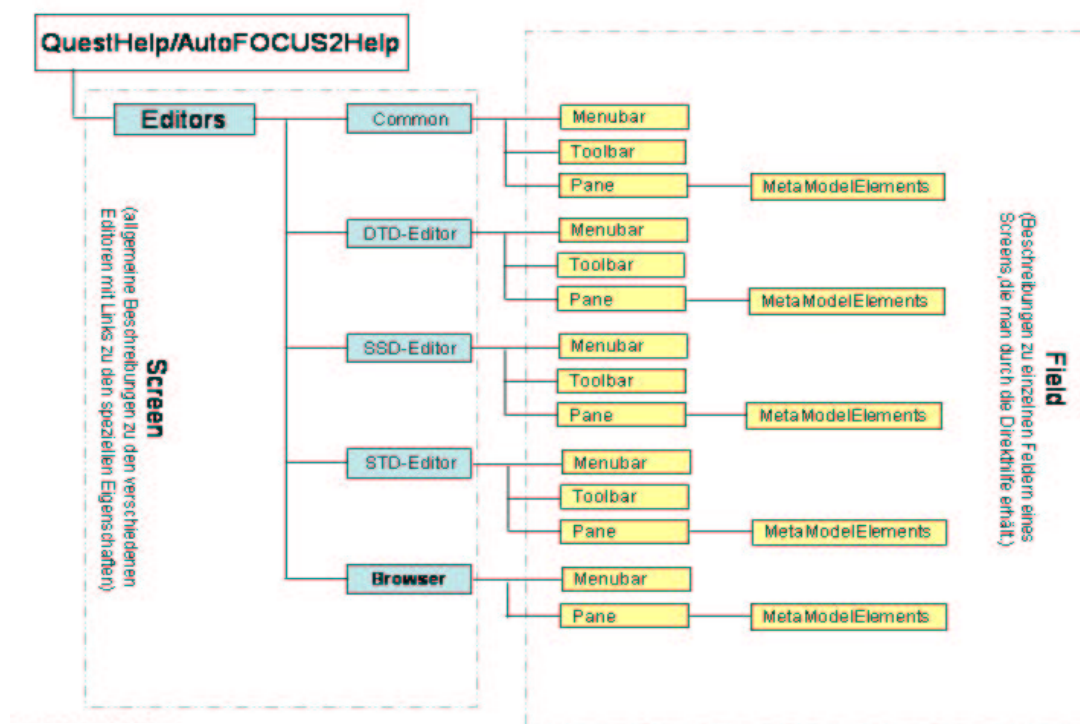


Abbildung 5: Die Verzeichnisstruktur für Hilfeinformationsdateien

Um den Anforderungen aus Kapitel 4 nachzukommen und hierbei im Speziellen der Forderung nach vollständiger Hilfe, ist es für das Anlegen der Verzeichnisstruktur im Hilfesystem notwendig, genauestens über die Editoren/Browser und deren Eigenschaften im System informiert zu werden und dementsprechend Hilfeinformationstexte zu hinterlegen. Die Verantwortung für diese Aufgabe sollte separat geklärt werden.

Sicht des Benutzers auf die Verzeichnisstruktur Der Benutzer bekommt eine spezielle Sicht auf das Modell, in der ihm verborgen bleiben soll, dass Hilfeinformationstexte nur einmalig im System zur Verfügung stehen. Für den Benutzer sieht es also so aus, als gäbe es für sämtliche Editoren/Browser jeweils einen separaten Hilfeinformationstext zu den einzelnen Menüpunkten. Selbst wenn der Benutzer zufällig beispielsweise sowohl im SSD-Editor, wie auch im DTD-Editor Hilfe zum Menüpunkt File->Exit anfordert und er feststellt, dass die Texte syntaktisch und semantisch sind, ist es für ihn nicht erkennbar, dass die angezeigten Informationen aus den gleichen Quellen im zugrunde liegenden Hilfesystem stammen.

Auf diese Weise wird die interne Modellstruktur dem Benutzer nicht in einer 1:1-Abbildung präsentiert, sondern als Sicht auf das Modell.

“Help-IDs“ Die abgebildete Verzeichnisstruktur weist für die verschiedenen Editoren/Browser eindeutige Pfadnamen auf, die im Folgenden als “Help-ID“ bezeichnet werden sollen. Help-IDs sind notwendig, um für den Benutzer die Zuordnung seines aktuellen Kontexts zur dazugehörigen Hilfeinformationsdatei herzustellen.

Der Mechanismus des Ladens der Datei in das Hilfefenster erfolgt mit Hilfe von JavaHelp und wird im nachfolgenden Kapitel kurz beschrieben.

5.6 Identifikation von Objekten der Modellierungstechnologie

Mit Objekten der Modellierungstechnologie sind Bestandteile des Systems gemeint, mit denen der Benutzer sein System in AutoFOCUS modelliert. Um Kontextsensitivität auf der kleinsten Granularitätsstufe realisieren zu können, ist es hierbei notwendig, diese Objekte zu identifizieren, um dem Benutzer der Anwendung korrekte Hilfeinformationstexte anzeigen zu können. Als direkte Zugangsart auf dieser Stufe der Granularität steht dem

Benutzer die “What is ..“-Hilfe (Direkthilfe) zur Verfügung, mit dessen Hilfe er direkt beispielsweise ein Element (oder Blatt) im Projektbrowser anklicken kann. Der Projektbrowser kann in seiner Eigenschaft als “Übersicht für den Benutzer“ sämtliche Objekte verwalten und anzeigen.

Anders ist es in den Editoren. Die verschiedenen Editoren ermöglichen dem Benutzer unterschiedliche Sichten auf sein System. Jede Sicht wird hierbei durch einen Editor dargestellt, der verschiedene Elemente zur Verfügung stellt. Beispielsweise ist bei der Arbeit mit dem System Struktur Diagramm Editor der Einsatz der Elemente “Components“, “Channels“ oder “Ports“ üblich, während es bei der Arbeit mit dem State Transition Diagram Editor zum Umgang mit Elementen wie “State“ oder “Transition“ kommt.

AutoFOCUS basiert auf einem sog. “Metamodel“, welches als Basis für die Entwicklung der Anwendung gilt. Gemäß dem “Model-View-Controller“ Pattern werden im (Meta-)Model die wichtigsten Daten des Systems gehalten und ein Zugriff auf das Modell ist nur indirekt möglich.

Dieser Zugriffsmechanismus soll verwendet werden, da über ihn die Namen der Elemente herausgefunden werden können, die zum Auffinden der jeweiligen Hilfeinformationsdateien notwendig sind. Wie der Zugriff auf Modell-daten geschieht, wird im nachfolgenden Kapitel 6 “Implementierung“ im Abschnitt 6.3.2 genauer erklärt.

5.7 Funktionsweise des Systems JavaHelp

Das System JavaHelp System beginnt bei seinem Aufruf mit dem Lesen des festgelegten “HelpSet“, welches die notwendigen Hilfeinformationen zu einem Projekt oder einer Anwendung (in diesem Falle das Projekt Auto-FOCUS) zur Verfügung zu stellt.³

5.7.1 Das HelpSet und die Organisation

Das HelpSet besteht aus

- **Organisations-/Navigationsdateien**

- * HelpSet.hs

- Die Datei kümmert sich um die Pfadangaben zu Map.jhm, Index.xml und TOC.xml und gilt als Anker im Hilfesystem. Die Datei wird als Initialdatei vom Hilfesystem benötigt..

³vgl. [LEW00]

- * Map.jhm
In dieser Datei wird eine Zuordnung von Help-IDs zu echten Pfadangaben vorgenommen.
- * Index.xml
In Index.xml wird definiert, mit welchem Bezeichner eine Hilfeinformationsdatei im Hilfefenster erscheinen soll.
- * TOC.xml
Die Datei "TOC.xml" legt die Struktur, das Inhaltsverzeichnis (TOC = Table Of Contents) für das Hilfefenster an.

– **Hilfeinformationsdateien**

Die Hilfeinformationsdateien sind nachfolgend lediglich exemplarisch angegeben.

- * Overview.html
- * EditorA.html
- * EditorB.html
- * Browser.html
- * ...

Es handelt sich also um herkömmliche html-Dateien, die die für den Benutzer sichtbaren Hilfeinformationstexte zu den jeweiligen Themengebieten enthalten.

Das Zusammenspiel der JavaHelp-Dateien also über spezielle Dateien organisiert. Der Benutzer hat die Möglichkeit, über die Verzeichnisstruktur im Hilfefenster zu navigieren.

Um die jeweilige Information darzustellen, werden vom JavaHelp-System die Navigationsdateien, die in der HelpSet-Datei aufgelistet sind, gelesen. Zusätzlich werden vom JavaHelp-System Informationen der Map-Datei verwendet, um die jeweiligen themenspezifischen Topic-Dateien des Help-Sets zu finden. Das Zusammenspiel der JavaHelp-System Dateien wird in der nachfolgenden Abbildung 6 grafisch dargestellt.

Die Ansteuerung der Hilfedatei in JavaHelp erfolgt, indem der genaue Bezeichner für die Hilfedatei aufgerufen wird. In dieser Realisierung des Hilfesystems entspricht die "Help-ID" des Hilfesystems genau dem Bezeichner in JavaHelp, der die Hilfeinformationstexte identifiziert. (In JavaHelp wird dieser Bezeichner "Map-ID" genannt.)

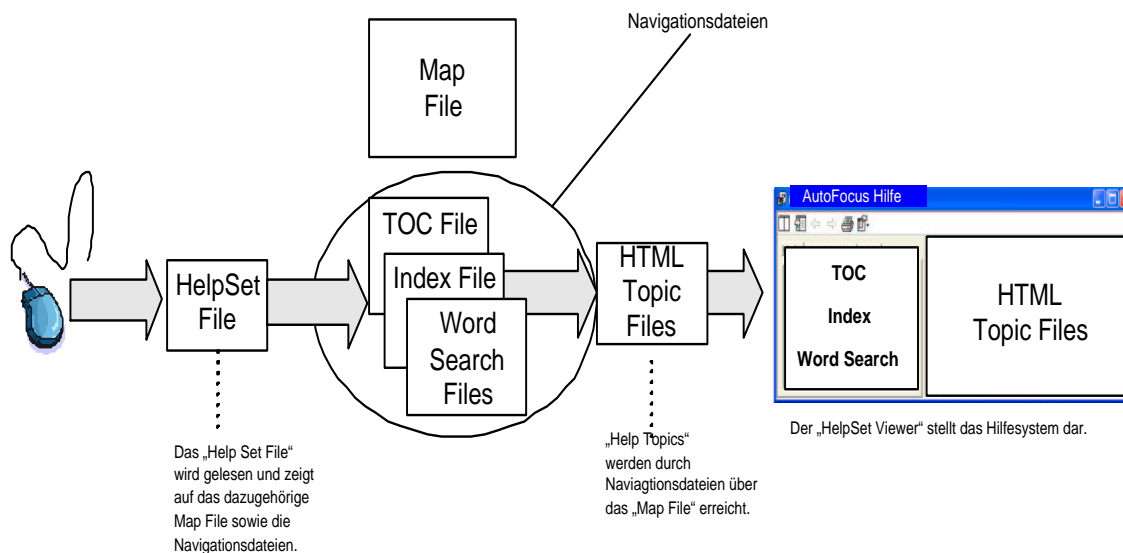


Abbildung 6: Das Zusammenspiel der JavaHelp-Dateien zu Erzeugung des Hilfesystems

5.7.2 Grundlagen XML

- XML ist die Abkürzung für “eXtensible Markup Language“ und ist eine “vereinfachte“ Version des Standard Generalized Markup Language (SGML). Die Entwicklung von XML begann 1996 und seit Februar 1998 ist XML ein W3C-Standard.
- XML soll es den Web-Programmieren erleichtern, SGML-Anwendungen zu schreiben und dabei eigene Dokumententypen (DTD) festzulegen.
- XML ist eine Methode, um strukturierte Daten in einer Textdatei darzustellen. Programme, die solche Daten erzeugen, können binär oder in einem Textformat / ASCII-Format speichern.
- XML verwendet Tags und Attribute. Diese werden jedoch lediglich zur Abgrenzung von Daten verwendet. Die Interpretation der Tags und Attribute erfolgt durch die Anwendung, die die Daten verarbeitet.
Im Projekt sind map-, TOC-, index- und HelpSet-Datei in XML-Format.

