Formbildung durch Muster: Leitmotive und Objekte
I: Ein Werkzeug zur leitmotivischen Analyse von *Der Ring des Nibelungen*

Diplomarbeit

am Arbeitsbereich Softwaretechnik
des Fachbereichs Informatik
der Universität Hamburg

Andreas Kornstädt

Inhaltsverzeichnis

Vorwort

Diese Arbeit ist keine Einzelleistung. Sie wäre ohne die Hilfe all derer, die mich im Laufe der Monate unterstützt haben, nicht möglich gewesen. Für diese Hilfe möchte ich mich herzlich bedanken.

Vielen Dank,

Heinz,
daß Du der Arbeit von Anfang an so offen gegenüberstandest und (fast) immer sofort für ein Gespräch Zeit gefunden hast,

Herr Seifert,
für Ihre enthusiastische Unterstützung,

Herr Floros und *Herr Petersen*,
für die viele Zeit und die Geduld, die Sie für mich erübrigt haben,

Ms. Selfridge-Field,
für Ihre außerordentliche Hilfe und Ihre Bemühungen in Stanford,

Mr. Grey,
daß Sie während Ihres so kurzen Deutschlandaufenthalts noch so viel Zeit für mich gefunden haben,

Mr. Huron,
für das fabelhafte *Humdrum*-Programmpaket,

Mr. Smith,
für die wertvollen *SCORE*-Informationen aus erster Hand,

Herr Süß,
daß Sie mir die Schott'schen Partiturdaten so bereitwillig zur Verfügung gestellt haben,

Herr Chen,
für das *SCORE*-Programm,

David Stifel,
für die selbsteingespielten MIDI-Daten,

Mr. Newcomb, *Mr. Renwick*, *Dr. Voß* und *Mr. Schottstaedt*
für Ihre Hilfe, Unterstützung und Ermutigungen

Herr Zannos, *Herr Göbel*, *Herr Furukawa* und *Mr. Page*,
daß Sie die Verbindungen geschaffen haben, ohne die ich bestimmt nicht so schnell ans Ziel gekommen wäre.

Mein besonderer Dank gilt schließlich noch *Herrn Hansmann*, ohne dessen anfängliche Ermunterung ich niemals so rasch den Mut zu dieser Arbeit gefunden hätte.

Hamburg im April 1995

1 Einleitung

Die von dem Architekten Christopher Alexander in seinem Buch *„Notes on the Synthesis of Form“*[Ale64] postulierten Entwurfsprinzipien für große Systeme scheinen mir von einer solchen Allgemeingültigkeit zu sein, daß sie den methodischen Kern zur Synthese von solch unterschiedlichen und unabhängig von Alexander entwickelten Systemen wie Software und Opern bilden. Dies tritt meiner Meinung nach besonders deutlich bei der Werkzeug-Aspekt-Material-Methode (WAM) zur objektorientierten Anwendunsgentwicklung und der von Richard Wagner geschaffenen Leitmotivtechnik hervor.

Um diese These zu stützen, müßte neben den existierenden Beschreibungen der WAM-Prinzipien auch eine leitmotivische Analyse z.B. von Wagners Operntetralogie *„Der Ring des Nibelungen“* vorhanden sein, um die vermuteten Parallelen aufzeigen zu können. Diese Analyse gilt unter Musikwissenschaftlern jedoch als „monumental task“[Coo68] und konnte aufgrund des gewaltigen Umfangs des Werkes bisher nicht durchgeführt werden.

Der erste Schritt ist daher die Entwicklung eines Softwarewerkzeugs, das Musikwissenschaftlern durch geeignete Unterstützung die Anfertigung genau dieser leitmotivischen Gesamtanalyse erlauben soll. Dieses Werkzeug und seine Entwicklung nach WAM sind Gegenstand dieser Arbeit.

Vor diesem Hintergrund stellte sich heraus, daß WAM, obwohl ursprünglich für qualifizierte Büroeinzelplatzarbeit ersonnen, eine viel universellere Methode ist und daß eine die Methode adäquat unterstützende Entwicklungsumgebung zwar noch fehlt aber verhältnismäßig leicht eingerichtet werden könnte. Ferner ergab es sich, daß durch die im Rahmen dieser Arbeit gewonnen speziellen Programme und Methoden eine gute Grundlage für allgemeine rechnergestützte musikwissenschaftliche Analysen geschaffen werden konnte.

In Kapitel 2 werden die Alexanderschen Prinzipien aus seinen allgemeinen Entwurfsvorschriften herausdestilliert und dann auf den Strukturbildungsprozeß bei Wagner und bei WAM bezogen.

Kapitel 3 schildert den Entwicklungsprozeß nach WAM aus drei verschiedenen Blickwinkeln. *Methodisch* wird festgestellt, daß mit einer leicht modifizierten WAM auch die nicht als Sachbearbeitertätigkeit einzustufende Leitmotivanalyse unterstützt werden kann und demzufolge der Schluß gezogen, daß WAM in Erweiterung der bisherigen Klassifizierung einen adaptierbaren Methodenkern für selbstbestimmte menschliche Arbeit darstellt. Bezüglich der *Entwicklungsdokumente* wird der Mißstand, daß deren bisherige textuelle Repräsentation ihrem stark vernetzten Charakter nicht Rechnung trägt, konstatiert und durch den Entwurf und die teilweise Umsetzung eines Hypertextdokumentationssystems behoben. Aus *fachlicher* Sicht werden die Schwierigkeiten des alten manuellen Analysevorgehens und deren Behebung durch die neue werkzeugunterstützte Analyseumgebung dargestellt.

