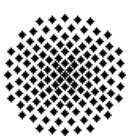
Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. Kurt Rothermel
Betreuer:	Dipl.-Inf. Jürgen Hauser
Begonnen am:	01.11.00
Beendet am:	30.04.01
CR-Nummer:	H.5.2, I.3.6, I.7.2

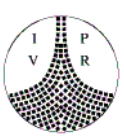
Diplomarbeit-Nr.: 1892

Konzeption und Realisierung von Vorlagen im MAVA-System

Jing Tian



Universität Stuttgart
Fakultät Informatik



Institut für Parallele und Verteilte
Höchstleistungsrechner
Breitwiesenstraße 20-22
D-70565 Stuttgart

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Arbeit wurde im Rahmen des Projektes MAVA (**M**ultimedi**A** document **V**ersatile **A**rchitecture) an der Universität Stuttgart durchgeführt. MAVA ist ein System für die Erstellung und Präsentation von Multimedia-Dokumenten, das in Java implementiert und somit plattformunabhängig und für die Integration ins Internet geeignet ist. Zunächst wird die Motivation für diese Arbeit gegeben. Danach folgt die Beschreibung der Aufgaben, die zu dieser Arbeit gehören. Im Anschluß daran wird eine Gliederung der Arbeit gegeben, die den Weg der Problemlösung beschreibt.

1.1 Motivation

Das MAVA-System besteht aus einem Präsentationssystem und einem Multimedia-Dokumenteneditor. Das Präsentationssystem ermöglicht anwendungsunabhängige Präsentationen von Multimedia-Dokumenten, was bedeutet, daß Multimedia-Dokumente aus allen unterschiedlichen Anwendungsgebieten mit einem System präsentiert werden können. Der MAVA-Dokumenteneditor vereinfacht wesentlich die Erstellung von Multimedia-Dokumenten, da der Autor keine Vorkenntnisse der Programmierung braucht, um Multimedia-Dokumenten für unterschiedlichsten Anwendungsgebieten zu erstellen.

Das Ziel dieser Arbeit ist, der MAVA-Dokumenteneditor um die Möglichkeit zur Erstellung und Verwendung von Vorlagen zu erweitern.

Was ist eine Vorlage? Im »Duden - Deutsches Universalwörterbuch A - Z« gibt es folgende Erklärung: Eine Vorlage ist “etw., was bei der Anfertigung von etw. als Muster, Grundlage, Modell o. Ä. dient”.

Vorlagen werden in vielen Arbeitsbereichen verwendet, um die Arbeit von Menschen zu erleichtern. Beispielsweise für häufig wiederkehrende Editieraufgaben sind Vorlagen sehr wichtig, um die sich immer wiederholende ähnliche Arbeiten zu sparen. Folgend werden

einige konkrete Beispiele gestellt, um die Nutzung der Vorlagen im Alltag zu verdeutlichen.

Zum Beispiel, schreibt man heutzutage Briefe immer öfter am Computer und läßt sie dann vom Drucker ausdrucken. Die normalen Briefe haben ein bestimmtes Layout, wie z.B. Briefkopf, Empfänger, Schrift, usw. Es ist nicht nur zeitaufwendig, sondern auch fehleranfällig, das gleiche Layout bei jedem Berief wieder vom Anfang an neu zu setzen. Offensichtlich ist es besser, eine Vorlage für Beriefe zu haben, die schon das richtige Berieflayout besitzt. Eine Briefvorlage sieht sowie ein normaler Brief aus, außer daß er noch keinen richtigen Inhalt hat, sondern nur einige voreingestellte Information als Platzhalter für den zukünftigen Inhalt besitzt. Die Struktur des Briefs ist schon von der Vorlage vorgegeben, d.h. wo sich der Briefkopf, das Datum und die Unterschrift auf der Seite befinden sollen, usw. Um einen Brief im richtigen Layout zu schreiben, braucht man nicht jedesmal komplett neu zu beginnen, sondern es genügt, eine Briefvorlage auszuwählen und die vordefinierten Platzhalter durch die Inhaltsinformation zu überschreiben. Solche Vorlagen sind heutzutage in Anwendungssoftware üblich und sehr hilfsreich in vielen verschiedenen Anwendungsbereichen.

Wenn eine Office Anwendungssoftware gestartet ist, beispielsweise Microsoft Word™, wird der Benutzer gefragt, was für einen Typ von Dokument er schreiben will. Für die Auswahl steht eine Reihe Vorlagen von verschiedenen Dokumententypen zur Verfügung, z.B. Brief, Fax, Bericht oder ganz normales Dokument. Der Benutzer kann diese Vorlagen auch vorab mit der Vorschau-Funktionalität ansehen, bevor er sich entscheidet, welche Vorlage er benutzen will. Falls der Benutzer ein Fax schreiben möchte, wählt er die Vorlage für Fax aus. Dann erscheint ein "halbfertiges" Fax im richtigen Layout auf dem Bildschirm für Verarbeitung. Da diese Fax-Vorlage schon die richtige Struktur vorgibt, muß der Benutzer nur noch den Inhallt hinzufügen. Dadurch werden nicht nur Fehler vermieden, die bei der Erstellung vom Layout eintreten können, sondern auch viel Zeit gespart.

Ein anderes Beispiel für die Verwendung von Vorlagen ist die Erstellungen von Vorträge mit Microsoft Powerpoint™. Nach dem Start von Powerpoint wird der Benutzer gefragt, was für einen Stil er seinem Vortrag vorgeben möchte. Der Benutzer kann aus eine Menge von vorgegebenen Stil-Vorlagen eine für seinen Vortrag auswählen. Mit der Vorschau

kann man alle Vorlagen sehen und sich zwischen unterschiedlichen Hintergrundbildern, Farben, Schriftzeichen entscheiden. Wenn eine Vielzahl von unterschiedlichen Stil-Vorlagen verfügbar sind, kann der Autor leicht eine geeignete Vorlage für seinen Vortrag finden. Er kann diese Vorlage direkt benutzen, oder modifizieren, wenn noch wenige Änderungen notwendig sind. Damit geht die Erstellung von einem Vortrag viel schneller und der Zeitbedarf wird daher deutlich reduziert.

Durch die obengenannten Beispiele kann man offenbar bemerken, wie Vorlagen die Erstellung von Dokumenten, egal ob normale Briefe, Fax oder Vorträge, deutlich vereinfachen können. Einerseits werden viele potentiale Fehler vermieden, während die sich immer wiederholende Arbeiten von Vorlagen automatisch gemacht sind. Andererseits wird viel Zeit durch die Verwendung von Vorlagen gespart, da der Autor viele Arbeiten nicht immer wiederholen muß. Insgesamt kann der Autor mit Vorlagen Dokumente viel effizienter und produktiver erstellen.

Aber nicht nur für das Editieren von Briefen, Faxen oder Vorträgen, die Vorlagen sind sogar noch nützlicher im Multimedia-Bereich, wo die Multimedia-Dokumente betrachtet werden sollen.

Im Vergleich zu herkömmlichen Dokumenten besitzen Multimedia-Dokumente nicht nur Text, Tabellen und Bilder, sondern viel mehr andere Typen von Medienelementen (z.B. Animationen, Audio- und Video-Sequenzen). Sowohl räumliche Dimension als auch zeitliche Dimension sollen für die Medienelemente in Multimedia-Dokumenten berücksichtigt werden. Dafür ist das Vorlagenkonzept, das man in obengenannten Beispiele von Word™ oder Powerpoint™ abgeleitet hat, im Multimedia-Bereich nicht mehr geeignet. Für die Erstellung von Multimedia-Dokumenten braucht man Vorlagen, die mehr als einfaches Layout für Briefe oder Vorträge anbieten können. Hier sind die Multimedia-Vorlagen gewünscht, die als Grundlagen für die Weiterverarbeitung zu Multimedia-Dokumenten funktionieren. Nicht nur Layout, sondern auch die räumliche und zeitliche Beziehungen zwischen den Medienelementen in Multimedia-Dokumenten sollen in Vorlagen dargestellt werden können.

Zum Beispiel, ein Computer-gestützten Training (engl. Computer Based Training, kurz CBT) bietet neben der Wissenvermittlung auch die Überprüfung der Verständnisse vom

Benutzer für die im CBT angeeigneten Wissens, indem einige konkrete Fragen durch Interaktionen mit dem Benutzer zu beantworten sind. Als Frageelement sind z.B. Multiple-Choice-Fragen sehr üblich, wobei der Benutzer zwischen einer oder mehreren gegebenen Antworten die richtigen auswählt. Der Computer akzeptiert die eingegebene Antworten vom Benutzer und gibt die Ergebnisse vom Test zurück. Diese Vorgehensweise ist beispielsweise durch die theoretische Führerscheinprüfung bekannt.

Da die Antwort vom Benutzer entweder richtig oder falsch sein kann, gibt es nur zwei entsprechende Reaktionen vom Computer: auf richtige Antwort und auf falsche Antwort. Dazu wird das Konzept für interaktive Multimedia-Dokumente um zwei Relationen erweitert (weiter-wenn-richtig und weiter-wenn-falsch). Diese zwei Relationen ermöglichen die Präsentation, auf die richtige oder falsche Antwort zu reagieren, d.h. die Präsentation kann einen alternativen Verlauf nach der Antwort vom Benutzer automatisch auswählen.

Diese Interaktion zwischen Computer und Benutzer sind bei allen Multiple-Choice-Fragen gleich, während die Unterschiede nur in den konkreten Multimedialen Frageelementen liegt. Für einen Test gibt es normalerweise eine Menge von Multiple-Choice-Fragen. Offensichtlich soll der Autor vom Test nicht die obengenannte Interaktion für jede Frage neu erstellen. Besser ist es, diese Interaktion in einer Multimedia-Vorlage zu integrieren, die der Autor für alle Multiple-Choice-Fragen verwenden kann. Die Multimedia-Vorlagen enthalten Muster, die sich lediglich in jeweils benutzten Medienelementen, nicht aber in ihrer Struktur unterscheiden.

Als ein Multimedia-Dokumenteneditor soll der MAVA-Editor die entsprechenden Multimedia-Vorlagen anbieten, um die Erstellung von Multimedia-Dokument für Autor so weit wie möglich zu vereinfachen. Zum Beispiel sollen die obengenannte Relation in den Vorlagen des MAVA-Editors verfügbar sein, um dem Autor des CBTs zu helfen, eine Menge von Fragen schnell und einfach zu erstellen. Durch die Verwendung von Vorlagen wird ein einfaches und schnelles Zusammenfügen von Multimedia-Dokumenten ermöglicht und damit die Effizienz der Arbeit von Autoren deutlich erhöht.

1.2 Aufgabebeschreibung

Bevor das Vorlagenkonzept für den MAVA-Editor spezifiziert wird, ist es sinnvoll, zuerst die Konzeptionen und implementierungen der Vorlagen in anderen Multimedia-Autorensystemen zu untersuchen. Durch diese Untersuchung kann man lernen, wie die Vorlagen in anderen Systemen konzipiert, welche Stärke und Schwachstellen sie haben, und wie es für den MAVA-Editor aussieht. Diese Untersuchung wird in Form von Literaturarbeit und praktischem Testen an verfügbaren Dokumentensystemen durchgeführt.

Nach der Untersuchung wird konzipiert, wie die Vorlagen im MAVA-Dokumenteneditor verwendet werden sollen. Dafür werden die Unterschiede und Vorteile zu existierenden Ansätzen in anderen Dokumentensystemen untersucht und dokumentiert. Die Kernidee der Erweiterbarkeit der Spezifikationssprache soll bei der Bearbeitung der Aufgabe im Vordergrund stehen.

Nach der Konzeption der Vorlagen wird die vorhandene Benutzeroberfläche des MAVA-Editors für den Umgang mit Vorlagen angepaßt. Dazu werden geeignete Menüs, Dialoge und Fenster definiert. Dies geschieht unter der Berücksichtigung, daß die Editerschritte für den Autor so einfach wie möglich gehalten werden sollen. Beispielsweise durch die Verwendung von Drag&Drop.

Anschließend werden die entwickelten Konzepte im MAVA-Dokumenteneditor, kurz MAVA-Editor, implementiert. Dazu stehen die aktuelle Version des MAVA-Editors inklusive der Dokumentenspeicher-Komponente zur Verfügung. Es soll die Benutzeroberfläche und das Speicherformat für die Verwendung von Vorlagen angepaßt werden. Die Implementierung erfolgt dabei basierend auf Java 2. Am Ende wird die Qualität der gewählten Lösung durch eine Demonstration des erweiterten MAVA-Dokumenteneditors gezeigt.

1.3 Aufbau der Arbeit

Im zweiten Kapitel wird einen Überblick über das MAVA-Projekt gegeben. Danach werden die Vorlagen aus anderen Dokumentensystemen im Kapitel drei untersucht. Kapitel vier beschreibt die Konzeption der Vorlagen für das MAVA-System. Die Anpassung des vorhandenen MAVA-Editors wird im Kapitel fünf behandelt. Anschließend kommt die Implementierung der MAVA-Vorlagen im Kapitel sechs. Nach der Implementierung wird

eine Bedienungsanleitung der MAVA-Vorlagen im Kapitel sieben gegeben. Am Ende dieser Arbeit gibt das achte Kapitel eine Zusammenfassung und Aussicht.

2 Einführung in das MAVA-Projekt

MAVA (**M**ultimedi**A** document **V**ersatile **A**rchitecture) ist ein plattformunabhängiges und erweiterbares Präsentation- und Autorensystem für Multimedia-Dokumente. Es wird als Teilprojekt in einem von der Deutschen Forschungsgemeinschaft (DFG) geförderten Forschungsprogramm “Verteilte Verarbeitung und Vermittlung Digitaler Dokumente (V3D2)” entwickelt. [V3D2]

In diesem Kapitel wird eine Einführung in das MAVA-System gegeben. Zuerst werden die Motivationen und Ziele des MAVA-Projekts vorgestellt. Anschließend wird das Dokumentenmodell von MAVA eingeführt. Am Ende wird das Autorensystem von MAVA, der MAVA-Editor, betrachtet.

2.1 Motivationen und Ziele des MAVA-Projekts

Traditionelle statische Dokumente beinhalten nur einfache Dokumentelemente wie beispielsweise Texte oder Bilder. Diese Dokumentelemente sind nicht zeitabhängig, sondern werden nur von der Größe, Farbe oder Position bestimmt. Deshalb werden sie als *statische Medienelemente* genannt. Bei der Erstellung von statischen Dokument muß man nur an das Layout von Dokumenten denken, d.h. die räumliche Anordnung der Dokumentelementen auf einer Seite.

Im Gegensatz zu traditionellen Dokumenten bestehen Multimedia-Dokumente aus beliebigen Typen von Medienelementen (z.B. Animationen, Audio- oder Videosequenzen). Da Audio- und Videosequenzen zeitabhängig sind, werden sie *zeitkontinuierliche Medienelemente* genannt. Daher besitzen Multimedia-Dokumente nicht nur eine räumliche Dimension, sondern auch eine zeitliche Dimension für die Darstellung der enthaltenen Medienelementen. Wenn zum Beispiel ein Multimedia-Dokument mehrere Videosequenzen beeinhaltet, muß die zeitliche Reihenfolge und die Dauer von Videosequenzen bei der

Erstellung vom Dokument festgelegt werden. Daher ist die Erstellung von Multimedia-Dokumenten viel komplexer als von traditionellen Dokumenten.

In der Vergangenheit wurden viele spezielle Präsentationssysteme für bestimmte Anwendungen entwickelt. Das Problem ist aber, daß diese Präsentationssysteme jeweils spezielles Transferformat und Editoren benötigen. Die Werkzeuge von solchen Systemen sind auch alle unterschiedlich, was jedesmal neuen Einarbeitungsaufwand vor der Verwendung bedeutet. Außerdem ist es sehr schwierig für den Autor, diese Präsentationssysteme zu erweitern oder anpassen, da die Systeme nur die bestimmte Formate von Medienelemente erkennen und bearbeiten können.

Im Vergleich zu solchen speziellen Präsentationssystemen wurden in der Vergangenheit auch mehrere generische Präsentationssysteme entwickelt. Solche generische Präsentationssysteme sind zwar nicht auf bestimmte Anwendungsgebiete beschränkt, stellen aber sehr hohe Anforderungen an den Autoren bezüglich den technischen Kenntnissen zur Erstellung von komplexen Multimedia-Dokumenten, da keine Unterstützung für spezielle Anwendungsgebiete vorhanden ist. Im gegensatz zu den speziellen Präsentationssystemen erlauben solche Systeme die Anpassung oder Erweiterung vom Autor. Aber solche Anpassung oder Erweiterung werden normalerweise durch die Integration einer Skript-Sprache durchgeführt. Da ein Autor in der Regel kein Programmierer, sondern Designer ist, ist die Anforderung von generischen Systemen an die Autoren zu hoch.

An dieser Stelle ist das Projekt MAVA angesiedelt. Das Ziel vom MAVA-Projekt ist, ein erweiterbares Multimedia-System zu entwickeln, um die Erstellung und Präsentation von Multimedia-Dokumenten möglichst zu vereinfachen.

Das Wort “erweiterbar” hier bedeutet, daß das MAVA-System nicht auf bestimmte Anwendungsgebiete eingeschränkt ist. Mit dem Dokumentenmodell vom MAVA-System kann der Autor stetig die anwendungsspezifische Erweiterungen einbauen. Einerseits kann der Autor die existierende Spezifikations-Konzepte um neue Aspekte erweitern, andererseits auch neue Anwendungskonzepte und Medienelemente in MAVA-System einfügen. Außerdem bietet das MAVA-System die gleiche Vorgehensweise und Werkzeuge für verschiedene Anwendungsgebiete. Damit muß der Autor nur einmal in den MAVA-Editor einarbeiten, was den Lernensaufwand deutlich reduzieren kann.

Mit der Erweiterbarkeit vom MAVA-System ist es möglich, Multimedia-Dokumente aus beliebigen Anwendungsgebieten innerhalb einem System zu erstellen. Die einzige Voraussetzung ist die entsprechende spezifische Erweiterung für jedes bestimmten Anwendungsgebiet. Wenn diese Erweiterung nicht vorhanden ist, muß sie basierend auf den Konzepten des MAVA-Systems von Entwicklern programmiert und anschließend den Autoren zur Verfügung gestellt werden. Aber es ist offensichtlich, daß mit der steigenden Verfügbarkeit von Anwendungskonzepten wird diese Notwendigkeit, neue Erweiterungen zu programmieren, immer seltener werden.

Grundlegende Konzepte aus dem Multimedia-Bereich muß in MAVA nicht neu entwickelt werden. Vielmehr sollen vorhandene Konzepte soweit wie möglich durch Integration, Verbesserung und Erweiterung weiter verwendet werden. Beispielsweise müssen keine neuen Konzepte für Beschreibung der räumlichen und zeitlichen Beziehungen zwischen Medienelementen im Multimedia-Dokument entwickelt werden. Da diese Konzepte schon vorhanden sind, sollen sie im MAVA-System einfach übernommen werden. Analog dazu sollen die vorhandenen Konzepte für Anwendungsgebiete wie CBT, Spiele oder Präsentation im MAVA-System auch weiter verwendet werden.

Das MAVA-System ist auf Java basiert, daher ist MAVA plattformunabhängig und kann auf allen Betriebsystemen, die Java Runtime Environment (JRE) unterstützt, wie z.B. Windows, Unix, Linux, Mac usw. laufen. Durch die Verwendung von XML als Transferformat und HTTP als Transportprotokoll ist die Integration von MAVA ins Internet einfach. Außerdem können MAVA-Dokumente auch durch Java-Applets direkt auf gewöhnlichen Internet-Seiten präsentiert werden, was offensichtlich ein großer Vorteil für die Verteilung der MAVA-Dokumente ist.

2.2 Das MAVA-Dokumentenmodell

Die Struktur eines Dokuments wird mittels eines sogenannten Dokumentenmodells beschrieben. Der Autor arbeitet im allgemeinen mit einer geeigneten Visualisierung des Dokumentenmodells. Das “What You See Is What You Get”-Prinzip ist ein im Zusammenhang mit der Dokumentenerstellung geprägter Begriff, der besagt, daß die Visualisierung bei der Erstellung schon mit der späteren Ausgabe (beispielsweise auf dem Drucker) übereinstimmt.

Mit statischen Dokumenten ist es kein Problem, da nur die räumlichen Anordnungen von Medienelementen betrachtet werden. Bei der Positionierung der Medienelementen kann der Autor den späteren Ausdruck des Dokuments schon auf dem Bildschirm festlegen. Das Layout der Seite auf dem Bildschirm soll mit dem Ausdruck des Dokuments immer übereinstimmen.

Die Schwierigkeit von Multimedia-Dokumenten ist dabei, daß hier zeitliche und interaktive Komponenten eintreten können. Bei der Darstellung von zeitlichen und interaktiven Komponenten ist das WYSIWYG-Prinzip nicht mehr haltbar. Hier ist es schwer, ein räumliches Layout für zeitliche Anordnung zu erzeugen. Obwohl es möglich ist, die zeitliche Konzepte mit Hilfe von gewissen Komponenten darzustellen (beispielsweise Balken über einer Zeitachse).

Die heutzutage existierende Ansätze haben das Problem, daß die beschriebenen Dokumentenmodelle immer von der zugehörigen Spezifikationssprache abhängig sind. Im Vergleich dazu soll das Dokumentenmodell vom MAVA folgende Ziele erreichen: Das Dokumentenmodell soll unabhängig von dem zu beschreibenden Anwendungskonzept bleiben, eine einfache Visualisierung auf hohem Abstraktionsniveau erlauben und die Erstellung von Dokumenten so weit wie möglich vereinfachen.

Gerade die Einfachheit der Dokumentenerstellung ist für die Praxis ein entscheidendes Argument, da Multimedia-Dokumente in der Regel nicht von Programmierer erstellt werden.

Um die obengenannten Ziele zu erreichen, wird ein Operatoren-Ansatz im MAVA Dokumentenmodell gewählt. Die Operatoren dienen dazu, die räumlichen, zeitlichen und anwendungsspezifischen Beziehungen und Abhängigkeiten zwischen den Medienelementen zu beschreiben. Der Hauptvorteil ist, daß Operatoren einfach zu verwenden sind und auf eine klare Art und Weise graphisch dargestellt bzw. manipuliert werden können. Dieser Ansatz liegt allen in MAVA verwendeten Anwendungskonzepten zu grunde, um eine sehr leichte Einarbeitung für den Autor zu ermöglichen.

In der graphischen Darstellung im MAVA-Editor wird ein Operator als ein Rechteck dargestellt. In der Abbildung 2-1 sieht man zwei Operatoren in rechtecke: ein räumlicher Operator genannt *setPosition* und ein zeitlicher Operator genannt *while*.

Die Operanden der Operatoren in MAVA-Dokumentenmodell sind die Medienobjekte. Die Medienobjekte repräsentieren die Informationseinheiten der Präsentation, und können von beliebigen Typen sein (Texte, Bilder, Audio- oder Videosequenzen, usw.). Ein Medienobjekt wird im MAVA-Editor als abgerundetes Rechteck veranschaulicht. In der Abbildung 2-1 kann man zwei Medienelemente sehen: ein Bild und eine Audiosequenz.

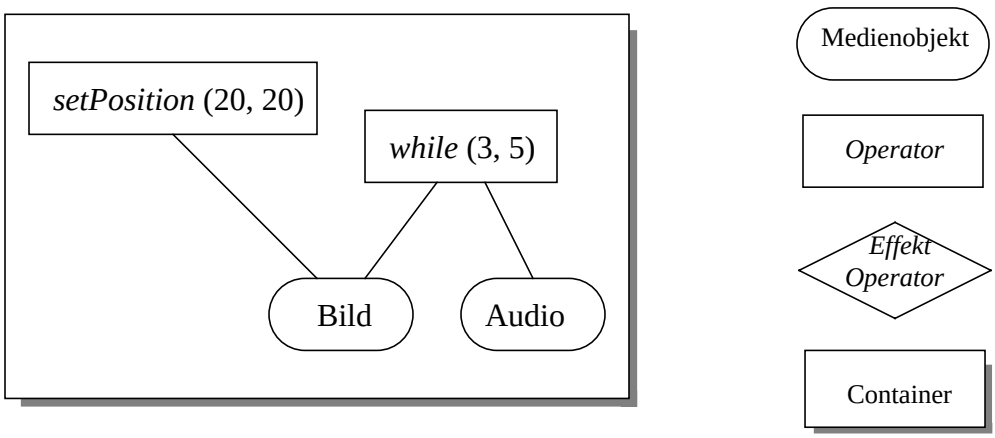


Abbildung 2-1. Elemente des MAVA-Dokumentenmodells

Für die Beschreibung des Dokumentenmodells werden die Mediendaten selbst nicht benötigt. Es ist ausreichend, die Attribute (z.B. Typ des Medienelements, Lokation der Mediendaten, usw.) zu betrachten. Daraus folgt, daß die Mediendaten im Dokumentenmodell nur als Attribute durch sogenannten Medienobjekte gespeichert werden.

Ein anderes Element im MAVA-Dokumentenmodell ist der Container. Ein Container umfaßt eine Menge von Medienelementen und Operatoren, und kann wiederum als Medienelement aufgefaßt werden, um rekursive Strukturen bilden zu können. Mittels Container kann ein Dokument in logische Einheiten aufgeteilt werden. Der Wurzelcontainer einer Hierarchie ist das Dokument. Im MAVA-Editor wird Container als ein Rechteck mit Schatten dargestellt, wie in der Abbildung 2-1 gezeigt wird.

Es gibt noch einen weiteren Typ von Element im MAVA-Dokumentenmodell, der sogenannte Effekt-Operator. Der Effekt Operator ist ein spezieller Operatorentyp und wird benutzt, um einen besonderen Effekt zu beschreiben (z.B. daß ein Medienelement ausge-

blendet werden muß). Ein Effekt-Operator wird immer zusammen mit einem zeitlichen Operator und einem Medienobjekt benutzt.

Das MAVA-Dokumentenmodell verwendet den Operatoren-Ansatz als ein Meta-Dokumentenmodell. Es beschreibt aber nicht den semantischen Aufbau, dazu braucht man konkrete Medienelemente oder Operatoren. Die Dokumentenmodelle für ein bestimmtes Anwendungsgebiet müssen sich an das Meta-Dokumentenmodell halten und demnach konkrete Medienelementen und Operatoren zur Verfügung stellen. Dadurch bleibt das MAVA-Dokumentenmodell unabhängig von bestimmten Anwendungskonzept.

Im rechten Teil der Abbildung 2-1 stehen die vier Typen von Elementen im MAVA-Dokumentenmodell: Medienobjekt, Operator, Effekt Operator und Container.

Links in Abbildung 2-1 ist ein einfaches MAVA-Dokument dargestellt. Das Dokument besteht aus einem (Wurzel-)Container, der wiederum zwei Medienelemente und zwei Operatoren umfaßt. Das erste Medienelement ist ein Bild, das zweite eine Audiosequenz. Der erste Operator bestimmt die Position vom Bild innerhalb der Präsentationsfläche des Dokuments. Die Parameter des Operators geben dabei die Position der linken oberen Ecke vom Bild an. Der zweite Operator beschreibt, daß während der Bildpräsentation eine Erklärung zu hören ist. Diese Semantik wird mit dem *while*-Operator beschrieben, bei dem es sich um einen sogenannten Intervall-Operator handelt. Die zwei Parameter des *while*-Operators beschreiben die Verzögerung zwischen dem jeweiligen Beginn der Präsentatiosintervalle der Medienelemente und dem Ende der Präsentationsintervalle. Der erste Parameter des Operators besagt, daß drei Sekunden nach dem Beginn der Bildpräsentation die Wiedergabe der Audiosequenz beginnen soll. Dementsprechend bedeutet das zweite Parameter, daß fünf Sekunden nach dem Ende der Audiosequenz auch die Bildpräsentation beendet werden soll.

Die wesentliche Eigenschaft im MAVA-Dokumentenmodell ist Trennung der Struktur eines Multimedia-Dokuments von der Realisierung der Semantik zur Präsentationszeit. Die Semantik, die das eigentliche Verhalten eines Multimedia-Dokuments beschreibt, wird ausschließlich durch die Operatoren definiert. Die konkrete Implementierung dieses Verhaltens ist in den sogenannten Managern enthalten. Diese befinden sich zusammen mit den ihnen zugeordneten Operatoren in einem Containerobjekt und realisieren deren Semantik.

2.3 Der MAVA-Editor

Nachdem das MAVA-Dokumentenmodell, das dem Präsentationssystem zugrundliegt, vorgestellt ist, wird nun das entsprechende Autorensystem betrachtet werden. Ziel des MAVA-Autorensystem ist, trotz Unabhängigkeit von den Anwendungskonzepten eine einfache und einheitliche Benutzungsoberfläche bereitzustellen. [Haus99]

Der Autor benutzt immer ein Autorensystem, um Multimedia-Dokumente zu erstellen. Und gerade die Verwendbarkeit von diesem Autorensystem ist entscheidend, ob der Autor den ganzen Ansatz schließlich akzeptieren wird. Ein kompliziertes und schwer verwendbares Autorensystem ist nicht attraktiv für den Autor und negativ für die Akzeptanz eines Ansatzes.

Das Autorensystem vom MAVA ist der MAVA-Editor (kurz MED). MAVA-Multimedia-Dokumente werden durch die Benutzung des MAVA-Editors durch einen Autor erstellt. [Haus00]

Der Hauptunterschied des MAVA-Editors im Vergleich zu existierenden Editoren für Multimedia-Dokumente anderer Ansätze sind seine Erweiterbarkeit und Anwendungsunabhängigkeit. Die meisten kommerziellen Multimedia-Autorensystemen sind auf den bestimmten Anwendungsgebieten beschränkt. Die Firma Macromedia allein bietet schon mehrere Autorensysteme in unterschiedlichen Anwendungsgebieten: Zum Beispiel *Authorware* produziert multimediale Lernanwendungen und Training, *Flash* kreiert vektorbasierte, animierte Web Seiten, *Director* steht für interaktive Multimedia-Inhalte fürs Web oder für CD-ROM, und *Dreamweaver* Verkürzt die Erstellung von Web Seiten, die HTML-Produktion und Verwaltung ihrer Seite usw. Im Vergleich dazu bleibt die Vorgehensweise der Erstellung von Multimedia-Dokumenten in verschiedenen Anwendungsgebieten mit dem MAVA-Editor immer gleich.

Ein weiteres Ziel des MAVA-Editors ist seine Einfachheit bei der Verwendung. Der Autor braucht nicht viel Zeit, um sich mit der Arbeitsumgebung vom MAVA-Editor vertraut zu machen. Da es im MAVA nur vier unterschiedliche Typen von Dokumentelementen gibt, besitzt der MAVA-Editor auch nur vier Symbole (engl. icon), um alle Dokumentelemente zu repräsentieren. Daher braucht der Autor nicht mehr als die Verwendung dieser vier Typen von Dokumentelementen zu kennen.

Der MAVA-Editor bietet drei unterschiedliche Ansichten auf Dokumente: die Symbol-Ansicht (engl. iconview), die Baum-Ansicht (engl. treeview) und die XML-Ansicht (engl. xmlview).

Die Erstellung und Bearbeitung des Dokuments wird in der Symbol-Ansicht durchgeführt. Der Autor benutzt die direkte Manipulation und Drag&Drop, um neue Medienelemente und Operatoren dem Dokumentegraph, der die Spezifikation des Dokuments darstellt, hinzuzufügen. Die Baum-Ansicht stellt eine logische und hierarchische Sicht von dem Dokument dar, damit der Autor immer eine baumförmige Übersicht der hierarchischen Struktur des Dokuments bekommen kann. Das von dem Autor erstellte Multimedia-Dokument wird in dem XML-basierten Format gespeichert. Mit der XML-Ansicht kann der Autor direkt die Dokumentinformationen in XML Darstellung sehen.