5.7.3 Funktionalitäten im Hilfefenster

Durch den Einsatz mit JavaHelp werden auch Funktionalitäten, wie Inhaltsübersicht, Volltextsuche und Indexierung zur Verfügung gestellt. Die Funktionalitäten werden im Wesentlichen unter Verwendung von von JavaHelp realisiert durch die in Abschnitt 5.7.1 bereits vorab eingeführten Dateien Organisations- und Navigationsdateien.

Alle Funktionalitäten finden sich für den Benutzer im Hilfefenster wieder, welches in Abbildung 7 bereits kurz vorgestellt wurde. Jede im Hilfefenster befindliche Registerkarte beinhaltet eine der Funktionalitäten.

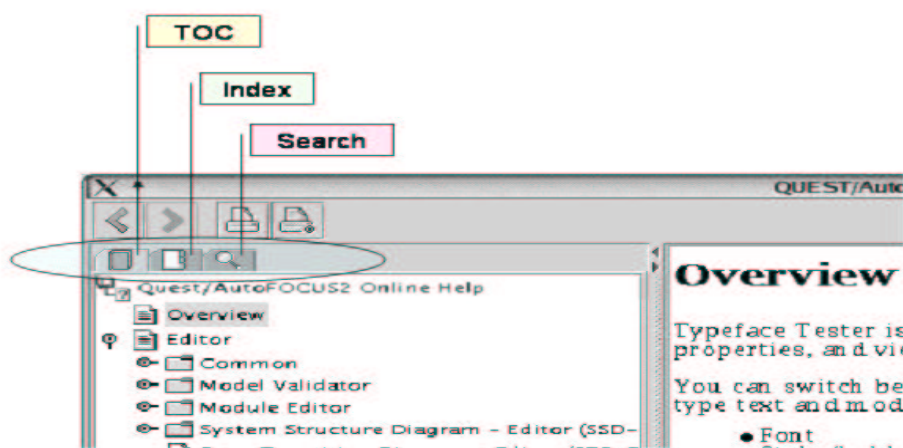


Abbildung 7: Funktionalitäten im Hilfefenster

Themenüberlick (Table of Contents) Der Themenüberblick gibt eine Inhaltsstruktur der im Hilfesystem verfügbaren Themen wieder. Verantwortlich für das Anlegen des Themenüberblicks ist die xml-Datei "TOC.xml", die Bestandteil des oben eingeführten notwendigen Helpsets in JavaHelp ist.

Volltextsuche (Word Search) Durch die Volltextsuche bekommt der Benutzer die Möglichkeit, gezielt nach dem Vorkommen von bestimmten Schlüsselwörtern in dem (umfangreichen) Hilfesystem zu suchen.

Der Benutzer bekommt nach einer erfolgreichen Suche sämtliche Dokumente (Dateien) aufgelistet, die das gesuchte Wort beinhalten. Die Suche wird durch eine von JavaHelp beigestellt "Search-Engine" realisiert.

Indexierung (Index) Zur Indexierung ist es notwendig, dass zu jedem html-Dokument im Hilfesystem ein Index bestimmt wird. Dieses wird in der Datei "Index.xml" vorgenommen, die bereits weiter oben kurz eingeführt wurde. Die Sortierung der Einträge übernimmt JavaHelp durch einen in der Spezifikation von JavaHelp beschriebenen Generierungsvorgang.

5.8 Standard-HelpSet: Die Hilfeinformationsverwaltung in AutoFOCUS

5.8.1 Verzeichnisstruktur

Um die HelpSet-Dateien zu erreichen, ist JavaHelp abhängig von einer sorgfältigen und gut strukturierten Verzeichnisstruktur. Die Verzeichnisstruktur wird in Kapitel 5.5.3 diskutiert. Verweise auf eine Datei anhand eines unkorrekten Bezeichners(Help-ID) oder der Angabe einer fehlerhaften Lokation im Speicher, führen unweigerlich zu Fehlverhalten im Hilfesystem.

Zur Entwicklung eines stabilen Hilfesystems sollte das Projekt mit einer gut durchdachten Verzeichnisstruktur versehen werden, deren grundsätzlicher Aufbau an die bereits eingeführte Verzeichnisstruktur angelehnt wird.

5.8.2 HelpSet- und Navigationsdateien

Es ist notwendig, dass eine Grobstruktur für die HelpSet-Dateien und darin enthaltenen Navigationsdateien festgelegt wird, auf dessen Grundlage der "View für den Benutzer gestaltet wird. Im vorliegenden Projekt wurde versucht die Verzeichnisstruktur aus Abbildung ?? weitestgehend zu übernehmen.

In einer neuen Diskussion um die hinsichtlich der Methodik und Didaktik für den Benutzer optimale Lösung müssen auf jeden Fall Entscheidungen zu den nachfolgend aufgelisteten Dateiinhalte getroffen werden.

* HelpSet File

- * Map File
- * Navigations Datei (xml-Datei)
- * Index Datei (xml-Datei)

5.8.3 Erweiterungen des Standard-HelpSets

Um die individuelle Hilfe für mindestens zwei Benutzergruppen (Anfänger/Geübte) anbieten zu können, müssen vorab hierfür Design-Entscheidungen getroffen werden.

In JavaHelp gibt es hierfür zum einen die Möglichkeit über zusätzliche Fenster neben dem eigentlichen Hilfefenster zusätzliche Hilfe anzubieten oder über die die Anpassung der Standard-Navigations- Möglichkeiten. Es wird hierbei auf die in der Anlage erwähnte Literatur [JAV02] verwiesen, in der die verschiedenen Möglichkeiten ausführlich diskutiert werden.

5.8.4 Erstellen von Hilfeinformationsdateien

Damit das Hilfesystem dem Benutzer von AutoFOCUS einen Mehrwert bei dessen Einsatz bringt, ist es notwendig, sich über den didaktischen sowie methodischen Charakter sowie den Inhalt der Hilfeinformationsdateien ein Konzept auszuarbeiten sowie die Realisierung dieser Hilfeinformationsdateien vorzunehmen. Es ist dabei darauf zu achten, dass die Hilfe individuell gemäß den Anforderungen aus Kapitel 4 gestaltet wird. Das bedeutet insbesondere, dass standardmäßig bei Aufrufe der Hilfe eine Kurz-Hilfeinformationsdatei aufgerufen wird und der Benutzer nach Bedarf die Möglichkeit hat, detailliertere Informationen zu diesem Thema anzufordern. Die Realisierung könnte hierbei über die Secondary oder Pop-Up-Windows (vgl. Kapitel 5.8.3) geschehen. Aus Benutzersicht handelt es sich im Wesentlichen um die Verfolgung eines Hyperlinks aus dem Hilfefenster heraus.

Bei der Ausarbeitung der Hilfeinformationsdateien sollte vor allem beachtet werden, dass Struktur und Inhalt der Dateien vom didaktischen wie auch methodischen Standpunkt her

- * ausführlich und vollständig definiert ,
- * im Detail ausgearbeitet, sowie
- * regelmäßig überprüft werden.

Während der Weiterentwicklung von AutoFOCUS sollte außerdem darauf geachtet werden, dass die Entwickler daran erinnert werden, zu jeder neuen Komponente eine Hilfeinformationsdatei anzulegen.

5.9 Das Hilfesystem aus Systemsicht

Initialisierung Mit Aufruf der Anwendung AutoFOCUS wird das Hilfesystem initialisiert. Dies bedeutet, dass zum Zeitpunkt des Aufrufs von AutoFOCUS festgelegt wird, welches das zu verwendende HelpSet (vgl. Abschnitt 5.2.3) ist.

Je nachdem, welches der drei Sichten (oder Granularitätsstufen) der Benutzer auswählt, öffnet sich das Hilfefenster des Hilfesystems mit der Registerkarte und die kontextsensitive Hilfeinformation wird angezeigt. Beim ersten Öffnen des Hilfefensters nach Neustart von AutoFOCUS wird die Registerkarte "Themenüberblick (Table Of Contents)" aufgerufen. Ändert der Benutzer die Registerkarte auf z.B. Volltextsuche und schließt anschließend das Hilfefenster, so wird beim nächsten Anfordern der Hilfe das Hilfefenster mit der Registerkarte "Volltextsuche" angezeigt. Die Verwaltung der jeweiligen Registerkarten wird über JavaHelp gesteuert. (Die Navigationsmöglichkeiten im Hilfefenster sowie unterschiedlichen Funktionalitäten, wie Volltextsuche, Inhaltsübersicht oder Indexierung unterliegen dem System JavaHelp und können in der Systemspezifikation nachgelesen werden.)

Kontextsensitive Hilfe

Aktivierung der kontextsensitiven Hilfe setzt voraus, dass im Hilfesystem jedem Objekt auf der jeweiligen Granularitätsstufe (also z.B. jedem möglichen Editor/Browser auf der mittleren Granularitätsstufe), für das Hilfeinformation gegeben werden soll, eine eindeutige Identifikation zugeordnet werden kann, die "Help-ID". Anhand der Help-ID kann dann das korrekte Hilfethema für den Benutzer zur Verfügung gestellt werden. (Auch hierbei wird im Wesentlichen der Automatismus von JavaHelp ausgenutzt.)

Wie bereits im Kapitel 5.6 angesprochen, kommt es lediglich bei der kleinsten Granularitätsstufe zu Herausforderungen, da die Funktionalität, die durch JavaHelp angeboten wird, beispielsweise nicht für Browserstrukturen geeignet ist. Um jedoch die Anforderung auf Konsistenz im System zu erfüllen, wurde hier der JavaHelp-Mechanismus durch ein eigenes Design ersetzt, auf das im nachfolgenden Kapitel 6 vertieft eingegangen wird.

6 Implementierung des Prototyps

Das vorliegende Kapitel 6 “Implementierung des Prototyps“ beschäftigt sich im Wesentlichen mit der Anbindung des Hilfesystems an die Hauptanwendung AutoFOCUS. Zur Erläuterung werden auszugsweise Codebeispiele gegeben. Der gesamte Programmcode findet sich im Anhang B

Die Editoren und Browser werden durch ein Editoren-Framework zur Verfügung gestellt. Da AutoFOCUS derzeit in der Entwicklung ist und sich das Editoren-Framework ebenfalls gerade in der Entwicklung befindet, wird bei der prototypischen Implementierung lediglich der Browser “Model Validator“ sowie der Editor “Module Editor“ für exemplarische Zwecke verwendet.