Kapitel 4 beschäftigt sich mit den technischen Aspekten der Arbeit. Zum einen werden das den Analysewerkzeugen zugrundeliegende *Humdrum* und das Konvertierungsprogramm *scr2hmd* vorgestellt. Ohne die umfassende Funktionalität des ersteren und die Versorgung mit Analysedaten durch das letztere wäre die Leitmotivanalyse nicht so rasch zu unterstützen gewesen. Zum zweiten wird die neuartige Art der Kapselung des *Humdrum*-Subsystems beschrieben und dann zunächst auf andere musikwissenschaftliche Analyseaufgaben und schließlich auf allgemeine WAM-Problematiken ausgeweitet.

Den Abschluß bildet Kapitel 5 mit einer kritischen Würdigung des Erreichten sowie einem Ausblick auf die Möglichkeiten, die sich aus dem erfolgreichen Abschluß dieser Arbeit ergeben.

2 Die Alexanderschen Entwurfsprinzipien

Große Systeme bedürfen der Strukturierung, um übersichtlich zu bleiben. Die Frage ist nur: Wie? Viele Strukturierungsmethoden verlieren im Laufe des Entwurfs den Bezug zum eigentlichen Problem und liefern daher keine befriedigenden Lösung desselben. Dieses Problem sieht auch Christopher Alexander und versucht, einen Weg aufzuzeigen, auf dem möglichst problemgerechte Strukturen gefunden werden können. Dieser Weg fußt auf gestalterhaltenden Analyse- und Syntheseschritten, die streng phasenorientiert durchlaufen werden. Während diese strenge Phasenorientierung erwiesenermaßen nur für einen sehr begrenzten Problembereich einsetzbar ist, so handelt es sich bei den Methodenteilen, die der Erhaltung der Gestalt dienen, meiner Meinung nach um allgemeine Prinzipien, die den Kern fast aller guten Verfahren in allen Disziplinen darstellen, die Probleme durch Analyse und Synthese zu lösen versuchen. Zu diesen zähle ich auch Wagners Leitmotivtechnik und die WAM-Methode, die Gegenstand der folgenden Kapitel dieser Arbeit sind.

Zu Beginn dieses Kapitels werden Alexanders Ansichten zum Verhältnis von Inhalt und Form geschildert, auf denen sein Verfahren fußt. Darauf aufbauend folgt eine Beschreibung des Verfahrens selbst in algorithmischer Form, begleitet von einem graphischen Beispiel. In der anschließenden Kritik wird der phasenorientierte Verfahrensteil als für die Methode nicht elementar identifiziert und lediglich der gestalterhaltende Kernteil unter der Bezeichnung „Alexandersche Entwurfsprinzipien“ zum Gegenstand der weiteren Überlegungen gemacht. Anhand dieser Prinzipien werden Qualitätsklassen für Strukturierungsverfahren im Allgemeinen abgeleitet und dann die oben erwähnte These aufgestellt, die alle auf den Alexanderschen Prinzipien aufbauenden Verfahren der höchsten Qualitätsstufe zuordnet. Da die These im Rahmen dieser Arbeit nicht überprüft werden kann, wird schließlich der Beweisansatz für Leitmotivtechnik und WAM-Methode kurz skizziert.

Christopher Alexander ist ein amerikanischer Architekt. Offensichtlich verärgert über die überbordende Betonung der Kreativität bei einigen Kollegen, die in seinen Augen nicht mehr zu problemgerechten architektonischen Lösungen sondern zu willkürlichen Konstruktionen führt, hat er die Grundlagen eines in seinen Augen guten Entwurfs in seinem Buch *„Notes on the Synthesis of Form"*[1][Ale64] niedergeschrieben. Er beruft sich dabei nicht explizit auf andere Theorien, jedoch spielt das Verhältnis von Form und Inhalt bei ihm eine zentrale Rolle.

Laut Alexander müsse der Inhalt die Form bestimmen und nicht vermeintliche architektonische Geniestreiche oder das starre Verhaften in alten Formen. Der Grund für die Flucht in eine dieser beiden Möglichkeiten liegt in seinen Augen in der Unfähigkeit einiger Architekten begründet, den immer komplexer werdenden Anforderungen beim Entwurf gerecht zu werden.[2] Mit seiner Entwurfsmethode versucht er, ein möglichst einfaches Verfahren aufzuzeigen, mit dem problemgerechte architektonische Lösungen erreicht werden können. Die Kreativität des Architekten solle dabei nicht ausgeschaltet sondern auf die tatsächlichen Probleme des Entwurfs gelenkt werden.

---

[1] Die von Alexander in späteren Werken *wie „The Timeless Way of Building“*[Ale79] und *„A Pattern Language“*[AIS79] aufgestellten Thesen finden in dieser Arbeit keine Berücksichtigung.