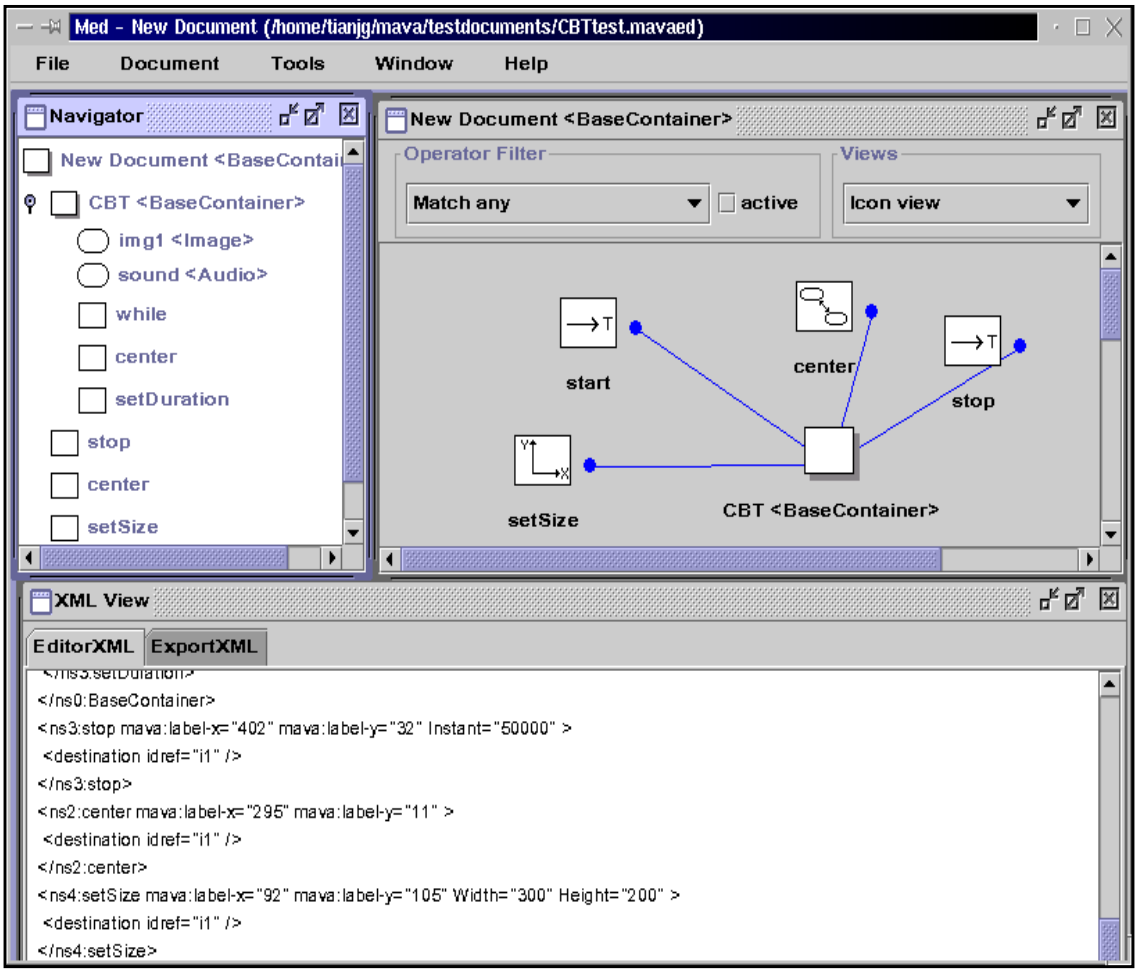


Abbildung 2-2. Die drei Ansichte des MAVA-Editors

In der Abbildung 2-2 kann man diese drei Ansichten in verschiedenen Fenstern sehen. Das linke obere Fenster zeigt die Baum-Ansicht, rechts daneben ist das Fenster für die Symbol-Ansicht, und darunter ist die XML-Ansicht von dem selben Dokument.

Die Erstellung und Bearbeitung von Multimedia-Dokumenten mit dem MAVA-Editor ist einfach und intuitiv. Das Wurzelelement eines Dokuments ist ein Container. Ein Autor erstellt ein Dokument, indem er zuerst einen Container auswählt, und dann die zugehörigen Medienelemente und Operatoren in diesen Container einfügt. Ein Container kann wiederum als Medienelement in den vorhandenen Container eingefügt werden, daher kann der Autor immer wieder neue Container in den aktuellen Container einfügen, um eine hierarchische Struktur in Dokumenten aufzubauen. Für das Hinzufügen von Dokumentelementen bietet der MAVA-Editor zwei separate Fenster jeweils für die Operatoren und Medienelemente. Mit dem Operator Fenster kann der Autor verschiedene Operatoren auswählen, die in unterschiedlichen Kategorien aufgeteilt sind, z.B. räumlich, zeitlich, usw. Für jede Kategorie kann man auch eine entsprechende Filterung aufrufen, um genau die gesuchte Operatoren zu zeigen. Für die Medienelemente gibt es eine analoge Filterung.

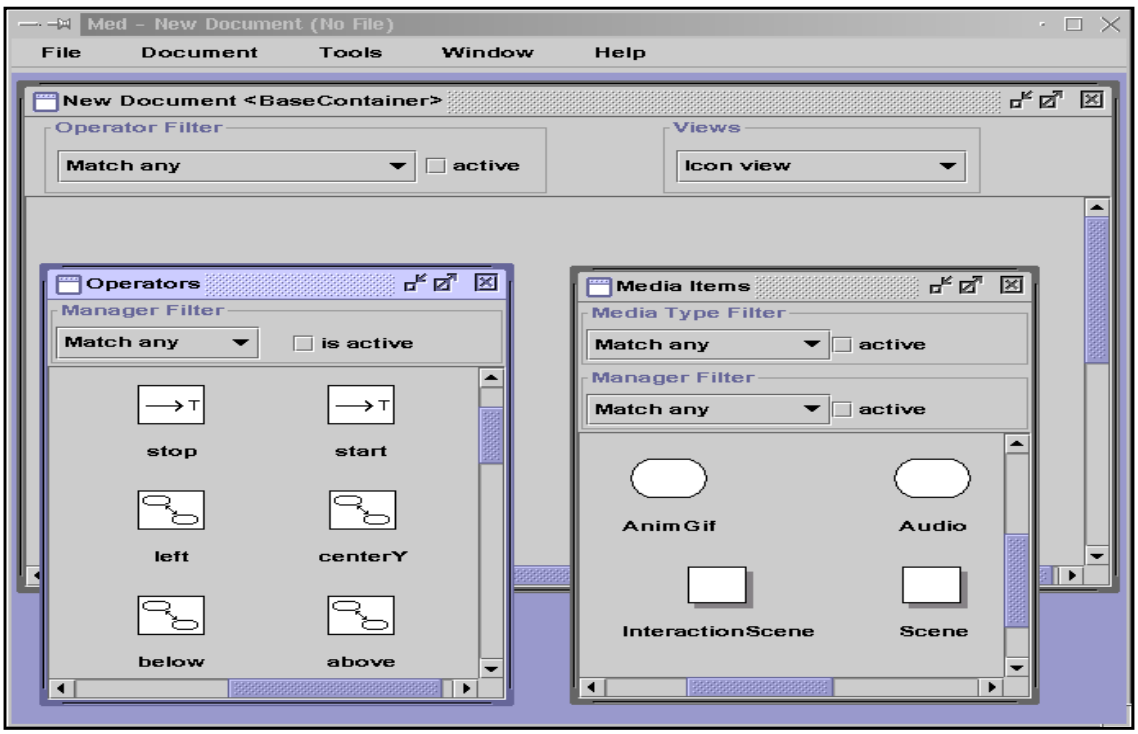


Abbildung 2-3. Die Erstellung vom Multimedia-Dokument mit dem MAVA-Editor

In der Abbildung 2-3 sieht man drei Fenster innerhalb des MAVA-Editors, vorne links steht das Fenster für Operatoren und rechts das Fenster für Medienelemente, dahinter befindet sich das Dokument-Fenster.

Um ein neues Dokument zu erstellen, wählt der Autor zuerst den Menüeintrag *New* von dem *File*-Menü des MAVA-Editors aus, ein "*New Document*"-Dialogfenster wird auf dem Bildschirm erscheinen. Darin kann der Autor einen Typ von Container als Wurzelelement des Dokuments auswählen und einen Name für das Dokument angeben. Anschließend erscheint ein leeres Fenster für das neue Dokument auf dem Bildschirm. Dann kann der Autor neue Medienelemente und Operatoren in diesem Fenster einfügen, indem er die Symbole von den Medienelement- und Operator-Fenstern ins Dokument-Fenster mittels Drag&Drop zieht.

Mit einem Doppelklick auf ein Container-Symbol kann der Autor ein neues Fenster für diesen Container öffnen und wiederum Medienelemente und Operatoren einziehen. Wenn der Autor mit der rechten Mausetaste auf das entsprechenden Symbol klickt, dann erscheint ein Kontextmenü. Mit der Hilfe von Kontextmenüs kann der Autor die Attribute von Medienelementen oder Operatoren überprüfen und ändern. Alle Änderungen, die der Autor in der Symbol-Ansicht macht, werden gleichzeitig auch in der Baum-Ansicht und XML-Ansicht gezeigt.

Mit Operatoren können Beziehungen zwischen Medienelementen beschrieben werden. Im MAVA-Editor verwendet der Autor die Operatoren durch Operator-Symbole. Der Autor verbindet das Operator-Symbol mit den Medienelement-Symbole, um die Beziehung zu setzen. Jedes Operator-Symbol hat einen oder zwei Punkte. Wenn ein Operator-Symbol nur einen Punkt rechts hat (in blauer Farbe), setzt dieser Operator eine Beziehung zwischen einem Ziel-Medienelement und dem Container-Element, in dem das Ziel-Medienelement liegt, z.B. die relative Position oder den Zeitpunkt zu Beginn einer Wiedergabe. Um diese Beziehungen zu setzen, drückt der Autor zuerst die linke Maustaste auf den Punkt rechts am Operator-Symbol und dann zieht die Maus zum Ziel-Medienelement-Icon. Eine Linie (in blauer Farbe) verbindet den Operator und das Ziel-Medienelement.

Hat ein Operator zwei Punkte, dann beschreibt der Operator eine Beziehung zwischen zwei Medienelementen. Um die Beziehung zu verdeutlichen wird ein Medienelement als Quell-

Medienelement genannt und das andere als Ziel-Medienelement genannt. Beispielsweise setzt der Operator *above* eine räumliche Beziehung zwischen einem Quell-Element und einem Ziel-Element und legt fest, daß das Quell-Element mit einem bestimmten Abstand über dem Ziel-Element positioniert werden soll. Um eine Beziehung zwischen zwei Medienelementen zu setzen, drückt der Autor zuerst die linke Maustaste auf den linken Punkt (in roter Farbe), und dann zieht die Maus zum Quell-Medienelement. Anschließend drückt der Autor die linke Maustaste auf den rechten Punkt (in blauer Farbe) und zieht zum Ziel-Medienelement. Danach wird eine Linie (in roter Farbe) zwischen dem Operator und dem Quell-Medienelement und eine andere Linie (in blauer Farbe) zwischen dem Operator und dem Ziel-Medienelement die Beziehung verdeutlichen.

So kann ein Autor ohne irgendwelche Programmierkenntnisse ein Multimedia-Dokument mit Drag&Drop in kurzer Zeit leicht erstellen.

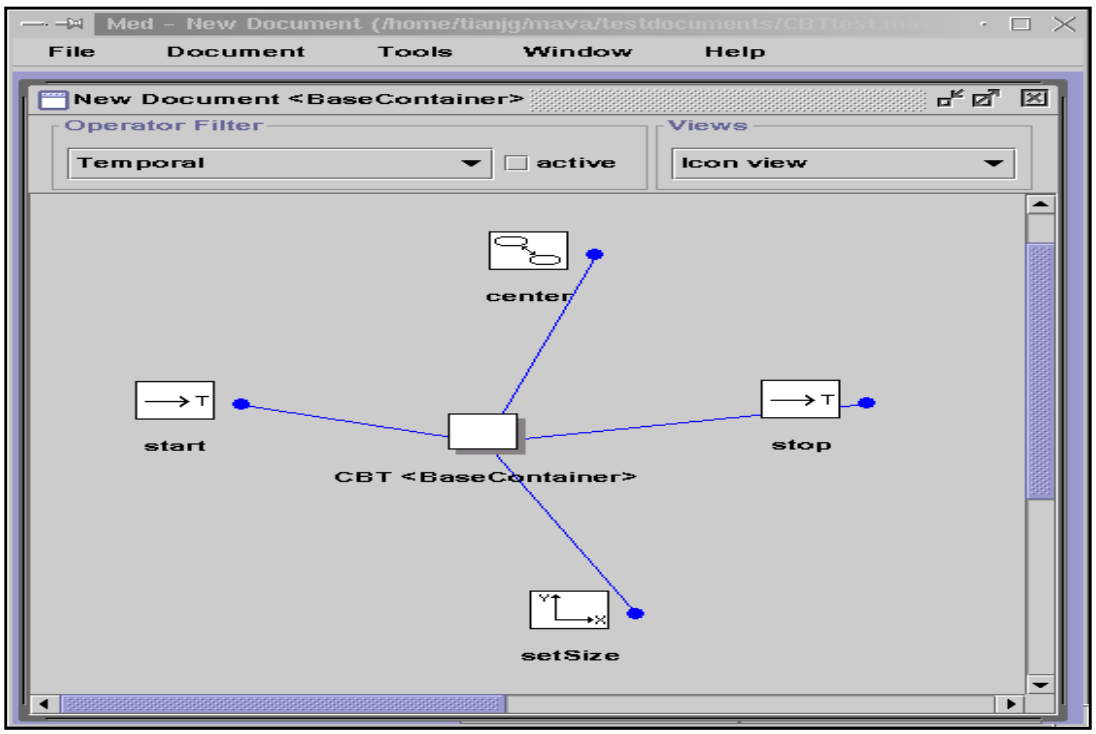


Abbildung 2-4. MAVA-Editor Symbol-Ansicht: ein Beispiel Dokument

Abbildung 2-4 zeigt die graphische Darstellung eines Dokuments im MAVA-Editor. In der Mitte befindet sich ein Container vom Typ Base Container mit dem Name CBT. Rund um

dem BaseContainer-Symbol stehen vier Operatoren, die durch die Linien mit dem Container verbunden sind.

Um das Dokument in der Abbildung 2-4 zu erstellen, öffnet der Autor zuerst ein Medienelement-Fenster, dann zieht ein BaseContainer-Symbol ins Dokument-Fenster ein. Danach öffnet der Autor ein Operator-Fenster, zieht die vier benötigte Operator-Symbole ins Dokument-Fenster ein und verbindet sie mit dem BaseContainer-Icon, um die Größe, die Position und den Anfang- und Endpunkt der Wiedergabe von BaseContainer zu definieren. Anschließend kann der Autor wieder auf das Container-Symbol doppelklicken und das zugehörige Container-Fenster öffnen, worin er wieder neue Medienelement- und Operator-Symbole einziehen und verbinden kann. Wenn noch weitere Container-Symbole im Container-Fenster vorhanden sind, kann er wieder neue Container-Fenster durch Doppelklick öffnen und diesen Vorgang weiter wiederholen.

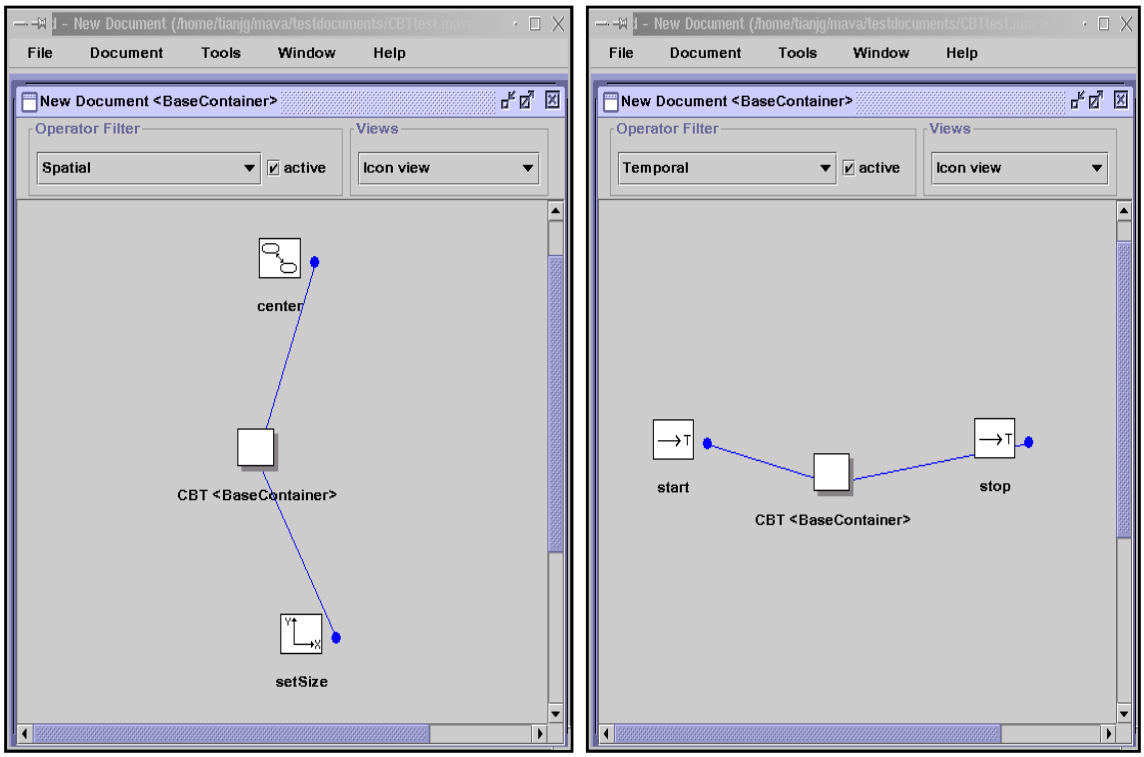


Abbildung 2-5. Symbol-Ansicht des Dokuments mit aktivierter Operator Filterung

Da bei einem größeren Dokument eine Vielzahl von Operatoren, Medienelemente und Linien verwendet werden, bietet das Autorensystem ein Filterungsmechanismus an, damit

der Autor auch im großen Dokumenten die Übersicht behalten kann. Der Autor vom MAVA-Editor kann verschiedene Filter wählen, um die Operatoren, die gewissen Kriterium nicht erfüllen, auszublenden.

Der Filterungsmechanismus erlaubt nur die Anzeige der Operatoren von einer gewissen Klasse (z.B. zeitliche Operatoren) oder einem gewissen Konzept (z.B. absolute Positionierung). Der Autor aktiviert eine Filterung, um alle Operatoren, die nicht zu einer Kategorie gehören, auszublenden. Um die Darstellung eines Dokuments zu vereinfachen, ist es sehr wichtig, verschiedene Filter aktivieren zu können. Nach der Filterung ist eine mögliche Verwirrung durch zuviele Operatoren im Dokumenten-Fenster vermeidbar. Dadurch kann der Autor sich auf gewisse Aspekte eines Dokuments konzentrieren. Die Übersichtlichkeit und Einfachheit des Editiervorgangs wird dadurch auch in großen und komplizierten Dokumenten garantiert.

In der Abbildung 2-5 ist dasselbe Dokument in zwei Fenster gezeigt, jeweiliges mit einem anderen aktivierten Operator-Filter. Links wird das Dokument mit einer räumliche-Filterung (*spatial*-Filterung) von Operatoren aktiviert, und nur die räumlichen Operatoren werden gezeigt. In dem rechten Fenster wird das Dokument mit der zeitliche-Filterung (*temporal*-Filterung) aktiviert und nur die zeitlichen Operatoren sind noch sichtbar.

Der Mava-Editor hat viele Eigenschaften: einfache Darstellung, flexible Ansicht, direkte Manipulation mit Drag&Drop, und Anwendungsunabhängigkeit.

Die Erstellung eines Dokuments mit dem MAVA-Editor ist zwar einfach, aber immer noch zeitaufwendig: Beispielsweise bei der Erstellung des Dokuments in der Abbildung 2-4 muß der Autor viele unkomplizierte einzelne Editierschritte immer wiederholen. Dieser sich immer oft wiederholende Vorgang ist sehr mühsam und kann auch sehr viel Zeit kosten. Dazu kommt, daß es nicht nur fehleranfällig, sondern auch die Effizienz der Arbeit von dem Autor vermindert.

Wie kann man die Zeit und den Aufwand bei der Dokumentenerstellung reduzieren? Ein Grundgedanke ist, die sich immer wiederholenden Arbeitschritte in eine wiederverwendbare Komponente zusammenzufassen und dann dieses Komponente als Einheit in dem Editiervorgang zu verwenden. Das wiederverwendbare Komponente wird als Vorlage (*template*) bezeichnet. Der Autor kann selbst Vorlagen erstellen oder die Vorlagen von

anderen Autoren wiederverwenden. Durch die Verwendung der Vorlagen braucht der Autor nicht mehr die gleichen Arbeiten immer zu wiederholen, sondern kann sich auf die Erstellung des aktuellen Dokuments konzentrieren.

Die Verwendung von Vorlagen ist effizient und produktiv für die Arbeit des Autors. Um die Erstellung des Multimedia-Dokuments möglichst zu vereinfachen, sollen Vorlagen in den MAVA-Editor eingebaut werden. Heutzutage bieten viele Multimedia-Autorensysteme Vorlagen schon als Standardkomponenten. Im nächstem Kapitel werden einige vertretenden Autorensysteme untersucht, um herauszufinden, ob diese Vorlagen unterstützen und wie Vorlagen in diesen Systemen integriert sind. Danach wird im Kapitel 4 konzipiert, wie die Vorlagen im MAVA-Editor erstellt und verwendet werden sollen.

3 Untersuchung der Vorlagen in Autorensystemen

Im letzten Kapitel wurde das multimediale Präsentation- und Autoren-System MAVA vorgestellt. Bei der Betrachtung des MAVA-Editors wurde festgestellt, daß viele wiederholende kleine Arbeitschritte bei der Erstellung von Multimedia-Dokumenten durch Vorlagen zusammengefaßt werden können, um den Aufwand und die benötigte Zeit bei der Dokumentenerstellungen deutlich zu reduzieren.

Heutzutage spielen Vorlagen eine sehr wichtige Rolle bei der Erstellung von Multimedia-Dokumenten: Einerseits helfen Vorlagen dabei, die Effizienz der Autoren bei der Erstellungen von Multimedia-Dokumenten beträchtlich zu erhöhen. Andererseits können die von verschiedenen Autoren erstellte Dokumente durch die Verwendung der Vorlagen gleiche Struktur behalten. Daher werden Vorlagen schon als Standardkomponenten in immer mehr modernen Multimedia-Autorensystemen, um die Arbeit der Erstellung von Multimedia-Dokumenten für die Autoren zu erleichtern.

In diesem Kapitel werden einige vertretende Multimedia-Autorensysteme untersucht. Autorensysteme, die Vorlagen bieten, werden in diesem Kapitel diskutiert und miteinander verglichen, um herauszufinden, wie Vorlagen in diesen Systemen integriert sind und welche Vorteile oder Nachteile das jeweilige Autorensystem besitzt. Zum Schluß findet eine Betrachtung der Spezifikation und Einsatz der Vorlage aus der Sichtweise vom MAVA-Editor statt.

3.1 Autorensysteme ohne Vorlagen

Obwohl die Verwendung der Vorlagen viele Vorteile besitzt und ein deutlicher Trend ist, gibt es immer noch einige Multimedia-Autorensysteme, die keine Vorlage-Unterstützung bietet. Beispielsweise sind folgende Multimedia-Autorensysteme noch ohne Vorlagen-Unterstützung:

- Macromedia Flash 4
- IBM Hotmedia 3.0
- Tribeworks iShell 2.0
- Crack für Windows 3.0

Hier wird Macromedia Flash 4 als ein Beispiel vorgestellt, wie der Autor im Multimedia-Dokument mit einem Autorensystem ohne Vorlagen-Unterstützung erstellen kann und welche Nachteile es für den Autor gibt.

Flash 4 ist ein Produkt der Firma Macromedia, um interaktive Vektor-Graphiken und Animationen, die sogenannte “Flash Movies”, für Webseiten im Internet zu erstellen.

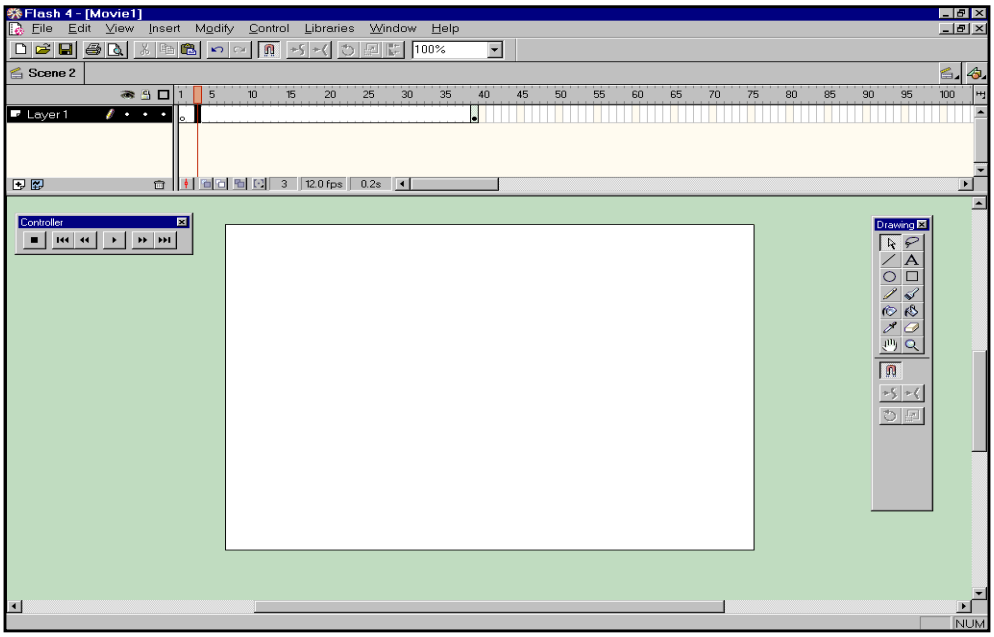


Abbildung 3-1. Macromedia Flash 4 Arbeitsfenster

Um einen neuen Flash-Movie zu erstellen, wählt der Autor den Eintrag *New* von dem File-Menü aus. Dann erscheint ein leerer Zeit-Rahmen (*time frame*) unter in der Mitte von dem Fenster, der einen Ausschnitt des Dokuments in bestimmtem Zeitpunkt zeigt. Der Autor kann die Medienelemente (z.B. Bilder, Audio- und Video-Sequenzen) in diesen Zeit-Rahmen einfügen. Auf einer Zeitachse kann der Autor die zeitliche Anordnungen der Medienelementen definieren. Und der Autor kann jederzeit den bearbeiteten Flash-Movie

mit Hilfe eines Kontroll-Fensters testen. Die Abbildung 3-1 zeigt das Arbeitsfenster vom Flash 4.

Zwar kann der Autor einige vorgefertigte Medienelemente aus dem Libraries-Menü wählen und wiederverwenden (z.B. Buttons, Bilder, Animationen, Audio- und Video-Sequenzen), mehr kann der Autor aber nicht. Es ist nicht trivial für einen Autor, der keine vorherige Erfahrung mit Flash 4 hat, ein neues Dokument schnell zu erstellen. Erst nachdem der Autor das mitgelieferte Online-Tutorial durchgelesen und vorhandene Demos ausprobiert hat, kann er mit der Erstellung von Dokumenten anfangen. Das ist ein übliches Problem bei allen Autorensystemen ohne Vorlagen: Eine relativ lange Einarbeitungsdauer ist bei solchen Autorensystemen unvermeidbar, bevor der Autor ein neues Dokument erstellen kann.

Ohne Vorlagen-Unterstützung kann der Autor auch nicht seine bearbeiteten Dokumente später effektiv wiederverwenden. Zwar existieren hier schon einige alternative Lösungen, damit der Autor seine Dokumente oder Medienelemente wiederverwenden kann, ist dieser Vorgang einfach für den Autor.

Eine Lösung für Wiederverwendung bei dem Flash 4 ist das Exportieren und Importieren des Dokuments. Es gibt zwei Export-Möglichkeiten im Flash 4. Der Autor kann "*Export Movie"* im File-Menü auswählen, um den ganzen Inhalt des Flash-Movie Dokuments zu exportieren, hier stehen außer Flash-Movie auch Quicktime, Windows AVI und andere Datenformate zur Verfügung. Oder kann der Autor "*Export Image"* im File-Menü auswählen, um nur die Bilder in Flash Dokument zu exportieren.

Links in der Abbildung 3-2 kann man das File-Menü von Flash 4 sehen, mit dem der Autor die Multimedia-Dokumente importiert oder exportiert. Wenn der Autor den Menüeintrag *Import* auswählt, erscheint ein *Import*-Dialogfenster, worin der Autor die hinzuzufügende Medienelemente auswählen kann (z.B. Bilder, Audio- und Video-Sequenzen in unterschiedlichen Formate). Mit einem Klick auf den Menüeintrag *Export Movie* ruft der Autor ein Dialogfenster auf, und definiert, in welchem Format das Dokument exportieren soll. Abbildung 3-3 zeigt den *Export Movie*-Dialog und den *Import*-Dialog im Flash 4.

Eine andere Alternative für die Wiederverwendung einzelner Bestandteile sind die *Copy* und *Paste* Befehle im Edit-Menü. Rechts in der Abbildung 3-2 kann man das Edit-Menü

sehen, mit dem der Autor eine oder mehrere Medienelemente im Quell-Zeitrahmen auswählen und kopieren, und dann im Ziel-Zeitrahmen wieder einfügen kann. Hier kann der Autor aber nur die in aktuellem Zeit-Rahmen gezeigte Medienelemente kopiert werden.

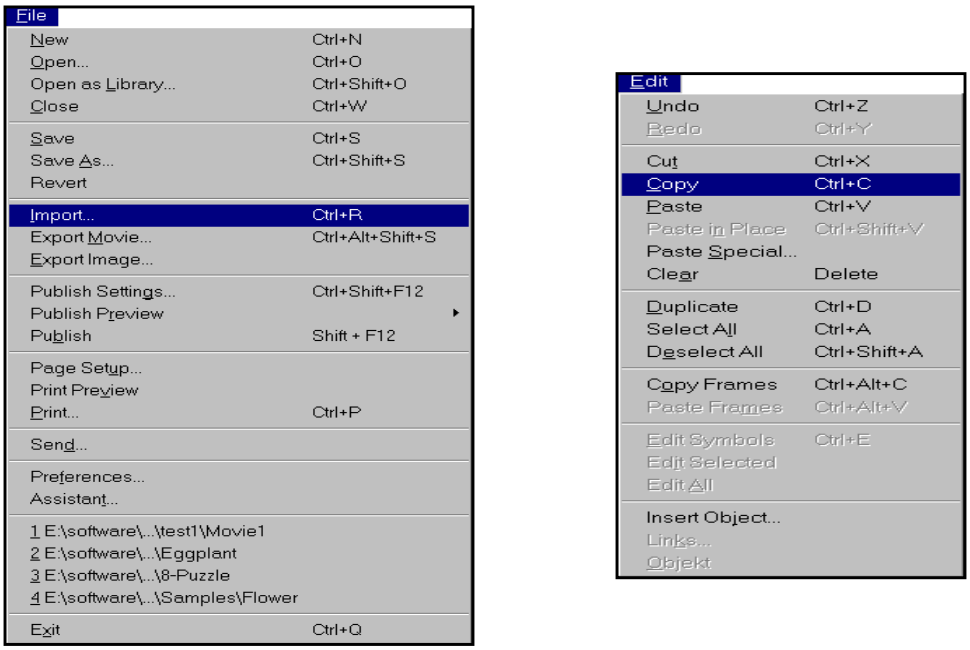


Abbildung 3-2. Das File-Menü (links) und das Edit-Menü (rechts)vom Flash 4

Weder das Import-Export-Konzept noch Copy-Paste-Konzept eignen sich zur Erstellung von Vorlagen. Aber diese zwei Lösungen ermöglichen schon einigermaßen die Wiederverwendung. Um einen Teil vom Dokument später in anderen Dokumenten wiederzuverwenden, muß der Autor zuerst das Dokument speichern können. Erst wenn die Persistenz der Arbeit (*persistency*) garantiert ist, d.h. das Dokument in einer Datei speichern zu können, kann der Autor einen Teil vom Dokument in Dateienexportieren oder speichern. Später bei der Erstellung von neuen Multimedia-Dokumenten kann der Autor das komplette alte Dokument aus der entsprechenden Datei importieren, oder die Dokument-Datei öffnen und Teile davon mit Kopieren und Einfügen wiederverwenden.

Da es sich bei Vorlagen eine Wiederverwendung von strukturelle Informationen behandelt, muß die Persistenz auch für Vorlagen garantiert werden. Das bedeutet, die Vorlagen müssen in einer Datei gespeichert werden können, um die strukturelle Informationen auch nach der Beendung des Autorenprogramms behalten zu können. Damit können später die

strukturelle Informationen aus den Vorlagen-Dateien geladen und wiederverwendet werden.