Besondere Herausforderung bei der Implementierung bedeutet die kontextsensitive Hilfe auf der kleinsten Stufe der Granularität, der so genannten “What is ...“-Hilfe (Direkthilfe). Auf diese Herausforderung wird gleich zu Beginn dieses Kapitels ausführlich eingegangen.

Die weiteren Realisierungen, wie sie im vorhergehenden Kapitel 5 angekündigt sind, werden in kompakter Form im Anschluss in Abschnitt 6.4 beschrieben.

Dieses Kapitel wird mit dem Abschnitt 6.5 abgeschlossen, welches sich mit dem Thema der zukünftigen Erweiterungsmöglichkeiten am Hilfesystem beschäftigt.

6.1 Programm-Bestandteile des Hilfesystems

Der Programmcode, in der die Funktionalität des Hilfesystems implementiert ist, ist mit “HelpSystem.java“ bezeichnet. Die dazugehörige Klasse findet sich in *quest.help.HelpSystem*; und kann durch den “import-Befehl“ in andere Implementierungen integriert werden.

6.1.1 Klassen

Das Hilfesystem besteht aus verschiedenen Klassen, wobei die Klasse “HelpSystem.java“ die Hauptfunktionalität übernimmt und die anderen Klassen als Hilfsklassen dienlich sind.

Helpsystem.java Zu Beginn des Programms müssen die notwendigen Klassen importiert werden, hierzu gehören unter anderem:

1. **import javax.help.***- Import der notwendigen Klassen der Hilfetechnologie JavaHelp.
2. **import quest.testing.application.browser.***-Import der Klassen zur Ansteuerung des in der Testversion eingesetzten Browsers von AutoFOCUS.
3. **import quest.testing.application.***- Import der Klassen zur Ansteuerung der Testapplikation für AutoFOCUS.
4. **import quest.testing.application.TiniTest**- Import der Klassen zur Steuerung von AutoFOCUS Testapplikation "TiniTest".

Die nachfolgende Abbildung 8 gibt grobskizziert den Aufbau der Datei HelpSystem.java wieder. Weitere Detailinformationen finden sich im Anhang B wieder.



Abbildung 8: Klassenstruktur HelpSystem.java

Zusätzlich verfügt die Datei HelpSystem.java noch über die Klassen "SwitchWhatIsHelp" und "WhatIsHelpDetector", die die Abwicklung der "What is ..."-Hilfe steuern. Der Ablauf des Mechanismus in der "What is..."-Hilfe wird im Kapitel 6.3.2 genauer geschildert und durch die Abbildung 11 in Form eines Sequenzablaufdiagramms dargestellt.

SwitchWhatIsHelp Diese Klasse ist für die Erkennung des Mauseklicks durch den Benutzer zuständig. Dieses ist auch Erkennbar an der Eigenschaft der Schnittstelle des ItemListeners (implements ItemListener).

WhatIsHelpDetector Diese Klasse ist für die Erkennung des Mauseklicks durch den Benutzer im Browser zuständig. Dieses ist auch Erkennbar an der Eigenschaft der Schnittstelle des TreeSelectionListener (implements TreeSelectionListener).

6.1.2 Aufrufende Klassen

Die Klassen "TiniTest" und "ModuleEditor" rufen beide unabhängig voneinander das Hilfesystem auf durch `HelpSystem helpsystem = new HelpSystem();`. (Diese Lösung ist nur suboptimal, aber für exemplarische Anschauungen im Prototyp ausreichend. Beispielsweise kann auf diese Weise ohne Probleme die Editorhilfe demonstriert werden.)

TiniTest.java TiniTest ist die Hauptklasse der Testanwendung für AutoFOCUS. Für sie wird das Hilfesystem als Instanz angelegt, wie bereits in den einleitenden Worten beschrieben.

Der Zugriff anderer Klassen auf das Hilfesystem erfolgt im Anschluss über die Methode "getHelpSystem()", die als Rückgabewert das Hilfesystem zurückgibt. `public HelpSystem getHelpSystem() return helpsystem;`

Des Weiteren ist es notwendig, dass das Hilfesystem sich am Browser registriert, der in TiniTest adressiert wird. Dieses erfolgt durch den Aufruf der Methode `helpsystem.initValidatorWhatIs(browser);`

ModuleEditor.java ModuleEditor legt eine eigene Instanz des Hilfesystems an und nist lediglich aus demonstrativen Zwecken im Rahmen des Prototypings so suboptimal gelöst. Beim ModuleEditor handelt sich im Gegensatz zu TiniTest um einen Editor und nicht um einen Browser.

6.2 Aufbau des Menüs "Hilfe" in Editor/Browser

Um im Editor/Browser das Hilfemenü aufzubauen, wird die Methode `insertHelpMenuBar(JMenuBar jMenuBar);` aufgerufen.

Der Aufbau des Menüs für die Testapplikation `TiniTest` wird in der Klasse `“TiniMenuBar“` vorgenommen. Hier kommt es dann zum Aufruf der genannten Methode durch die Variable `“tiniTest“` der Klasse `TiniTest`. Die nachfolgende Methode `“getHelpSystem“` gibt eine Variable vom Typ `HelpSystem` zurück, welche wiederum die Methode `insertHelpMenuBar` mit dem übergebenen Parameter `“this“` vom Typ `JMenuBar` aufruft.

```
tiniTest.getHelpSystem().insertHelpMenuBar(this);
```

Der Aufbau des Menüs ist dann in der Klasse `HelpSystem` festgelegt. Über den Menüpunkt auf der Menüleiste hinaus

```
JMenu helpMenu = new JMenu(“JavaHelp“);
```

gibt es dann noch die drei Unterpunkte für die Hilfe in den drei Granularitätsstufen:

```
JCheckBoxMenuItem whatIsHelpItem;
```

```
JMenuItem specificHelpItem helpItemTOC;
```

Durch Einbindung in die bereits bestehende Menüleiste ist der Befehl `jMenuBar.add(helpMenu)`; notwendig. Selbstverständlich gilt dieses ebenso für die drei Menüpunkte.

Damit auf den Mouseclick des Benutzers reagiert werden kann, ist es notwendig einen `ActionListener` zu implementieren. Der `ActionListener` verlässt dann im Laufe der Anwendung auf den Mouseclick des Benutzers eine Aktion. In diesem Fall besteht die Aktion darin das richtige `HelpSet` anzusteuern, mit dem gearbeitet werden soll. Für die weiteren Schritte ist im Wesentlichen `JavaHelp` verantwortlich.

```
ActionListener helper = new CSH.DisplayHelpFromSource(hb);
```

```
helpItemTOC.addActionListener(helper);
```

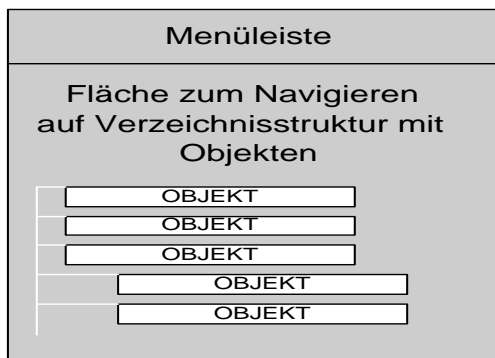
```
specificHelpItem.addActionListener(helper);
```

6.3 Realisierung der `“What is ...“-Hilfe (Direkthilfe)`

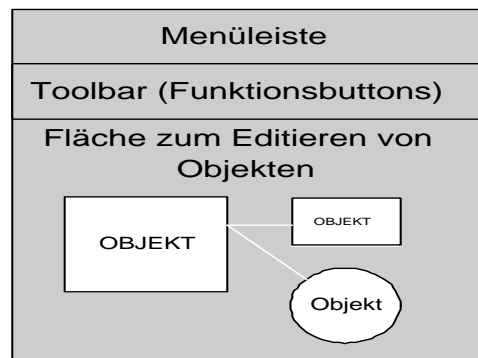
In der nachfolgenden Abbildung 9 ist schematisch dargestellt, wie Browser und Editoren aufgebaut sind. Diese Abbildung dient lediglich dazu zu vermitteln, dass es durch `JavaHelp` sehr leicht ist, typische Java-spezifische Bereiche in Editoren und Browsern abzudecken, jedoch nicht die Java-unspezifischen Eigenschaften, wie Objekte in Editoren oder Browsern.

Selbstverständlich sieht die `“What is...“-Hilfe` des Hilfesystems so-

Browser-Struktur



Editor-Struktur



Erklärung der Farben:

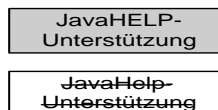


Abbildung 9: Struktureller Aufbau von Browser und Editor

wohl für die grau unterlegten Bereiche wie Menüleiste oder Toolbar eine Hilfe vor, wie auch für die AutoFOCUS-spezifischen Objekte. Der Unterschied ist lediglich die Unterstützung des Systems JavaHelp.

6.3.1 Menüleisten und Funktionsbuttons

Ein Browser-Fenster besteht aus Menüleiste(n) und einem Feld, in dem in der angebotenen Struktur navigiert werden kann. Die Navigationsstruktur ist über den Datentyp "JTree" realisiert. So ist es auch in AutoFOCUS gelöst.

Um die in JavaHelp angebotene Direkthilfe nutzen zu können, sind bestimmte Konventionen einzuhalten, wie beispielsweise bestimmte Datentypen, für die Direkthilfe unterstützt wird. Version 1.1.3 von JavaHelp unterstützt derzeit die Datentypen

- * "MenuItem" und
- * "Component".

Zusammen mit der so genannten Map-Id für das Auffinden der dazugehörigen Hilfeinformationsdatei werden in JavaHelp hierfür die Methoden "setHelpIDString" bzw. "getHelpIDString" zur Verfügung gestellt.

- * public static void setHelpIDString(MenuItem *comp*, String *helpID*)
- * public static void setHelpIDString(Component *comp*, String *helpID*)

Während JavaHelp also ohne Probleme Menüleisten und darin enthaltene Einträge unterstützt (Datentyp MenuItem) und auch Funktionsbuttons und ähnliches unterstützt (Datentyp Component), werden für Browser-Strukturen zur Navigation bisher keine Unterstützung zur Direkthilfe angeboten. (Die Überführung von JTree-Datenstrukturen auf die von JavaHelp geforderten Datentypen sind nicht möglich.)

6.3.2 Browser: Objekte in Navigationsstruktur

Da die “What is...“-Hilfe (Direkthilfe) für Objekte in der Navigationsstruktur nicht durch JavaHelp unterstützt wird, diese Funktionalität aber unabdingbar für das zu realisierende Hilfesystem ist, muss diese Funktion durch eigene Implementierung unterstützt werden.

Hierzu ist es besonders wichtig, dass für den Benutzer keine Veränderungen in der Bedienung der Hilfe gefordert werden. Um dieses zu gewährleisten ist es für die Implementierung zum einen wichtig, dass das Hilfesystem zum einen bemerkt, dass Hilfe durch den Benutzer angefordert wird und zum anderen ist es notwendig, dass die korrekte Hilfeinformation zu dem ausgewählten Objekt angegeben wird.

Anschauliche Schritte zur Realisierung Nachfolgende Abbildung 10 zeigt die wesentlichen Schritte zur Realisierung der “What is ...“-Hilfe für Objekte in der Navigationsstruktur. Die Erläuterungen zu der in der Skizze enthaltenen Nummerierung finden sich in der untenstehenden Tabelle.