[2] vgl. [Ale64, S.11]

Alexanders Grundannahme ist, daß im Rahmen des Machbaren jedes Problem nahezu optimal gelöst werden kann, d.h. zu jedem Inhalt die passende Form gefunden werden kann, wenn das Problem nur genau bekannt ist. Entwürfe scheiterten zumeist daran, daß der Entwerfer das Problem nicht genau genug erfaßt hat. Um ein Problem griffiger beschreiben zu können, rät er deshalb zunächst zu einer negativen anstelle einer positiven Problembeschreibung:

Eine positive Problembeschreibung ist eine Liste von sinnvoll erscheinenden Lösungskomponenten („Wir brauchen...“). Beschreibungen dieser Art müssen in seinen Augen immer unvollständig bleiben, da jedes Problem - je nach Standpunkt- eine unendliche Anzahl von Ausprägungen habe, die niemals vollständig angegeben werden könne. Ohne eine vollständige Charakterisierung des Problems könne aber auch keine optimale Lösung gefunden werden, weshalb eine positive Problembeschreibung nicht sinnvoll sei.

„Wir brauchen einen neuen Menüeintrag“

„Verschreiben Sie mir bitte eine Flasche Hustensaft“

Eine negative Problembeschreibung besteht aus einer Mängelliste, die die Abweichungen des momentanen Zustands vom optimalen, mängelfreien Lösungszustand enthält. Dadurch wird das Problem nicht wie im positiven Fall durch eine willkürliche Auswahl von Aspekten aus einer unendlichen Liste ausgedrückt, sondern als sein eigentlicher Kern: Die Abweichung vom Optimum.

„Unser Abrechnungsverfahren hat sich geändert“

„Ich habe seit kurzem ein merkürdiges Kratzen im Hals“

Mit Hilfe einer möglichst vollständigen negativen Problembeschreibung kann das Alexandersche Entwurfsverfahren, von ihm „Programm“ genannt, durchgeführt werden. Es besteht aus einer Vorbereitungs- und drei Hauptphasen, von denen die erste und die letzte zyklisch wiederholt werden. In der Vorbereitungsphase wird das Problem als Menge von Abweichungen vom Optimum (***a***s) dargestellt, wobei die ***a***s durch gewichtete Kanten, die das Verhältnis(***v***) der as zueinander ausdrücken, verbunden werden. Positive Gewichte drücken Komplementarität, negative Konflikt aus. Unverbundene ***a***s beeinflussen sich nicht gegenseitig. In der ersten Phase wird das Problem zyklisch in atomare Subprobleme zerlegt. In der zweiten Phase wird jedes atomare Subproblem einzeln gelöst. In der dritten Phase werden diese Sublösungen (***l***s) zyklisch zur Gesamtlösung verschmolzen. Das besondere des „Programms“ liegt dabei in der Art und Weise, nach der das Problem zerlegt, in Teilen gelöst und wieder zusammengesetzt wird. In jeder Phase soll sichergestellt werden, daß die Struktur des Problems und nicht willkürliche Entscheidungen den Entwurf bestimmen sollen.

Das Programm läßt sich so zusammenfassen:

Definitionen:

Das Gesamtproblem (***GP***) wird als ein ungerichteter Graph ***GP(A,V)*** angesehen. Das Problem wird charakterisiert durch die Menge der Abweichungen vom Optimum (***a***)  und deren Verhältnis (***v***)  zueinander. Die Gesamtlösung (***GL***) ist isomorph zu ***GP***, wobei die ***a***s durch atomaren Teillösungen (***l***) ersetzt werden (***GL(L,V)***).

Vorbereitungen:

- Wähle so viele ***a***s wie möglich.

  Nur was als ein ***a*** identifiziert worden ist, kann auch durch ein ***l*** gelöst werden.

- Wähle die ***a***s so, daß sie alle bekannten Teile des Problems vollständig abdecken.

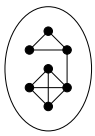
  Lücken in der Problembeschreibung können zu Lücken in der Lösung führen.

- Gib die ***v***s so vollständig wie möglich an und achte darauf, daß es sich wirklich immer um kausale Beziehungen handelt.

  Fehlende oder falsche erkannte Verhältnisse führen zu einer Verzerrung der Problemstruktur.

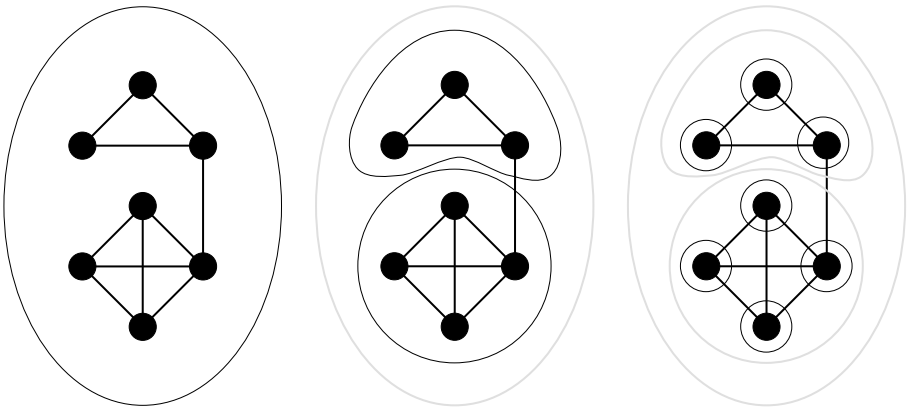
- Versuche, die ***a***s so zu wählen, daß sie von ähnlicher Komplexität sind, wobei gilt: Je kleiner desto besser.