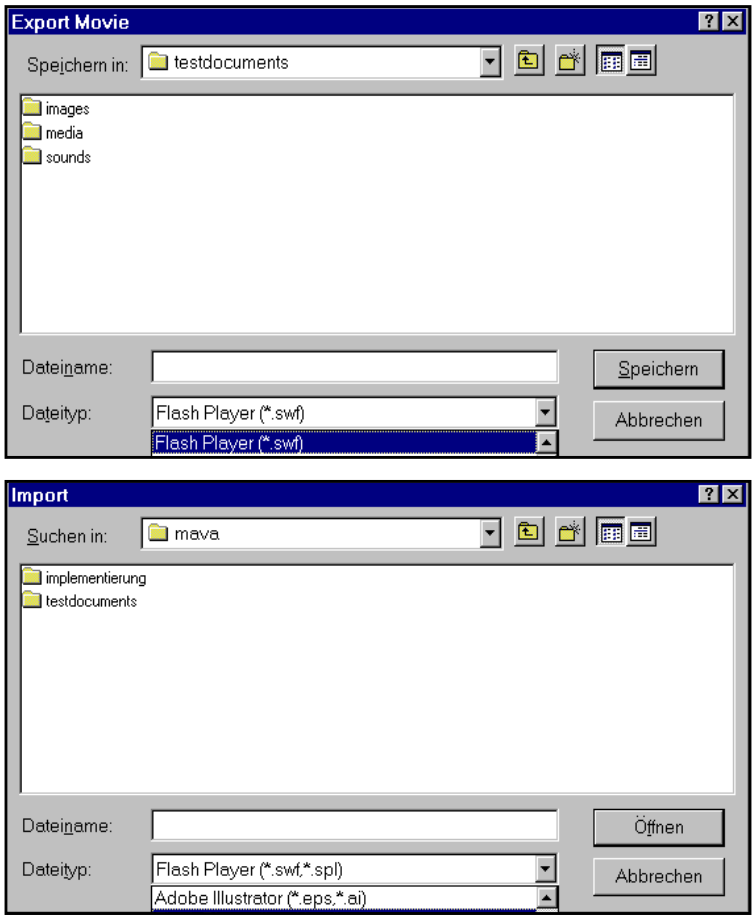


Abbildung 3-3. Der Export Movie-Dialog (oben) und der Import-Dialog (unten) von Flash4

Die anderen obengenannten Autorensysteme funktionieren in ähnlicher Weise wie Flash 4. Das Kopieren und Einfügen von einzelnem oder mehreren Medienelementen ist bei allen Autorensystemen möglich. In IBM Hotmedia und Tribeworks iShell ist Kopieren und Einfügen die einzige Lösung für die Wiederverwendung der Arbeit vom Autor. Bei anderen Autorensystemen beispielsweise Multimedia Builder, stehen ähnliche Import- und Export-Methoden zur Verfügung, um dem Autor zu helfen, einfache und kleinere Multimedia-Dokumente wiederzuverwenden.

Die Nachteile der Multimedia-Autorensystemen ohne Vorlagen-Unterstützung sind hier offensichtlich: großer Aufwand bei der Erstellung und weniger Flexibilität für die Wiederverwendung.

3.2 Autorensysteme mit Vorlagen

Die obengenannten Autorensysteme ohne Vorlage-Unterstützungen bieten zwar die grundlegende Funktionalität für die Erstellung von Multimedia-Dokumenten, erfordert aber lange Einarbeitungsdauer vom Autor, wenn er keine vorherige Erfahrungen mit solchen Autorensystemen hat.

Die Funktionalität der Wiederverwendung in diesen Autorensystemen ist auch schwach. Die Lösungen wie Kopieren-Einfügen oder Importieren-Exportieren sind immer nicht intuitiv genug und leiden an wenig Flexibilität im Vergleich zur Vorlagen. Zum Beispiel mit Kopieren-Einfügen kann der Autor nur die Medienelemente und logische Struktur in aktuellem Zeit-Rahmen wiederverwenden, und mit Importieren kann nur der ganze Dokumentinhalt wiederverwendet werden. Ohne die Unterstützung für Vorlagen ist es für den Autor unmöglich, beliebige viele Medienelementen und deren logische Struktur auszuwählen und wiederzuverwenden.

Daher bieten immer mehr Multimedia-Autorensysteme Vorlagen als Standardkomponente. Hier werden folgende vertretende Autorensysteme diskutiert, um herauszufinden, wie Vorlagen in diesen Systemen konzipiert und integriert sind, und welche Vorteile oder Nachteile diese Systeme jeweils besitzen.

3.2.1 Microsoft Powerpoint 2000

Powerpoint ist ein Produkt speziell für die Erstellung der elektronischen Bildschirmpräsentationen, Overheadfolien, Vortragsnotizen usw. Zwar ist die Nutzung vom Powerpoint mehr auf die Erstellung von Präsentationsseiten von Vorträge beschränkt, sind jedoch eine große Menge von multimedialen Medienelementen in Powerpoint verfügbar. Da sich damit einfache Multimedia-Präsentationen erstellen lassen, wird Powerpoint hier als ein Multimedia-Autorensystem betrachtet.

Powerpoint gibt dem Autor folgende Möglichkeiten, eine neue Präsentation zu erstellen:

- Wenn der Autor Powerpoint nicht gut auskennt, kann er mit einem Autoinhalt-Assistent anfangen. Dadurch beginnt er mit einer Präsentation, bei der Inhalt und Layout vorgeschlagen werden.

- Der Autor kann eine Entwurfsvorlage auswählen, die zwar das Layout, aber nicht den Inhalt der Präsentation festlegt.
- Eine Präsentation kann erstellt werden, indem der Autor von einer bereits vorhandenen Präsentation ausgeht und diese entsprechend Ihren Anforderungen anpaßt.
- Der Autor kann mit einer leeren Präsentation beginnen, ohne Inhalt oder Layout Vorschläge.

In der Abbildung 3-4 ist das Dialogfenster des Autoinhalt-Assistenten zu sehen. Schritt für Schritt wird der Autor nach den Präsentaionstyp, den Präsentationsformat und den andere Präsentationsoptionen gefragt, die er mit Hilfe des Autoinhalt-Assistenten auswählen und definieren kann.

Der Autoinhalt-Assistent bietet dem Autor eine visuelle Schnittstelle, worin der Autor eine Menge von Parameter festlegen kann. Dadurch wählt der Autoinhalt-Assistent eine entsprechende Vorlage für den Autor aus. Zum Beispiel wählt der Autor zuerst seinen Präsentationstyp aus ein paar Kategorien wie "*Alle*", "*Allgemein*" oder "*Projekte*". In jeder Kategorie gibt es Vorlagen für unterschiedlicher Präsentationstypen.

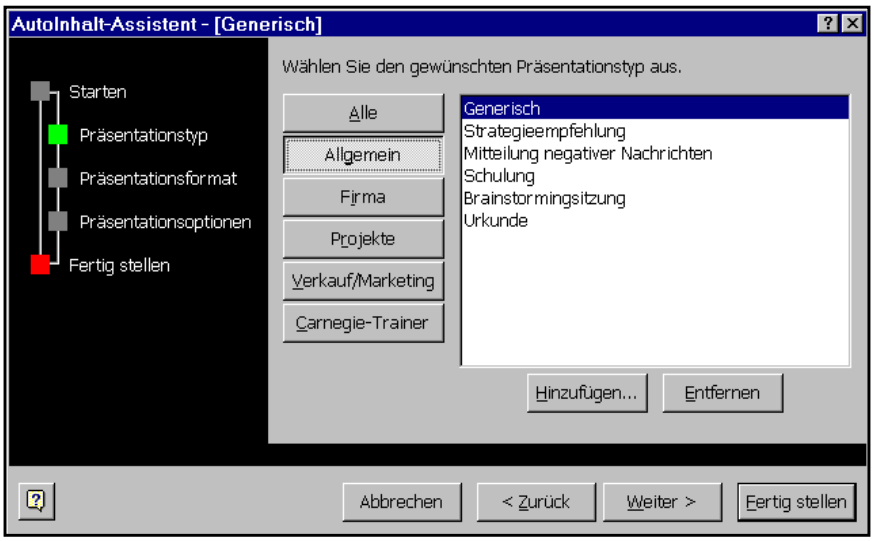


Abbildung 3-4. Erstellung von Präsentationen mit dem Autoinhalt-Assistent in Powerpoint

Mit dem Autoinhalt-Assistent kann der Autor eine Vorlage nach seiner Anforderung auswählen. Abbildung 3-5 zeigt dem Autor eine Vorlage mit Inhalt- und Layout-Vorschläge. Links im Arbeitsfenster sieht man eine Gliederung des Inhalts mit vordefinierter inhaltli-

chen Struktur. Rechts kann man das Layout der Präsentation sehen, wo die Schriftart, Hintergrundfarbe und räumliche Anordnung der Vorlage vorgegeben sind.

Diese Präsentation in Abbildung 3-5 ist aus Sicht des Autors noch nicht fertig, da die vorgegebene inhaltliche Struktur sehr allgemein ist und noch weitere Bearbeitung vom Autor erfordert. Aber der Autor kann diese Präsentation schon als “Prototyp” auf dem Bildschirm ausprobieren, um einen Eindruck des Aussehen der späteren Präsentation zu bekommen. Der Autoinhalt-Assistent selbst ist keine Vorlage, sondern nur eine visuelle Benutzerschnittstelle. Aber mit Hilfe des Autoinhalt-Assistents kann der Autor die Vorlagen von Powerpoint nach seinem Bedarf schnell ausfindig machen und konfigurieren. Damit kann er in kurzer Zeit eine Präsentation erstellen und ausprobieren, auch wenn er keine vorherige Erfahrung mit Powerpoint hat. Der Einarbeitungsaufwand ist daher deutlich reduziert.

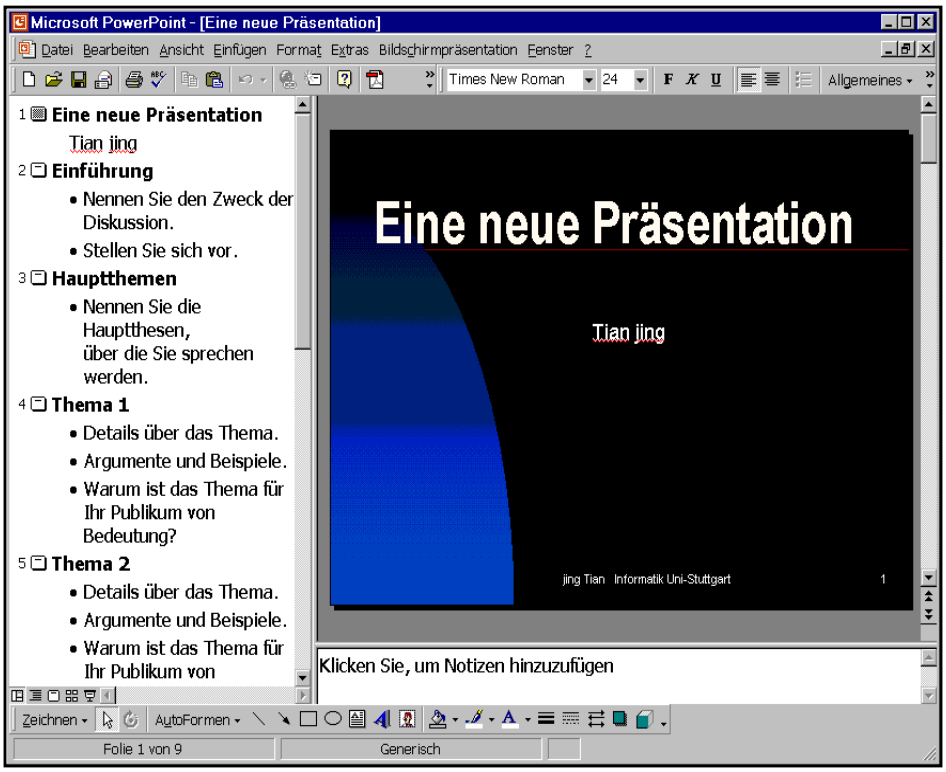


Abbildung 3-5. Eine Prototyp Powerpoint-Datei mit vorgegebene logische Struktur und Layout

Wenn der Autor keine vorgegebene inhaltliche Struktur braucht, kann er zu einer Entwurfsvorlage greifen. In Powerpoint sind zahlreiche Entwurfsvorlagen verfügbar, die jeweils

eigene Schriftarten, Hintergrundelemente und räumliche Anordnung auf der Seite vorgeben.

Abbildung 3-6 zeigt das Dialogfenster, in dem der Autor eine Entwurfsvorlage auswählen kann. Mit der Vorschau rechts kann der Autor sofort die gewählte Vorlage sehen. Wenn der Autor der Präsentation eine Entwurfsvorlage zuweist, wird die Präsentation durch dieses Layout ersetzt. Nach dem Anwenden einer Entwurfsvorlage erhält jede hinzugefügte Folie dasselbe Erscheinungsbild wie die Vorlage. Die Entwurfsvorlage enthält keine strukturelle Information, sondern nur die Layout Information der Seite. Mit der Entwurfsvorlage kann der Autor schnell das Layout der Präsentation fertigstellen, und spart Zeit, um sich auf den Inhalt der Präsentation zu konzentrieren.

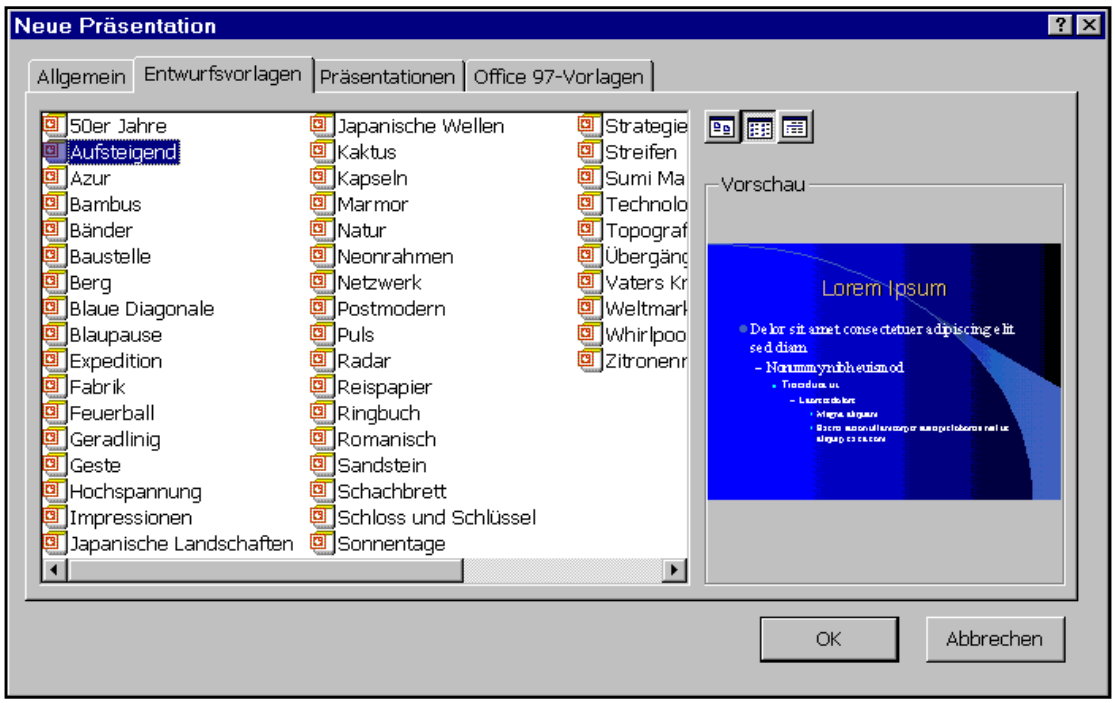


Abbildung 3-6. Entwurfsvorlage Dialog in Powerpoint

Eine besondere Flexibilität der Vorlagen in Powerpoint ist, daß der Autor jederzeit die Entwurfsvorlage der Präsentation ändern kann. Um die Entwurfsvorlage der Präsentation zu ändern, klickt der Autor auf der Präsentationsseite mit der rechten Maustaste. Ein Kontextmenü wird erscheinen, wie links in der Abbildung 3-7 zu sehen, indem der Autor den Eintrag “Entwurfsvorlage übernehmen” auswählen kann. Nachdem der Autor diesen

Menüeintrag ausgewählt hat, wird ein Dialogfenster auf dem Bildschirm erscheinen, wie rechts in der Abbildung 3-7 zeigt, in dem der Autor neue Entwurfsvorlage für die Präsentation auswählen kann.

Das dynamische Wechseln der Entwurfsvorlage ist besonders flexibel und nützlich bei der Erstellung von Präsentationen. Damit braucht der Autor nicht das Layout von jeder Präsentationsseite individuell zu ändern, eine einmalige Änderung der Entwurfsvorlage der ganzen Präsentation ist ausreichend. Mit ein paar Klicks der Maustaste kann der Autor jederzeit das Layout der Präsentation von neuer Entwurfsvorlage übernehmen. Damit bleibt der Editiervorgang sehr flexibel.

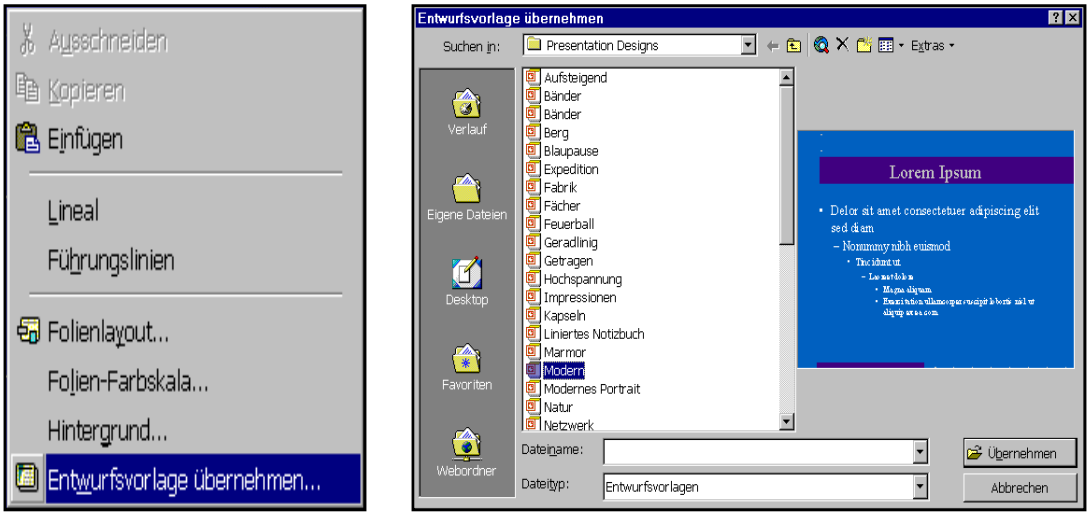


Abbildung 3-7. Übernahme der Entwurfsvorlage in Powerpoint

Nachdem eine Vorlage ausgewählt ist, kann der Autor diese Vorlage gemäß seinen Anforderungen anpassen, indem er mit der Maus direkt auf den Medienelementen (beispielsweise. Textfelder, Bilder, Icons von Audio- und Video-Sequenzen) auf der Seite klickt und deren Eigenschaften ändert.

In PowerPoint stehen dem Autor folgende zwei Typen von Vorlagen zur Verfügung:

- Entwurfsvorlagen: Entwurfsvorlagen enthalten vordefiniertes Layout für Präsentationsseite. Der Autor kann Entwurfsvorlage in beliebiger Präsentation anwenden, um ein individuelles Erscheinungsbild zu verleihen.

- Inhaltsvorlagen: Inhaltsvorlagen besitzen nicht nur die vorgefertigtes Layout der Präsentationsseite, sondern auch die inhaltliche Struktur. Wie Abbildung 3-5 zeigt, sind die Inhalte und die Reihenfolge aller Präsentationsseiten auch vorgegeben.

Der Autor kann nicht nur sämtliche Vorlagen entsprechend seinen Anforderungen anpassen, sondern neue Vorlagen auf der Grundlage einer bereits erstellten Präsentation automatisch erstellen lassen. Außerdem kann er eigene Vorlagen dem AutoInhalt-Assistenten hinzufügen, um diese Vorlagen nächstes Mal im Dialogfenster vom Autoinhalt-Assistent anzeigen zu lassen.

Um eigene Vorlage zu erstellen, erstellt der Autor eine neue Präsentation mit oder ohne Hilfe des Autoinhalt-Assistent oder Entwurfsvorlage. Nach der Bearbeitung kann der Autor diese Präsentation statt in einer normalen Powerpoint Präsentation-Datei in einer Vorlage-Datei speichern.

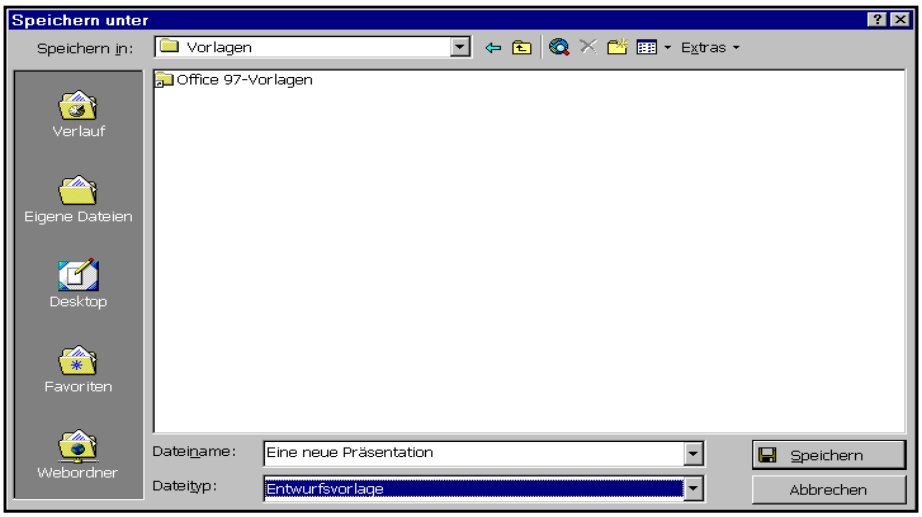


Abbildung 3-8. Die Präsentation als eigener Vorlage in Powerpoint zu speichern

Wählt der Autor den Eintrag "*Speichern unter*" im Datei-Menü, erscheint ein Dialogfenster wie in der Abbildung 3-8 zeigt, worin der Autor die *Entwurfsvorlage* als Dateityp wählen kann. Nachdem die Präsentation in einer Vorlage-Datei gespeichert ist, kann der Autor sie jederzeit als Entwurfsvorlage verwenden. Andererseits können die eigenen Vorlagen dem Autoinhalt-Assistent hinzugefügt werden, indem der Autor mit der Maus auf die "Hinzufügen" Schaltfläche im Autoinhalt-Assistent Dialogfenster klickt. Dann wird ein Dialogfenster erscheinen, worin der Autor seine eigene Vorlage auswählen und hinzufügen kann.

Nach dem Hinzufügen der eigenen Vorlage kann der Autor sie zukünftig im Dialogfenster vom Autoinhalt-Assistent sehen, auswählen und benutzen.

Die Powerpoint-Vorlage wird in Vorlagedatei mit der Datei-Erweiterung (.pot) anstatt (.ppt) für normaler Präsentationsdatei gespeichert. Powerpoint speichert alle Vorlagen im Verzeichnis “Vorlage”, aber der Autor kann seine Vorlage-Datei auch in einem anderen Verzeichnis speichern. Alle Dateienmit der Datei-Erweiterung *“.pot”* können direkt mit Powerpoint geöffnet und als normale Präsentationsdateien auf dem Bildschirm vorgeführt werden.

Vorlagen werden durch Powerpoint gut unterstützt. Die Einarbeitung mit Hilfe des Autoinhalt-Assistenten ist sogar für Autoren ohne Nutzungserfahrung von Powerpoint relativ leicht und schnell. Die Vorlagen in Powerpoint können direkt zum Laufen gebracht und auch einfach geändert werden. Der Autor kann jederzeit die Vorlage der Präsentation wechseln, und damit das Layout aller Präsentationsseiten auf einmal zu ändern. Außerdem kann der Autor eigene Vorlagen erstellen und dem Autoinhalt-Assistent hinzufügen, damit er die eigene Vorlage später leicht finden und wiederverwenden kann.

Die Schwachstelle von Vorlagen in Powerpoint sind ebenso auch offensichtlich: Es gibt mehr als genug Vorlagen für das Layout einzelner Folien, aber fast keine für Interaktionen. Die Erstellung von Multimedia-Dokumenten mit komplexen zeitlichen Beziehungen oder Interaktionen ist in Powerpoint entweder schwierig oder unmöglich. Beispielsweise gibt es in Powerpoint keine Vorlagen mit zeitlichen Beziehungen von Medienelementen, daher kann der zeitliche Operator *“while”*, der beispielsweise im MAVA-System bereits integriert ist, in Powerpoint nicht dargestellt werden.

Als Folgerung ist Powerpoint zwar ein gutes Produkt für die Erstellung von multimediale Präsentationen, aber für komplexe Multimedia-Anwendungen sowie interaktive Lernprogramme oder Spiele wird die Grenze schnell erreicht. Es ist wünschenswert, daß die Vorlagen im MAVA-Editor auch die Einfachheit und Flexibilität der Vorlagen in Powerpoint besitzt. Aber die Benutzung von Vorlage im MAVA-Editor soll nicht nur auf das Layout beschränkt sein, sondern auch für interaktive Multimedia-Programme wie CBT, einfache Spiele, usw. verfügbar sein.

3.2.2 Macromedia Authorware 5.2

Authorware ist ein Multimedia-Autorensystem mit Vorlagen-Unterstützung. Der Haupteinsatzbereich von Authorware ist die Erstellungen von multimedialen Lernanwendungen und Training.

Wenn der Autor Authorware startet, erscheinen ein Dokument-Fenster und ein "*Knowledge Objects*"-Fenster im Arbeitsfenster, wie die Abbildung 3-9 zeigt. Das Fenster links ist das Dokument-Fenster, in dem alle Medienelemente und Operatoren stehen. Das Dokument in Authorware ist Zeit-basiert, daher sind alle Icons vom Dokument in einem Ablaufsdiagramm mit dem Pfeil eingetragen. Links vom Dokument-Fenster ist die sogenannte "Symbol Palette", die aus einer Reihe von unterschiedlicher Icons besteht. Auf der "Symbol Palette" sind die oft verwendete Elemente zu sehen, z.B. Icons für Bilder, Audio- und Video-Sequenzen. Rechts vom Dokument-Fenster ist das "*Knowledge Objects*"-Fenster, in dem alle "Knowledge Objects", die Vorlagen-Objekte in Authorware, in verschiedenen Kategorien gezeigt sind.

Um ein neues Dokument zu erstellen, kann der Autor die Element-Icons vom "*Symbol Palette*" oder die "*Knowledge Object*"-Icons vom "*Knowledge Objects*"-Fenster in das Dokument-Fenster einziehen. Danach kann er mit der Maus auf die Icons klicken, und die Eigenschaften von diesen Objekten in dem erscheinenden Dialogfenster ändern.

Es gibt zwei Typen von Vorlagen in Authorware: "Modell" und "Knowledge Objects". Ein Modell besteht aus einer Menge von Medienelementen und logischen Strukturen. Zum Beispiel, wenn der Autor ein Modell für Multiple Choice erstellt, sollen nicht nur die Frageelemente, sondern auch die Interaktion bei der Auswahl eingetragen werden. Der Autor kann die Medienelemente zusammen mit der struktureller Information der Dokumente im Modell speichern und wiederverwenden.

Es ist sehr einfach, ein eigenes Modell zu erstellen. Der Autor öffnet ein neues Dokument oder ein Dokument von einem anderen Autor. Dann bearbeitet er das Dokument nach der Anforderungen. Anschließend selektiert er die Icons im Fenster, die er wiederverwenden möchtet und wählt den Eintrag "*Save in model*" im File-Menü, um diese Medienelemente und Strukturen in ein Modell zu speichern.

Um das Modell wiederzuverwenden, soll die Modell-Datei in dem Verzeichnis "*Knowledge Objects*" gespeichert werden. Später wird das Symbol für das Modell im "*Knowledge Objects*"-Fenster gezeigt, worin der Autor dieses Modell auswählen und in das Dokument-Fenster einziehen kann, wo der Autor das Modell nach der Anforderungen bearbeitet.

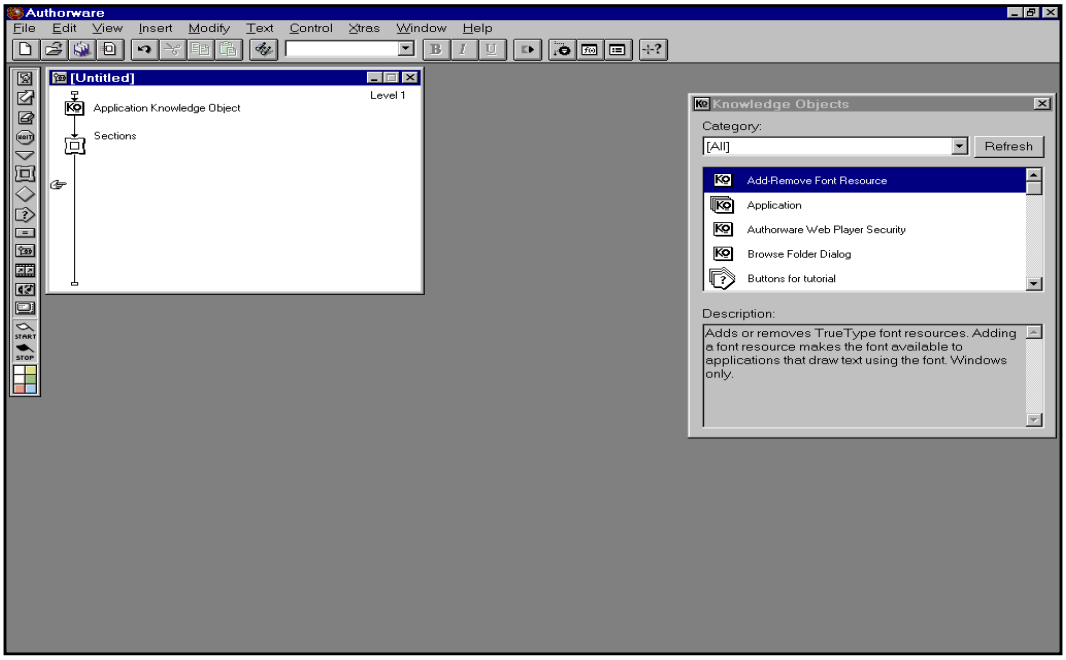


Abbildung 3-9. "New File" Dialog (links) und Die Arbeitsumgebung (rechts) von Authorware 5.2

Der andere Vorlagen-Typ von Authorware ist das "*Knowledge Object*". Der Unterschied zwischen einem Modell und ein "*Knowledge Object"* ist, daß jedes "*Knowledge Object"* mit einem Wizard verbunden ist. Der Wizard entspricht dem Autoinhalt-Assistent in Powerpoint und bietet eine Benutzerschnittstelle, damit der Autor das damit verbundene Modell schrittweise ändern und konfigurierten kann.

Zum Beispiel, wenn der Autor ein "*Quiz"*-Symbol im "*Knowledge Object"*-Fenster auswählt und in das Ablaufsdiagramm im Dokument-Fenster einfügt, wird der mit diesem "*Quiz"* verbundene Wizard automatisch gestartet. Und der Autor kann mit Hilfe des Wizards das "*Quiz"* Schritt für Schritt bearbeiten.

In Authorware sind eine Menge von "*Knowledge Objects"* in mehreren Kategorien (beispielsweise *Assessment, File, Interface Components, Tutorial*, usw.) vorhanden, die der

Autor im *"Knowledge Object"*-Fenster auswählen und ins Dokument-Fenster einfügen kann.