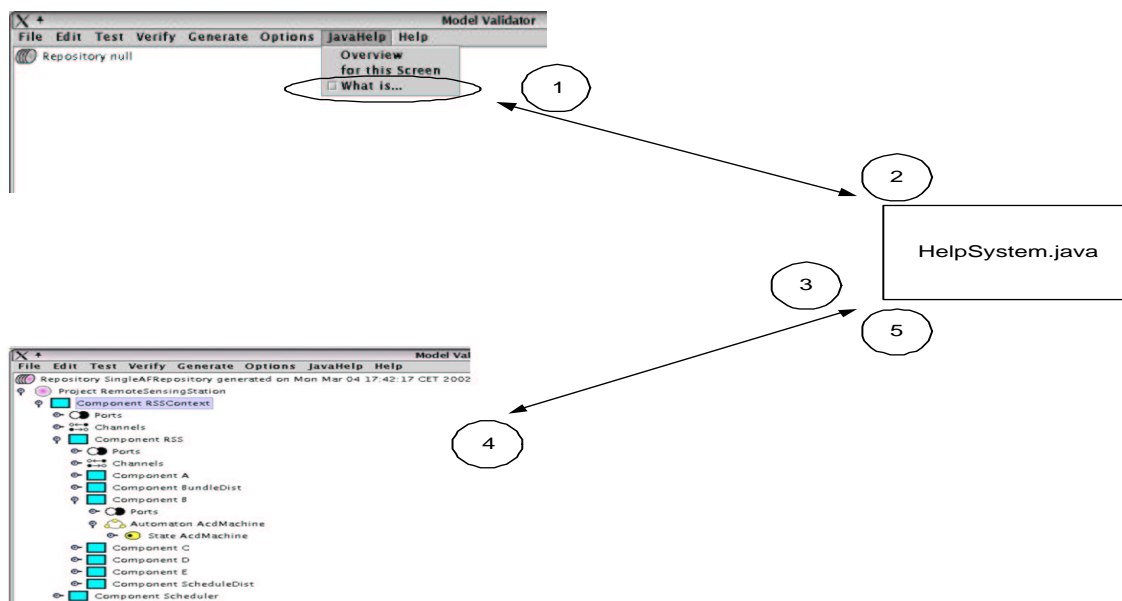


Abbildung 10: Zugriffsmöglichkeiten auf das Hilfesystem über die Menüleiste

Nummerierung	Erläuterung
1	Der Benutzer wählt den Menüpunkt "What is..." aus.
2	Das Hilfesystem erkennt die Auswahloption und
3	veranlasst eine Registrierung am "TreeSelectionListener", damit eine Benachrichtigung mit Übergabe des Objektnamens durch den Listener an das Hilfesystem erfolgt, sobald der Benutzer ein Objekt ausgewählt hat.
4	Nachdem der Benutzer dann das Objekt ausgewählt hat, gibt der TreeSelectionListener den Namen des Objekts zurück.
5	Das Hilfesystem öffnet das Hilfefenster und zeigt den Hilfeinformationstext passend zum ausgewählten Objekt an.

Schematischer Ablauf im Programmcode Mit dem Erkennen des Objektnamens, ist die für das Auffinden der korrekten Hilfeinformationsdatei die eindeutig zuordnungsbarer HelpID notwendig. Die HelpID basiert auf der im vorherigen Kapitel 5 eingeführten Verzeichnisstruktur, die angedeutet in Abbildung 5 gezeigt wird.

Identifikation der Objektbezeichner in AutoFOCUS Um den Namen von Objekten in der Browserstruktur oder der Modellierungsebe-

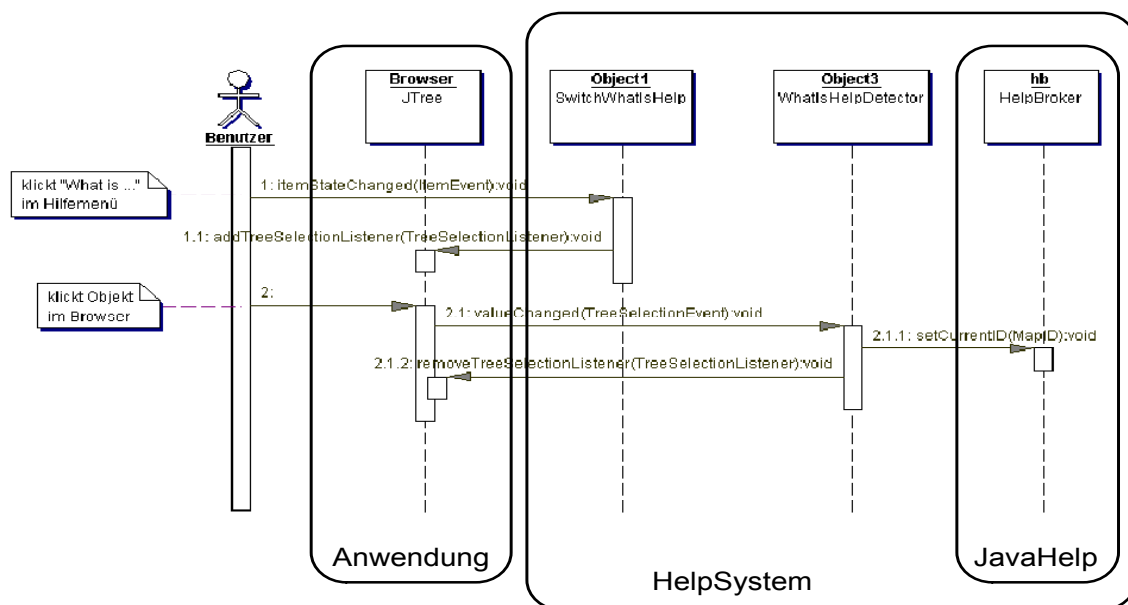


Abbildung 11: Realisierung der “What is ...“-Hilfe

ne von AutoFOCUS für das korrekte Zuweisen von Hilfeinformati-
ontexten zu finden, ist ein lesender Zugriff auf die AutoFOCUS Modell-
struktur notwendig. Hierfür ist der Import der folgenden “packages“
notwendig:

- * quest.metamodel.model.*;
- * quest.mmm.model.*;

Der sequenzielle Ablauf der aufrufenden Klassen wird in der Abbil-
dung 12 dargestellt. Der dazugehörige Programmcode findet sich un-
terhalb der Abbildung.

Nachfolgend ist ein Programmcode-Auszug des lesenden Zugriffs auf
den Objektamen in der Browserstruktur angegeben:

```
MMElement myMMElement = (MMElement) selectedNode.getUserObject();
MetaModelElement myMetaModelElement = myMMElement.getMetaModelElement();
String myName = myMetaModelElement.getName();
```


zu finden, einen lesenden Zugriff auf die AutoFOCUS Modellstruktur durchzuführen. Die nachfolgenden Zeilen in kursiver Schrift beschreiben den Zugriff.

Eine andere Art der Beschreibung in Form eines Klassendiagramms kann der nachfolgenden Abbildung 13 entnommen werden.

```
import quest.metamodel.model.*;
import quest.mmm.relation.*;
import quest.mmm.model.*;
```

-----snip-----

```
MetaModel myMM = quest.metamodel.model.MyMetaModel.getMetaModel();
```

```
for(Enumeration elems = myMM.getClasses(); elems.hasMoreElements();)

```

```
Object myMME = (Object)elems.nextElement();
```

-----snip-----

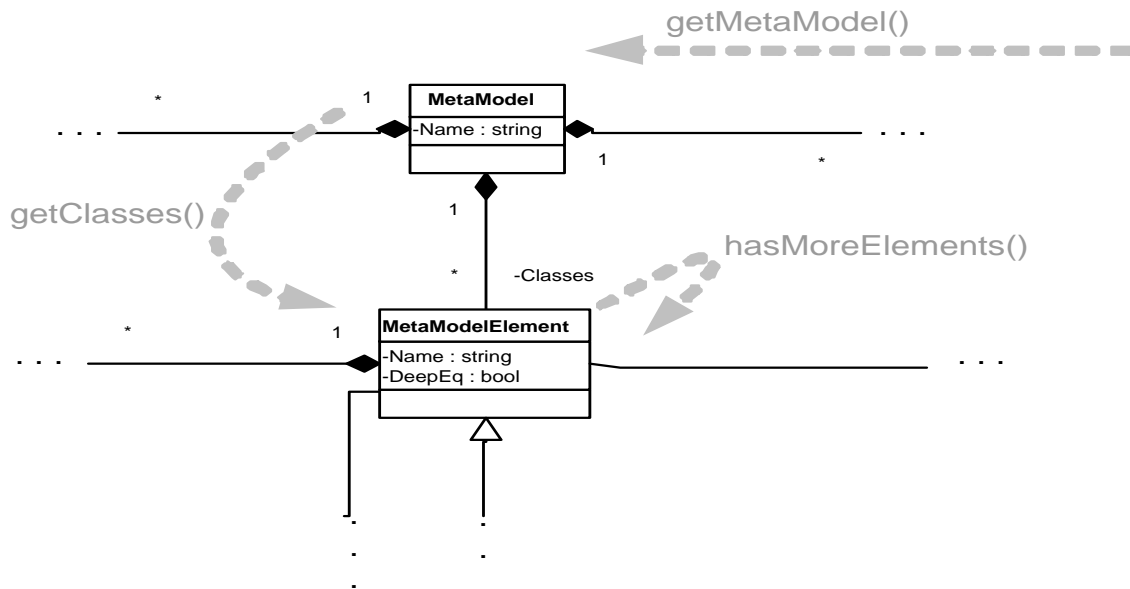


Abbildung 13: Klassendiagramm, welches den Weg zum Auslesen von Objektnamen darstellt

6.3.4 Menüleiste

Eine Menüleiste ist in der Programmiersprache Java vom Datentyp "MenuBar", welches gemäß Java API eine spezielle Form des Datentyps "Component" darstellt und somit von JavaHelp und den darin zur Verfügung stehenden Methoden unterstützt wird. Ebenso ist es bei den Menüeinträgen in der Menüleiste. Diese sind vom Datentyp "MenuItem" und werden direkt durch JavaHelp unterstützt.

Um den richtigen Hilfeinformationstext dem jeweiligen Menüeintrag zuzuordnen kann sich an den Namen im Menü orientiert werden. Es gibt beispielsweise den Menüpunkt File -> Open. Zusammen mit dem jeweiligen Editor stellt dieser Pfad einen eindeutigen Bezeichner für eine HelpID dar, wie er auch in der Verzeichnisstruktur in Abbildung 5 aufgezeigt wird.

6.3.5 Toolbar

Da bisher keine Editoren in AutoFOCUS zur Verfügung stehen, kann die "What is..."-Hilfe für Toolbars bisher nicht implementiert werden. Generell wird die Hilfe für Toolbars vollständig durch die JavaHelp-Technologie unterstützt, da Elemente von Toolbars, die sog. "Funktionsbuttons" vom Datentyp "Component" sind und somit wie bereits zuvor beschrieben durch JavaHelp unterstützt werden.