  Große Teilprobleme sind durch versteckte innere Abhängigkeiten schlecht lösbar.
  In kleinen und knapp formulierte ***a***s können solche Abhängigkeiten besser erkannt und vermieden werden.



Zerlegungsphase. Unterteile das Gesamtproblem solange in Teilprobleme, bis alle Teilprobleme atomar sind. Gehe dabei so vor, daß Teilprobleme möglichst stark im Inneren und möglichst schwach mit anderen Teilproblemen verbunden sind.

Durch eine Zerlegung entlang den erkannten Grenzen von Teilproblemen wird die Problemstruktur gewahrt. Eine Zerlegung gemäß problemfremder Konzepte verschleiert sie. Eine gute Zerlegung erkennt man an einer guten Beschreibbarkeit der Teilprobleme.

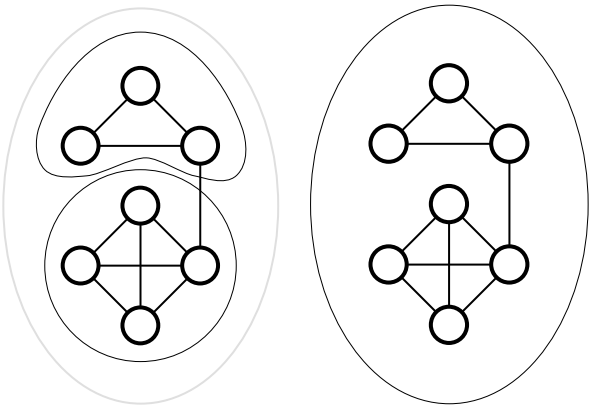


Teillösungsphase. Löse jedes atomare Teilproblem so, daß die Teillösung sich strukturell möglichst eng an das Teilproblem anschmiegt. Die ***l***s sollen ebenso wie die zugrundeliegenden ***a***s keine willkürlich hinzugefügten Elemente enthalten.

> Nicht exakt zum Teilproblem passende Teillösungen bergen ebenso wie unnötige Konkretisierungen die Gefahr einer willkürlichen Lösung. Je konkreter eine Teillösung ist, desto schwieriger ist sie später mit anderen kombinierbar.

Kombinationsphase. Setze die Teillösungen gemäß der in der Zerlegungsphase vorgenommen Unterteilung solange zu größeren Teillösungen zusammen, bis eine Gesamtlösung vorliegt. Die größeren Teillösungen sollen dabei aus den zugrundeliegenden kleineren Teillösungen herleitbar sein.

> Die größeren Teillösungen verzerren das Problem, wenn Elemente der kleineren Teillösungen nicht mit einbezogen werden oder wenn unnötige Konkretisierungen hinzugenommen werden. Die konkretesten Teillösungen sollten am ehesten mit anderen kombiniert werden, da sie zu Beginn noch am flexibelsten sind.



Das „Programm“ gibt einen verführerisch trivial klingenden Weg zur Strukturierung großer Systeme an. Dabei wird die Allgemeingültigkeit des Verfahrens durch die Verquickung von Systemstruktur und Verfahrensstruktur jedoch stark eingeschränkt. Aus der hierarchischen Systemstruktur leitet Alexander eine Phasenorientierung ab, nach der das Problem zunächst hierarchisch von oben nach unten zerlegt und danach die Lösung in umgekehrter Reihenfolge wieder zusammengesetzt wird. Er geht also davon aus, daß jede Phase des Programms, wenn auch mit Zyklen im Phaseninneren, nur einmal durchlaufen wird. Das bedeutet zum Beispiel, daß die Problemgestalt vor Beginn der Teillösungsphase eindeutig und vollständig vorliegen muß und dementsprechend nach einmaligem Durchlaufen der weiteren Synthesephasen das fertige System vollständig spezifiziert werden kann.

Diese Eindeutigkeit und Vollständigkeit kann aber nur bei mathematischen Problemen gewährleistet werden. In allen anderen Bereichen ist die vorgeschlagene negative Problembeschreibung zwar ein gutes Hilfsmittel, um Teilprobleme zu identifizieren, aber Eindeutigkeit und Vollständigkeit können auf diese Weise nicht si-

chergestellt werden, zumal sich das Problem jedem Betrachter unter einem anderen Aspekt darstellen kann.[3] Die initiale Problembeschreibung kann daher nur der Ausgangspunkt eines evolutionären Entwicklungsprozesses sein, in dem nach einmaliger Analyse und Synthese die Ergebnisse kritisch betrachtet werden müssen und darauf aufbauend weitere, neu aufgetauchte oder erst zu diesem Zeitpunkt offensichtlich gewordene Probleme erneut analysiert und zur Grundlage einer verbesserten Synthese gemacht werden. Diese Analyse-Synthese-Kritik-Zyklen können so lange durchgeführt werden, bis eine an intersubjektiven Kriterien festgemachte Übereinkunft der am Entwicklungsprozeß Beteiligten darüber besteht, daß vorerst kein weiterer Verbesserungsbedarf besteht. Eine mathematisch-objektive Feststellung des Entwicklunsabschlusses im Sinne von „quod erat demonstrandum“ kann außerhalb des mathematischen Bereichs nicht erfolgen. Damit scheidet der vorgehensorientierte Anteil des „Programms“ als Grundlage für ein allgemeines Strukturierungsverfahren aus.