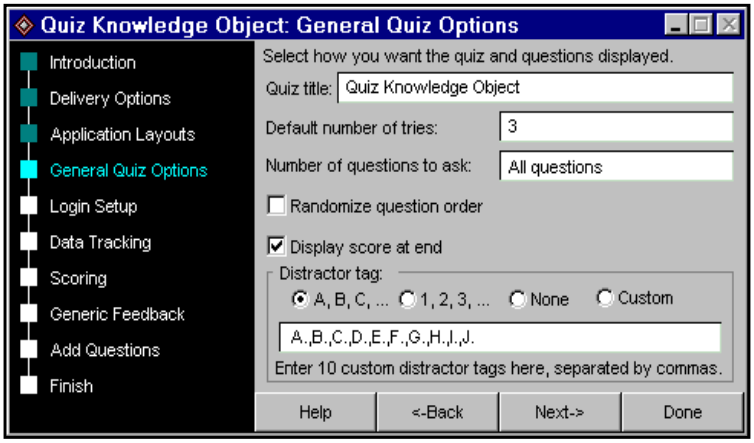


Abbildung 3-10. Das Wizard vom Quiz-*"Knowledge Object"* in Authorware 5.2

Außerdem kann der Autor auch eigene *"Knowledge Objects"* erstellen. Die Erstellung von *"Knowledge Objects"* ist aber komplizierter als die Erstellung eines eigenen Modell. Der Autor klick auf dem Eintrag *"Knowledge Object Icon"* im Menü *"Insert"*, um ein *"Knowledge Object"*-Symbol zu dem Ablaufsdiagramm vom Dokument hinzufügen. Danach klickt er mit der Maus auf das Symbol und öffnet ein neues Fenster vom diesem Objekt, worin er die Medienelement- und Funktion-Icons einziehen kann, um das Objekt nach der Anforderungen zu bearbeiten.

Da jedes *"Knowledge Object"* mit einem Wizard verbunden werden ist, muß der Autor auch ein Wizard Paket-Datei für dieses Objekt erstellen. Der Autor soll die Benutzerschnittstelle und logische Struktur des Wizards erstellen und dann zusammen mit allen benötigte Medienelementen in ein Paket einpacken. Die Wizard-Datei wird mit der Datei-Erweiterung *".a5r"* gespeichert. Anschließend soll der Autor das *"Knowledge Object"*-Symbol mit der Wizard-Datei zusammenverbinden.

Um diese Verbindung zu setzen, klickt der Autor das *"Knowledge Object"*-Symbol mit rechter Maustaste und wählt dem Kontextmenü Eintrag *"Properties"*. Dann erscheint ein Dialogfenster, wie in Abbildung 3-11 gezeigt, in dem der Autor die Wizard-Datei Name eingeben kann. Am Ende speichert der Autor das Objekt in ein Modell. Gleich wie beim

Modell wählt der Autor den Eintrag "*Save in model*" im "*File*"-Menü, und speichert das Objekt in einer Datei mit der Datei-Erweiterung "*.a5d*" in dem Verzeichnis "*Knowledge Objects"*.

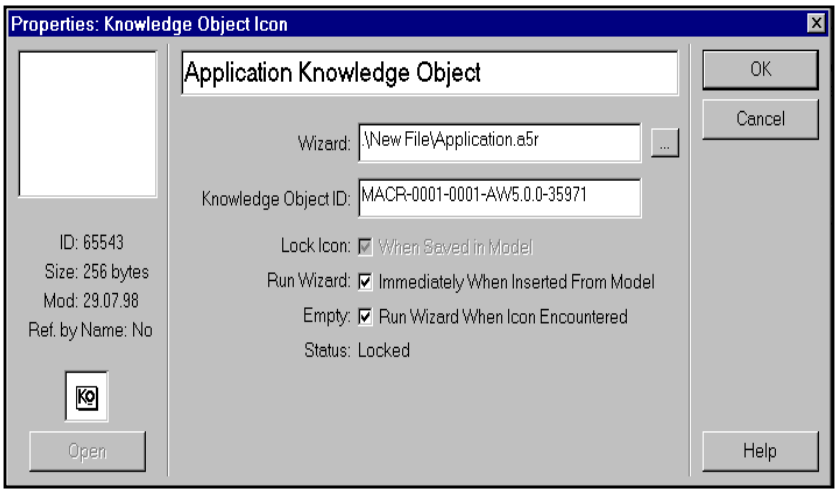


Abbildung 3-11. Das *Properties* Dialogfenster vom *"Knowledge Object"*

Die Objekte werden in verschiedenen Kategorien gespeichert, z.B. *"All"* steht für alle Typen der Objekte, *"File"* beihält alle Objekte, die Datei verarbeiten, *"New File"* Objekte werden nach dem Start von Authorware automatisch gezeigt, in dem der Autor den zu bearbeitenden Dokument-Typ auswählen kann. Jede Kategorie entspricht ein Unterverzeichnis im Verzeichnis *"Knowledge Objects"*. Je nach dem Typ des erstellten *"Knowledge Object"*, speichert der Autor die Objekt-Datei in ein entsprechendem Unterverzeichnis. Nachdem die eigene Objekte in dem Unterverzeichnis gespeichert sind, werden die auch in dem Fenster gezeigt.

Der Autor kann auch ein neues Unterverzeichnis erstellen, das später auch im "*Knowledge Objects"*-Fenster, wie in der abbildung 3-12 zu sehen, erscheinen kann.

Die Wiederverwendung des *"Knowledge Object"* ist ganz einfach, der Autor wählt das Objekt-Symbol aus dem *"Knowledge Objects"*-Fenster und zieht es in dem Dokument-Fenster, das Wizard vom Objekt wird automatisch gestartet, Schritt für Schritt kann der Autor die Eigenschaften des Objekts nach der Anforderungen setzen. Auch später kann der Autor diese Eigenschaften jederzeit ändern, indem er das Objekt-Symbol klickt und den

Wizard nochmal startet. Die Bearbeitung des Objekts im Dokument ändert das originale "*Knowledge Object*" nicht, nur eine Kopie davon wird verändert.

Das Vorlagen-Konzept wird von Authorware sehr gut unterstützt, mit Hilfe von Modellen und "*Knowledge Objects*" kann man nicht nur Medienelemente, sondern auch die logische Struktur wiederverwenden. Der "*Knowledge Object*" Wizard hilft dem Autor, die Eigenschaften des Objekts in der Benutzerschnittstelle schrittweise zu setzen oder ändern. Der Autor kann nicht nur die vorhandene Vorlagen von Authorware benutzen, sondern eigene Vorlagen erstellen.

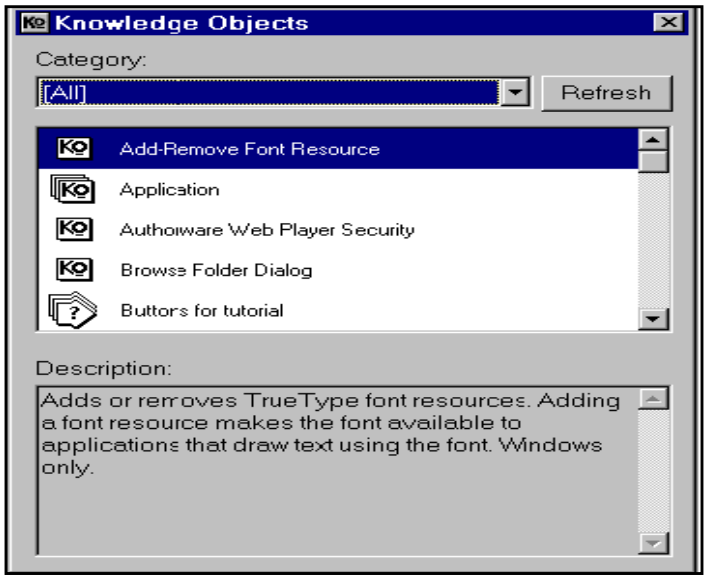


Abbildung 3-12. das "Knowledge Objects" Dialogfenster in Authorware

Allerdings besitzt Authorware eine große Menge von System Variablen und Funktionen, die der Autor gut kennen muß, um die Vorlagen effektiv zu benutzen, oder eigene "*Knowledge Objects*" und die dazugehörende Wizard zu erstellen. Daher ist hier eine lange Einarbeitungszeit für den Autor ohne vorheriger Erfahrungen mit Authorware unvermeidlich.

Im Vergleich zu Powerpoint sind die Vorlagen in Authorware nicht auf das Layout der Präsentationsseiten eingeschränkt. Die Vorlagen in Authorware sind für mehrere Typen von Multimedia-Dokumenten vorhanden. Die Konzepte wie Modell, "*Knowledge Object*" und Wizard von Authorware sind sehr interessant und können auch im MAVA-Editor gut eingesetzt werden. Damit können alle Multimedia-Medienelemente und Operatoren im

MAVA-Editor leicht wiederverwendet werden, und der Autor kann auch selbst eigene Vorlagen erstellen. Aber die zusätzliche Belastungen wie System Variablen und Funktionen von Authorware werden zukünftig im MAVA vermieden, um die Einarbeitungszeit für neuen Autor möglichst zu reduzieren.

3.2.3 Multimedia ToolBook 4.0 CBT Edition

Multimedia ToolBook ist ein Produkt der Firma Asymetrix für die Erstellungen von multimediale Anwendungen, z.B. Präsentationen, Kiosk Information Systeme, Spiele usw. Die CBT Edition von ToolBook 4.0 ist in der Lage, interaktive Lernanwendungen zu erstellen.

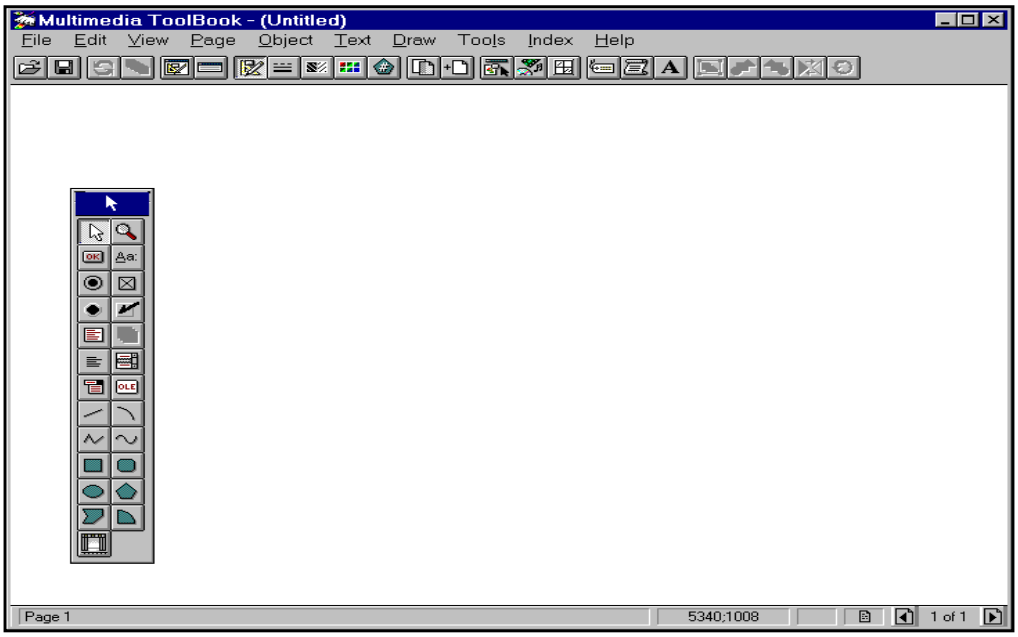


Abbildung 3-13. Das Arbeitsfenster in ToolBook 4.0 CBT Edition

Ein *Buch* mit der Datei-Erweiterung (.tbk) ist die grundlegende Datei, die durch ToolBook erstellt wird. Alle multimediale Anwendungen, die von ToolBook erstellt werden, sind in *Buch*-Dateien gespeichert. Ein Autor baut ein *Buch* auf, indem er zuerst die Seiten vom *Buch* erstellt, dann die Objekte in den Seiten einfügt und anschließend die Eigenschaften der Objekte setzt und die Aktionen der Objekte in *OpenScript* programmiert.

CBT Edition von ToolBook hat einen sogenannte *Buch Specialist*, um dem Autor zu helfen, das grundlegende Struktur-Skelett zu erstellen. Ein *Buch Specialist* ist eine Benutzerschnittstelle ähnlich wie *Wizard* in Authorware oder *Assistent* in Powerpoint. Der Autor

setzt die Parameter oder wählt die Konfiguration, das *Buch Specialist* wird die entsprechende Vorlagen benutzen, um eine grundlegende Struktur zu erstellen, und die notwendigen Skripte einzusetzen. In der Abbildung 3-14 kann man sehen, wie der Autor die Struktur durch *Buch Specialist* setzen kann.

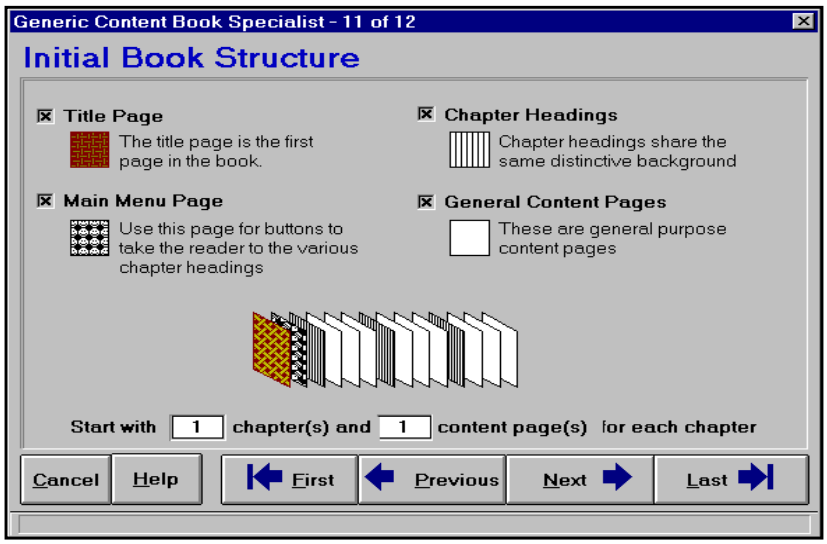


Abbildung 3-14. Das *Buch Specialist* von ToolBook 4.0 CBT Edition

Der *Buch Specialist* selbst ist eine Multimedia ToolBook Buch-Datei mit der Datei-Erweiterung (.spb), und wird im Verzeichnis "*SPCLST*" gespeichert. Der Autor kann zwar eigene *Buch Specialist* Datei erzeugen, aber dafür ist ein sehr gründliches Verständnis der *OpenScript*-Sprache von ToolBook ist hier erforderlich. Daher ist die Erstellung eines *Buch Specialist* keine Arbeit für normale Autoren, sondern für Programmierer.

Die CBT Edition von ToolBook unterstützt Vorlagen durch zwei Konzepte: *Layout Templates* und *Widgets*. Wobei ist *Layout Template* eine Vorlage für die Hintergründen der einzelnen Seiten in *Buch*-Dokumenten, und *Widget* ein Multimedia ToolBook Objekt mit vorgegebenen Skript oder Eigenschaften.

Die *Layout Templates* sind eine Sammlung der Hintergründen, die der Autor zu seiner Seite importieren kann. Der Hintergrund hier beinhält nicht nur die Hintergrundfarbe und Schriftart, sondern auch manche allgemeine Objekte wie Textfelder und Buttons. ToolBook liefert eine Menge von *Layout Template*s, die in unterschiedlichen Unterverzeichnissen des Verzeichnis *Template* gespeichert sind. Der Autor kann eine *Layout Template*

solwohl in *Buch Specialist* auswählen als auch später in seine Seit importieren. Außerdem kann der Autor auch eigene *Layout Templates* erstellen, indem er ein neue Vorlage entwikkelt oder eine vorgegebene Vorlage modifiziert. Danach soll dieses neue *Layout Template* in Verzeichnis *Template* mit der Datei-Erweiterung (.ptp) gespeichert werden.

Die Widgets in ToolBook sind vorgefertigte Objekte, die der Autor direkt benutzen kann, z.B. graphische Elemente, Action Buttons, Questions, usw. ToolBook bietet eine große Menge von unterschiedlichen Widgets, die der Autor direkt benutzen kann. Alle diese Widgets Objekte sind nach verschiedenen Kataloge im Verzeichnis *Widgets* gespeichert.

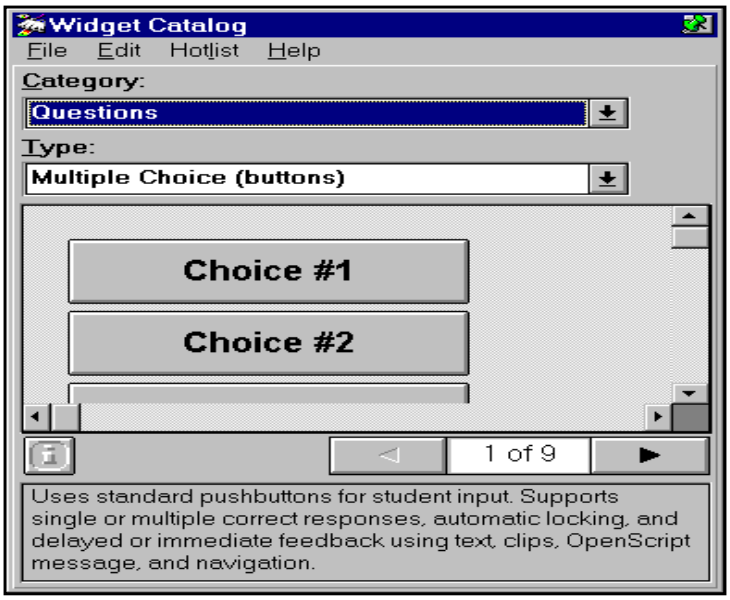


Abbildung 3-15. Das Widget-Dialogfenster von ToolBook 4.0 CBT Edition

Wenn der Autor den Eintrag *New Widget* im Menü *Object* wählt, erscheint das Widget-Dialogfenster, wie die Abbildung 3-15 zeigt, in dem der Autor ein Objekt auswählen kann. Nachdem der Autor das Objekt ausgewählt hat, kann er dieses Objekt mit Drag&Drop in die Seite einziehen. Da die Eigenschaften und Aktionen der Widget Objekte schon vorgefertigt sind, kann der Autor viel Zeit sparen, indem er direkt die Widget Objekte benutzt.

Der Autor kann auch selbst einen neuen Widget Katalog erstellen und eigene Widget Objekte einsetzen. Wenn der Autor neue Widget Objekte erstellt hat, soll er diese Objekte im Verzeichnis *Widgets* mit der Datei-Erweiterung (.wbk) speichern. Später werden die neue erstellte Kategorie und Objekte ebenfalls im Widget-Dialogfenster gezeigt. Aber die

Erstellung von neuen Widget Objekte ist nicht einfach, da es OpenScript-Programmierung erfordert, was für einen Autor ohne Programmierkenntnisse sehr schwierig ist.

Die Vorlagen sind in ToolBook 4.0 CBT Edition gut unterstützt. Einerseits bietet ToolBook eine Vielzahl von verschiedenen Widget Objekten und Layout Vorlagen, die für allgemeine Arbeit sehr nützlich sind. Andererseits kann der Autor nach eigenem Bedarf neue Widgets und Layout-Vorlagen erstellen. Und die Vorlagen in ToolBook sind nicht nur für Seitenlayout, sondern auch komplexe Dokumente verfügbar.

Aber die Anforderung an OpenScript-Programmierung ist immer eine große Beschränkung für die Autoren, wenn der Autor eigene Vorlagen oder komplexe Dokumente erstellt. Beispielsweise muß der Autor die Aktionen in OpenScript schreiben, wenn er neue Widget Objekte erstellt. Damit ist die Anforderungen an den Autor in ToolBook immer noch zu hoch. Da MAVA gar keine Skript-Sprache benutzt, gibt es keine solche Anforderung.

3.2.4 Weitere Untersuchungen

Weitere in dieser Arbeit untersuchte Multimedia-Autorensysteme waren z.B. Matchware Mediator 5 und Question Mark Designer.

Mediator 5 ist ein Produkt der Firma Matchware für Multimedia-Präsentation, Computer Based Training (CBT) usw. und bietet schon einige Vorlagen in unterschiedlichen Kategorien, z.B. Präsentationen, Foto-Alben und CBTs. Der Autor kann jederzeit das Wizard-Fenster aufrufen, wie in der Abbildung 3-16 zu sehen, um eine Vorlage aus der vorhandenen Liste zu wählen. Alle Vorlagen im Mediator sind eigentlich vorgefertigte Mediator Dokumente und werden in der Dateien mit der Datei-Erweiterung wie normale Dokumente gespeichert. Daher kann eine Vorlage direkt geöffnet und zum Laufen gebracht werden. Der Autor kann zwar diese Vorlagen benutzen und entsprechend eigenen Anforderungen ändern, er kann aber keine Vorlagen selbst erstellen. Nur die vorhandenen Vorlagen, die im Wizard-Fenster gezeigte sind, stehen dem Autor zur Verfügung, daher ist die Erweiterbarkeit von der Vorlagen im Mediator unmöglich. Diese Beschränkung, daß Vorlagen nicht von dem Autor selbst erstellt werden können, soll im MAVA vermieden werden.

Question Mark Designer ist ein Produkt der Firma Question Mark Computing für die Erstellungen von CBTs. Die Erstellung von Question Mark Dokumenten ist Vorlage-

basiert. Für jede Frage muß der Autor eine passende Vorlage im Vorlage-Fenster auswählen, wie die Abbildung 3-17 zeigt, und dann diese Vorlage nach der Anforderungen bearbeiten. Der Autor kann die bearbeitete Frage wieder als eigene Vorlage speichern. Später werden diese erstellten Fragen-Vorlagen auch im Vorlagen-Fenster gezeigt. Die Wiederverwendung und Bearbeitung der Vorlagen ist relativ intuitiv. Allerdings werden alle Vorlage im Question Mark Designer in einer einzigen Datei genannt *"Style.qds"* gespeichert. Wenn ein Autor die Vorlagen von einem anderen Autor benutzen möchtet, muß er die *"Style.qds"* Datei von diesem Autor in eigenem Verzeichnis kopieren. Aber so wird seine eigene Vorlagen-Datei übergeschrieben, und alle die Vorlagen von ihm selbst werden verloren.

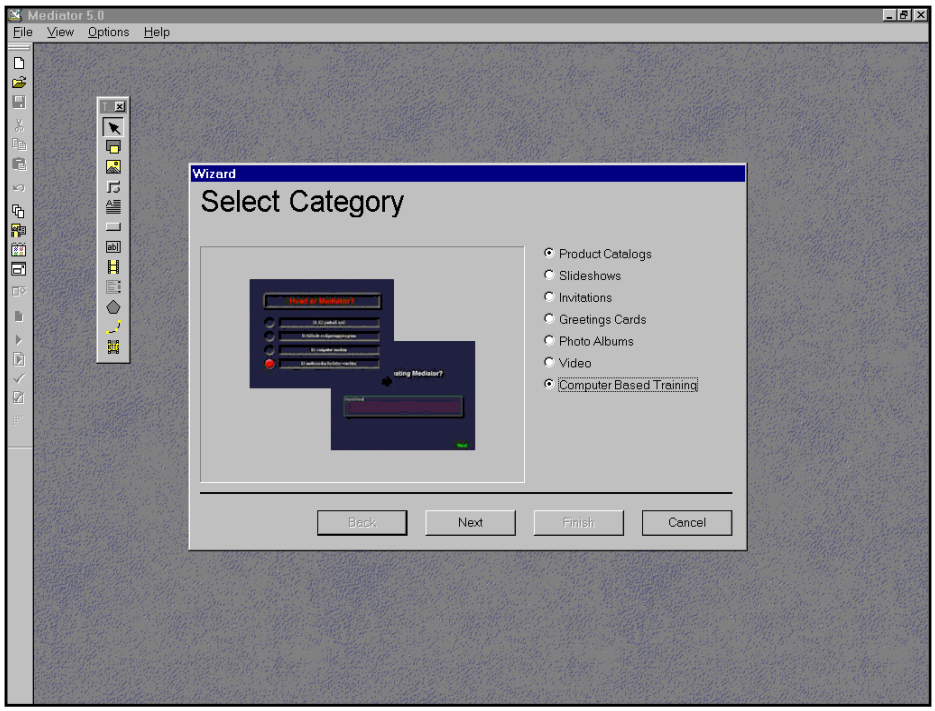


Abbildung 3-16. Das Wizard Fenster im Mediator 5

Da die Vorlagen in Question Mark Designer nicht getrennt gespeichert werden können, gibt es einer große Nachteil bei der Verteilung der Vorlagen. Zum Beispiel, wenn einer Autor eine Vorlage von einem anderen Autor benutzen möchte, muß er die ganze Vorlagen-Datei von dem anderen Autor statt nur die benötigte Vorlage kopieren. Diese Nachteile sollen im MAVA unbedingt vermieden werden.

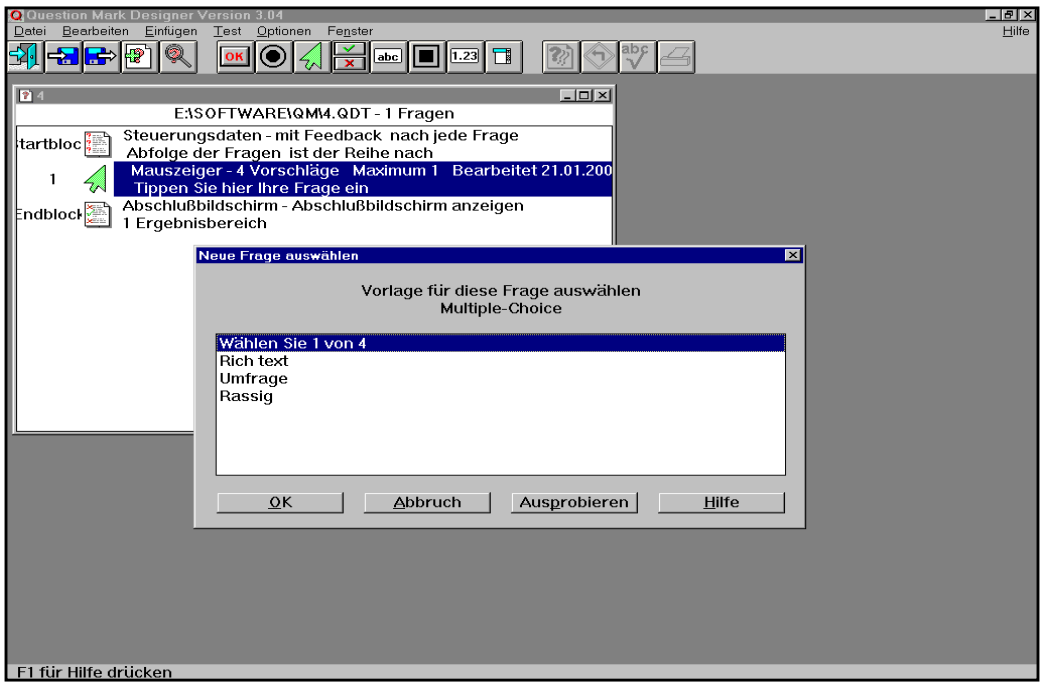


Abbildung 3-17. das Question Mark Designer Arbeitsfenster

3.3 Zusammenfassung

In diesem Kapitel wurden einige vertretende Multimedia-Autorensysteme mit und ohne Vorlagen-Unterstützung untersucht. Durch Untersuchungen kann man feststellen, daß Vorlagen die Effizienz und Flexibilität der Arbeit von Autor bei Erstellung der Multimedia-Dokumenten deutlich erhöht hat. Die Autorensysteme mit Vorlagen haben jeweils Vorteile und Nachteile, z.B. Powerpoint bietet zwar gute Vorlagen, aber nur in sehr beschränktem Anwendungsgebiet (Präsentationen). Im Vergleich dazu können Authorware und ToolBook Vorlagen für verschiedenen Anwendungsbereiche flexibel einsetzen, aber die beiden leiden unter zu hoher Anforderungen an die Skript-Programmierung. In nächstem Kapitel werden die Untersuchungsergebnisse in diesem Kapitel weiter diskutiert und die Konzeption der Vorlagen im MAVA-System durchgeführt.

4 Konzeption von Vorlagen für den MAVA-Editor

Im Kapitel 3 wurden mehrere Multimedia-Autorensystemen untersucht, um die Konzeption und Realisierung von Vorlagen in diesen Systemen zu studieren. In diesem Kapitel werden zuerst die Anforderungen an die Vorlagen im MAVA-System anhand der Untersuchungsergebnisse aus Kapitel 3 festgestellt. Anschließend werden die Vorlagen für das MAVA-System konzipiert. Am Ende dieses Kapitels wird eine kurze Zusammenfassung gegeben.

4.1 Anforderungsanalyse

Im Kapitel 3 wurden einige exemplarische Multimedia-Autorensysteme, die Vorlagen anbieten, bereits untersucht. Da diese Autorensysteme jeweils in speziellen Anwendungsgebieten eingesetzt werden, sind die Anforderungen an deren Vorlagen auch sehr unterschiedlich. Zum Beispiel behandeln die Vorlagen in Powerpoint das Layout der Präsentationsseiten. Im Vergleich dazu legen die Vorlagen in Authorware mehr Wert auf die Struktur des Dokuments und die Interaktionen zwischen Benutzern und Medienelementen. Wie im Kapitel 3 gesehen, kein Vorlagenkonzept der untersuchten Autorensystemen den Anforderungen in allen Aspekten erfüllen.

Das MAVA-System ist ein erweiterbares multimediale Präsentation- und Autorensystem, das in verschiedenen Anwendungsgebieten eingesetzt werden kann. Daher sollen die Vorlagen im MAVA-System unabhängig von Anwendungen bleiben. Das bedeutet, die Vorlagen im MAVA-System sollen nicht nur für bestimmte vorgesehenen Anwendungsgebieten wie z.B. die Präsentationen oder interaktive Lernanwendungen, sondern auch alle möglichen Anwendungsgebieten nutzbar sind.

Im folgenden werden zuerst zwei fundamentale Anforderungen betrachtet, die für alle Vorlagen wesentlich sind und auch als minimale Anforderungen für MAVA-Vorlagen gelten:

4.1.1 Persistenz von Vorlagen

Persistenz von Vorlagen bedeutet, daß die Informationen in Vorlagen lange erhalten bleiben. Die Informationen in Vorlagen gehen nicht verloren, auch wenn das Autorenprogramm beendet wird oder der Computer ausgeschaltet wird. Um die Persistenz-Anforderung zu erfüllen, muß man die Vorlagen in Dateien speichern können.

Folgend wird ein Beispiel genommen, um die Nachteile zu zeigen, wenn die Persistenz der Vorlagen im MAVA-System nicht unterstützt wird:

Wenn der Autor ein Autorenprogramm benutzt, möchte er immer die vorhandenen Vorlagen benutzen, um die wiederholende Arbeit zu sparen und die Effizienz der Arbeit zu erhöhen. Ohne Persistenz kann keine Vorlage in Dateien gespeichert werden. Daher muß der Autor selbst alle Vorlagen von Anfang an erstellen, und die Vorlagen nur bis zur Beendigung des gerade laufenden Programm benutzen. Da die Vorlagen nicht in Dateien gespeichert werden können, gehen alle Vorlagen verloren, sobald das Autorensystem beendet wird.

Das Beispiel zeigt, daß die Persistenz der Vorlagen eine minimale Anforderung an die Verwendung der Vorlagen ist. Alle untersuchten Autorensysteme im Kapitel 3 können diese Anforderung erfüllen. Und diese Anforderung muß im MAVA-Editor auch erfüllt sein.

Persistenz von Vorlagen ist auch wesentlich im Hypermedia-Bereich, “Name and save a template for future use” gilt als die fundamentale Anforderung an die Hypermedia-Vorlagen [Catl91]. Allerdings gibt es Unterschiede zwischen Vorlagen in Hypermedia- und Multimedia-Autorensystemen. Die Hypermedia-Vorlagen legen mehr Werte auf die Verweise (engl. hyperlink) zwischen verschiedenen Teilen von Dokumenten, um den Leser zu helfen, innerhalb einer großen Menge von Dokumenten sich zu orientieren. Im Vergleich dazu umfassen Multimedia-Vorlagen nicht nur Verbindungen zwischen Dokumenten, sondern auch die Struktur der Dokumenten selbst.

4.1.2 Modifizierbarkeit vorhandener Vorlagen

Modifizierbarkeit bedeutet, daß der Autor die vom Autorensystem vorgegebenen Vorlagen jeweils nach seinem Bedarf verändern kann.

Die vom Autorensystem vorgegebenen Vorlagen sind normalerweise nur für allgemeine Anwendungsfällen geeignet. Daher muß der Autor Vorlagen in der Regel modifizieren, um sie für seinen konkreten Fall anzupassen.