6.4 weitere Implementierungen

6.4.1 Realisierung der "for this Screen ..." -Hilfe (Editor/Browser-Hilfe)

Die mittlere Granularitätsstufe "for this Screen ..." wird im Moment so realisiert, dass jeder Editor Browser eine eigene Instanz des Hilfesystems hat, die jeweils auf das identische HelpSet zugreift. Durch die eigene Instanz kann *prototypisch* sichergestellt werden, dass für jeden Editor/Browser die jeweils korrekte Hilfeinformationsdatei im Hilfefenster geöffnet wird. (Der "anschauliche" Charakter der Hilfe kann auf diese Weise dargestellt werden.)

6.4.2 Realisierung des "Overview"s

Der Overview stellt die größte Stufe der Granularität dar und bedeutet lediglich die Ansteuerung einer festen Hilfeinformationsdatei.

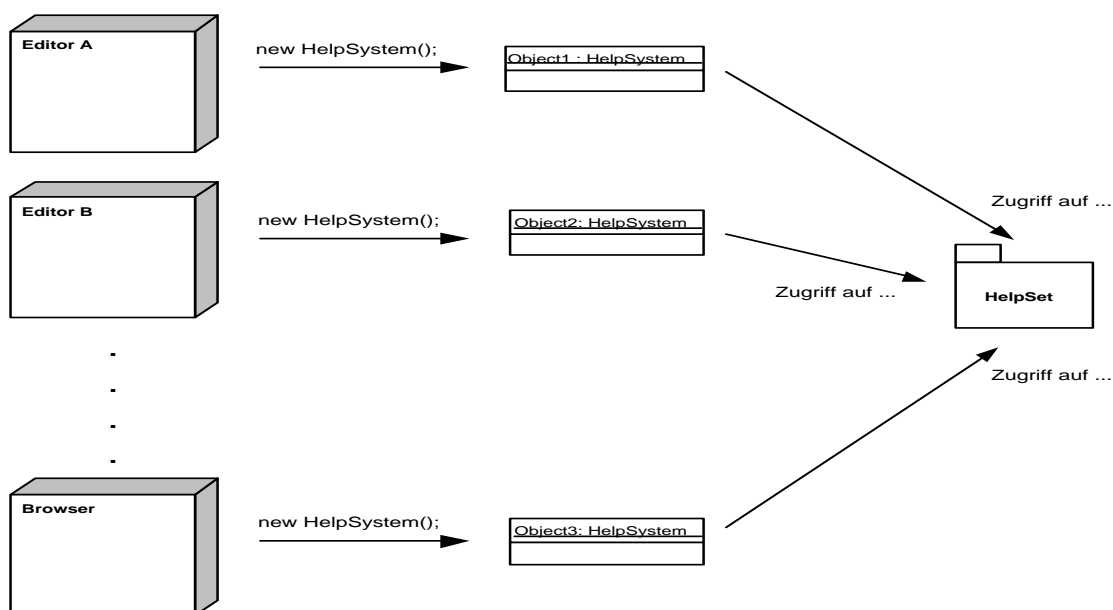


Abbildung 14: Zugriff der verschiedenen Instanzen des Hilfesystems auf dasselbe HelpSet

Dieses wird vollständig durch JavaHelp unterstützt und kann durch folgende Zeile programmiertechnisch beschrieben werden:

```
CSH.setHelpIDString(helpItemTOC, "overview");
```

Der erste Parameter "helpItemTOC" bezeichnet hierbei den im Hilfesystem verwendeten Namen für den Menüpunkt "Overview". Die Bezeichnung "overview" steht für die Hilfeinformationsdatei, die aufgerufen werden soll.

6.5 Erweiterungen am Hilfesystem

Es ist grundsätzlich möglich, Erweiterungen am Hilfesystem vorzunehmen. Diese Erweiterungen können sich zum einen auf zusätzliche Funktionalitäten beziehen oder auf inhaltliche Erweiterungen, wenn beispielsweise neue Editoren dazukommen oder ähnliches.

Bei Letzterem ist zu beachten, dass

- * (a) Hilfeinformationstexte hinterlegt werden müssen,
- * (b) die die Navigationsstruktur der jeweiligen Hilfetechnologie angepasst werden muss (z.B. Änderung der verantwortlichen XML-Dateien.)

- * (c) und das die eindeutige Adressierung gegeben sein muss, damit die unter (b) erwähnte Navigation gestartet werden kann.

Erweiterungen bezüglich der Funktionalität werden im nachfolgenden Kapitel 7 "Ausblick" diskutiert.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Bei der Entwicklung des modellbasierten Hilfesystems für AutoFOCUS ist nach der Einführung der wichtigsten Begrifflichkeiten im Themengebiet Benutzerunterstützungssysteme ein Konzept angefertigt worden, welches zum einen die Anforderungen an das Hilfesystem in AutoFOCUS definieren, dessen Umfang sowie die verschiedenen Zugangsmöglichkeiten. Im anschließenden Design des Systems wurden Entscheidungen zur Umsetzung des Systems entwickelt und in der anschließenden Realisierung des Prototyps zum Teil implementiert.

Das Projekt gibt Bestätigung darüber, dass ein modellbasiertes Hilfesystem für AutoFOCUS mit Hilfe gängiger Hilfetechnologien realisiert werden kann. Es wird dafür auf ein im Rahmen der Konzeption entwickelte Modell zurückgegriffen, welches im Wesentlichen der Ablagestruktur für den Entwickler der Hilfeinformationstexte entspricht. Der Zugriff auf das Modell und die für den Benutzer sichtbare Visualisierung der Hilfe erfolgt durch Einsatz der frei auf dem Markt erhältlichen Hilfetechnologie JavaHelp, auf die in diesem Projekt im Rahmen des Prototyping zurückgegriffen wird.

Beim Thema "Kontextsensitivität" des Hilfesystems wird besonders der Bereich des zugrunde liegenden Modells von AutoFOCUS betrachtet, auf den der Benutzer der AutoFOCUS-Anwendung durch Modellierung seines Systems indirekten Zugriff erhält. Die Schnittstelle zwischen dem AutoFOCUS-Modell und dem Hilfesystem ist im Zuge dieses Projekts kurz behandelt worden. Bei der Realisierung des Prototyps konnte als Herausforderung im Zusammenspiel mit JavaHelp die Umsetzung der kontextsensitiven Hilfe festgestellt werden. JavaHelp unterstützt klassische Swing-Objekte aus Java, kann jedoch beispielsweise bezüglich des Datentyps "JTree" keine Verfeinerungen vornehmen, so dass es hier gilt, alternative Lösungen zu entwickeln, die die Konsistenz des Hilfesystems in der gesamten zugrunde liegende Hauptanwendungen AutoFOCUS sicherstellen.

Zusätzlich zum rein konzeptionellen und teilweise realisierten technischen Erfolg des Projekts, muss abschließend noch festgestellt werden, dass es sich im Laufe des Projekts als notwendig und hilfreich herausgestellt hat, das Hilfesystem auch methodisch und didaktisch bezüglich seines Inhaltes zu gestalten. Hierzu könnte beispielsweise ein "Content-Management-System" als Hilfsmittel eingesetzt werden.

7.2 Ausblick

Im folgenden Ausblick wird das Potential vom modellbasierten Hilfesystem in AutoFOCUS hinsichtlich einer Auswahl von Aspekten beleuchtet und kann somit als motivierender Teil für die weiteren Arbeiten in AutoFOCUS dienlich sein.

7.2.1 Aktives Hilfesystem

Ein aktives Hilfesystem liegt vor, sobald das System von sich aus aktiv wird, um dem Benutzer Hilfe anzubieten. Aus Sicht des Benutzers kann es einige Vorteile aufweisen, die ihn bei seiner Arbeit mit AutoFOCUS in Hinblick auf z.B. seine Effizienz unterstützen. Um diese Aussage zu stützen, seien nachfolgend drei Beispiele genannt.

– ***Einbindung der AF-Konsistenzsicherung***

Eine solche Situation ist zum Beispiel erreicht, sobald der Benutzer fehlerhafte Eingaben im System vornimmt, die nicht mit einer erfolgreichen Aktion abgeschlossen werden können. Hierzu zählen Verletzungen von Konsistenzbedingungen im AutoFOCUS-Metamodell oder auch. Es wäre hilfreich und unterstützt die effiziente Arbeitsweise des Benutzers, wenn das Hilfesystem von sich aus frühzeitig aktiv wird und den Benutzer auf seine Fehler hinweist.

Hierbei ist dann eine sehr nahe Zusammenarbeit mit der Hauptanwendung notwendig, die laufend Konsistenzüberprüfungen durchführen muss und Inkonsistenzen zeitnah an das Hilfesystem meldet. Die Hilfeinformationstexte könnten je nach eingetretenem Fehlergrad und der Art des Fehlers gegebenenfalls vom Hilfesystem generiert werden.

– ***Eingabeunterstützung***

Wird der Benutzer vom System aufgefordert Eingaben im System zu machen, so könnte ein aktives Hilfesystem eine Auswahl der möglichen Eingabedaten in der korrekt geforderten Form zur Verfügung stellen. Auf diese Weise vermeidet der Benutzer fehlerhafte Eingaben und die Konsistenz seines Systems bleibt erhalten.

– ***Optimierung von Arbeitsschritten***

Führt der Benutzer bestimmte Tätigkeiten in AutoFOCUS auf sehr umständliche Art und Weise durch, so könnte das Hilfesystem das Verhalten beobachtet haben und ihn beim erneuten ineffizienten Umgang mit dem System auf eine Optimierung seiner Arbeitsschritte hinweisen.

Hierbei könnte der Einsatz von Agentensystemen sinnvoll sein.

7.2.2 Prozessunterstützung

Während das Hilfesystem derzeit für die Unterstützung des Benutzers vorwiegend auf den Umgang mit Editoren/Browsern abzielt, wäre ebenfalls eine prozessunterstützende Hilfestellung denkbar. (Die in AutoFOCUS notwendigen "Arbeitsschritte" zum korrekten und erfolgreichen Umgang mit dem Werkzeug bedeuten hierbei die verschiedenen Prozessschritte, die z.B. an verbreiteten und üblichen Entwicklungsprozessen orientiert sein können.)

Begonnen werden könnte hierbei direkt beim Start der AutoFOCUS-Anwendung, indem der Benutzer im System begrüßt wird und auf die ersten Prozessschritte hingewiesen wird. Nachfolgende Abbildung gibt einen Überblick über die verschiedenen Möglichkeiten so einer unterstützenden Hilfemaßnahme im "Einstiegs"-Prozess.