Der von seinem vorgehensorientierten Teil befreite strukturelle Kernteil des Verfahrens macht nun keinerlei Aussagen mehr über die zeitliche Abfolge von Entwicklungsschritten. Es handelt sich bei ihm vielmehr um eine Sammlung von Prinzipien, die helfen sollen, bei der Durchführung von Anaylse- und Syntheseschritten die Gestalt des Problems nicht aus den Augen zu verlieren und willkürliche Entwicklungsentscheidungen zugunsten problemgerechter zu vermeiden. Lediglich eine implizite logische Kopplung zwischen den Schritten (z.B. „ein Teillösungsschritt kann nicht erfolgen, bevor das ihm zugrundeligende Problem analysiert worden ist.“) bleibt bestehen.

Dieser Kern des Alexanderschen Verfahrens, der im folgenden als „die Alexanderschen Entwurfsprinzipien“ bezeichnet wird, kann wie folgt angegeben werden (ohne Wiederholung der Definitionen):

Analyse

- **Teilproblemermittlung (A1).** Ermittle ausreichend viele ***a***s.

  Die Problemgestalt besteht aus den ***a***s und deren ***v***s untereinander. Bei sehr wenigen ***a***s ist keine Gestalt erkennbar.

- **Teilproblemgestaltung (A2).** Versuche, die ***a***s so zu wählen, daß sie von ähnlicher Komplexität sind, wobei gilt: Je kleiner desto besser.

  Große Teilprobleme sind durch versteckte innere Abhängigkeiten schlecht lösbar. In kleinen und knapp formulierte ***a***s können solche Abhängigkeiten besser erkannt und vermieden werden.

- **Problemgestaltermittlung (A3).** Gib die ***v***s so vollständig wie möglich an und achte darauf, daß es sich wirklich immer um kausale Beziehungen handelt.

  Die Problemgestalt besteht aus den ***a***s und deren ***v***s untereinander. Fehlende oder falsche erkannte ***v***s führen zu einer Verzerrung der Problemstruktur.

[3] Gryczan zeigt dies am Beispiel der Programmentwicklungsmethode „Strukturierte Programmierung“ in [Gry95, S.16-25].

Synthese

- **Teillösungsermittlung (S1).** Löse ***a***s so durch ***l***s, daß das ***l*** sich strukturell möglichst eng an das ***a*** anschmiegt und die durch ***v***s gegebene Einbettung in das System so gut wie möglich gewahrt bleibt.

  Nicht gut auf die ***a***s abgestimmte ***l***s bergen ebenso wie Vernachlässigung der ***v***s die Gefahr einer willkürlichen, nicht gestalterhaltenden Lösung.

- **Teillösungsgestaltung (S2).** Die ***l***s sollen ebenso wie die zugrundeliegenden ***a***s keine willkürlich hinzugefügten Elemente enthalten.

  Je komplexer ein ***l*** ist, desto schwieriger ist es später mit anderen kombinierbar.

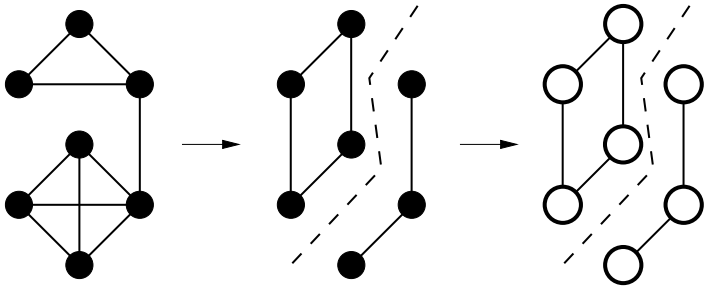
Die Richtlinien versuchen sicherzustellen,

- daß die Gestalt des Problems erkannt werden kann (A1),
- daß die Problemgestalt bei der Durchführung von Analyseschritten (A3) und Syntheseschritten (S1) erhalten bleibt und
- daß die Systemteile übersichtlich und klar beschrieben werden (A2, S2).

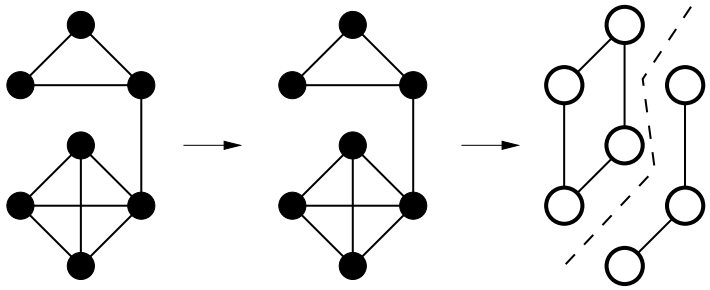
Das streng phasenorientierte „Programm“ ist *eine mögliche* Konkretisierung dieser Richtlinien. Eine andere wäre ein evolutionäres Verfahren, mit dem schrittweise ***a***s ermittelt und darauf basierend ***l***s entworfen werden können.