Wenn die vorhandenen Vorlagen nicht modifizierbar sind, muß der Autor sie unverändert in Dokumenten einsetzen. Das bedeutet, nicht nur Medienelemente in Vorlagen wie Text, Bilder und Audio- und Video-Sequenzen, sondern auch Beziehungen zwischen solchen Medienelementen müssen wie vorgegeben bleiben. Die Nachteile hier sind offensichtlich: Einerseits ist es sehr schwierig, früh zu bestimmen, welche Teile im Dokument später unverändert wiederzuverwenden sind. Andererseits muß der Autor viel mehr kleine Vorlagen anstatt wenige große Vorlagen erstellen, denn je größer und komplizierter eine Vorlage ist, desto schwieriger es ist, sie ohne Modifizierung später wiederzuverwenden.

Alle vorhandenen Vorlagen in den untersuchten Autorensystemen sind modifizierbar. Bei der Verwendung von Vorlagen erzeugen die Autorensysteme automatisch eine Kopie der Vorlage und fügen diese Kopie ins Dokument ein und nicht die originale Vorlage. Nach dem Einsatz wird die Kopie der Vorlage ein Teil des Dokuments und unterscheidet sich nicht von den anderen Elementen des Dokuments, daher kann der Autor die Kopie der Vorlage wie normale Elemente des Dokuments behandeln und nach seinem Bedarf modifizieren. Die Duplikation von Vorlagen bietet einen anderen Vorteil, daß die originale Vorlage durch die Duplikation geschützt wird und unverändert bleibt. Damit kann der Autor immer auf die originale Vorlage zurückgreifen. Alle Vorlagen in untersuchten Autorensysteme im Kapitel 3 sind modifizierbar. Um die Modifizierbarkeit der MAVA-Vorlagen zu erfüllen, wird dieselbe Methode benutzt.

Modifizierung der Vorlagen durch Duplikation ist ebenfalls eine fundamentale Anforderung in Hypermedia-Autorensystemen: wenn der Autor eine vorhandene Hypermedia-Vorlage benutzen möchte, kopiert er zuerst diese Vorlage, und dann bearbeitet er die Kopie anstatt der originalen Vorlage. “When the user duplicate the template, the system should make a copy of everything in it.” [Catl91].

Bisher sind die zwei grundlegenden Anforderungen an Vorlagen genannt. Wie alle andere Multimedia-Autorensysteme muß das MAVA-System diese Anforderungen an Vorlagen erfüllen, um das Vorlagen-Konzept richtig realisieren zu können. Jedoch besitzt das

MAVA-System noch ein paar besondere Eigenschaften, die bei der Konzeption von Vorlagen im MAVA-System berücksichtigt werden sollen.

Folgend werden zusätzliche Anforderungen aufgestellt, die bei der Konzeption der Vorlagen im MAVA-System betrachtet werden müssen.

4.1.3 Hinzufügen benutzerspezifischer Vorlagen

Das Hinzufügen benutzerspezifischer Vorlagen ermöglicht dem Autor, außer der vorhandenen Vorlagen auch eigene Vorlagen zu erstellen.

Die vorgegebenen Vorlagen, die selbst eine geschlossene Menge bilden und festgelegt sind, können nicht von dem Autor geändert werden. Wenn die vorgegebene Vorlagen für neue Anwendungen nicht mehr ausreichend sind, oder für manche Anwendungen nicht geeignet sind, soll der Autor neue Vorlagen selbst erstellen und hinzufügen können. Vorgegebene Vorlagen können nicht die neuen Anforderungen erfüllen, wenn das Hinzufügen neuer Vorlagen nicht möglich ist. Wenn die Vorlagen nicht andauernd aktualisiert werden können, werden sie nicht mehr für neue Anwendungen geeignet sein.

Nicht alle Autorensysteme, die im letzten Kapitel untersucht werden, ermöglichen das Hinzufügen eigener Vorlagen. Beisielsweise besitzt Mediator nur eine feste Menge von vorgegebenen Vorlagen, die nicht ergänzt werden kann. Im Gegensatz dazu unterstützen Authorware und ToolBook benutzerdefinierte Vorlagen, die der Autor nach seinem Bedarf selbst erstellen kann.

Da das MAVA-System Anwendungsunabhängig ist, haben die Autoren in verschiedenen Anwendungsbereichen sehr unterschiedliche Anforderungen an MAVA-Vorlagen. Beispielsweise stellt der Autor von interaktiver Lernanwendungen die Anforderung, daß Vorlagen die interaktive Beziehungen von Medienelementen enthalten sollen. Und für den Autor von Präsentationen ist es wichtig, daß das Layout von Präsentationsseiten in Vorlagen verfügbar ist. Außerdem ist es normal, daß ein Autor seine Anforderungen an Vorlagen stetig ändern kann. Und nicht zuletzt werden immer neue Anwendungsbereiche erscheinen werden, deren Anforderungen man zur Zeit nicht vorsehen kann. Daher ist es nicht ausreichend, eine abgeschlossene Menge von Vorlagen zu stellen. Der Autor vom MAVA-Editor

soll in der Lage sein, stetig neue Vorlagen zu dem Editor hinzuzufügen, um neue Anforderungen erfüllen zu können.

4.1.4 Einfachheit bei der Erstellung

Die Unterstützung für die Einfachheit bei der Erstellung von Vorlagen in den untersuchten Multimedia-Autorensystemen ist ein Dilemma:

Einerseits ist die Erstellung der Vorlagen in manchen Systemen relativ intuitiv und einfach, wie z.B. in Powerpoint und Question Mark Designer. Aber viele nützliche Funktionalität werden für diese Einfachheit geopfert, da keine Skriptsprache eingesetzt wird, kann der Autor keine mächtige Skript-Funktionen in den Vorlagen einbauen.

Andererseits ist die Erstellung von Vorlagen zu kompliziert in den Autorensystemen wie Authorware und ToolBook. Die Vorlagen in solchen Systemen sind zwar sehr flexibel und mächtig mit vielen nützlichen Funktionalität, aber sehr schwierig und aufwendig für den Autor, da der Autor sehr gute Programmierkenntnisse von Skript-Sprachen wie Lingo oder OpenScript besetzen muß.

Eines der wichtigsten Ziele des MAVA-Systems ist, die Erstellung von Multimedia-Dokumenten so weit wie möglich zu vereinfachen. Die meiste Arbeit soll in der Form von der sogenannten “Visueller Programmierung” erledigt werden können. Die Visueller Programmierung ist die Manipulation von visuellen Informationen und Interaktionen, um die Programmierung in visueller Präsentationen durchzuführen [Goli90].

Anstatt das Programm von Tastatur einzutippen benutzt der Autor hauptsächlich die Maus bei der Erstellung von Dokumenten, wie z.B. Drag&Drop oder Klicken mit der Maus. Daher ist die Benutzung von Vorlagen für die Autoren intuitiver und einfacher, und der Autor vom MAVA-Editor braucht keine Programmierkenntnisse zu haben, um Vorlagen im MAVA-Editor zu erstellen.

Aber für Vorlagen im MAVA-System sollen Flexibilität und Mächtigkeit nicht zu Gunsten die Einfachheit geopfert werden. Es ist wünscherswert, daß die Vorlagen im MAVA-System nicht nur einfach, sondern auch sehr leistungsfähig sind.

4.1.5 Modularität und Struktur bei der Speicherung

Die Modularität von Vorlagen ist ein wichtiges Thema, was bedeutet, daß das Speichern und Laden von einzelnen Vorlagen unabhängig von anderen Vorlagen ist. Jede Vorlage soll ein separates Modul sein. Außerdem soll die Struktur von Vorlagen bei der Speicherung erkennbar sind. Zum Beispiel, Vorlagen mit ähnlichen Eigenschaften sollen in gleichem Verzeichnis oder Unterverzeichnis gespeichert werden. Damit kann der Autor die Vorlagen leicht nach dem Datei- und Verzeichnis-Name finden.

Die untersuchten Multimedia-Autorensysteme im letzten Kapitel haben unterschiedliche Vorgehensweisen beim Speichern und Laden von Vorlagen: Manche Autorensysteme, beispielsweise Powerpoint, speichern jede Vorlage als eine einzelne Datei in einem bestimmten Verzeichnis. Damit kann der Autor leicht die einzelnen Vorlagen finden, bearbeiten und mit anderen Autoren austauschen. Um die Struktur von Vorlagen zu verdeutlichen, speichern manche Autorensysteme, wie z.B. Authorware und ToolBook, die Vorlagen je nach Kategorie in verschiedenen Unterverzeichnisse, was bei der Strukturierung von vielzähligen Vorlagen von großem Vorteil ist.

Manche Systeme, wie beispielsweise Question Mark Designer, speichern alle Vorlagen, egal ob es sich dabei vorgegebene oder benutzerdefinierte handelt, in einer einzigen Datei. Das bringt einen großen Nachteil bei der Verteilung der einzelnen Vorlagen. Zum Beispiel, wenn der Autor A eine einzelne Vorlage von dem Autor B braucht, muß A die ganze Vorlagen-Datei von B kopieren. Dabei bekommt A zwar die benötigte Vorlage, aber gleichzeitig auch alle andere Vorlagen, die für A uninteressante sind. Außerdem wird die originale Vorlagen-Datei vom A durch die Vorlagen-Datei vom B übergeschrieben, daher gehen alle eigene Vorlagen von A verloren, da beide Dateien gleichen Name haben. Auch die Struktur von Vorlagen kann hier nicht dargestellt werden. Da alle Vorlagen in einer Datei gespeichert sind, spielt die Name von Datei oder Verzeichnis hier auch keine Rolle mehr. Daher ist es nicht möglich, die Vorlagen mit ähnlichen Eigenschaften in eigenem Unterverzeichnis zu speichern.

Die einzelnen Vorlagen im MAVA-System sollen in jeweils getrennten Dateien, und wenn nötig, auch in verschiedenen Unterverzeichnissen gespeichert werden können, um die Modularität von Vorlagen zu erhalten. Durch die Verteilung der Vorlagen auf verschiede-

nen Verzeichnisse kann der Autor leicht die gesuchten Vorlagen-Dateien erkennen, da Vorlagen mit ähnlichen Eigenschaften normalerweise in gleichem Verzeichnis gespeichert werden.

Außerdem ist die Modularität der Grundlage für die einfache Verteilung der Vorlagen. Da Vorlagen in einzelnen Dateien gespeichert werden, kann der Autor die einzelne Vorlage als Ergänzung zum MAVA-Editor zum Beispiel auf eine Webseite bereitstellen. Und die Autoren können die benötigte Vorlage ausfindig machen und herunterladen, was weniger Bandbereite und Zeit benötigt als eine einzige große Vorlagen-Datei herunterzuladen.

4.1.6 Sichtbarkeit und Erhaltung struktureller Informationen

Die Sichtbarkeit und Erhaltung struktureller Informationen ermöglichen dem Autor, die innere Struktur von Dokumenten direkt auf dem Bildschirm zu sehen, und diese Struktur auch nach der Änderungen von betroffenen Medienelementen zu erhalten.

Das ist eine spezielle Eigenschaft vom MAVA-System, da die Medien- und Struktur-Informationen im MAVA-System deutlich getrennt werden sind. Kein Autorensystem, das im letzten Kapitel untersucht wurde, hat diese Eigenschaft, daher kann auch kein andere Autorensystem außer MAVA diese Anforderung erfüllen.

Zum Beispiel erstellt der Autor eine Präsentationsseite in Powerpoint, auf der sich ein Bild und eine Textzeile befinden. Wenn das Bild zentriert überhalb einem Text dargestellt werden soll, kann diese einfache Beziehung in Powerpoint nicht angezeigt werden. Der Autor kann zwar die relative Position von Bild und Text festlegen, aber diese Beziehung ist mit den betroffenen Medienelementen eng verbunden und kann nicht separat angezeigt werden. Noch wichtiger ist, wenn der Autor eine Vorlage von dieser Seite erstellen und später wiederverwenden möchte, geht diese Beziehung verloren, sobald ein Medienelement davon geändert oder ausgetauscht wird.

Das bedeutet, daß die strukturelle Informationen in Powerpoint weder separat sichtbar gemacht noch getrennt erhalten werden. Das gleiche Problem existiert auch in anderen Autorensystemen wie Authorware und ToolBook. Da die strukturellen Informationen bei diesen Autorensystemen auch mit den Medienelementen eng verbunden sind, kann diese Information nicht von Medienelementen getrennt erhalten oder angezeigt werden.

Da im MAVA-Editor alle strukturelle Informationen in Operatoren gespeichert und als Operator-Icons gezeigt werden, sind alle strukturellen Informationen für den Autor direkt sichtbar. Das bringt den Vorteil, daß der Autor die Struktur von Dokument durch das Operator-Icon sehr leicht erkennen kann. Außerdem bleiben die strukturelle Informationen erhalten, auch wenn die entsprechende Medienelemente geändert oder ausgetauscht werden. Da alle strukturelle Informationen in Operatoren gespeichert sind, sind die Information von Medienelementen getrennt gespeichert. Damit ist die strukturelle Information in MAVA-Vorlagen sichtbar und kann leicht erhalten bleiben.

4.2 Konzeption der Vorlagen im MAVA-System

Nach der Analyse der obengenannten Anforderungen werden folgende Eigenschaften der Vorlagen im MAVA-System festgelegt:

- **Persistenz der Vorlagen**

  Die Vorlagen im MAVA-System sollen in einzelnen Dateien gespeichert werden können. Damit geht die Vorlagen nicht verloren, auch wenn das Autorenprogramm beendet wird.

- **Modifizierbarkeit vorhandener Vorlagen**

  Bei der Verwendung einer Vorlage soll eine Kopie der originalen Vorlage erzeugt und ins Dokument eingefügt werden, damit die originale Vorlage unverändert bleibt. Außerdem soll der Autor diese Kopie wie andere normale Dokumentelemente modifizieren können.

- **Hinzufügen benutzerspezifischer Vorlagen**

  Die Vorlagen sollen nicht festgelegt sein und selbst eine geschlossene Menge bilden. Der Autor soll in der Lage sein, nach seinem Bedarf neue Vorlagen zu erstellen und den gegebenen Vorlagen hinzuzufügen.

- **Einfachheit bei der Erstellung**

  Trotz der Flexibilität und Mächtigkeit soll die Erstellung von Vorlagen für den Autor so einfach und intuitiv wie möglich bleiben. Zum Beispiel soll die Bearbeitung von Vorlagen in der Form von visueller Programmierung erledigt werden, und daher keine Programmierkenntnisse erforderlich sein.

- **Modularität und Struktur bei der Speicherung**

  Jede einzelne Vorlage soll in einer eigenen Datei, oder wenn nötig auch in einem eigenen Verzeichnis gespeichert werden, um die Modularität und Struktur der Vorlagen zu erhalten und damit die Verwaltung und Verteilung der Vorlagen zu vereinfachen.

- **Sichtbarkeit und Erhaltung struktureller Informationen**

  Die strukturelle Information zwischen Medienelementen soll möglichst deutlich sichtbar sein. Außerdem soll diese Information erhalten bleiben, auch wenn die entsprechenden Medienelemente geändert oder ausgetauscht werden.

Alle obengenannte Anforderungen sind im MAVA-System schon erfüllt. Und kein andere Autorensystem, wie im letzten Kapitel untersucht, kann allein alle Anforderungen erfüllen. Um das Vergleich von MAVA-System und anderen Autorensystemen zu verdeutlichen, werden folgend eine übersichtliche und präzise Bewertung gegeben.

4.3 Vergleich und Bewertung

In diesem Kapitel wurden die Anforderungen an die Vorlagen im MAVA-System anhand der Untersuchungsergebnisse aus dem Kapitel 3 festgelegt.

Anforderung an Vorlagen	**Mediator 5**	**Question Mark Designer**	**Power-point 2000**	**Author-ware 5.2**	**Tool-Book II**	**MAVA**
Persistenz der Vorlagen	Ja	Ja	Ja	Ja	Ja	Ja
Modifizierbarkeit vorhandener Vorlagen	Ja	Ja	Ja	Ja	Ja	Ja
Hinzufügen benutzerspezifischer Vorlagen	Nein	Ja	Ja	Ja	Ja	Ja
Einfachheit bei der Erstellung	Ja	Ja	Ja	Nein	Nein	Ja
Modularität und Struktur bei der Speicherung	Nein	Nein	Ja	Ja	Ja	Ja
Sichtbarkeit und Erhaltung struktureller Informationen	Nein	Nein	Nein	Nein	Nein	Ja

Tabelle 4-1. Das Vergleich der untersuchten Autorensystemen nach der Anforderungen

Tabelle 4-1 gibt eine Übersicht für das Ergebnis des Vergleichs. Aus dem Ergebnis, das in der Tabelle 4-1 gezeigt ist, kann man die Vorteile von MAVA-Vorlagen gegen allen anderen Vorlagen der untersuchten Autorensysteme leicht sehen. Der MAVA-Editor ist das ein-

zige Autorensystem, das alle in der Tabelle aufgelisteten Anforderungen an Vorlagen erfüllen kann. Alle anderen Autorensysteme können zwar die allgemeine Anforderungen erfüllen, haben aber jeweils eigene Schwachstellen:

- Die Vorlagen in **Mediator** sind festgelegt und selbstgeschlossen. Es ist nicht möglich für den Autor, neue Vorlagen nach eigenem Bedarf zu erstellen und hinzuzufügen. Daher ist die Nutzung von Vorlagen in Mediator sehr beschränkt und unflexibel.
- Die Vorlagen in **Question Mark Designer** sind nicht festgelegt. Aber die Speicherung aller Vorlagen in einer einzigen Datei ist sehr problematisch: Mit allen Vorlagen in einer Datei können Autoren nicht einzelne Vorlagen untereinander austauschen. Die Verwaltung und Verteilung von einzelnen Vorlagen ist dabei sehr unübersichtlich.
- Die Erstellung von Vorlagen in Powerpoint ist relativ einfach. Der Autor kann eigene Vorlagen erstellen und hinzufügen. Die Speicherung von Vorlagen ist modular. Aber Powerpoint ist kein generisches Multimedia-Autorensystem, die meisten Vorlagen in Powerpoint sind nur für das Layout der Präsentationsseiten gedacht. Daher sind die Flexibilität und der Einsatzbereich von Vorlagen in Powerpoint sehr beschränkt.
- Die Vorlagen in Authorware und ToolBook sind nicht festgelegt. Der Autor kann beliebige neue Vorlagen erstellen und hinzufügen. Alle Vorlagen können auch modular gespeichert werden, damit die Struktur der Vorlagen sichtbar ist. Aber die Erstellung von Vorlagen in diesen zwei Autorensystemen ist zu schwierig für normalen Autoren, da hier gute Programmierkenntnisse von Skriptsprachen (Lingo bzw. OpenScript) erforderlich sind.

Außerdem sind die strukturelle Informationen in Vorlagen von allen obengenannten Autorensystemen weder sichtbar noch erhaltbar. Die strukturelle Beziehungen sind mit den betroffenen Medienelementen eng verbunden. Daher können solche Informationen weder separat gezeigt noch von Medienelementen getrennt gespeichert werden.

Im Gegensatz dazu ist MAVA das einzige Autorensystem, dessen Vorlagen alle obengenannten Anforderungen erfüllen kann. Die Vorlagen im MAVA sind offen und der Autor kann selbst neue Vorlagen erstellen und hinzufügen. Außerdem können die Vorlagen im MAVA-System modular gespeichert werden. Die Erstellung von Vorlagen im MAVA ist

einfach und intuitiv, und erfordert keine Programmierkenntnisse von Autoren. Und die strukturelle Informationen in Vorlagen sind direkt sichtbar, und bleiben getrennt von Medienelementen erhalten.

4.4 Zusammenfassung

In diesem Kapitel wurden die Anforderungen an Vorlagen in Autorensystemen festgelegt. Danach wurde die Konzeption von Vorlagen in MAVA angegeben. Am Ende wurden die Vorlagen in MAVA und allen anderen untersuchten Autorensystemen nach den gegebenen Anforderungen analysiert und verglichen, eine Bewertung wurde ebenfalls gegeben. In diesem Kapitel wurden die Vorlagen in MAVA konzipiert.

5 Anpassung des MAVA-Editors zur Unterstützung von Vorlagen

Im letzten Kapitel wurden die Anforderungen an die Vorlagen in MAVA-System analysiert und eine Konzeption der MAVA-Vorlagen angegeben. Um Vorlagen zu unterstützen, ist eine Anpassung des vorhandenen MAVA-Editors notwendig. In diesem Kapitel wird zuerst der vorhandene MAVA-Editor vorgestellt. Danach wird diskutiert, wie die Benutzungsoberfläche und das Speicherformat im MAVA-Editor angepaßt werden sollen. Eine kurze Zusammenfassung wird am Ende dieses Kapitel gegeben.

5.1 Der vorhandene MAVA-Editor

In Kapitel 2 wurde der MAVA-Editor (kurz MED) schon kurz vorgestellt. Da die Anpassung des MAVA-Editors nur die Benutzungsoberfläche und das Speicherformat betrifft, werden sie hier ausführlicher betrachtet. Später in diesem Kapitel werden die notwendige Anpassung ausführlich erläutert.

5.1.1 Die Benutzungsoberfläche

Der Autor interagiert mit dem MAVA-Editor allein und einzig durch eine Schnittstelle: die Benutzungsoberfläche vom MAVA-Editor. Daher muß die komplette Funktionalität des MAVA-Editors in der Benutzungsoberfläche dargestellt werden.

Die wichtigsten Komponenten der Benutzungsoberfläche sind Menüs, Dialoge und die Symbolleiste. In der Abbildung 5-1 sieht man das Hauptfenster des MAVA-Editors: Ganz oben in dem Fenster befindet sich die Leiste von Menüs, hier gibt es z.B. ein File-Menü, ein Edit-Menü usw. Jedes Menü besitzt mehrere Menüeinträge, die mit einer Funktion verbunden sind. Wenn der Autor einen Menüeintrag auswählt, wird die damit verbundene Funktion ausgeführt. Unter der Menüleiste liegt die Symbolleiste mit mehreren Schaltflächen (engl. butttons). Jede Schaltfläche auf der Symbolleiste ist mit einer Funktion verbunden. Klickt der Autor auf eine Schaltfläche, wird die entsprechende Funktion aufgerufen.

Da alle mit einer Schaltfläche verbundenen Funktionen auch durch Menüs aufrufbar sind, dient die Schaltfläche auf der Symbolleiste nur als Abkürzung für häufig benutzte Funktionen. Damit braucht der Autor nicht jedesmal nach dem Menüeintrag zu suchen, sondern er kann direkt auf die entsprechende Schaltfläche der Funktion in der Symbolleiste klicken.

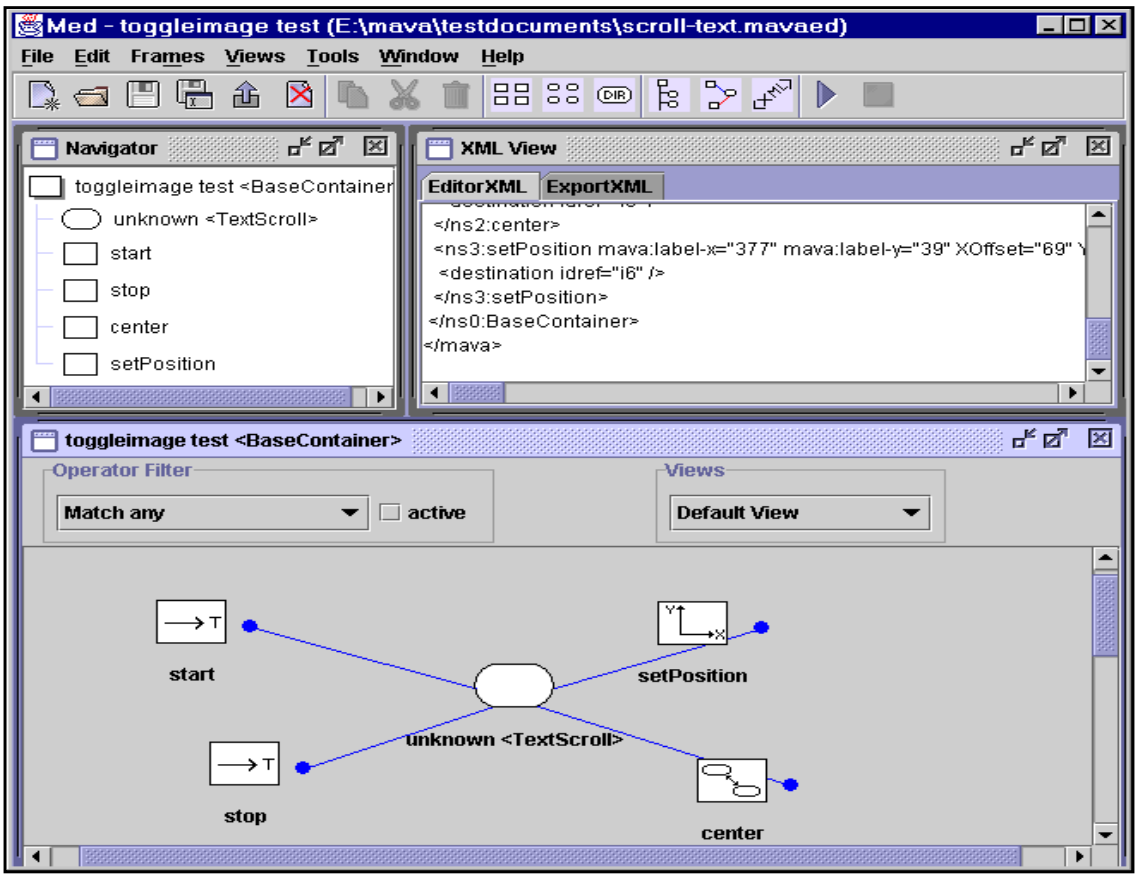


Abbildung 5-1. Das Hauptfenster vom vorhandenen MAVA-Editor

Folgend werden die Komponenten vorgestellt, die entweder relevant für Vorlagen sind oder für die Verwendung von Vorlagen angepaßt werden müssen.

- **Das Edit-Menü**

Abbildung 5-2 zeigt das Edit-Menü des MAVA-Editors. Alle Funktionen, mit den der Autor das Dokument bearbeitet, sind mit den Menüeinträgen vom Edit-Menü verbunden. Hier sind die Funktionen wie Kopieren, Ausschneiden und Löschen von Dokumentelementen jeweils mit dem Menüeintrag “Copy”, “Cut” und “Delete” verbunden.

Mit dem Edit-Menü kann der Autor nicht nur das Dokument bearbeiten, sondern auch die Eigenschaften eines Dokuments prüfen und verändern. Die Eigenschaften vom Dokument sind hier z.B. der Name vom Dokument und die Größe vom Präsentationsfenster des Dokuments, usw. Wählt der Autor den Menüeintrag “Document Properties” des Edit-Menüs aus, erscheint ein Dialogfenster auf dem Bildschirm, worin der Autor diese Eigenschaften einstellen kann.

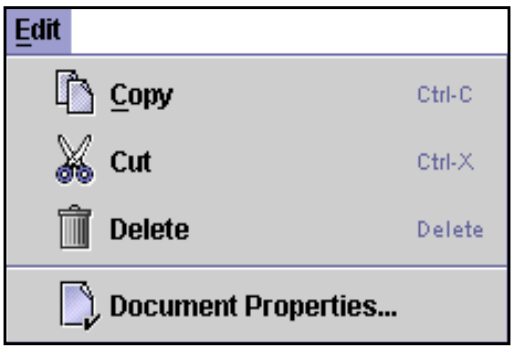


Abbildung 5-2. Das Edit-Menü von MAVA-Editor (Vor der Anpassung)

- **Das Frame-Menü**

Das Frame-Menü wird benutzt, um verschiedene Fenster (engl. frame) im MAVA-Editor zu öffnen.Wenn der Autor einen Menüeintrag auswählt, wird ein entsprechendes Fenster geöffnet, z.B. das Fenster für verfügbare Operatoren und verfügbare Media-Elementen Typen. Abbildung 5-3 zeigt das Frame-Menü von MAVA-Editor.

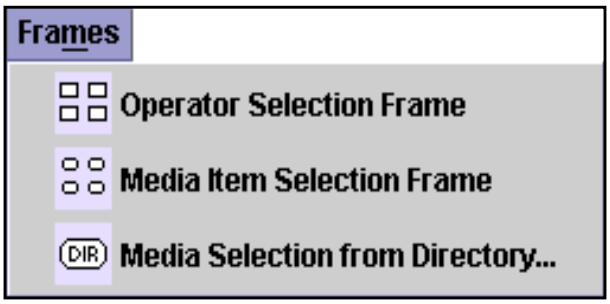


Abbildung 5-3. Das Frame-Menü von MAVA-Editor

Wenn der Autor den Menüeintrag “Operator Selection Frame” oder der Menüeintrag “Media Item Selection Frame” auswählt, erscheint ein Fenster auf dem Bildschirm, worin alle verfügbare Operatoren und vorhandene Typen von Media-Elementen als Symbole dar-

gestellt sind. Der Autor kann diese Operatoren oder Media-Typen ins Dokument einfügen, indem er die entsprechenden Symbole selektieren und ins Symbol-Ansicht Fenster vom Dokument mit der Maus per Drag&Drop einziehen.

- **Das Fenster für die Auswahl der Medien-Dateien**

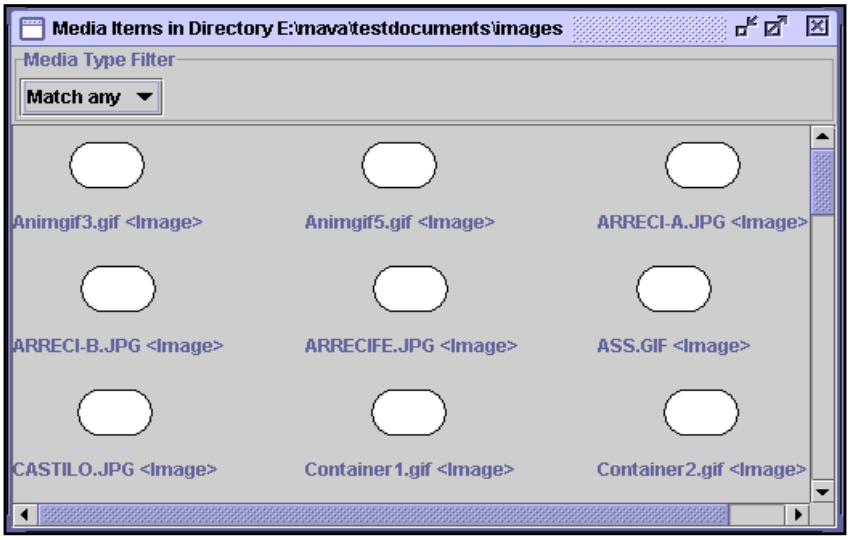


Abbildung 5-4. Das Fenster für die Auswahl der Medien-Dateien in einem Verzeichnis

Wählt der Autor den Eintrag “Media Selection from Directory” im Frame-Menü, kommt ein neues Dialogfenster auf dem Bildschirm, worin der Autor ein Verzeichnis auswählen kann. Danach werden alle Medien-Dateien in dem ausgewählten Verzeichnis, wie z.B. Bilder, Audio- und Video-Sequenzen, als Symbole in einem separaten Fenster gezeigt. Abbildung 5-4 zeigt das “Media Selection From Directory” Fenster, wo alle Medien-Dateien in einem Verzeichnis als Symbole dargestellt sind. Danach kann der Autor die Medien-Dateien, die in diesem Fenster gezeigt sind, leicht ins Dokument einfügen, indem er die Symbole von Medien-Dateien per Drag&Drop direkt ins Symbol-Ansicht Fenster (engl. IconView) vom Dokument einzieht.

- **Die Kontextmenüs**

Um die Dokumente zu bearbeiten, braucht der Autor nicht jedesmal das Edit-Menü zu öffnen und nach dem richtigen Menüeintrag im Edit-Menü zu suchen. Mit der rechten Maustaste kann der Autor jederzeit ein Menü auf dem Bildschirm bringen, gerade wo die Maus sich befindet. Da diese Menüs nicht an bestimmter Stelle liegen, sondern immer an

der Position erscheinen, wo sich die Maus gerade befindet, werden sie Kontextmenüs genannt.

Die Kontextmenüs sind leicht und intuitiv zu bedienen: wenn der Autor die Elemente vom Dokument bearbeiten möchte, wie z.B. kopieren, einfügen oder löschen, muß der Autor nicht in das Edit-Menü gehen und nach Menüeinträgen zu suchen. Im Gegenteil kann er mit der Maus die zu bearbeitende Dokumentelemente selektieren und dann die rechte Maustaste klicken, um das Kontextmenü an der Stelle aufzurufen, wo sich die zu bearbeitenden Dokumentelemente befinden.