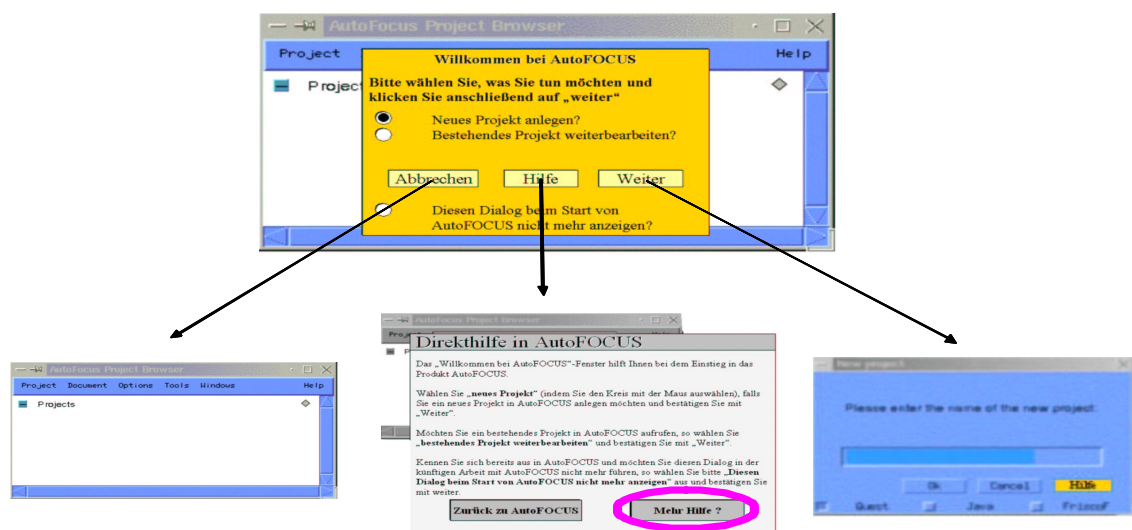


Abbildung 15: Beispielhafte Benutzerführung beim Einstieg in die Hauptanwendung AutoFOCUS

Das hierbei entstehende System ist dem eines *Beratungssystems* gleichzusetzen, da der Benutzer beim Umgang mit der Anwendung durch das System beraten wird.

Einbindung der Tutorhilfe "Nelli" Bevor obige Erweiterung zur Erweiterung des Hilfesystems angestrebt wird, sollte mit Vorrang die Integration des bestehenden Online-Tutorsystems "Nelli" in das konzipierte Hilfesystem vorgenommen werden und damit in die Anwen-

dung AUTOFOCUS einzubinden. Nelli wird zur Zeit unabhängig von der Anwendung zur Einarbeitung in die Anwendung eingesetzt.

7.2.3 Inspektion von Objekten

Der Benutzer des Systems AutoFOCUS könnte daran interessiert sein, verschiedene eingesetzte Objekte in seinem modellierten System genauer inspizieren zu wollen, indem er die Möglichkeit hat, auf Assoziationen des zugrunde liegenden Metamodells zurückzugreifen.

Das Hilfesystem könnte demnach eine Inspektionsmöglichkeit anbieten, durch die der Benutzer Verknüpfungen in seinem System besser verstehen kann. Beispielsweise könnte es beim Klicken auf die Komponente A die Möglichkeit geben, die an dieses Objekt direkt anliegenden Objekte genauer inspizieren zu können. Die Informationen hierfür liefert das Metamodell und die dazugehörigen Hilfeinformationstexte kommen aus dem Hilfesystem selbst.

7.2.4 Wartung und Erweiterung

Um die Konsistenz im Hilfesystem auch bei Weiterentwicklung des Systems zu gewährleisten ist es notwendig, einen Prozess für die Wartung und Weiterentwicklung zu definieren, der Entwicklern neuer Komponenten im System AutoFOCUS dazu "zwingt", Hilfeinformation an einer zu definierenden Stelle im System abzulegen.

Hilfeinformation korrekt angelegt Existiert der Hilfeinformationstext des Entwicklers und ist dieser korrekt abgelegt, so ist ein Automatismus zu entwickeln, der die notwendigen Daten im Rahmen des Kompilierungsvorgangs in das Hilfesystem einspeist. Hierzu gehören beispielsweise das Angleichen der Navigationsstrukturen innerhalb des Hilfesystems, die Aktualisierung der Indexierung sowie der Suchfunktion im Hilfefenster, wie auch die Integration in das Inhaltsverzeichnis.

Hilfeinformation nicht korrekt angelegt Fehlen die Hilfetextinformationen, so könnte beispielsweise der Kompilierungsvorgang abgebrochen werden und eine entsprechende Nachricht an den Entwickler gegeben werden mit der Bitte um Behebung des Mangels.

7.2.5 Automatische Generierung

Um die Modellstruktur des Hilfesystems stabil zu erhalten, könnte es angebracht sein, die Hilfeinformationstexte der Entwickler neuer Komponenten im System AutoFOCUS nicht direkt im zugrunde liegenden Modell einpflegen zu lassen, sondern an einer anderen festzulegenden Lokation. Ein Generierungsprozess könnte dann im Anschluss die Aufgabe verfolgen, diese Hilfeinformationstexte "einzu-sammeln" und in das Modell zu integrieren. Nachfolgende Abbildung 16 stellt skizziert den Generierungsablauf dar.

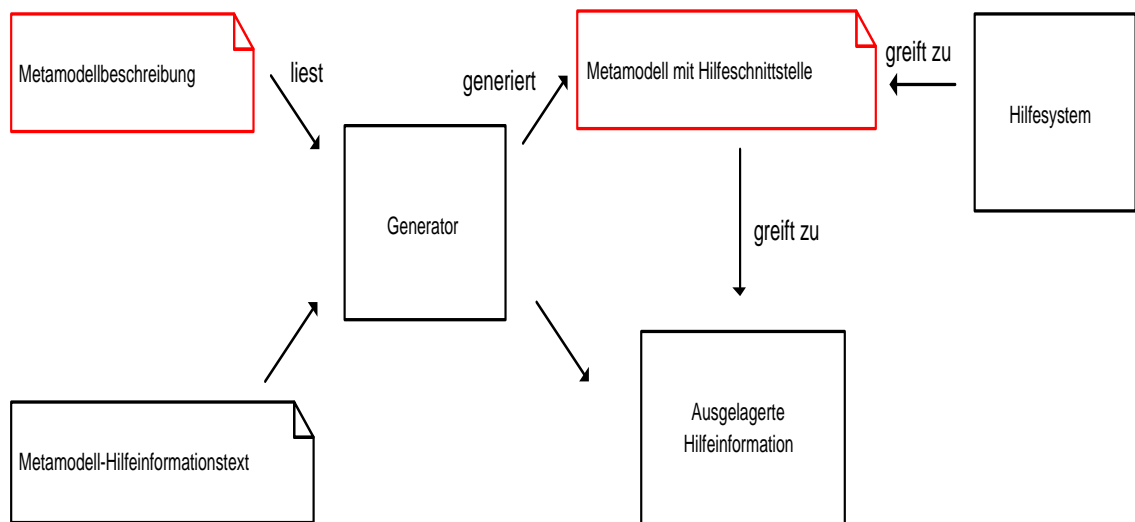


Abbildung 16: Automatische Generierung und Erweiterung des Hilfesystems

7.2.6 Hilfesystem für Entwickler/innen

Das Hilfesystem ist derzeit stark am Benutzer orientiert, könnte aber in Zukunft auch für den Entwickler als Unterstützung dienen, indem beispielsweise eine spezielle Entwicklungsumgebung in Form eines Views zur Verfügung gestellt wird. Innerhalb dieser Entwicklungsumgebung könnte dann Hilfeinformation ebenfalls mit Hilfetechnologie, wie beispielsweise JavaHelp zur Verfügung gestellt werden.

Die Hilfeinformationstexte würden in diesem Zusammenhang andere Inhalte aufweisen als die der Benutzer. Allerdings ist die Verzeichnisstruktur an dieser Stelle leicht erweiterbar, so dass eine Erweiterung keine Herausforderung bedeutet.

Um die Hilfeinformationstexte dem Benutzer nicht in Form des in der Hilfetechnologie zur Verfügung gestellten Inhaltsverzeichnisses sichtbar zu machen, könnte beispielsweise ein neues Objekt des Hilfesystems erweitert um Hilfeinformationen für den Entwickler in Folge objektorientierter Vererbung zur Verfügung gestellt werden.

Bei Aufruf der Entwicklungsumgebung wird das erweiterteSSystem (anstatt des normalen Benutzersystems) eingebettet.

A Pflichtenheft

Zielbestimmungen

Musskriterien (Minimum)

Bei den Musskriterien wird versucht, dass Minimum an den von Balzert[BAL98] vorgeschlagenen Inhalten in einem Hilfesystem zu orientieren. Hier-nach sollten bei den Inhalten des Hilfesystems im Wesentlichen die folgenden Punkte berücksichtigt werden:

- * Inhalte zu den im aktuellen Kontext wählbaren Objekten, Funktionen bzw. Kommandos sowie Optionen zu Objekten und Funktionen
- * Bedeutung von Funktionstasten, insbesondere von mehrfach belegten Funktionstasten und Mausknöpfen

Balzert erwähnt in diesem Zusammenhang noch weitere Punkte, die in diesem Projekt zunächst nicht den Minimalanforderungen entsprechen und daher im nachfolgenden Kapitel A plaziert sind.

Über die zuvor beschriebenen, eher allgemeinen Ausführungen zu Inhalten in Hilfesystemen hinaus, sind im Rahmen dieses Projekts die folgenden Funktionalitäten geplant:

- * **passive Hilfe**

Die passive Hilfe bedeutet in der Realisierung, dass der Benutzer über verschiedene Zugriffsmechanismen die Möglichkeit hat, von sich aus Hilfe anzufordern. Das Hilfesystem wird demnach nicht selbständig aktiv, sondern muss durch den Benutzer der AutoFOCUS-Anwendung aktiviert werden.

- * **dynamische Hilfe (kontextsensitiv)**

Der Benutzer der AutoFOCUS-Anwendung bekommt mit dem zu realisierenden Hilfesystem die Möglichkeit, kontextbezogene Hilfeinformation anzufordern. Kontextsensitivität wird im Rahmen der ersten Realisierung auf die Fenster- bzw. Editor-Ebene sowie die Objekte innerhalb desgleichen beschränkt. Anschaulich bedeutet das also, dass der Benutzer im Falle der Hilfeanforderung aus dem SSD-Editor, Hilfe zu diesem speziellen Editor bekommt. Fordert der Benutzer innerhalb des SSD-Monitors mit Hilfe der Funktion "Direkthilfe" (auch bekannt als "What is?" oder "?") Information zu einer bestimmten Komponente an, so wird das Hilfesystem Information zu dieser speziellen Komponente die Eigenschaften anzeigen.

* **uniform & individuell**

Auf der einen Seite, wird das zu realisierende Hilfesystem uniforme Hilfeinformation enthalten. Die Individualisierung wird auf der anderen Seite dadurch zu einem bestimmten Grad erreicht, dass verschiedene Detaillierungsebenen der uniformen Information angestrebt werden. Als Annahme werden unterschiedliche Benutzergruppen (vgl. Kapitel 4) definiert, deren Anforderungen an Hilfeinformation durch die Detaillierungsebenen abgebildet werden sollen.

Während sich die Eigenschaft der Uniformität in diesem Zusammenhang auf die immer gleichen Textpassagen bezieht, ist mit der Eigenschaft der Individualität die Berücksichtigung verschiedener Benutzergruppen gemeint.

* **Integration von Tutorsystem "Nelli" in das Hilfesystem**

Neben den standardisierten Hilfeinformationen, die aus der Fenster-/Editor-Umgebung bzw. innerhalb dieser angefordert werden können, hat der Benutzer zusätzlich die Möglichkeit das Tutorsystem "Nelli" aufzurufen. Der Zugriff zu diesem Tutorsystem kann aus den Hilfesystemtexten heraus realisiert werden oder durch Direktzugriff in einem Menüpunkt angesteuert werden.