Daß eine Strukturierung gemäß dieser Prinzipien nicht so einfach ist, beweisen die vielen existierenden Entwurfsmethoden, die entweder bei Analyse- oder bei Syntheseschritten den Bezug zum eigentlichen System verlieren und, angelehnt an problemfremde Konzepte, zu einer nicht problemgerechten Lösung führen. Demnach gibt es drei Qualitätsklassen von Entwurfsmethoden:

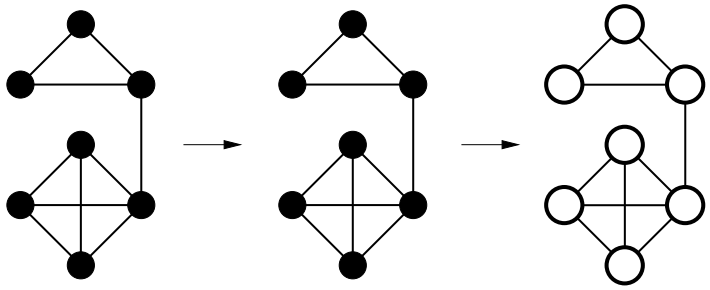
Zur Klasse der Entwurfsmethoden der geringsten Qualität gehören diejenigen Methoden, die bereits bei der Durchführung von Analyseschritten den Bezug zum System, wie es sich den Betroffenen darstellt, verlieren. Ob die Synthese sich auf diese Analyse bezieht oder nicht: Sie kann die Qualität nicht mehr verbessern.



Die Klasse der Entwurfsmethoden von mittlerer Qualität umfaßt Methoden, die zwar ein dem System angemessenes Analyseergebnis liefern, deren Syntheseschritte sich aber nicht problemgerecht auf dieses beziehen. Immerhin kann auf dem Analyseergebnis basierend eine verbesserte Synthese durchgeführt werden.



Zur Klasse der hochwertigen Entwurfsmethoden gehören solche Methoden, die sich über Analyseschritte hinaus auch bei der Durchführung von Syntheseschritten an der Gestalt des Problems orientieren und eine bruchlose Synthese anstreben. Diese Methoden liefern Ergebnisse, die zum Problem passen wie ein Handschuh zur Hand.



Meine These ist nun die folgende:

> ***Die Alexanderschen Entwurfsprinzipien stellen den Kern vieler hochwertiger Entwurfsmethoden in zahlreichen Disziplinen dar, in denen es das Ziel ist, problemgerechte Lösungen auf dem Wege der Analyse und der Synthese zu erzeugen. Leitmotivtechnik und WAM-Methode sind nichts anderes als fachspezifische Ausprägungen dieser Prinzipien.***

Die Prüfung dieser These ist nicht Gegenstand dieser Arbeit. Eine mögliche Beweisführung soll hier deshalb nur äußerst kurz skizziert werden:

Leitmotivtechnik. Im Gegensatz zu vorher üblichen Kompositionstechniken gibt die Struktur des Dramas und nicht abstrakte, rein musikalische Prinzipien die Struktur der Musik vor. Das Analyseergebnis ist das Libretto, das sich nicht an der musikalischen Form der zukünftigen Oper wie z.B. Arie-Rezitativ-Schema sondern am tatsächlichen Verlauf der dramatischen Handlung orientiert. Die Synthese stellt die Vertonung des Librettos dar; an ihrem Ende steht die Partitur. In der Analyse werden die dramatischen Handlungselemente identifiziert und miteinander in Beziehung gesetzt. In Syntheseschritten wird jedem dramatischen Handlungselement ein

kurzes Leitmotiv zugeordnet und die Leitmotive unter Wahrung der dramatischen Gestalt zu einer unendlichen Melodie verschmolzen.

WAM. Im Gegensatz zu vorher üblichen Verfahren geben die Arbeitsabläufe und Gegenstände der bestehenden Anwendungswelt und nicht fachfremde Konzepte wie z.B. „Trennung in Daten und Funktionen“ die Struktur des zukünftigen Softwaresystems vor. Analyseschritte sind die Erstellung des Glossars und der Szenarios. Die Erstellung der Systemvisionen und der Prototypen stellen Syntheseschritte dar, die in ein Softwareprodukt münden. In den Analyseschritten werden in aus Interviews, Dokumenterweiterungen und deren Evaluation bestehenden Autor-Kritiker-Zyklen Glossar und Szenarios erstellt. In Syntheseschritten werden zunächst aufbauend auf den Szenarios möglichst strukturgleiche Systemvisionen und dementsprechende Prototypen erstellt und in Autor-Kritiker-Zyklen solange kritisiert und modifiziert, bis die Produktreife erreicht ist. Die Reihenfolge von Analyse- und Syntheseschritten ist nicht fest vorgegeben, sondern wird von den am Entwicklungsprozeß beteiligten Personen gestaltet.