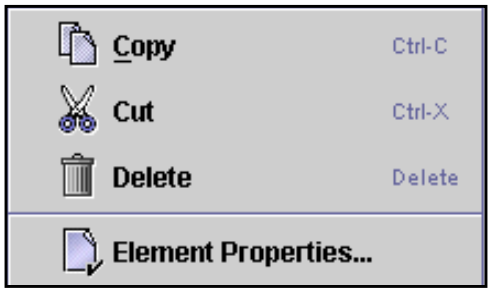


Abbildung 5-5. Das Kontextmenü vom Medienelement im Symbol-Ansicht Fenster (Vor der Anpassung)

Wenn der Autor mit der rechten Maustaste das Symbol eines Dokumentelements in der Symbol-Ansicht klickt, erscheint das Kontextmenü wie in der Abbildung 5-5 gezeigt. Mit diesem Menü kann der Autor die selektierten Elemente kopieren, ausschneiden oder löschen.

Jedes Element eines Dokuments besitzt Eigenschaften, die auch Attribute genannt sind. Zum Beispiel besitzt der zeitliche Operator “start” den Anfangszeitpunkt als Attribut, und der räumliche Operator “setPosition” die absolute Position als Attribut. Für die Media-Elemente wie Bilder, Audio- und Video-Sequenzen steht die Eigenschaften für den Datei-Name, die Datei-URL, usw. Um die Eigenschaften der Dokumentelementen zu prüfen und die Argumente der Eigenschaft zu verändern, kann der Autor den Menüeintrag “Element Properties” vom Kontextmenü aufrufen.

Wenn der Autor mit der rechten Maustaste nicht auf ein Symbol, sondern direkt auf die Grundfläche im Symbol-Ansicht Fenster klickt, erscheint das Kontextmenü wie in der Abbildung 5-6 gezeigt. Dieses Menü besitzt zusätzliche Menüeinträge: Mit dem Eintrag

“Select All” kann der Autor alle Dokumentelemente im aktuellen Fenster selektieren, und mit Menüeintrag “Paste” können die Dokumentelemente, die bereits kopiert wurden, ins Symbol-Ansicht Fenster eingefügt werden. Wählt der Autor den Menüeintrag “Close Icon-View Frame”, wird das aktuelle Symbol-Ansicht Fenster geschlossen.

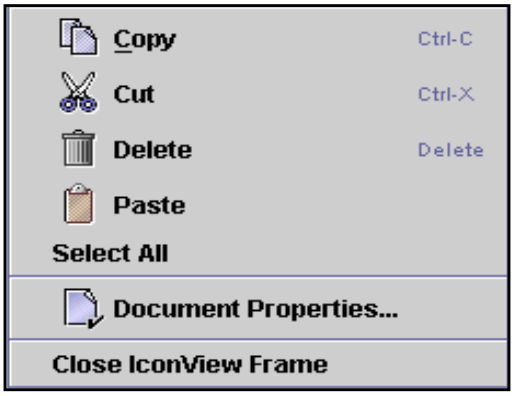


Abbildung 5-6. Das Kontextmenü vom Symbol-Ansicht Fenster (Vor der Anpassung)

Die Kontextmenüs werden nicht nur im Symbol-Ansicht Fenster, sondern auch in das Baum-Ansicht-Fenster (engl. TreeView) integriert. Die Baum-Ansicht dient dazu, die Struktur von Dokumenten in der Form von einem Baum zu verdeutlichen. Alle Container-Elemente, die noch andere Elemente behalten, werden als Knoten dargestellt. Und alle Elemente, die kein Container-Objekt sind, werden als Blätter vom Baum dargestellt. In der Abbildung 5-1 wurde ein Symbol-Ansicht Fenster mit dem Titel “Navigator” gezeigt.

Wenn der Autor mit der rechten Maustaste auf die Blätter vom Baum klickt, erscheint ein Kontextmenü, das gleich wie bei den Symbole in der Symbol-Ansicht und in der Abbildung 5-5 gezeigt. Damit der Autor das selektierte Dokumentelement bearbeiten und die Eigenschaft davon prüfen und verändern kann.

Klickt der Autor auf einen Knoten in der Baum-Ansicht, der ein Container-Element des Dokuments darstellt, wird das Kontextmenü wie in der Abbildung 5-7 gezeigt, auf dem Bildschirm erscheinen. Das Kontextmenü vom Knoten bietet alle Funktionalität, die im Kontextmenü der Blätter im Baum vorhanden sind. Außerdem kann der Autor mit diesem Kontextmenü den Typ von dem selektierten Container-Objekt verändern. Wenn der Autor den Menüeintrag “Change Container Type” vom Kontextmenü auswählt, wird ein Dialog-fenster auf dem Bildschirm erscheinen, worin der Autor den Container-Typ ändern kann.

Zum Beispiel ist diese Funktionalität wichtig, wenn der Knoten vom Typ BaseContainer ist, und während der Arbeit ändert der Autor seine Meinung und möchte statt BaseContainer ein ButtonContainer-Typ haben. Mit der Hilfe vom Kontextmenü braucht der Autor nicht ein neues ButtonContainer-Elements zu erstellen, sondern nur den Container-Typ dieses Container-Elements vom BaseContainer ins ButtonConainer zu wechseln. Damit bleiben alle Elemente, die in diesem Container-Element enthalten sind, unverändert. Der Autor muß diese Elemente nicht vom Anfang an nochmal erstellen und kann deshalb viel Zeit sparen.

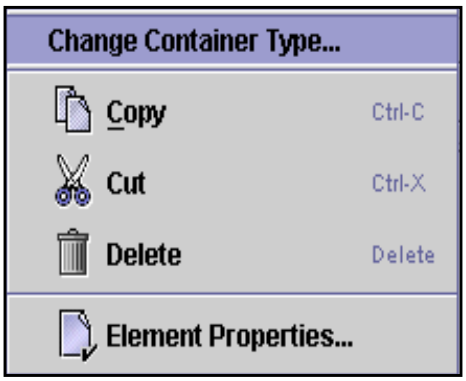


Abbildung 5-7. Das Kontextmenü vom Knoten in der Baum-Ansicht (vor der Anpassung)

- Die Symbolleiste

In der Abbildung 5-8 wird die Symbolleiste des MAVA-Editors gezeigt. Die Symbolleiste dient dazu, die Bearbeitung mit dem MAVA-Editor zu vereinfachen und beschleunigen. Jede Schaltfläche der Symbolleiste ist mit einer Funktion verbunden, um das Dokument zu bearbeiten.

Wenn der Autor mit der Maus auf eine Schaltfläche klickt, wird die damit verbundene Funktion aufgerufen. Die Funktionen in der folgenden Symbolleiste sind (von links nach rechts): neues MAVA-Dokument erstellen, vorhandenes MAVA-Dokument öffnen, MAVA-Dokument speichern, MAVA-Dokument unter anderen Name speichern, MAVA-Dokument nach MAVA-Meiden Dokument exportieren, aktuelles MAVA-Dokument schließen, Dokumentelemente kopieren, Dokumentelemente ausschneiden, Dokumentelemente löschen, Operatoren-Fenster öffnen, Medienelement-Fenster öffnen, Medien-Dateien Fenster öffnen, Baum-Ansicht Fenster öffnen, Symbol-Ansicht Fenster öffnen,

XML-Ansicht Fenster öffnen, aktuelles Dokument in MAVA-Präsentationssystem testen und das MAVA-Präsentationssystem zu stoppen.

Alle obengenannten Funktionen kann der Autor auch mit Menüs aufrufen, aber mit den Schaltflächen auf der Symbolleiste geht es direkter und schneller, da der Autor nicht nach dem Menüeintrag in Menüs suchen muß. Das ist besonders einfach und schnell für die Funktionen, die sehr oft aufgerufen werden, wie z.B. das Speichern und Öffnen von Dokumenten.

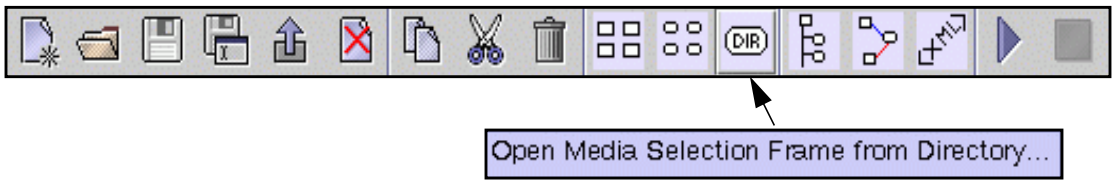


Abbildung 5-8. Die Symbolleiste des MAVA-Editors

5.1.2 Das Speicherformat

Um das Dokument in einer Datei zu speichern, selektiert der Autor den Menüeintrag “Save” oder “Save As...” im File-Menü. Danach erscheint ein Dialogfenster, in dem der Speicherort und der Name der Dateien angegeben werden können.

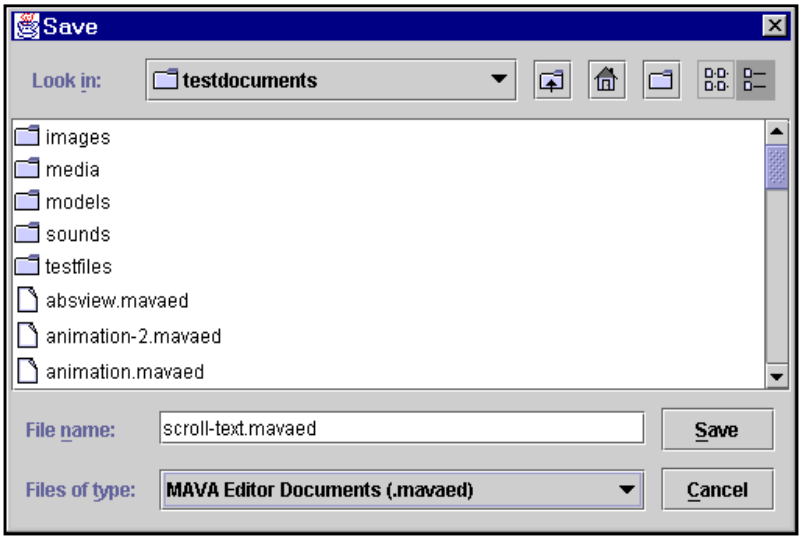


Abbildung 5-9. Das Save-Dialog von MAVA-Editor

Abbildung 5-9 zeigt das “Save” Dialogfenster. Im Dialog kann der Autor das Verzeichnis angeben, wohin die Dokumentdatei gespeichert werden soll. Ebenfalls im Dialog legt der Autor den Name und Datei-Typ von Dokumentdatei fest. Hier werden alle MAVA-Dokumente in die Dateien mit der Erweiterung “mavaed” gespeichert.

5.2 Die Anpassung des vorhandenen MAVA-Editors

Um Vorlagen zu unterstützen, müssen die obengenannte Komponenten der Benutzungsoberfläche im vorhandenen MAVA-Editor angepaßt werden. Die anzupassende Komponente der Benutzungsoberfläche sind Menüs, Kontextmenüs, Symbolleiste und Dialogfenster. Sie werden im folgenden einzeln betrachtet.

5.2.1 Edit-Menü

Wie bereits beschrieben, ist das Edit-Menü für die Bearbeitung der MAVA-Dokumenten zuständig. Nachdem das Vorlage-Konzept in dem MAVA-System eingeführt ist, gibt es eine neue Funktion bei der Bearbeitung der Dokumenten: Der Autor soll den selektierten Teil eines Dokuments als eine Vorlage speichern können.

Um diese neue Funktion zu benutzen, soll das Edit-Menü durch einen neuen Menüeintrag erweitert werden. Dazu wird ein neuer Menüeintrag “Save As Template” in das Edit-Menü integriert. Abbildung 5-10 zeigt das Edit-Menü nach der beschriebener Erweiterung.

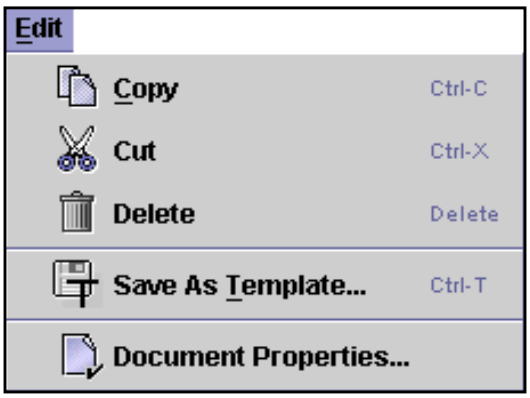


Abbildung 5-10. Das Edit-Menü von MAVA-Editor (Nach der Anpassung)

Wenn der Autor den Menüeintrag “Save As Template” in dem Edit-Menü auswählt, erscheint ein Dialogfenster auf dem Bildschirm, worin der Autor festlegen kann, wo und

unter welchen Namen eine Vorlage gespeichert wird. In Abbildung 5-11 wird dieses Dialogfenster gezeigt. Oben im Dialog kann der Autor das Verzeichnis feststellen, wohin die Vorlage-Datei gespeichert werden soll. Unter im Dialog kann der Autor den Name von Vorlage-Datei angeben. Alle Vorlagen-Dateien werden mit der Erweiterung “mavatmp” gespeichert. Damit kann der Autor die Vorlagen-Dateien von normalen MAVA-Dokumentdateien leicht anhand der Datei-Erweiterung unterscheiden.

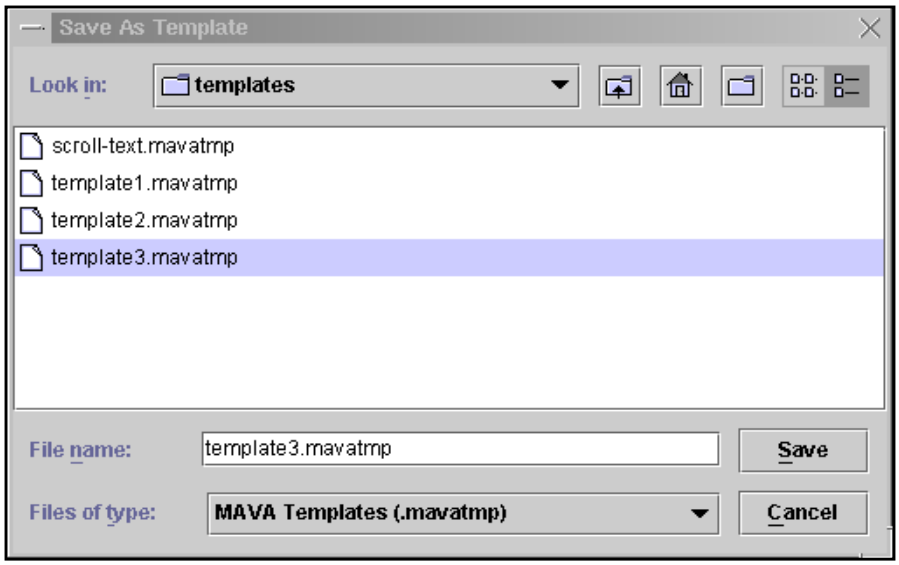


Abbildung 5-11. Das “Save As Template” Dialogfenster von MAVA-Editor

5.2.2 Frame-Menü

In der Abbildung 5-3 wurde das Frame-Menü des MAVA-Editors bereits gezeigt. Nach der Einführung des Vorlagen-Konzepts in das MAVA-System, wird das Aussehen des Frame-Menüs nicht geändert. Jedoch wird die mit dem Eintrag “Media Selection From Directory” verbundene Funktionalität erweitert.

Wählt der Autor jetzt den Eintrag “Media Selection From Directory” aus, kommt ein Dialogfenster auf dem Bildschirm, worin der Autor das Verzeichnis festlegen kann. Der Unterschied liegt darin, daß in dem Verzeichnis außer Medien-Dateien auch Vorlagen-Dateien sich befinden können. Und die Vorlagen-Dateien werden gleich wie Medien-Dateien als Symbole auf der Grundfläche in einem Fenster dargestellt. Der Autor kann die Vorlagen in das Dokument einfügen, indem er die Vorlage-Symbole einfach in das Symbol-Ansicht Fenster vom Dokument per Drag&Drop einzieht.

5.2.3 Anpassung des Fenster für die Auswahl der Medien-Dateien

Abbildung 5-12 zeigt das Fenster für die Auswahl der Medien-Dateien nach der Anpassung, wo Vorlagen-Dateien in einem Verzeichnis auch als Symbole dargestellt werden können. Und das Einfügen einer Vorlage ins Dokument ist identisch wie bei einer Medien-Datei: Drag&Drop per Maus. Damit hat es sich nichts geändert, wenn der Autor eine Vorlage statt einer Medien-Datei ins Dokument einfügt, deshalb ist die Benutzung der Vorlage für den Autor intuitiv und einfach.

Im Kapitel 4 wurde die Modularität der Speicherung von Vorlagen als eine Anforderung festgestellt, was bedeutet, daß die Vorlagen nach der Kategorien in verschiedenen Unterverzeichnissen gespeichert werden können. Deshalb wird das Fenster in Abbildung 5-12 erweitert, damit der Autor zwischen dem aktuellen Verzeichnis und alle seiner direkten Unterverzeichnisse innerhalb Fenster umschalten kann. Wie in Abbildung 5-12 zeigt, besitzt das momentan gewählte Verzeichnis vier direkte Unterverzeichnisse: CBT, Medienelements und Presentation. Das bedeutet, momentan sind die Vorlagen-Dateien in diesen vier Kategorien gespeichert. Und diese Verzeichnisse werden alle in der Combo-Box rechts oben im Fenster dargestellt. Wählt der Autor ein Verzeichnis in diesem Combo-Box, werden alle Medien-Dateien oder Vorlagen in dieser Kategorie im Fenster gezeigt.

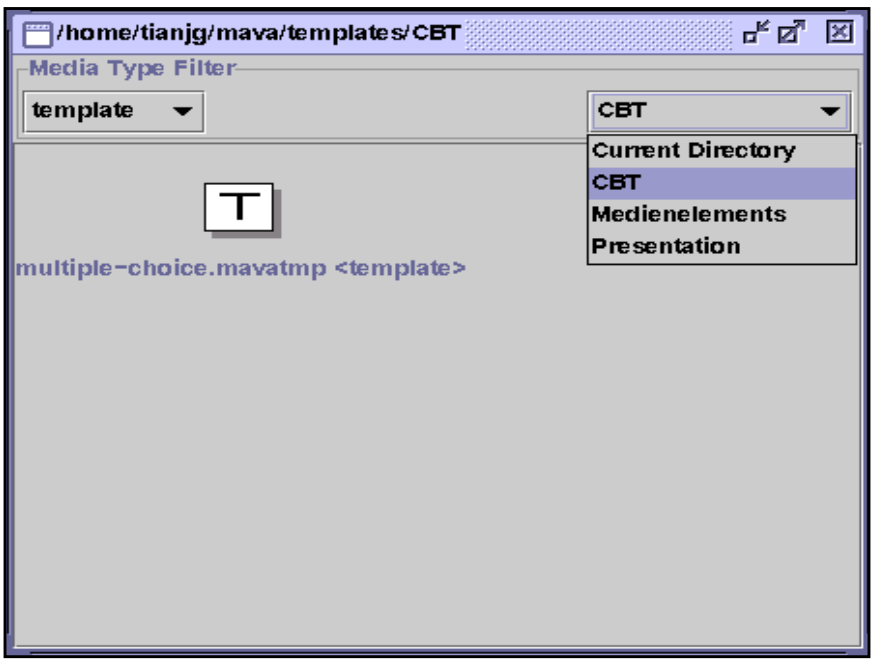


Abbildung 5-12. Das Fenster für die Auswahl der Medien-Dateien (nach der Anpassung)

5.2.4 Kontextmenüs

Bei der Anpassung zur Unterstützung von Vorlagen werden die Kontextmenüs sowohl im Symbol-Ansicht Fenster als auch Baum-Ansicht Fenster durch den Menüeintrag “Save As Template” erweitert. Damit kann der Autor die selektierten Elemente im aktuellen Fenster mit Hilfe vom Kontextmenü sofort in Vorlagen-Dateien speichern. Abbildung 5-13 zeigt die Kontextmenüs im Symbol-Ansicht Fenster nach der Erweiterung. Wenn der Autor mit der rechten Maustaste auf das Dokumentelement im Symbol-Ansicht Fenster klickt, kommt das rechte Kontextmenü auf dem Bildschirm. Klickt der Autor direkt auf das Symbole-Fenster, wird das linke Kontextmenü gezeigt.

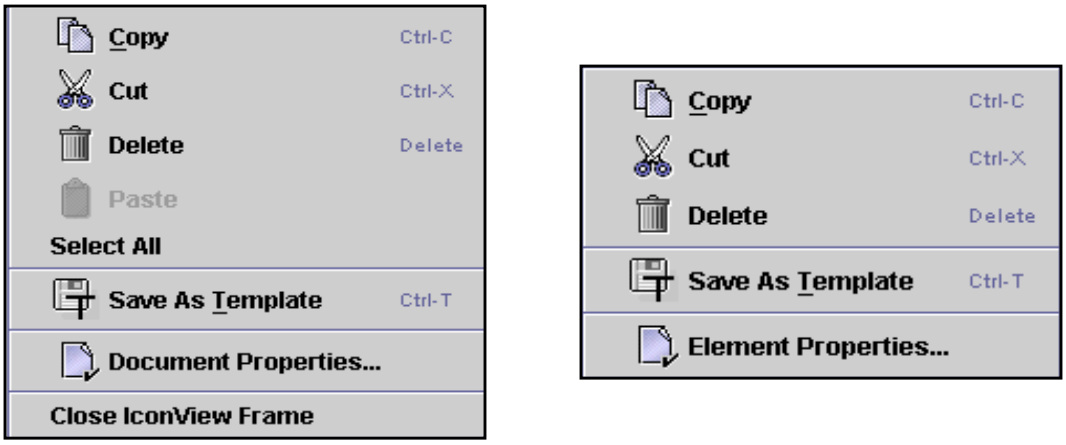


Abbildung 5-13. Die Kontextmenüs im Symbol-Ansicht Fenster von MAVA-Editor (nach der Anpassung)

Die Kontextmenüs im Baum-Ansicht Fenster werden ebenfalls durch den “Save As Template...” Eintrag erweitert. Wenn der Autor ein Knoten-Element oder Blatt-Element vom Baum im Baum-Ansicht Fenster selektiert, kann man mit der rechten Maustaste das Kontextmenü aufrufen. Danach kann Autor mit dem Menüeintrag “Save As Template...” das selektierte Element in einer Vorlage-Datei speichern.

Abbildung 5-14 zeigt die Kontextmenüs im Baum-Ansicht Fenster. Selektiert der Autor ein Knoten-Element im Baum, nämlich ein Container-Element des Dokuments, erscheint das linke Kontextmenü auf dem Bildschirm. Wenn er ein Blatt-Element vom Baum selektiert, d.h. ein nicht-Container Element des Dokuments, wird das rechte Kontextmenü der Abbildung 5-14 aufgerufen.

Durch die Erweiterung der Kontextmenüs ist es möglich, die selektierten Elemente der aktuellen Symbol-Ansicht Grundfläche in eine Vorlage-Datei zu speichern, gerade wo sich das Maus befindet. Dadurch wird die Speicherung der Vorlagen für den Autor schneller und intuitiver. Was ebenso die Benutzerfreundlichkeit bei der Speicherung von Vorlagen erhöht.

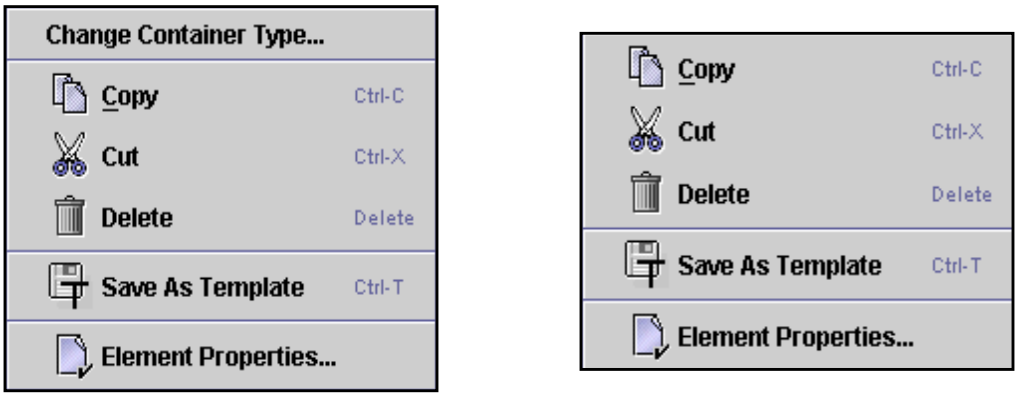


Abbildung 5-14. Die Kontextmenüs im Baum-Ansicht Fenster von MAVA-Editor (nach der Anpassung)

5.2.5 Anpassung der Symbolleiste von MAVA-Editor

Da die Schaltfläche “Open Media Selection Frame from Directory” eigentlich wie eine Abkürzung des Frame-Menü Eintrag “Media Selection From Directory” funktioniert, sind die beide mit der gleichen Funktionalität verbunden. Wie bereits beschrieben, ist die mit dem Menüeintrag “Media Selection From Directory” verbundene Funktion schon erweitert. Das Aussehen von der Symbolleiste wird bei der Anpassung nicht geändert.

5.2.6 Anpassung des Speicherformats

Um die Vorlagen in Dateien speichern zu können, muß das Speicherformat der Vorlagen im MAVA-Editor vorgegeben werden. Dafür gibt es folgende zwei Möglichkeiten:

Erste Möglichkeit ist, das vorhandene MAVA-Dokumentformat zu nehmen. Dafür gibt es einen Vorteil: Wenn Vorlagen im MAVA-Dokumentformat gespeichert sind, können sie als normale MAVA-Dokumente direkt in MAVA-Editor geöffnet und editiert werden.

Es ist zwar einfach, die Vorlagen in MAVA-Editor direkt öffnen zu können, aber eine große Beschränkung ist auch damit eng verbunden. Um die Vorlagen im MAVA-Dokumentformat zu speichern, müssen alle Vorlagen zuerst gültige MAVA-Dokument sein. Das

bedeutet, jede Vorlage muß die Voraussetzung für MAVA-Dokument erfüllen, einen Wurzel-Container zu besitzen, was natürlich eine sehr große Beschränkung für Vorlagen ist. Zum Beispiel, wenn der Autor nur eine Gruppe von Medienelemente und Operatoren als Vorlage speichern möchte. So ist dies nicht möglich, wenn das MAVA-Dokumentformat als Vorlagen-Format benutzt wird.

Ein anderer Nachteil ist, wenn die Vorlagen im MAVA-Dokumentformat gespeichert werden, daß die Vorlagen-Dateien die gleiche Datei-Erweiterung (“.mavaed”) wie MAVA-Dokumentdateien haben. Damit kann der Autor die Vorlagen-Dateien von den normalen MAVA-Dokumentdateien nicht direkt unterscheiden.

Die zweite Möglichkeit ist, ein spezielles Format für Vorlagen zu benutzen. Um beliebige Elemente ohne Wurzel-Container in die Vorlage speichern zu können, muß das Format von Vorlagen dem Autor erlauben, die Vorlage ohne das Wurzel-Container Element zu speichern. Damit ergibt sich mehr Flexibilität bei der Speicherung von Vorlagen.

Eine eigene Datei-Erweiterung für das neue Format wird ebenfalls festgelegt. Alle Vorlagen-Dateien enden mit der Datei-Erweiterung “.mavatmp”. Anhand dieser Datei-Erweiterung ist es viel einfacher, die Vorlagen-Dateien von normalen MAVA-Dokumentdateien zu unterscheiden.

5.3 Zusammenfassung

In diesem Kapitel wurden die notwendigen Anpassungen des MAVA-Editors für die Unterstützung von Vorlagen beschrieben.

Einerseits soll die Kompatibilität zum vorhandenen MAVA-Editor bei der Anpassungen berücksichtigt werden: Es sollen möglichst wenige Änderungen an dem vorhandenden MAVA-Editor durchgeführt werden. Alle vorhandene Funktionen und Benutzerschnittstellen sollen unverändert bleiben, solange sie nicht durch neue Funktionalität und Benutzerschnittstellen ersetzt werden sollen. Damit kann der Autor mit dem erweiterten MAVA-Editor immer noch arbeiten, ohne zu wissen, daß der MAVA-Editor durch Vorlagen erweitert wurden.

Andererseits soll die neue Funktionalität der Vorlagen im MAVA-Editor möglichst intuitiv und einfach zu benutzen, um die Einarbeitungsdauer zu verkürzen. Dies wurde erreicht durch direkte Manipulation der Symbole und Drag&Drop.

Nach in diesem Kapitel beschrieben wurde, wo in dem existierenden Editor Änderungen durchgeführt werden müssen, wird im nächsten Kapitel die Implementierung der Erweiterungen beschrieben.

6 Implementierung der Vorlagen für den MAVA-Editor

Im Kapitel 5 wurde die Anpassung des vorhandenen MAVA-Editors (kurz MED), für die Unterstützung der Vorlagen behandelt. Dieses Kapitel betrachtet die Implementierung der Vorlagen. Zuerst wird die interne Struktur vom MED vorgestellt. Im Anschluß daran wird diskutiert, welche Anpassungen für die Integration der Vorlagen in dem vorhandenen MED notwendig sind, und wie diese Anpassungen implementiert wurden. Am Ende wird eine kurze Zusammenfassung gegeben.

6.1 Die interne Struktur vom MED

In diesem Abschnitt wird die interne Struktur vom MED beschrieben. Zuerst wird das Konzept von Java-Paketen erklärt, damit man das Prinzip der Klassen-Gruppierung verstehen kann. Danach wird die Struktur vom MAVA-Projekt erläutert. Und am Ende kommt die Beschreibung der internen Struktur vom MED.

6.1.1 Das Paket-Konzept

Um die Struktur des MAVA-Editors zu verstehen, sollte man zunächst das Verständnis vom Paket-Konzept in Java besitzen. Das Paket-Konzept in Java bedeutet die Gruppierung von Java-Klassen. Ein Paket enthält typischerweise zusammenhängende Java-Klassen, die für gleiche oder ähnliche Nutzung entwickelt sind. [Bill00]

Abbildung 6-1 zeigt einige standardmäßige Pakete von Java: Zum Beispiel enthält das `lang`-Paket alle wesentlichen Java-Klassen, die für jedes Java-Programm gebraucht werden. Das `applet`-Paket dient dazu, Java-Applets für Anwendungen in Internet zu schreiben. Um plattformunabhängige Graphic User Interface (GUI) zu entwickeln, kommt das `swing`-Paket ins Spiel. Der Vorteil von Paketen ist, daß die Struktur von komplexen Programm viel übersichtlicher ist, indem man eine große Menge von Klassen in nur wenigen Pakete zusammenfaßt.

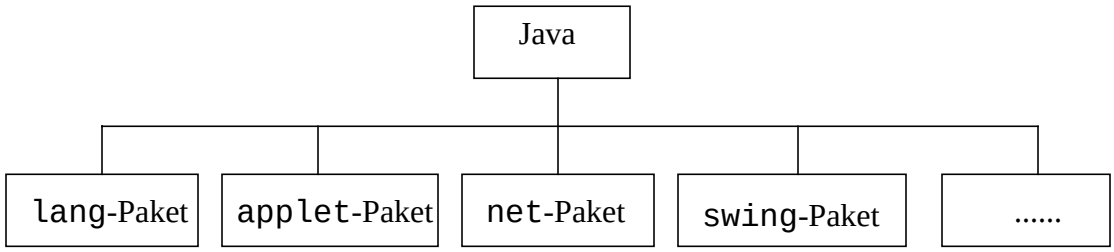


Abbildung 6-1. Einige standardmäßigen Java-Pakete

6.1.2 Die interne Struktur vom MAVA-Projekt

Das MAVA-Projekt besteht aus vier Hauptpaketen, dem `mava`-Paket, dem `mavax`-Paket, dem `med`-Paket, und dem `medx`-Paket. In der Abbildung 6-2 wird diese Struktur vom MAVA-Projekt gezeigt. Jedes Paket im MAVA-Projekt funktioniert als eine eigenständige Programm-Einheit mit eigener Funktionalität. Beispielsweise steht das `mava`-Paket für das MAVA-Präsentationssystem, und das `mavax`-Paket als die "Standard Erweiterung" davon. Das `med`-Paket ist zuständig für den MAVA-Editor, und das `medx`-Paket ist die "Standard Erweiterung" des `med`-Pakets.