* **Einleitende Führung für den neuen Benutzer in AF2**

Es soll die Möglichkeit geben, neue Benutzer des AutoFOCUS-System eine einleitende Sequenz in das System zur Verfügung zu stellen. Hierbei ist zu beachten, dass es sich lediglich um den ersten Schritt der AutoFOCUS-Anwendung handelt und in keiner Weise, den Benutzer anleitet, erfolgreich ein System zu modellieren.

Der Benutzer soll durch die Einleitende Führung unter anderem auch auf das ihm zur Verfügung stehende Hilfesystem aufmerksam gemacht werden.

* **Tooltips**

Bei der Realisierung des eigentlichen Hilfesystems ist zusätzlich darauf zu achten, dass die in der Anwendung zur Verfügung stehenden Tooltips vollständig zur Verfügung stehen und den Anwender des Systems inhaltlich bei der Arbeit mit dem System unterstützen.

* **Individuelle Ergänzungen der Hilfetexte durch den Benutzer**

Dem Anwender des Systems wird mit dieser Funktionalität die Möglichkeit gegeben, eigene Hilfeinformation im System zu ver-

fassen.

Wunschkriterien

Wie bereit in den Musskriterien im vorangegangenen Unterkapitel begonnen, sollten nach Balzerts⁴ Vorstellungen über die bereits genannten Anforderungen an die Inhalte eines Hilfesystems hinaus auch die folgenden Punkte in der Realisierung berücksichtigt werden.

* Unterstützung bei Eingabedialogen

z.B.:

- mögliche Eingaben zur Auswahl anbieten
- automatische Vervollständigung von Eingaben
- Spezifische Fehlermeldungen und Fehlererklärungen, ggf. mit Angabe einer Methode zur Fehlerkorrektur
- Erläuterungen zu Ergebnissen von Funktionsausführungen

Darüber hinaus wäre es im Rahmen dieses Projekts auch angebracht, dem Benutzer im Rahmen des Hilfesystems eine Prozessunterstützung anzubieten, die den Anwender durch seine Arbeit begleitet und dabei unterstützt, indem ihm die nächsten Schritte jeweils als Orientierung vorgegeben werden.

Abgrenzungskriterien

- Software-Agents für Beratungssystem (-> Wizards)

Produkteinsatz

Anwendungsbereiche

Unterstützung bei Benutzung der Anwendung AF2

Zielgruppen

Anwender von AutoFOCUS (Anfänger und Geübte)

Produktübersicht

Aufzeigen der wesentlichen Prozesse, die mit dem System unterstützt werden sollen.

⁴vgl. ebenda

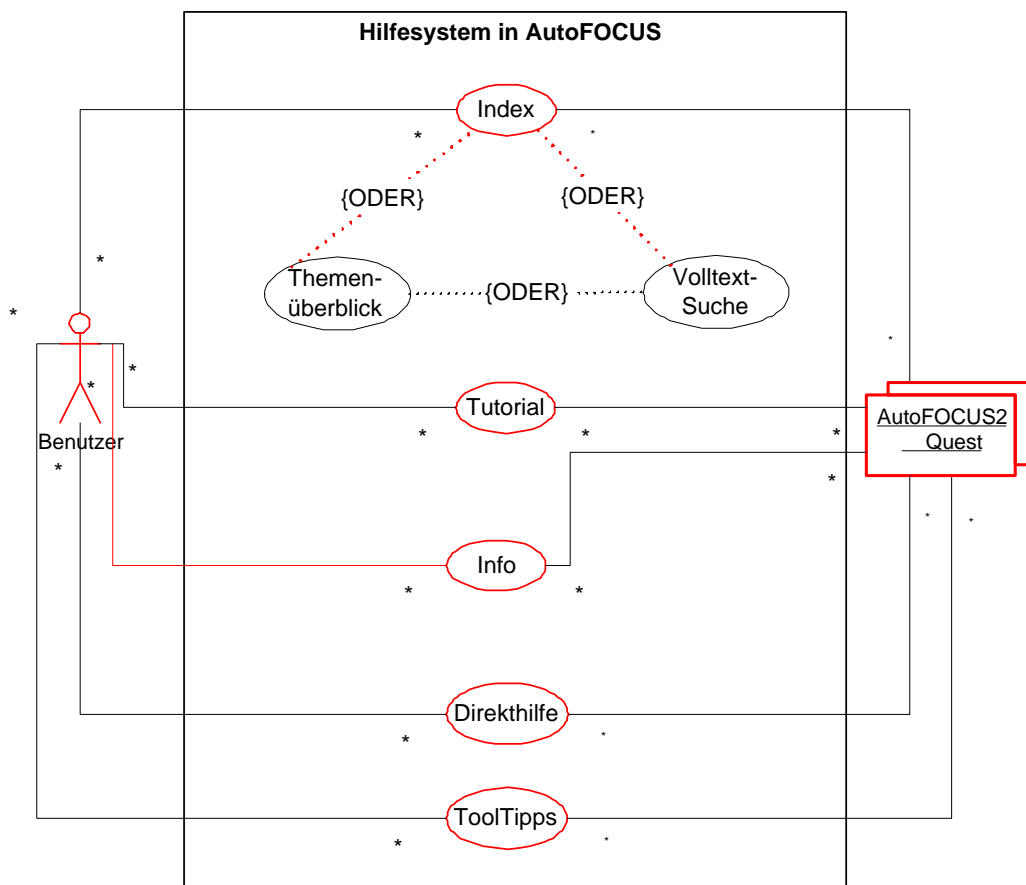


Abbildung 17: Übersicht der zu realisierenden Anforderungen

Produktfunktionen

(Detaillierte Auflistung der angebotenen Funktion durch das Hilfesystem) - Einführung für den neuen Benutzer - Zugangsfunktionen zum Tutorsystem - auflisten - Zugangsfunktionen zum Hilfesystem - auflisten - Volltext-Suchfunktion in Hilfesystem (Registerkarten-> JavaHelp) - Index-Suchfunktion (Registerkarten-> JavaHelp) - Inhaltsverzeichnis / Überblick der Hilfethemen(Registerkarten-> JavaHelp) - Tooltips - kontextsensitive Hilfe über rechte Maustaste zu View-Objekten (Komponenten der Modellierung)

Produktdaten

- Kurze und ausführliche Hilfetexte in Html - Struktur der Daten in XML beschreiben - Tutortool Nelli (-> html-Dokument) - Tooltips-Texte

Qualitätsanforderungen

- konsistente Struktur - verschiedene Zugriffsmöglichkeiten auf das Produkt - "sinnvolle" Hilfe (z.B. Eigenschaften von Komponenten könnten recht langweilig sein!) - Berücksichtigung verschiedener Benutzergruppen (Anfänger, Geübte)

Benutzeroberfläche

- innerhalb der Anwendung: angelehnt an Microsoft Hilfesystemen mit Fenstertechnik, Popup-Windows .

Nichtfunktionale Anforderungen

- Gründliche Dokumentation der Konzeption und der Implementierung - Code-Dokumentation in Englisch - Dokumentation der SEP-Arbeit: Deutsch oder Englisch - Vorbereitende Maßnahmen zur Wiederverwendung: - z.B. modularer/objektorientierter Entwurf und Realisierung, - Achten auf Erweiterbarkeit - insbesondere der Nutzung im Eventmechanismus - Weitgehende Verwendung von vorhandenen Entwicklungsbausteinen - z.B. Hilfesysteme, Eventmechanismen

Technische Produktumgebung

Spezielle Anforderungen an die Entwicklungsumgebung

Software

- * Implementierung erfolgt in Java
- * Vorerst erfolgen die Implementierungen nicht direkt in AutoFOCUS, sondern in quest.

Hardware

Orgware

Entwicklungs-Schnittstellen

Gliederung der Teilprodukte

Ergänzungen

B Programmcode

Nachfolgend finden sich die für das bestehende Hilfesystem notwendigen Dateien. Neben den Programmcode-Kommentaren befindet sich in Kapitel 6: "Implementierung" eine nähere Erläuterung zur Verwendung des Codes.

HelpSystem.java

```
package quest.help;
//JavaHelp-specific Package
import javax.help.*;
import javax.help.FlatMap;

//for resolving URLs
import java.net.*;

import java.io.*;
import java.awt.event.*;
import java.awt.*;
import java.awt.Component;
```

```
import java.awt.TextComponent;
import java.util.*;

import javax.swing.*;
import javax.swing.event.*;
import javax.swing.JMenu;
import javax.swing.tree.*;

import quest.mmm.relation.*;
import quest.mmm.model.*;
import quest.mmm.model.MetaModelElement.*;
import quest.metamodel.model.*;

import quest.testing.application.browser.*;
import quest.testing.application.*;
import quest.testing.application.TiniTest;

public class HelpSystem
//HelpSystem-Variables
public HelpBroker hb;
private HelpSet hs;
private JMenu helpMenu = new JMenu("JavaHelp");
private JCheckBoxMenuItem whatIsHelpItem;
public JMenuItem specificHelpItem, fieldLevelHelpItem;

//-----//

public void insertHelpMenuBar(JMenuBar jMenuBar)

//add JavaHelp Menu
jMenuBar.add(helpMenu);

//embedding JavaHelp: show help
JMenuItem helpItemTOC; // additional menu Help-Component
helpItemTOC = new JMenuItem("Overview");
helpItemTOC.setToolTipText("show Help-Overview");
specificHelpItem = new JMenuItem("for this Screen");
specificHelpItem.setToolTipText("shows contextsensitive help (for this
```

```
screen");
whatIsHelpItem = new JCheckBoxMenuItem("What is...");

// open HelpSet, send console message
// hardcoded location: "HelpSet.hs" in (for example) "QuestHelp" sub-
// directory

try
URL hsURL = new URL((new File("../Editor/Help")).toURL(),
"QuestHelp/HelpSet.hs");
hs = new HelpSet(null, hsURL);
System.out.println("Found help set at " + hsURL);

catch (Exception ee)
System.out.println("HelpSet not found"+ee+" :-(");
System.exit(0);

// create HelpBroker from HelpSet
hb = hs.createHelpBroker();
System.out.println("hb ist jetzt:" + hb);//for debugging

//show help
//activate the help menu item
ActionListener helper = new CSH.DisplayHelpFromSource(hb);
helpItemTOC.addActionListener (helper);
specificHelpItem.addActionListener (helper);

helpMenu.add(helpItemTOC); // add MenuItem
helpMenu.add(specificHelpItem);
helpMenu.add(whatIsHelpItem);

//_____

CSH.setHelpIDString(helpItemTOC, "overview");//sets HelpIDString
for complete HelpSystem

//Call for method "createCSHS()", which realizes CSHS (ContextSen-
sitiveHelpScreen) for Screens
createCSHS(jMenuBar);
```

```
//assign mapIDs for field-Level context-sensitive help
CSH.setHelpIDString(jMenuBar, "menuBar");
// CSH.setHelpIDString(Component, "menuBar");

//end insertHelpMenuBar()

public void initValidatorWhatIs(Browser browser)
whatIsHelpItem.addItemListener(new SwitchWhatIsHelp(browser));
//end initValidatorWhatIs()

public void createCSHS(JMenuBar jMenuBar)
//System.out.println("jMenuBar.getClass().
toString ist:>" +jMenuBar.getClass().toString()+"<"); //for debugging

if ( jMenuBar.getClass().toString().equals
("class quest.testing.application.TiniMenuBar"))

CSH.setHelpIDString(specificHelpItem, "Editor/modelvalidator");

else if (jMenuBar.getClass().toString().equals("class javax.swing.JMenuBar"))

CSH.setHelpIDString(specificHelpItem, "Editor/moduleeditor");
else

CSH.setHelpIDString(specificHelpItem, "no_match");
//end createCSH()

/**FieldLevel Help
Listen for when the selection in the browser changes.
/
class SwitchWhatIsHelp implements ItemListener

private WhatIsHelpDetector whatIsHelpDetector;
```

```
private Browser browser;

//Constructor
public SwitchWhatIsHelp(Browser browser)
this.browser = browser;
whatIsHelpDetector = new WhatIsHelpDetector(browser);
//end Constructor

public void itemStateChanged(ItemEvent e)

if (e.getStateChange() == ItemEvent.DESELECTED)
browser.removeTreeSelectionListener(whatIsHelpDetector);
//end if

else
browser.addTreeSelectionListener(whatIsHelpDetector);
//end else
//end itmeStateChanged()

//end class "SwitchWhatIsHelp"

class WhatIsHelpDetector implements TreeSelectionListener

private Browser browser;

public WhatIsHelpDetector(Browser browser)
this.browser = browser;

public void valueChanged(TreeSelectionEvent e)
TreePath[] paths = e.getPaths();
for (int i=0; i<paths.length; i++)
if(e.isAddedPath(paths[i]))
ModelTreeNode selectedNode = (ModelTreeNode) paths[i].
getLastPathComponent();

MMElement myMMElement = (MMElement) selectedNode.getUserObject();
MetaModelElement myMetaModelElement = myMMElement.
getMetaModelElement();
String myName = myMetaModelElement.getName();
```

```
String newName = "Editor/modelvalidator/pane/"+myName.toLowerCase();
System.out.println("MetaModelElement ist: "+myName);
System.out.println("Vollst-bändiger Name ist: "+newName);

// wir testen die direkte Hilfeausgabe
javax.help.Map.ID testMapID = javax.help.Map.ID.create
(newName.toString(), hs);
try
hb.setCurrentID(testMapID);
catch (Exception ee)
hb.setDisplayed(true);

//unselect WhatIs Help
browser.removeTreeSelectionListener(this);
whatIsHelpItem.setState(false);

//if
//for
// valueChanged()

//end class "WhatIsHelpDetector"
```