Da eine vollständige Auswertung des Verhältnisses von Alexanderschen Entwurfsprinzipien, Leitmotivtechnik und WAM im Rahmen dieser Arbeit nicht stattfinden kann, wird der Geschlossenheit des Textes zuliebe gänzlich darauf verzichtet. Um jedoch das Augenmerk auf auffällige Parallelen zu lenken, befinden sich am Rande der folgenden Kapitel Anmerkungen, aus denen die Art der Ähnlichkeit hervorgeht. Eine vermutete Parallele zur Art und Weise der Teilproblemgestaltung (A2) sieht z.B. wie folgt aus:

A2

keine willkürlichen Elemente

Art und Weise. Dadurch, daß die Szenarios auf der Grundlage von Interviews mit den Anwendern erstellt werden und solange diskutiert werden, bis Anwender ihre Arbeitssituation richtig beschrieben sehen, ist sichergestellt, daß das Analyseergebnis auch tatsächlich mit dem Anwendungssystem korrespondiert und daß die Analyse keine Objekte enthält, die die Entwickler aus ihrer Sicht in die Anwendungswelt hineininterpretiert haben. Diese Strukturtreue bleibt auch bei der Erstellung der Systemvisionen gewahrt, die sich gestaltlich eng an die Szenarios anschmiegen und anfangs lediglich Objekten des Anwendungssystems Objekte im antizipierten Softwaresystem gegenüberstellen.

3 Der Entwicklungsprozeß nach WAM

Softwaresysteme sollten so aufgebaut sein, daß sie die Anwender in ihrer Tätigkeit unterstützen und für Anwender und Entwickler verständlich sind. Auf dieser Grundlage wurde von Heinz Züllighoven und Guido Gryczan in Zusammenarbeit mit Reinhard Budde, Karl-Heinz Sylla und Dirk Bäumer die Softwareentwicklungsmethode WAM ersonnen. Daß WAM bisher nur in einem kleinen Entwicklerkreis eingesetzt wird, liegt meiner Meinung nach an zwei Gründen:

1. WAM hat sich noch nicht als Universalmethode bewiesen. Obwohl WAM für das weite Feld der „qualifizierten und kooperativen menschlichen Tätigkeit“ geeignet sein soll, wurde sie bisher nur zur Unterstützung der kleinen Untermenge von Sachbearbeitertätigkeiten in Banken und Versicherungen eingesetzt.
2. Außer einigen Klassenbibliotheken gibt es keine Entwicklungsumgebung für WAM.

Durch meine Arbeit im Leitmotivanalyse-Projekt bin ich zu der Überzeugung gelangt, daß beide Tatsachen sich lediglich aus der WAM-Entstehungsgeschichte ergeben, nicht aber methodenbedingte Notwendigkeiten sind: Die musikwissenschaftliche Tätigkeit der Leitmotivanalyse konnte mit einer leicht modifizierten WAM ebenso gut unterstützt werden wie die eines Sachbearbeiters. Außerdem konnte eine Entwicklungsumgebung konzipiert und teilweise verwirklicht werden. Somit ergeben sich die beiden Thesen, deren Bestätigung im Mittelpunkt dieses Kapitels steht:

1. ***WAM ist eine Universalmethode für qualifizierte und kooperative menschliche Tätigkeit.***
2. ***Eine geeignete Entwicklungsumgebung für WAM fehlt, ist aber in greifbare Nähe gerückt.***

Im Projekt fanden drei ineinander verwobene Entwicklungsprozesse statt: WAM wurde den Gegebenheiten des Anwendungsgebiets angepaßt und stellte damit seinen Universalmethodencharakter unter Beweis; die zur Dokumenterstellung verwendete Entwicklungsumgebung wurde geschaffen; und das Produkt Leitmotivanalysewerkzeug wurde prototypisch erstellt. Dementsprechend ist dieses Kapitel in die drei inhaltlichen Hauptkapitel „Methode“, „Dokumente“ und „Anwendungsgebiet“ gegliedert. In jedem dieser Hauptkapitel wird zunächst die Situation zu Projektbeginn und dann der Erkenntnisstand bei Projektende beschrieben. Den Zwischenteil zu allen Hauptkapiteln bildet eine Beschreibung des Entwicklungsprozesses selbst.

3.1 Ausgangssituation

3.1.1 Methode

Die Werkzeug-Aspekt-Material-Methode (WAM)[4] wird als eine Methode zur Erstellung von Softwaresystemen zur Unterstützung qualifizierter und kooperativer menschlicher Tätigkeit beschrieben,[5] wurde aber bisher meistens[6] auf existierende Sachbearbeiteraufgaben angewendet, für die es schon zuvor partielle Softwarelösungen -meist ablausteuernder Natur- gab.

Die dieser Arbeit zugrundeliegende Erstellung eines Werkzeugs zur leitmotivischen Analyse stellt demzufolge in dreierlei Hinsicht methodisches Neuland dar:[7] Zum ersten stand der zu modellierende Arbeitsablauf nicht (mehr) eindeutig fest und mußte zunächst rekonstruiert werden. Zum zweiten existierte bisher überhaupt keine Rechnerunterstützung für die Leitmotivanalyse, so daß auf Erfahrungen der Musikwissenschaftler mit anderer Software nicht zurückgegriffen werden konnte. Zum dritten war die Entwicklung die erste Anwendung von WAM auf ein geisteswissenschaftliches Gebiet überhaupt.