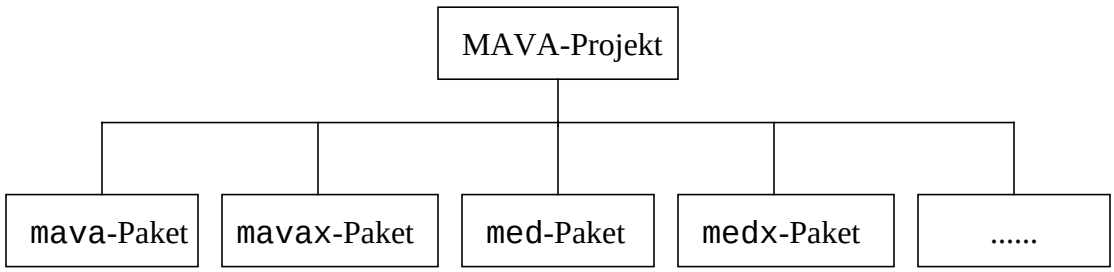


Abbildung 6-2. Die logische Struktur vom MAVA-Projekt

Davon ist `med`-Paket das einzige Paket, das für die Implementierung des MAVA-Editors zuständig ist. Also ist nur das `med`-Paket für die Implementierung von MAVA-Vorlagen interessant. Daher kommt später in diesem Kapitel nur noch das `med`-Paket in Betrachtung.

6.1.3 Die interne Struktur vom `med`-Paket

Das `med`-Paket ist zuständig für die Implementierung vom MAVA-Editor und besteht selbst wieder aus mehreren Unterpaketen. Jedes Unterpaket vom `med`-Paket faßt mehrere Java-Klassen zusammen und funktioniert als eine logische Einheit. Abbildung 6-3 zeigt die interne Struktur vom MAVA-Editor, wobei alle Unterpakete des `med`-Pakets angegeben sind.

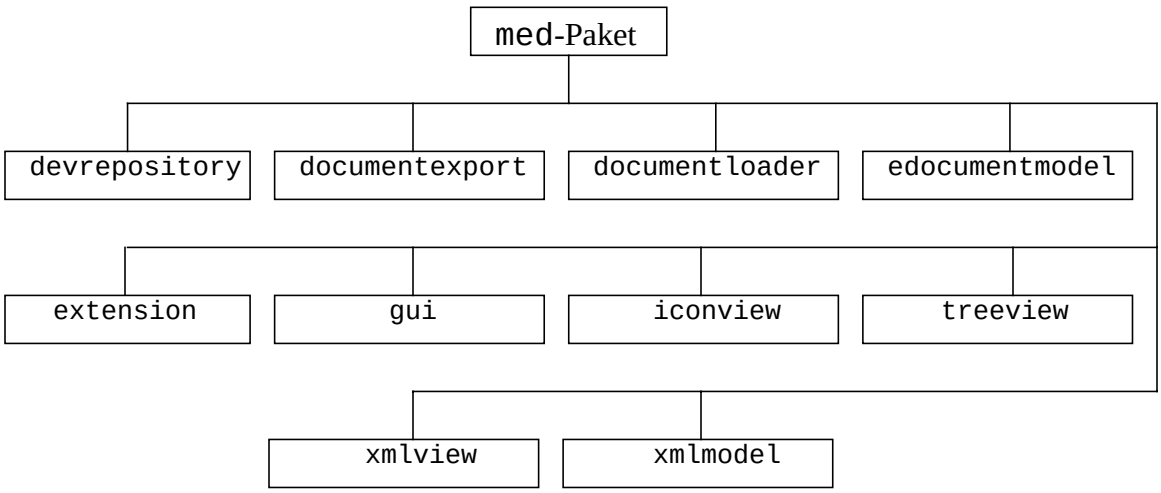


Abbildung 6-3. Die interne Struktur vom `med`-Paket

Folgend wird die Funktionalität von jedem einzelnen Unterpaket kurz vorgestellt:

- **Das `gui`-Paket**

Das `gui`-Paket besteht aus eine Menge von Klassen, die für graphische Benutzeroberfläche (engl. GUI) vom MAVA-Editor zuständig sind. Die GUI-Komponente vom MAVA-Editor sind z.B. Dialoge, Fenster, Button usw. Zum Beispiel, die Klasse `MediaFromDirectoryFrame` im `gui`-Paket befasst sich mit dem Fenster für die Auswahl aller Media-Dateien in einem bestimmten Verzeichnis.

- **Das `iconview`-Paket**

Die Klassen im `iconview`-Paket sind für die Symbol-Ansicht eines Dokuments zuständig. Alle Komponenten im Symbol-Ansicht-Fenster und alle Interaktionen mit diesem Fenster werden im `iconview`-Paket implementiert. Zum Beispiel steht die

`IconViewFrame` Klasse für das Symbol-Ansicht-Fenster, und die Klasse `Draggable-Label` ermöglicht das Drag&Drop von Elementsymbolen (engl. label) im Symbol-Ansicht-Fenster mit der Maus.

- **Das `treeview`-Paket**

Das `treeview`-Paket ist zuständig für die Darstellung der Baum-Ansicht auf ein Dokument. Die Struktur vom Dokument wird in Form eines Baum im Baum-Ansicht-Fenster dargestellt. Beispielsweise ist die Klasse `DocumentTreeFrame` zuständig für das Baum-Ansicht-Fenster. Alle Interaktionen mit der Baum-Ansicht werden in der Klasse `TreeViewListener` implementiert.

- **Das `xmlview`-Paket**

Das `xmlview`-Paket ist zuständig für die XML-Ansicht eines Dokuments, in der das Dokument in der Form von XML dargestellt wird. Sobald das Dokument im Symbol-Ansicht-Fenster vom Autor geändert wurde, wird die XML-Ansicht auch entsprechend aktualisiert. Dafür sind z.B. die Klassen `XMLViewFrame` und `XMLViewPanel` zuständig.

- **Das `edocumentmodel`-Paket**

Das `edocumentmodel-`Paket ist zuständig dafür, alle Elemente im MAVA-Dokumentmodell im MED abzubilden. Zum Beispiel stehen die Klassen `EContainer`, `EOperator` und `EElement` im `edocumentmodel-`Paket jeweils für den Container-, Operator- und Medienobjekt-Typ im MAVA-Dokumentmodell.

- **Das `devrepository`-Paket**

Das `devrepository`-Paket ist zuständig für die zentrale Aufbewahrung und Konfiguration aller Daten im MAVA-Editor. Die Attributen oder Identität aller MAVA-Editor Komponenten werden mit Hilfe vom `devrepository`-Paket deklariert und konfiguriert. Zum Beispiel, alle Beschreibungen und Veränderungen der Attribute eines Operators werden von der Klasse `OperatorDescription` implementiert.

- **Das `extension`-Paket**

Das `extension`-Paket definiert die Schnittstellen für das `medx`-Paket, was als die Standard Erweiterung vom `med`-Paket funktioniert. Beispielsweise dient die Klasse `Media-`

`Inspector` als die Schnittstelle für die Klassen `AudioInspector`, `GifMediaInspector` und `MovieInspector` im `medx`-Paket. Diese Inspector-Klassen haben die Funktionalität, die Attributen-Informationen des Medienelements aus den entsprechenden Media-Dateien automatisch zu ermitteln, was das Editieren erleichtert.

- **Das `xmlmodel`-Paket**

Das `xmlmodel`-Paket steht für die Erstellungen von XML-Darstellungen, in denen die Dokumente oder der Zustand vom MAVA-Editor gespeichert sind. Zum Beispiel ist die Klasse `XMLModel` zuständig für die Erstellung von XML-Darstellung des Dokuments, während die Klasse `XMLMEditorProperties` die XML-Darstellung für die MED Zustand-Informationen implementiert.

- **Das `documentexport`-Paket**

Die Funktionalität vom `documentexport`-Paket ist, die vom `xmlmodel`-Paket erstellten XML-Darstellungen in XML-Dateien zu speichern.

- **Das `documentloader`-Paket**

Das `documentloader`-Paket ist zuständig für das Laden und Analysieren von Dokumenten, die in den vom `documentexport`-Paket erstellten XML-Dateien gespeichert sind.

6.2 Die Implementierung der Anpassungen im MED

Die interne Struktur vom `med`-Paket und die Funktionalität von dessen Unterpaketen wurden bereits beschrieben. Um die Vorlagen in den MAVA-Editor zu integrieren, müssen einige Unterpakete des `med`-Pakets angepaßt werden. Folgend wird die Implementierung der Anpassungen in diesen Unterpaketen einzeln betrachtet.

6.2.1 Das `gui`-Paket

Die Klasse `MediaFromDirectoryFrame` im `gui`-Paket hat die Funktionalität, alle Media-Dateien in einem bestimmten Verzeichnis als Symbole in einem Fenster anzuzeigen. Diese Funktionalität soll für die Unterstützung der Vorlagen erweitert werden, damit die Vorlagen-Dateien im Verzeichnis ebenso als Symbole im Fenster angezeigt werden können. Um die Vorlagen von den Media-Dateien zu unterscheiden, wird neues Symbol für

Vorlagen benutzt werden. Darum muß die dafür zuständige Klasse `MediaFromDirectoryFrame` durch folgende Methoden erweitert werden:

- Der Konstruktor `MediaFromDirectoryFrame(File dir)` ist erweitert, um die Umschaltung zwischen Verzeichnissen zu ermöglichen. Eine Combo-Box wurde oben rechts in das Fenster eingefügt, um diese Funktionalität zu bieten. Im Kapitel 4 wurde bereits beschrieben, daß die Vorlagen nach der Kategorie in verschiedenen Unterverzeichnisse zu speichern sind. Die Combo-Box ist dafür zuständig, alle Unterverzeichnisse, die jeweils einer Kategorie entsprechen, darzustellen und die Umschaltung dazwischen zu ermöglichen. Wenn der Autor die Vorlagen einer Kategorie auswählen möchte, braucht er nicht das aktuelle Fenster zu schließen, sondern nur das Verzeichnis dieser Kategorie in der Combo-Box auszuwählen. Und die Vorlagen-Dateien in diesem Verzeichnis werden auf der Grundfläche angezeigt.
- Die Methode `addMediaItems(JPanel mediaPanel, File directory)` war zuständig für das Auslesen aller Media-Dateien von einem Verzeichnis der Parameter (`directory`) und Anzeigen von solchen Dateien als Symbole auf der Grundfläche (`mediaPanel`). Vor der Anpassung konnte diese Methode nur Media-Dateien anzeigen, die der MAVA-Editor kennt. Nach der Erweiterung kann diese Methode auch die Vorlagen-Dateien im Verzeichnis erkennen und als Symbole auf der Grundfläche anzeigen.
- Die Methode `rescanDirectory(File dir)` ist neu in der Klasse, um die Umschaltung zwischen verschiedenen Unterverzeichnisse in der Combo-box zu implementieren. Wenn das ausgewählte Verzeichnis in der Combo-box gewechselt wird, wird diese Methode aufgerufen, um alle Dateien in diesem Verzeichnis zu suchen.

6.2.2 Das `iconview`-Paket

Die Klassen `IconViewPanel` und `IconViewLabel` im `iconview`-Paket sind zuständig für die Implementierung der Grundfläche (engl. panel) in der Symbol-Ansicht und die Elementsymbole (engl. label) auf der Grundfläche im MED.

Im Kapitel 5 wurde die Anpassungen der Kontextmenüs betrachtet. Nach der Einführung der Vorlagen müssen die Kontextmenüs bei der Grundflächen und Elementsymbole erwei-

tert werden (siehe Abbildung 5-13). Deshalb müssen diese beiden Klassen angepaßt, damit die Kontextmenüs von der Grundfläche und Elementsymbole durch den neuen Menüeintrag “Save As Template...” erweitert werden.

- Die Methode `createPopupMenu()` ist zuständig für die Erstellung der Kontextmenüs in beiden Klassen. Deshalb wurde diese Methode modifiziert, damit die Kontextmenüs der Grundfläche und Elementsymbole durch den neuen Menüeintrag “Save As Template...” erweitert. Wenn der Autor diesen Eintrag auswählt, wird die Methode `doSaveTemplate()` von der Klasse `MEditor` aufgerufen, um die selektierten Dokumentelemente in Vorlagen zu speichern.

6.2.3 Das `treeview`-Paket

In Kapitel 5 wurde festgelegt, daß die Kontextmenüs im Baum-Ansicht-Fenster für die Unterstützung der Vorlagen erweitert werden müssen (siehe Abbildung 5-14). Da diese Kontextmenüs von der Klasse `TreeViewListener` im `treeview`-Paket implementiert sind, ist es erforderlich, die Klasse `TreeViewListener` wie folgend zu erweitern:

- Die Methode `createPopupMenu()` in der Klasse `TreeViewListener` ist zuständig für die Erstellung des Kontextmenü im Baum-Ansicht-Fenster und die Funktionalität der Menüeinträge. Identisch wie beim `iconview`-Paket wird diese Methode auch durch den neuen Menüeintrag “Save As Template...” modifiziert.

6.2.4 Das `xmlmodel`-Paket

Alle Dokumente werden im XML-Format gespeichert. XML ist eine Abkürzung von Extensible Markup Language[XML]. XML wurde vom World Wide Web Consortium (W3C) für den Einsatz im WWW entwickelt. [W3C] Mit den Fähigkeiten wie die Darstellung der Dokumentstruktur und die Selbstbeschreibung wurde XML für das Transferformat von Dokumenten eingesetzt. In der Diplomarbeit vom Rod Groß [Groß99] findet sich eine Einführung in das Format der XML-Dateien von Dokumenten, was aber im vorhandenen MED stark verändert wurde.

Die Klasse `XMLModel` im `xmlmodel`-Paket erstellt die XML-Darstellung von Dokumenten. Abbildung 6-4 zeigt die Struktur der XML-Darstellung von einem Dokument.

Wie in der Abbildung gezeigt, sollen die XML-Dokumente mit einer XML-Deklaration beginnen, die die verwendete XML-Version und Kodierung spezifiziert. Die Versionsnummer "`1.0`" hier muß benutzt werden, um Konformität zu dieser Version der Spezifikation anzuzeigen, und die Kodierungsdeklaration "`UTF-8`" [UTF-8] deklariert den Kodierungstyp des Dokuments.

Die XML-Dokumenttyp-Deklaration enthält oder verweist auf Markup-Deklarationen, die eine Grammatik für eine Klasse von Dokumenten bilden. Diese Grammatik ist bekannt als Dokumenttyp-Definition, kurz DTD. Die Dokumenttyp-Deklaration kann entweder auf eine externe Teilmenge (eine besondere Art eines externen Entity) verweisen, die Markup-Deklarationen enthält, oder sie kann Markup-Deklarationen direkt in einer internen Teilmenge enthalten oder beides.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mava [
......
]>

<mava>

<!-- Start of the document content -->
<ns0:BaseContainer id="i0" xmlns:mava="http://www.mava.de"
...... xmlns:ns0="de.uni-stuttgart.informatik.vs"
xmlns:ns1="de.uni-stuttgart.informatik.vs.timeline"
...... >
  <ns0:TextScroll id="i1" Name="unknown" Width="150"
Height="250" URL="./testfiles/Mava.html" Infinite="true" />
  <ns1:start Instant="0" >
   <destination idref="i1" />
  </ns1:start>
  ......
</ns0:BaseContainer>
......

</mava>
```

Abbildung 6-4. Die XML-Darstellung eines Dokuments

Im Fall von MAVA ist die Dokumenttyp-Deklaration der XML-Darstellung direkt in das Transferformat enthalten. Sie beginnt mit der Markierung "`<!DOCTYPE mava [`" und

endet mit der Markierung “]>”. Zwischen den Anfangs- und Ende-Markierungen von DTD befinden sich die Beschreibungen der Typen und Attributeinformationen von allen Dokumentelementen. Da diese Informationen nicht Vorlagen-relevant sind, werden sie von der Abbildung 6-4 weggelassen.

Mit der Markierung “<mava>” beginnt der Abschnitt vom Dokument-Inhalt, der wieder mit der Markierung “</mava>” beendet.

Der Inhalt vom Dokument beginnt mit dem Kommentar “<!-- Start of the document content -->”. Alle Dokumentelemente werden hier abgebildet. Zu beachten ist, daß das Wurzel-Container Element (engl. root-container) immer ganz am Anfang vom Dokument-Inhalt dargestellt ist. Bei dem Beispiel in Abbildung 6-4 wird das BaseContainer-Element zuerst beschrieben, zusammen mit der Beschreibung des Namesraum für jedes Element in diesem Dokument. Danach werden alle Dokumentelemente einzeln beschrieben.

Nach der Einführung der Vorlagen muß der MAVA-Editor auch in der Lage sein, die XML-Darstellung für die Vorlagen im MAVA-Editor zu erstellen. Deshalb ist eine Erweiterung von der Klasse XMLModel hier notwendig. Jedoch ist hier zu bemerken, daß eine Vorlage nicht immer ein Wurzel-Container Element besitzt, und daher auch nicht immer ein gültiges MAVA-Editor Dokument ist.

```
<?xml version="1.0" encoding="UTF-8" ?>

<mava_template>

<!-- Start of mava template content -->

<NameSpaceInfo xmlns:ns1="de.uni-stuttgart.informa-
tik.vs.abspositioning" ......>
</NameSpaceInfo>

 <ns0:center mava:label-x="265" mava:label-y="190" >
  <destination idref="i25" />
 </ns0:center>
 ......