JavaHelp spezifische Navigationsdateien

Die nachfolgenden Navigationsdateien Helpset.hs, Map.jhm, TOC.xml und Index.xml sind bisher nur für Präsentationszwecke im Rahmen des Testings am Prototyp geschrieben. Die Vollständigkeit der Dateien ist daher in keiner Weise gegeben und die Ansteuerung der Hilfeinfortionstexte funktioniert ebenfalls nicht, da die Dateien zum Teil noch nicht existieren.

HelpSet.hs

```
<helpset version="1.0">
<title>QUEST/AutoFOCUS2 Online Help</title>
<maps>
<mapref location="Map.jhm"/>
```

```

<homeID>overview</homeID>
</maps>
<view>
<name>TOC</name>
<label>Text Utility TOC</label>
<type>javax.help.TOCView</type>
<data>TOC.xml</data>
</view>
<view>
<name>Index</name>
<label>Text Utility Index</label>
<type>javax.help.IndexView</type>
<data>Index.xml</data>
</view>
<view>
<name>Search</name>
<label>Text Utility Word Search</label>
<type>javax.help.SearchView</type>
<data engine="com.sun.java.help.search.DefaultSearchEngine">
JavaHelpSearch
</data>
</view>
</helpset>

```

Map.jhm

```

<!-- Image Files -->

<mapID target="toplevelfolder" url="Images/toplevel.gif" />
<mapID target="category" url="Images/category.gif" />
<mapID target="topic" url="Images/topic.gif" />

<mapID target="overview" url="Editor/overview.html" />

<!-- Level: Editor -->
<mapID target="Editor/modelvalidator" url="Editor/modelValidator/
modelvalidator.html"/>

<mapID target="Editor/moduleeditor" url="Editor/moduleEditor/

```

```

moduleeditor.html"/>

<mapID target="Editor/dtdeditor" url="Editor/dtdEditor/dtdeditor.html"/>

<!-- Level: modelvalidator_pane -->
<mapID target="Editor/modelvalidator/pane" url="Editor/modelValidator/
pane.html"/>

<mapID target="Editor/modelvalidator/pane/repository"
url="Editor/modelValidator/pane/repository.html"/>

<mapID target="Editor/modelvalidator/pane/project"
url="Editor/modelValidator/pane/project.html"/>

<mapID target="Editor/modelvalidator/pane/component"
url="Editor/modelValidator/pane/component.html"/>

<mapID target="Editor/modelvalidator/pane/module"
url="Editor/modelValidator/pane/module.html"/>

<!-- Level: modelvalidator_menu -->
<mapID target="Editor/modelvalidator/menu/edit/attributes"
url="Editor/modelValidator/menu/edit/attributes.html"/>

```

TOC.xml

```

<!-- oberste Ebene -->

<tocitem image="toplevelfolder" text=" Quest/AutoFOCUS2 Online
Help">
<tocitem image="topic" target="overview" text="Overview"/>

<!--—————>
<!--—————EDITORS(begin)—————>
—>
<!--—————>

<!-- Editor-Ebene -->

```

```
<tocitem image="topic" target="Editor" text="Editor ">
<!--—————>
<!-- common-Ebene-->
<tocitem target="Editor/common" text="Common">

<!-- common-Menu-Ebene-->

<tocitem target="Editor/common/menu"
text="Menu">

<tocitem target="Editor/common/menu/edit"
text="Menu: Edit ">

<tocitem target="Editor/common/menu/file"
text="Menu File ">

<tocitem target="Editor/common/menu/options"
text="Menu Options ">

<tocitem
target="Editor/common/menu/help" text="Menu Help ">
</tocitem><!--common-Menu-->

<!-- common-Toolbar-Ebene-->

<tocitem target="Editor/common/toolbar"
text="Toolbar">

<tocitem target="Editor/common/toolbar/select"
text="Toolbar: select ">

<tocitem target="Editor/common/toolbar/resize"
text="Toolbar: resize ">

</tocitem><!--common-toolbar-->

<!-- common-Pane-Ebene-->

<tocitem target="Editor/common/pane"
text="Pane">
```

```
<tocitem target="Editor/common/pane/Channel"  
text="Element: Channel" />  
  
<tocitem target="Editor/common/pane/Port"  
text="Element Port" />  
  
</tocitem><!--common-toolbar-->  
  
</tocitem><!--common-->  
  
<!--—————>  
  
<tocitem target="Editor/modelvalidator" text="Model Validator">  
  
<!-- Model Validator-Ebene-->  
  
<tocitem target="Editor/modelvalidator/menu"  
text="Menu of Model Validator">  
  
<tocitem target="Editor/modelvalidator/menu/edit"  
text="Menu: Edit" >  
  
<tocitem  
target="Editor/modelValidator/menu/edit/attributes"  
text="Menu Edit: Attributes" />  
  
</tocitem><!--modelvalidator/menu/edit-->  
  
<tocitem target="Editor/modelvalidator/menu/file"  
text="Menu File" />  
  
<tocitem target="Editor/modelvalidator/menu/test"  
text="Menu Test" />  
  
<tocitem target="Editor/modelvalidator/menu/verify"  
text="Menu Verify" />  
  
<tocitem target="Editor/modelvalidator/menu/generate"  
text="Menu Gererate" />
```

```
<tocitem target="Editor/modelvalidator/menu/options"
text="Menu Options "/>

<tocitem
target="Editor/modelvalidator/menu/help" text="Menu Help "/>
</tocitem><!--modelvalidator-Menu-->

<tocitem target="Editor/modelvalidator/pane"
text="Pane of Model Validator">

<tocitem target="Editor/modelvalidator/pane/repository"
text="Pane: Repository "/>

<tocitem target="Editor/modelvalidator/pane/project"
text="Pane: Project "/>

<tocitem target="Editor/modelvalidator/pane/component"
text="Pane: Component "/>

<tocitem target="Editor/modelvalidator/pane/module"
text="Pane: Module "/>

</tocitem><!--modelvalidator-Pane-->
</tocitem><!--modelvalidator-Editor-->

<!--—————>

<tocitem target="Editor/moduleeditor" text="Module Editor">

<!-- Module Editor-Ebene-->

<tocitem target="Editor/moduleeditor/menu"
text="Menu of Module Editor">

<tocitem target="Editor/moduleeditor/menu/edit"
text="Menu: Edit "/>

<tocitem target="Editor/moduleeditor/menu/file"
text="Menu File "/>
```

```

<tocitem target="Editor/moduleeditor/menu/test"
text="Menu Test "/>

<tocitem target="Editor/moduleeditor/menu/verify"
text="Menu Verify "/>

<tocitem target="Editor/moduleeditor/menu/generate"
text="Menu Generate "/>

<tocitem target="Editor/moduleeditor/menu/options"
text="Menu Options "/>

<tocitem
target="Editor/moduleeditor/menu/help" text="Menu Help "/>
</tocitem><!--moduleeditor-Menu-->
</tocitem><!--moduleeditor-Editor-->

<!--—————>

<tocitem target="Editor/ssdEditor"
text="System Structure Diagram - Editor (SSD-Editor)">

<!-- SSD-Editor-Ebene-->

<tocitem target="Editor/ssdEditor/menu"
text="Menu of System Structure Diagramm - Editor">

<tocitem target="Editor/ssdEditor/menu/edit"
text="Menu: Edit "/>

<tocitem target="Editor/ssdEditor/menu/file"
text="Menu File "/>

<tocitem target="Editor/stdEditor/menu/options"
text="Menu Options "/>

<tocitem target="Editor/ssdEditor/menu/help"
text="Menu Help "/>
</tocitem><!--ssd-Menu-->
</tocitem><!--SSD-Editor-->

```

```
<!-->
<tocitem target="Editor/std_editor"
text="State Transition Diagram - Editor (STD-Editor)"/>
<!-->
<tocitem target="Editor/eet_editor"
text="Extended Event Trace - Editor (EET-Editor)"/>
<!-->
<tocitem target="Editor/dtd_editor"
text="Data Type Definition - Editor (DTD-Editor)"/>
<!-->

</tocitem><!--Editor-->

<!-->
<!--EDITORS(end)-->
-->
<!-->

<!-->
<!--Tutorial(begin)-->
>
<!-->

<tocitem image="topic" target="tutorial" text="Nelly - Das Online
Tutorial"/>

<!-->
<!--Tutorial(end)-->
<!-->

<!-->
<!--Processes(begin)-->
>
<!-->

<tocitem image="topic" target="processes"
text="Arbeitsprozesse mit Quest/AutoFOCUS2"/>

<!-->
```

```
<!--Processes(end)-->
>
<!-->

</toc>
```

Index.xml

```
<index version="1.0">

<indexitem target="overview" text="overview"/>

</index>
```

C Literaturverzeichnis

Literatur

- [BAL98] Balzert, Helmut (Software-Technik, 1998): Lehrbuch der Softwaretechnik. Bd. 2, Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung, Heidelberg (Spektrum Akademischer Verlag), 1998, S. 668-669.
- [LEW00] Lewis, Kevin (JavaHelp, 2000): Creating Effective JAVAHELP, Sebastopol (O'Reilly Associates), 2000.
- [JAV02] <http://java.sun.com/products/javahelp>