</mava_template>
```

Abbildung 6-5. Die XML-Darstellung einer MAVA-Vorlage

Einen neuen Konstruktor der Klasse, nämlich `XMLModel(Selection sel, IconViewModel ivm)`, wird eingefügt für die Vorlagen. Die drei Methoden `getXML(String documentBaseURL)`, `getXMLRec(EElement element)` und `searchNameSpaces()` mußten angepaßt werden, um die XML-Darstellung für die Vorlagen zu erzeugen.

Abbildung 6-5 zeigt die Struktur der XML-Darstellung für MAVA-Vorlagen. Die XML-Deklaration bleibt gleich wie bei normalen Dokumenten. Jedoch ist zu beachten, daß in der XML-Darstellung von Vorlagen keine Dokumenttyp-Definition (DTD) eingesetzt wird. Da MAVA-Vorlagen keine gültige MAVA-Editor Dokumente sind, sind die DTDs hier nicht erforderlich. Nachdem die Vorlagen in den vorhandenen Dokumenten eingefügt sind, wird die Dokumenttyp-Definition von Dokumenten aktualisiert, damit die Typen von den Klassen der Vorlagen-Elementen auch deklariert werden.

Der Abschnitt einer MAVA-Vorlage beginnt mit der Anfangsmarkierung "`<mava_template>`" und endet mit der Endmarkierung "`</mava_template>`". Der größte Unterschied in den XML-Darstellungen zwischen Dokumenten und Vorlagen liegt daran, daß die XML-Darstellungen von Dokumenten immer ein Wurzel-Container Element besitzt, bei den Vorlagen aber nicht. Daher gibt es spezielle Anfangsmarkierung "`<NameSpaceInfo`" und Endmarkierung "`</NameSpaceInfo>`", um die Namesraum-Informationen aller Vorlagen-Elementen darzustellen.

6.2.5 Das `documentloader`-Paket

Die Klasse `DocumentLoader` im `documentloader`-Paket ist zuständig für das Laden von Dokumenten. Das bedeutet das Auslesen und Analysieren von XML-Dateien, in denen die Dokumente gespeichert sind, um das Dokument im MED darzustellen.

Nach der Einführung von Vorlagen werden die Vorlagen auch in den XML-Dateien gespeichert. Daher ist es notwendig, die Klasse `DocumentLoader` zu erweitern, um Vorlagen aus Dateien laden zu können.

Dafür wird einen neuen Konstruktor `DocumentLoader(File fileName, MEditor mEditor, IconViewPanel iconViewPanel, Point dropLocation)` eingefügt. Dieser Konstruktor akzeptiert eine Vorlagen-Datei mit dem Name "`file-`

`Name`", und fügt diese Datei auf die Position "`dropLocation`" der Symbol-Ansicht Grundfläche "`iconViewPanel`" vom dem MAVA-Editor "`mEditor`".

Die Methode `startElement(String name, org.xml.sax.AttributeList attrs)` der Klasse `DocumentLoader` wird ebenfalls erweitert, um die Vorlagen-Elemente und ihre Attribute in der XML-Darstellung einzeln zu analysieren.

6.3 Zusammenfassung

In diesem Kapitel wurde die Implementierung der Vorlagen im MAVA-Editor betrachtet. Zuerst wurde die interne Struktur von MED vorgestellt, danach wurde die notwendigen Anpassungen des `med`-Paket zur Unterstützung der Vorlagen festgestellt, und folglich die Implementierung der Anpassungen beschrieben. Nach der Implementierung wird eine Bedienungsanleitung im nächsten Kapitel gegeben, um zu erklären, wie man die MAVA-Vorlagen benutzen kann.

7 Bedienungsanleitung für MAVA-Vorlagen

Nachdem die Implementierung des Vorlagen-Konzepts in Kapitel 6 betrachtet wurde, wird eine Bedienungsanleitung für MAVA-Vorlagen in diesem Kapitel angegeben. Dafür wird ein konkretes Beispiel hier genommen, um den ganzen Ablauf der Erstellung einer Vorlage zu beschreiben: von der ursprünglichen Motivation zu der Erstellung, und bis zu der Benutzung. Mit der Hilfe von dem Beispiel sollte man erkennen, in welchem Zustand es sinnvoll ist, eine Vorlage einzusetzen. Und, wenn eine Vorlage für die Arbeit nötig ist, wie kann man sie erstellen und folglich benutzen. Am Ende dieses Kapitel gibt es eine kurze Zusammenfassung.

7.1 Die Motivation der Vorlage

In diesem Abschnitt wird die Motivation der Vorlage beschrieben. Um es leicht verständlich zu machen, wird das folgende typische Szenario als Beispiel genommen:

Der Autor möchte mit dem MAVA-Editor ein Computer-gestützten Training Programm (CBT) erstellen, um dem Benutzer des Programms das Wissen über ein bestimmtes Thema zu vermitteln. Als Überprüfung der Verständnisse vom Benutzer für das angeeignete Wissen stellt das CBT-Programm dem Benutzer einige konkrete Fragen. Hier ist Multiple-Choice-Frage ein sehr übliches Frageelement.

Um die Multiple-Choice-Frage zu beantworten, wählt der Benutzer zwischen einer oder mehreren gegebenen Antworten die richtigen aus. Das Programm akzeptiert die eingegebene Antworten vom Benutzer und reagiert unterschiedlich je nach der Antwort. Zum Beispiel, wenn die Antwort richtig ist, dann gibt das CBT-Programm eine Grüße zurück. Wenn die Antwort jedoch falsch ist, wird die richtige Lösung der Frage auf dem Bildschirm gezeigt.

Eine Multiple-Choice-Frage braucht drei Präsentationsszenen: eine für die Darstellung der Frage, zwei für die Reaktionen auf die jeweils richtige bzw. falsche Antwort. Wenn der Autor mit dem MAVA-Editor diese Frage erstellt, braucht er zuerst drei Container-Elemente: ein Container-Element von dem Typ “Question”, was die Präsentationsseite der Frage entspricht, und zwei Container-Elemente von dem Typ “Scene”, die für die Darstellung der Reaktionen in Präsentationsszenen zuständig sind. Da jedes Container-Element eine Präsentationsseite entspricht, muß der Autor die Größe und Farbe der Seiten als Attribute des Container-Elements angeben, indem er das Kontextmenü von jedem Container-Element aufruft, um die Attribute zu setzen.

Drei Präsentationsszenen sind noch nicht ausreichend für die Multiple-Choice-Frage. Um die Reaktion des CBT-Programms richtig zu setzen, braucht der Autor noch zwei Operatoren: ein Operator vom Typ “correct”, und der andere vom Typ “false”. Der “correct”-Operator verbindet die Frageseite und die Reaktion auf die richtige Antwort, während der “false”-Operator die Frageseite zu die Reaktion auf die falsche Antwort führt. Damit wird die logische Struktur der Reaktion auf die Beantwortung einer Multiple-Choice-Frage vervollständigt.

Um die Struktur der Reaktion in einer Multiple-Choice-Frage zu spezifizieren, muß der Autor jedes Mal folgendes machen: Zuerst setzt der Autor drei Container-Elemente ein, jedes Element steht für eine Präsentationsseite. Danach gibt der Autor die Attribute von jedem Container-Element an, um die Größe und Farbe der Präsentationsseite festzustellen. Schließlich verbindet der Autor die Container-Elemente mit den zwei Operatoren, um die Reaktion des CBT-Programms zu definieren.

Obwohl die Inhalte von verschiedenen Multiple-Choice-Fragen sehr unterschiedlich sind, bleiben die obengenannten Arbeitsschritte für alle Fragen identisch, da die grundlegende Reaktion auf die Beantwortungen in aller Multiple-Choice-Fragen immer gleich sind: drei Präsentationsszenen für die Darstellung der Frage und Reaktionen, und zwei Operator für die Reaktionen auf die richtige und falsche Antwort. Für alle Multiple-Choice-Frage in einem CBT-Programm gilt es, daß die Attribute wie Größe und Farbe von Präsentationsszenen normalerweise gleich sind.

An dieser Stelle kommt die Frage, warum soll der Autor jedes Mal die Struktur der Multiple-Choice-Fragen von Hand neu erstellen, wenn die Struktur aller Fragen identisch sind? Einer Grundgedanke ist, diese Arbeitsschritte bei der Erstellung von Struktur einer Multiple-Choice-Frage als eine Vorlage zu speichern. Danach benutzt der Autor diese Vorlage, um die Erstellung der Reaktion auf die Beantwortung in aller Multiple-Choice-Fragen zu automatisieren. Damit werden viele Handarbeit vom Autor gespart, und die Effizienz der Arbeit wird ebenfalls stark erhöht.

7.2 Die Erstellung der Vorlage

Oben wurde ein typisches Szenario genommen, um es leicht verständlich zu machen, wann es sinnvoll ist, eine Vorlage bei der Erstellung von Multimedia-Dokumenten einzusetzen. Durch das Beispiel ist es verdeutlicht, daß eine Vorlage für Multiple-Choice-Fragen im MAVA-Editor gewünscht ist. In diesem Abschnitt wird die Erstellung der Vorlagen vorgestellt. Um eine Vorlage erstellen zu können, sind folgende Schritte erfordert:

7.2.1 Neues MED-Dokument öffnen

Um die Erstellung einer Vorlage zu beginnen, soll der Autor zuerst ein neues MED-Dokument öffnen, um die Elemente der Vorlage einfügen zu können.

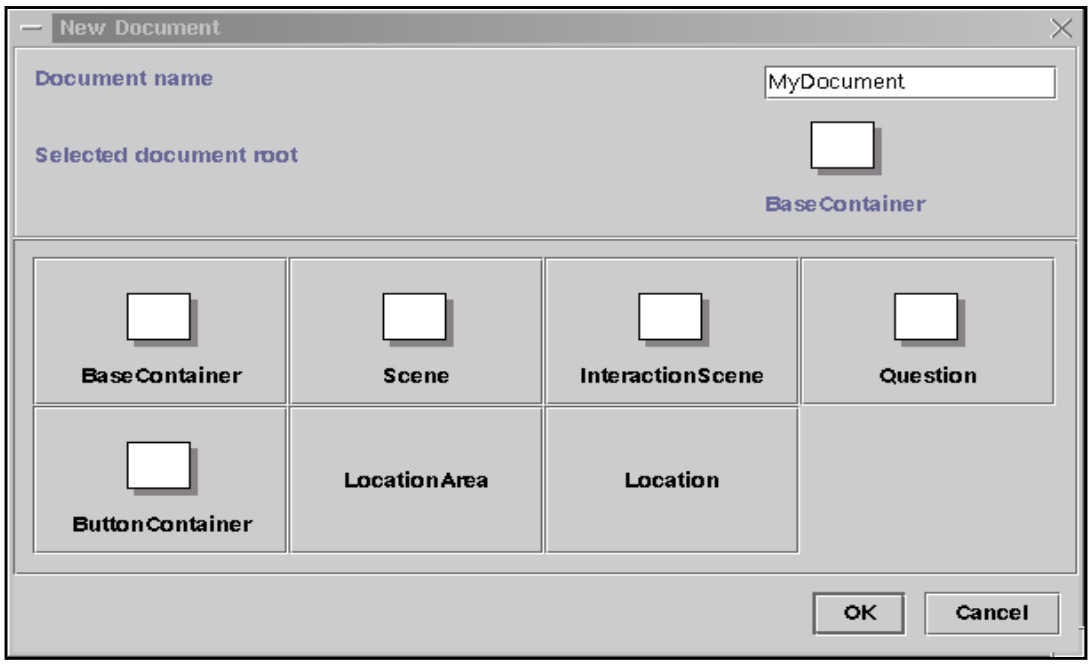


Abbildung 7-1. Das Dialogfenster zur Erstellung neues MED-Dokuments

Wenn der Autor ein neues MED-Dokument öffnen möchte, kann er entweder den Eintrag “New” im File-Menü auswählen, oder mit der Maus auf die Schaltfläche “New Document” in der Symbolleiste klicken. Dann erscheint ein Dialogfenster auf dem Bildschirm, worin der Autor den Name des Dokuments angeben kann. Außerdem kann der Autor in diesem Fenster den Typ des Wurzel-Container Element vom Dokuments auswählen.

Das Dialogfenster wird in der Abbildung 7-1 gezeigt. Oben im Textfelder “Document name” gibt der Autor den Name vom Dokument an. Darunter im Fenster werden alle verfügbare Typen von Wurzel-Elementen als Schaltfläche auf einer Grundfläche (engl. Panel) gezeigt. Der Autor kann einen Typ davon auswählen, indem er mit der Maus auf die entsprechende Schaltfläche klickt. Wählt der Autor den Element-Typ “BaseContainer” aus und klickt anschließend auf die Schaltfläche “OK”, wird das BaseContainer-Element als Wurzel-Container des zu erstellenden Dokuments festgelegt.

7.2.2 Aufbau der Vorlage

Der Wurzel-Container im MAVA-Editor wird als ein leeres Fenster gezeigt, wohin der Autor die benötigten Elemente einfügen, um Vorlage aufbauen zu können.

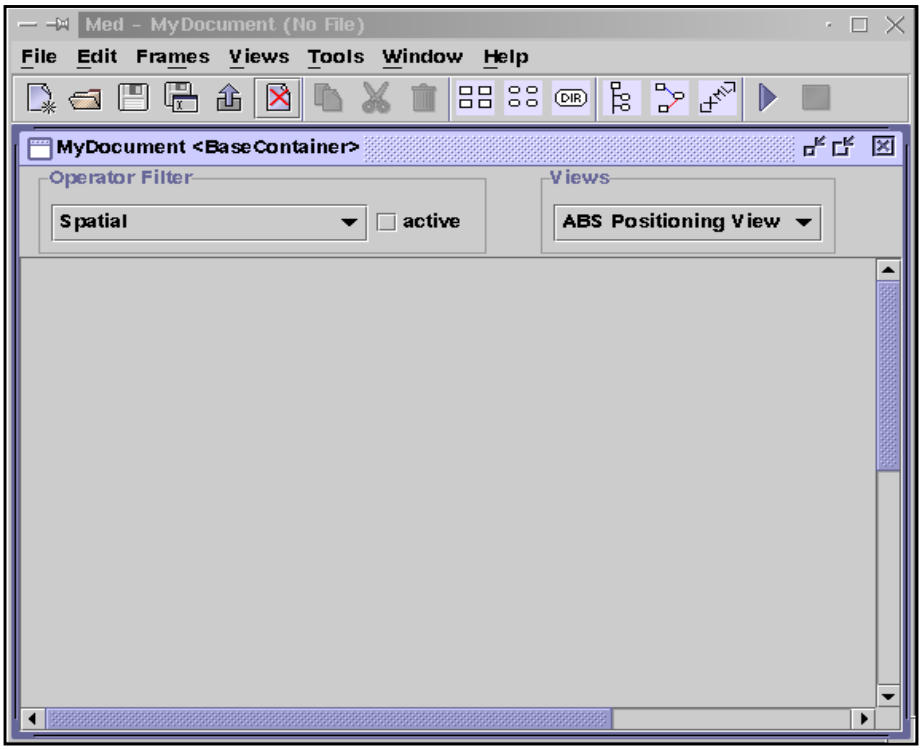


Abbildung 7-2. Das leere Fenster von dem Wurzel-Container des Dokuments

Abbildung 7-2 zeigt das Fenster von dem Wurzel-Container Element. Das Einfügen von Medienelementen oder Operatoren von der Vorlage in das Fenster geht gleich wie bei der Erstellung von einem Dokument, damit die Benutzung der Vorlagen so weit wie möglich intuitiv und einfach für den Autor bleiben.

Wenn der Autor einen Operator ins Dokument einfügen möchte, klickt er mit der Maus auf die Schaltfläche “Open Operator Selection Frame” in der Symbolleiste, oder wählt den Eintrag “Operator Selection Frame” im Frame-Menü. Dann erscheint ein Fenster auf dem Bildschirm, in dem alle verfügbare Operatoren als Symbole gezeigt sind. Wenn der Autor ein Medienelement braucht, klickt er auf die Schaltfläche “Open Media Item Selection Frame”, oder wählt den Eintrag “Media Selection Frame” im Frame-Menü. Und alle verfügbare Medienelemente werden als Symbole in einem anderen Fenster gezeigt.

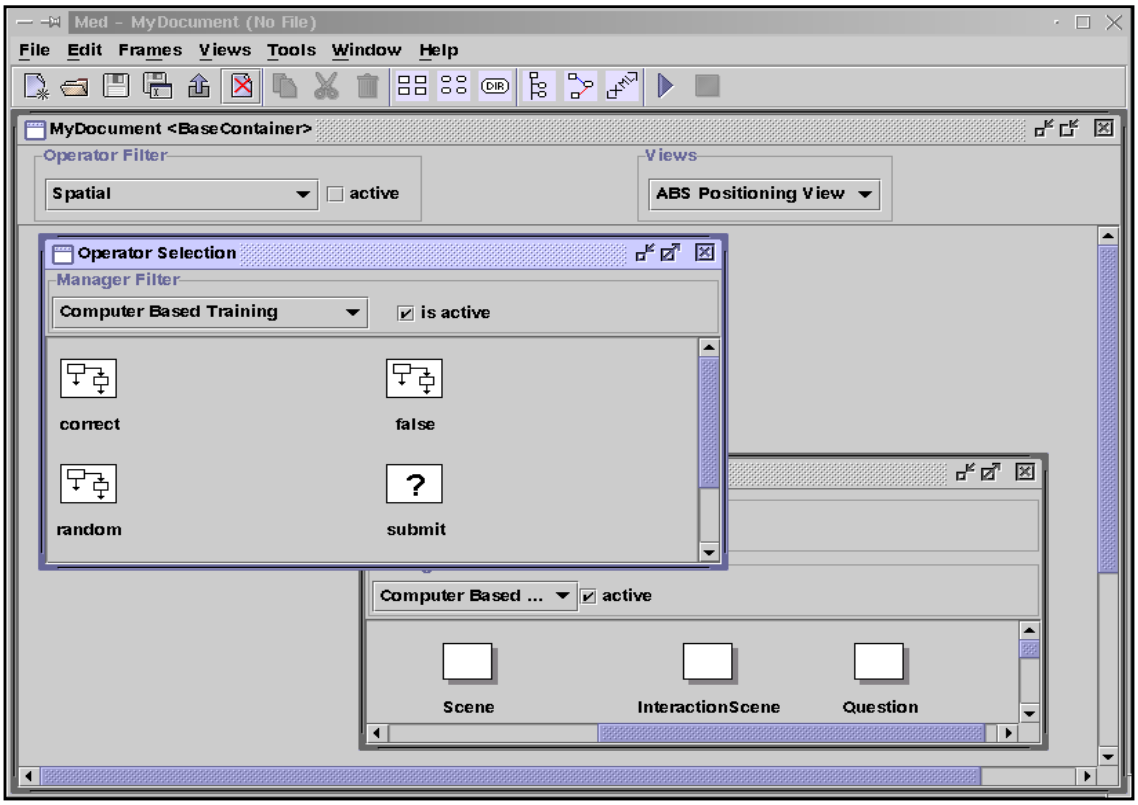


Abbildung 7-3. Die Fenster für die Auswahl der Operatoren und Medienelementen

Die beiden Fenster für Auswahl von Medienelemente und Operatoren bieten Funktionalität der Filterung, damit nur die Elemente von einer Kategorie im Fenster gezeigt werden. Wie

in der Abbildung 7-3 zeigt, ist die Filterung “Computer Based Training” in beiden Fenster aktiviert, daher werden nur die Operatoren und Medienelemente in CBT Kategorie hier gezeigt, und der Autor erhält eine bessere Übersicht.

Hat der Autor das Symbol des gesuchten Element im Fenster gefunden, dann kann er das Symbol mit der Maus selektieren, und anschließend in das Fenster vom Wurzel-Container per Drag&Drop einziehen.

Nachdem das Symbol des Element im Fenster vom Wurzel-Container eingezogen ist, werden die entsprechenden Elemente kopiert und in das Dokument eingefügt. Hier fügt der Autor zuerst drei Container-Elemente ins Dokument: Der Question-Container mit dem Namen “Frage” steht für die Frage. Der Szene-Container mit dem Name “Richtige Antwort” steht für die Reaktion auf die richtige Antwort, und der andere für die Reaktion auf die falsche Antwort. Der Autor kann die Attribute von Container-Elementen feststellen, indem er die Kontextmenüs der Elemente aufrufen, und anschließend den Eintrag “Element Properties...” auswählt. Ein Dialogfenster erscheint auf dem Bildschirm, worin der Autor die Größe, Farbe und andere Attribute von Präsentationsseite überprüfen und ändern kann.

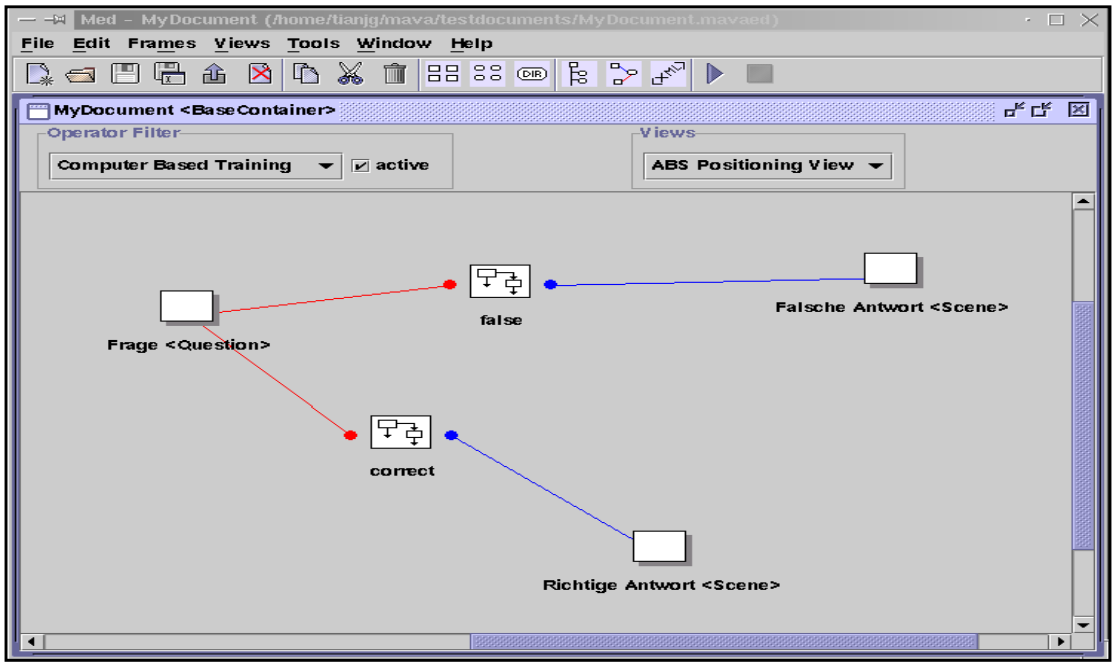


Abbildung 7-4. Die Vorlage für die Reaktion auf eine Multiple-Choice-Frage

Danach fügt der Autor noch zwei Operatoren ein: den “correct”-Operator und den “false”-Operator, und verbindet die Operatoren mit den Container-Elementen zusammen, um die logische Beziehung für die Reaktion zu erstellen.

Somit ist der Aufbau der Vorlage vervollständigt, alle Elemente der Vorlage sind eingefügt und die Attribute sind auch richtig gesetzt. In nächstem Abschnitt wird die Speicherung der Vorlage behandelt.

An dieser Stelle ist es jedoch zu bemerkt, daß diese Vorlage nur für die Reaktion auf die Beantwortung zuständig ist. Da jede Frage oder Reaktion selbe komplexe Struktur besitzt, kann der Autor Vorlage ebenfalls auf einer niedrigen Stufe einsetzen.

Zum Beispiel besetzen alle normalen Frageszene immer gleiche Struktur: der Titel der Frage, der Inhalt der Frage, die Kandidatantworten, die Kästchen neben den Antworten für Auswahl und eine Schaltfläche für Bestätigung der Antwort. Obwohl verschiedene Frage unterschiedliche Inhalte haben, bleibt die Struktur der Fragen sehr ähnlich. Daher ist es sinnvoll, eine Vorlage für die Struktur der normalen Frageszene zu erstellen.

Solange es sinnvoll ist, um die wiederholende Arbeit zu sparen, kann der Autor die Vorlagen an jeder Stelle erstellen. Daher ist es normal, daß es innerhalb Vorlagen noch Vorlagen gibt, wie zum Beispiel eine Frageszene-Vorlage in einer Reaktion-Vorlage einzusetzen.

7.2.3 Als Vorlage speichern

Nachdem alle Elemente in das Dokument eingefügt und deren Attribute richtig gesetzt sind, kann der Autor die drei Container-Elemente zusammen mit zwei Operatoren als Vorlage speichern.

Um die Elemente im Fenster als Vorlage zu speichern, muß der Autor zuerst diese Elemente mit der Maus selektieren. Danach kann der Autor entweder den Eintrag “Save As Template...” im Edit-Menü auswählen, oder das Kontextmenü in dem Symbol-Ansicht Fenster aufrufen und den gleichen Eintrag wählen, um das Dialogfenster für die Speicherung der Vorlage aufzurufen. Außerdem kann er mit der rechten Maustaste auf das Wurzel-Container Element in der Baum-Ansicht klicken, um das Kontextmenü aufzurufen und anschließend den Eintrag “Save As Template...” zu wählen. Danach erscheint das gleiche Dialogfenster auf dm Bildschirm.

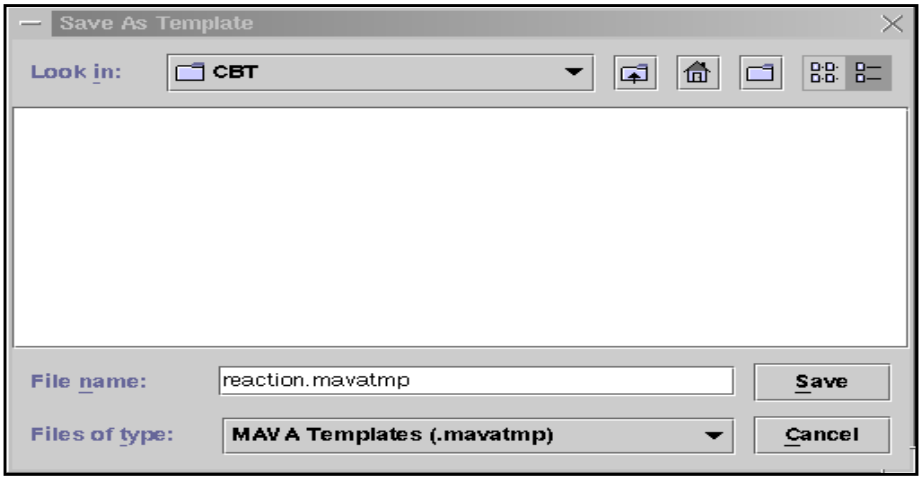


Abbildung 7-5. Das Dialogfenster für Speicherung der Vorlage

Abbildung 7-5 zeigt das Dialogfenster für die Vorlage-Speicherung. Oben im Fenster kann der Autor das Verzeichnis festlegen, wohin die Vorlage gespeichert werden soll. Da die Vorlage für CBT-Programm entwickelt ist, wird hier das Unterverzeichnis “CBT” vom Verzeichnis “templates” als Speicherort genommen. Unten im Fenster kann der Autor noch den Name für die Vorlage angeben, hier im Beispiel wird “reaction” als Name gegeben. Die Datei-Erweiterung der Vorlage ist “mavatmp”, die auch unter dem Dateiname angezeigt wird.

7.3 Die Benutzung der Vorlage

Bisher wurde beschrieben, wie der Autor eine Multiple-Choice-Frage Vorlage erstellt. In diesem Abschnitt wird nun die Benutzung dieser Vorlage vorgestellt.

Nachdem die Vorlage gespeichert wurde, kann diese Vorlage später beliebig oft wiederverwendet werden, wenn eine CBT erstellt werden soll.

Abbildung 7-6 zeigt das Symbol-Ansicht Fenster eines Dokuments im MAVA-Editor. Das Dokument ist ein CBT-Programm. Der Autor hat gerade eine “Scene”-Container als die Einleitung ins Dokument eingefügt. Anschließend möchte der Autor eine Multiple-Choice-Frage ins Dokument einsetzen. Da es schon eine Vorlage dafür gibt, muß der Autor nicht mehr die Struktur der Reaktion in Frage von Hand aufbauen.

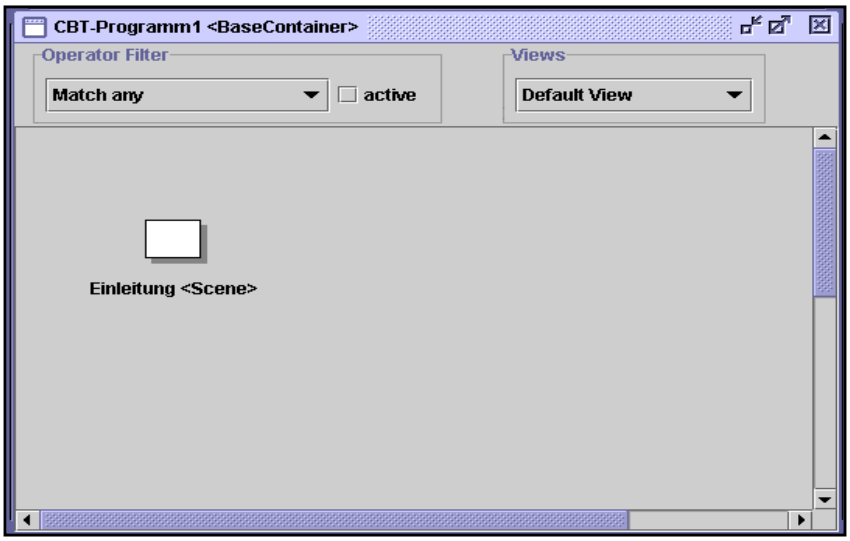


Abbildung 7-6. Das neue CBT-Programm mit Einleitungsszene

An dieser Stelle verwendet der Autor eine Vorlage, indem er den Eintrag “Media Selection from Directory...” im Frame-Menü auswählt, oder mit der Maus auf die Schaltfläche “Open Media Selection from Directory...” in Symbolleiste klickt. Dann wird ein Dialogfenster wie in Abbildung 7-7 gezeigt auf dem Bildschirm erscheinen, worin der Autor das Verzeichnis angeben kann, wo die Vorlagen gespeichert ist.

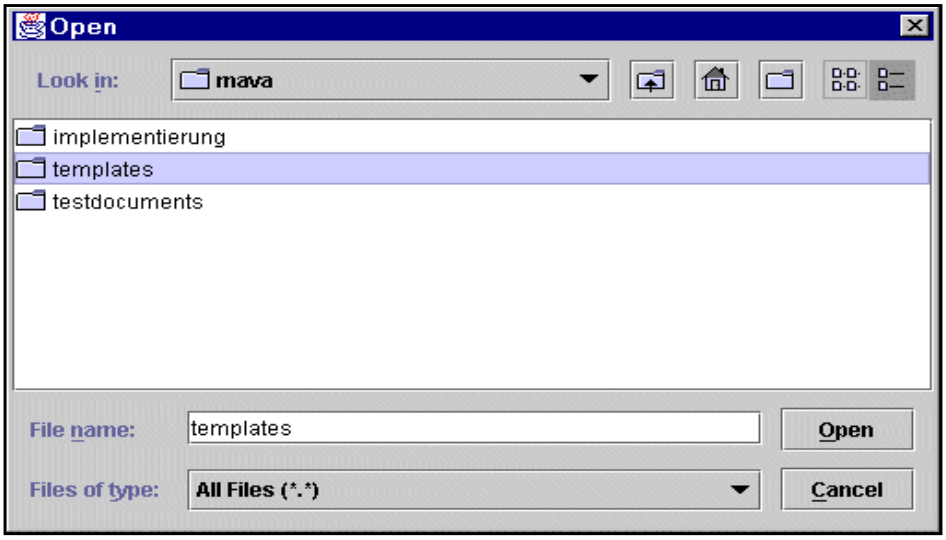


Abbildung 7-7. Das Dialogfenster für öffnen von Vorlagen

Nachdem der Autor das Verzeichnis ausgewählt hat, in dem die Vorlagen gespeichert ist, werden alle Vorlagen-Dateien in diesem Verzeichnis als Symbole in einem Fenster dargestellt, wie in der Abbildung 7-8 gezeigt ist.

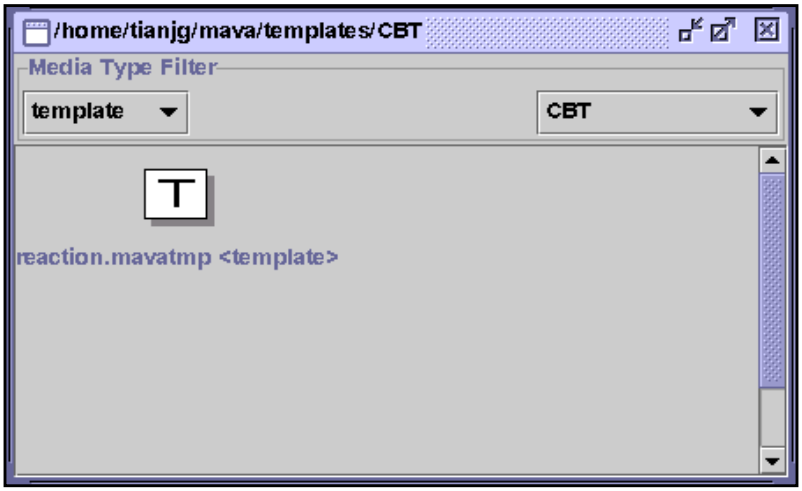


Abbildung 7-8. Das Fenster für die Auswahl der Vorlagen- und Media-Dateien

Links oben im Fenster befindet sich ein Combo-Box, damit man die Filterung-Funktionalität benutzen kann, um nur bestimmte Datei-Typen anzeigen zu lassen. Zum Beispiel, wenn der Autor im Combo-Box den Medien-Typ “template” auswählt, werden nur die Vorlagen-Dateien im Fenster gezeigt. Alle Dateien von anderen Typen, wie z.B. Bilder, Audio- oder Video-Sequenzen, werden ausgeblendet, damit der Autor gute Übersicht kriegen kann.

Der Autor kann die Vorlage ins Dokument einfügen, indem er das Symbol der Vorlage mit der Maus per Drag&Drop auf die Symbol-Ansicht Grundfläche vom Dokument einzieht. Alle Elemente von Vorlage werden kopiert und in das Dokument eingefügt. Diese Symbole von den kopierten Elementen werden alle auf der Grundfläche dargestellt. Wie Abbildung 7-9 zeigt, ist die Struktur der Reaktion in der Multiple-Choice-Frage schon im Fenster mit der Hilfe von Vorlage automatisch dargestellt. Und der Autor muß sich jetzt nur um den Inhalt der Frage kümmern.

Der Autor kann diese eingefügten Elemente wie normale Dokumentelemente behandeln, wie z.B. kopieren, löschen, verändern, usw. Dadurch braucht der Autor nicht neu einzuar-

beiten, um Vorlagen zu benutzen, den Vorgang hat er schon bei der Erstellung von normalen MED-Dokumenten kennengelernt.

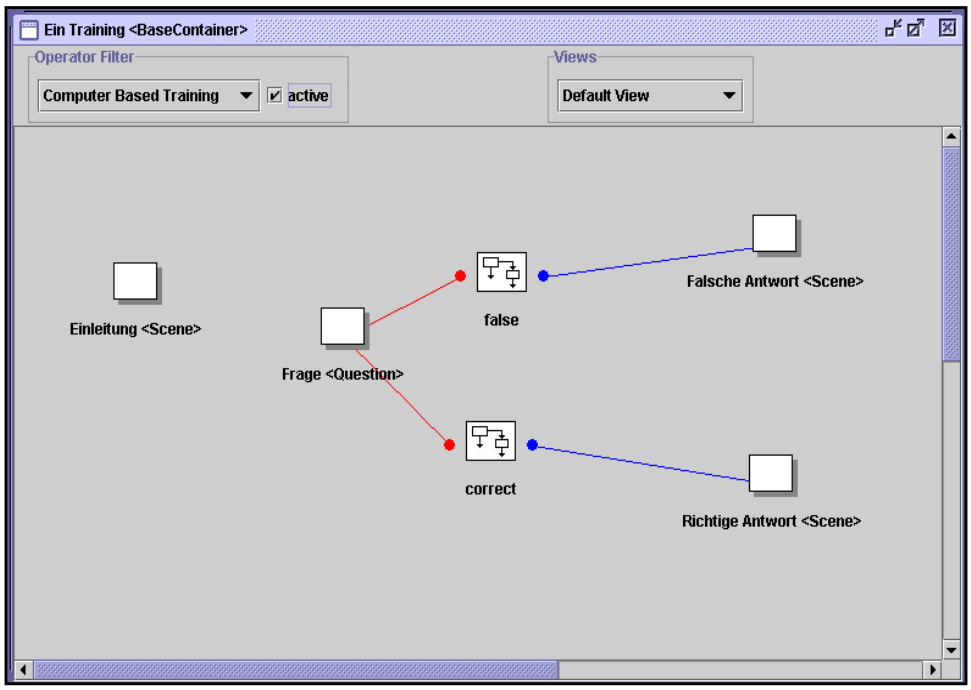


Abbildung 7-9. Das CBT-Programm mit der eingefügten Vorlage für Multiple-Choice-Frage

7.4 Zusammenfassung

In diesem Kapitel wurde die Bedienungsanleitung anhand eines Beispiels für MAVA-Vorlagen gegeben. Eine typische Szene wurde genommen, um es leicht verständlich zu machen, wann eine Vorlage Sinn macht, und wie der Autor eine Vorlage erstellen und benutzen kann. Durch die Beschreibung wurde verdeutlicht, daß die Erstellung und Benutzung von MAVA-Vorlagen identisch wie bei normalen Dokumenten ist, und daher für den Autor sehr intuitiv und einfach ist. Deshalb wird die extra Einarbeitungsdauer für die Vorlagen deutlich reduziert.

8 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es, Vorlagen für den vorhandenen MAVA-Editor (kurz MED) zu konzipieren und zu implementieren. Vorlagen werden benutzt, damit die Wiederverwendbarkeit der MED-Dokumenten erhöht wird und der Autor von sich immer wiederholenden ähnlichen Arbeiten befreit werden kann. Um den MAVA-Editor durch Vorlagen zu erweitern, sollte der vorhandene Editor für die Unterstützung der Vorlagen angepaßt werden.

Im Allgemein funktionieren Vorlagen wie ein Muster, nach dem man etwas herstellt. Im täglichen Leben verwendet man oft Vorlagen, zum Beispiel beim Schreiben von Dokumenten, Aufbauen von Maschinen oder Programmieren von Anwendungen. Durch die Verwendung von Vorlagen werden sich immer wiederholende Arbeiten automatisiert. Damit ergeben sich viele Vorteile: Einerseits spart man viel Zeit und Bemühung, um sich auf die neue Arbeit zu konzentrieren. Andererseits werden viele potentielle Fehler durch die Verwendung von Vorlagen vermieden.

Im Bereich der Spezifikation von Multimedia-Dokumenten wird die Verwendung der Vorlagen immer wichtiger, um die Erstellung der Multimedia-Dokumenten so weit wie möglich zu vereinfachen. Das Ziel von Vorlagen in der Erstellung von Multimedia-Dokumenten ist: Der Autor muß sich nur um die neuen Arbeiten kümmern, und alle sich wiederholenden Arbeiten sollen von Vorlagen automatisch erledigt werden. Mit Hilfe der Vorlagen kann der Autor schneller und effizienter Multimedia-Dokumenten erstellen. Daher unterstützen zur Zeit immer mehr Multimedia Autorensysteme Vorlagen, wie zum Beispiel Powerpoint, Authorware, ToolBook, Mediator, usw.

Nach der Untersuchung von einigen repräsentativen Autorensystemen wurde festgestellt, daß Vorlagen in all diesen Autorensystemen Schwachestellen haben: Die Vorlagen in manchen Autorensystemen wie zum Beispiel Mediator und Question Mark Designer sind zwar einfach zu bedienen, besitzen aber nur relativ wenig Funktionalität, und nicht oder nur

schwer zu verwalten. In machen Autorensystemen wie beispielsweise Powerpoint sind Vorlagen leicht zu bedienen und erweiterbar, können aber die strukturellen Informationen weder sichtbar machen noch erhalten. Die Vorlagen in Authorware und ToolBook besitzen viele nützliche Funktionalität und sind leicht zu erweitern, können aber ebenfalls die strukturellen Informationen nicht erhalten. Außerdem ist die Erstellung von Vorlagen hier für den Autor ohne Programmiereerfahrungen besonders schwierig, da hier gute Kenntnisse von Skriptsprachen wie Lingo oder OpenScript gefordert werden.

In Gegensatz dazu sind im MAVA-System keine Schwachestellen wie in oben beschriebenen Autorensystemen vorhanden. Die Erstellung und Verwendung von MAVA-Vorlagen wird durch eine graphisch-interaktive Arbeitsweise erledigt, d.h. visuelle Programmierung durch direkte Manipulation. Die Benutzung der Vorlagen bleibt intuitiv und einfach, und der Autor braucht keine Programmierkenntnisse, um Vorlagen zu erstellen. Trotz der Einfachheit sind die MAVA-Vorlagen sehr flexibel, und der Autor kann immer eigene Vorlage erstellen und einfügen, daher bleiben die MAVA-Vorlagen unabhängig von bestimmten Anwendungsgebieten, und die Funktionalität von MAVA-Vorlagen ist unbegrenzt. Das Dokumentmodell von MAVA-System garantiert die Trennung der Struktur von der Medienelementen der Dokumenten. Also bleiben die strukturellen Informationen in MAVA-Vorlagen auch erhalten.

Die Implementierung der MAVA-Vorlagen hat die Kompatibilität zu dem vorhandenen MAVA-Editor berücksichtigt. Die Funktionalität der Vorlagen wurde in den MAVA-Editor integriert, ohne die vorhandenen Funktionalität des MAVA-Editors zu ändern. Der MAVA-Editor wurde erweitert, und die Benutzungsoberfläche vom MAVA-Editor wurde zur Unterstützung der Vorlagen angepaßt. Damit kann der Autor immer noch mit dem erweiterten Editor arbeiten, ohne etwas über Vorlage zu wissen.

Um die Benutzerfreundlichkeit des MAVA-Editors und damit die graphisch-interaktive Arbeitsweise weiter zu verbessern und zu beschleunigen zu können, wären folgende erweiternde Funktionalität denkbar bzw. wünschenswert:

- AutoInhalt-Assistent

Wie im Kapitel 3 untersucht, bietet Powerpoint einen AutoInhalt-Assistent, um dem Autor bei der Erstellung von Multimedia-Dokumenten weiter zu helfen. Der Autor stellt die rich-

tigen Eigenschaften vom Dokument in ein paar Dialogfenstern zusammen, und der Auto-Inhalt-Assistent wird je nach der Eingabe vom Autor die entsprechenden Vorlagen für das Dokument auswählen. Damit braucht der Autor nicht jedes Mal selbst nach der Vorlagen zu suchen. Das ist besonders nützlich, wenn der Autor die vorhandenen Vorlagen vom Autorensystem nicht gut kennt.

Wenn später eine große Menge von Vorlagen im MAVA-Editor vorhanden sind, braucht der Autor viel Zeit, sich zuerst mit den vorhandenen Vorlagen vertraut zu machen, um sie richtig benutzen zu können. Eine AutoInhalt-Assistent im MAVA-Editor hilft dem Autor, die Vorlagen benutzen zu können, ohne viel Kenntnisse von MAVA-Vorlagen zu besitzen. Damit geht die Benutzung der Vorlagen im MAVA-Editor noch schneller und intuitiver.

- Vorschau von Vorlagen

Um die richtige Vorlage aus einer großen Menge der Vorlagen schnell auszuwählen, ist die Vorschau-Funktionalität wünschenswert. Damit kann der Autor eine schnelle Übersicht von Vorlagen bekommen, besonders wenn die Vorlage um das Layout von Präsentationsseiten geht. Für manche Vorlagen, besonders die Vorlagen mit zeitlichen Operatoren, ist ein Vorschau-Bild weniger nutzbar. Trotzdem ist es sinnvoll, die Vorschau-Funktionalität für alle Layout-Vorlagen anzubieten, um die Benutzerfreundlichkeit von MAVA-Vorlagen zu erhöhen.

Literaturverzeichnis

[Bill00] Bill Joy, Guy Steele, James Gosling, Gilad Bracha; >>The Java Language Specification, Second Edition<<, Addison-Wesley Verlag, 2000.

[Catl91] Karen Smith Catlin, L. Nancy Garrett; "Hypermedia Templates: An Author's Tool", Proceedings of the third annual ACM conference on Hypertext, 1991, Pages 147-160.

[Groß99] Groß Rod, "Speicherung eines erweiterbaren Dokumentenmodells in MAVA", Diplomarbeit an der Universität-Stuttgart, Informatik, 1999.

[Gust97] Gustavo Rossi, Daniel Schwabe and Alejandra Garrido; "Design Reuse in Hypermedia Applications Development", Proceedings of the eighth ACM conference on Hypertext, 1997, Pages 57 - 66.

[Haus01] Jürgen Hauser and Kurt Rothermel; "Document Specification and Dissemination with an Extensible Multimedia System", Proceedings of Interactive Distributed Multimedia Systems and Telecommunication Services (KiVS2001), Springer, 2000.

[Haus00] Jürgen Hauser; "Multimedia Authoring with MAVA", Proceedings of the 8. International Conference on Digital Documents and Electronic Publishing (DDEP2000), München, Springer Verlag , 2000.

[Haus99] Jürgen Hauser; "Realization of a Extensible Multimedia Document Model", in: Correia, Nuno / Chambel, Teresa / Davenport, Glorianna (ed.), Multimedia '99 - Media Convergence: Models, Technologies and Applications, 1999.

[Lynd93] Lynda Hardman, Guido van Rossum and Dick C. A. Bulterman; "Structured multimedia authoring", Proceedings of the first ACM international conference on Multimedia, 1993, Pages 283 - 289.

[Marc98] Marc Nanard, Jocelyne Nanard and Paul Kahn; "Pushing Reuse in Hypermedia Design: Golden Rules, Design Patterns and Constructive Templates ", Proceedings of the ninth ACM conference on Hypertext and hypermedia, 1998, Pages 11 - 20.

[Mark98] Mark Bernstein; "Patterns of Hypertext", Proceedings of the ninth ACM conference on Hypertext and hypermedia, 1998, Pages 21 - 29.

[Mich96] Michael D. Rabin and Michael J. Burns; "Multimedia authoring tools", Proceedings of the CHI ’96 conference companion on Human factors in computing systems: common ground, 1996, Pages 380 - 381.

[Nana96] S. Fraïssé, J. Nanard, M. Nanard; "Generating Hypermedia from Specificationjs by Sketching Multimedia Templates", Proceedings of the fourth ACM international conference on Multimedia, 1996, Pages 353 - 364.

[Notz99] Notz Martin; "Entwicklung eines Editors für MAVA-Dokumente", Diplomarbeit an der Universität-Stuttgart, Informatik, 1999.

[UTF-8] Unicode Transformation Format, 8-bit encoding form, siehe Website http://www.unicode.org.

[W3C] World Wide Web Consortium (W3C), siehe Website http://www.w3c.org.

[V3D2] DFG-Schwerpunktprogramm "Verteilte Verarbeitung und Vermittlung Digitaler Dokumente" (V3D2). Siehe Webseit http://www.dfg.de/foerder/formulare/1_53.htm.

[XML] Extensible Markup Language (XML), siehe Website http://www.w3.org/TR/2000/REC-xml-20001006.