Daß alle diese Punkte keine Schwierigkeiten verursachten, die nicht im Rahmen von WAM gelöst werden konnten, zeigt, daß WAM tatsächlich nicht nur eine Sachbearbeiter-Spezialmethode ist, sondern das die These

WAM ist eine Universalmethode für qualifizierte und kooperative menschliche Tätigkeit

zutrifft. Da WAM im Leitmotivanalyse-Projekt wie auch im Gebos-Projekt spezielle Anpassungen an das Anwendungsgebiet erfahren hat, wird zudem die zusätzliche These aufgestellt, daß WAM weder eine feste Methode noch ein abstrakter Methodenrahmen ist:

WAM ist ein adaptierbarer Methodenkern für selbstbestimmte menschliche Tätigkeit

Zu Beginn wird WAM in ihrer bisher verwendeten Form und unter Ausklammerung der technischen Aspekte, die nicht Gegenstand dieser Arbeit sind, vorgestellt. Im Anschluß daran werden die Beweise zur Stützung der beiden Thesen skizziert und schließlich in Kapitel 3.3.1 auf der Grundlage des in Kapitel 3.2 geschilderten Entwicklungsprozesses bestätigt.

WAM ist eine Methode zur bruchlosen Unterstützung des objektorientierten Softwareentwurfs. Sie basiert auf einem Leitbild, das durch dazu passende Entwurfsmetaphern konkretisiert und durch Entwurfsmuster realisiert wird. Außerdem beschreibt sie eine evolutionäre Vorgehensweise, nach der der Entwicklungsprozeß gestaltet werden soll. Leitbild, Metaphern und Muster sowie die Vorgehensweise werden im Folgenden vorgestellt.

---

[4] vgl. [Gry95, S.3]
[5] vgl. [Gry95, S.219]
[6] Weitere Bereiche sind der Einsatz von WAM zur Automatisierung eines medizinischen Großlabors und zur Erstellung von Telephoniesystemen. Über die noch laufenden Projekte liegen bisher nur wenig verwertbare Informationen vor.
[7] Gemäß der Gebos(**Ge**nossenschaftliches **B**ürokommunikations- und **O**rganisations**S**ystem)-Klassifikation ein „High Risk Projekt“[GZ95, S.21f]

Leitbild, Entwurfsmetaphern und -muster

Ziel des Softwareentwicklung ist es, durch Analyse des Anwendungsbereichs und Synthese der dabei gewonnenen Erkenntnisse ein Anwendungssystem zu schaffen, das die Schwächen des bisherigen Systems beseitigt ohne die Anwender vor neue rechnerbedingte Probleme zu stellen. Für Analyse und Synthese wurden im Laufe der Geschichte der Informatik zahlreiche Methoden entwickelt, die dieses Ziel unter oft wechselnden Maximen -von Korrektheit über Sicherheit bis zu Anwenderfreundlichkeit- zu erreichen suchten. Dieser Wechsel des Hauptaugenmerks sollte jedoch nicht als willkürliche Laune der Entwickler angesehen werden, die, einer Mode folgend, alle bisherigen Erkenntnisse verwarfen, nur um einem neuen Ideal zu folgen. Vielmehr sollte jeder Methodenwechsel einen Übergang auf ein höheres Niveau darstellen, das größtenteils die Errungenschaften der Vorstufen beinhaltet. Der Wechsel sollte erst dann eintreten, wenn die Entwicklung auf der bisherigen Stufe als weitgehend problemlos angesehen werden kann: solange Programme nicht mit Gewißheit korrekt zum Laufen gebracht werden konnten, waren Gedanken über Rechnersicherheit und Anwenderfreundlichkeit Zukunftsmusik[8]. Der letzte erfolgte Wechsel war der von der strukturierten Programmierung zur objektorientierten Programmierung.

Die strukturierte Programmierung entstand als Versuch zur Behebung der sogenannten Softwarekrise der 60er Jahre. Bis dahin wurden Programme meist „aus einem Guß“ von einem oder wenigen Programmierern erstellt, und waren für andere Personen so gut wie undurchschaubar. Im Rahmen der strukturierten Programmierung wurden Anwendungssysteme meist schrittweise in Teilprobleme zerlegt, die dann nach Lösung und Umsetzung der atomaren Teilprobleme in Funktionen und Prozeduren zu einem aus Funktionen und Moduln bestehenden Programmsystem zusammengesetzt wurden.[9] Die Gestaltung der Programmsysteme orientierte sich dabei an der Vorstellung, daß die menschlichen Tätigkeiten im Rahmen des bisherigen Anwendungssystems prinzipiell auch von Maschinen ausgeführt werden könnten und die (potentiell fehlerhaften) menschlichen Interaktionen mit dem Rechner auf Datenein- und ausgabe reduziert werden sollten.[10]

Alexandersches „Programm“ (S.5-6)

Der Schwachpunkt dieser Methode liegt in der impliziten Ausrichtung an mathematisch-technischen Problemen, die unreflektiert auf Probleme übertragen wurde, in deren Zentrum Menschen stehen. Zum einen lassen sich solche Anwendungssysteme nur gewaltsam in Funktions- bzw. Prozedurform pressen. Zum anderen wird aus der Tatsache, daß Maschinen Aufgaben im rechnerischen Bereich schneller, exakter und im größerem Umfang ausführen können als Menschen, fälschlicherweise der Schluß gezogen, daß Maschinen den Menschen auch in sehr vielen anderen Bereichen überlegen sind und dementsprechend die Steuerung von Aufgaben am besten auf Maschinen zu übertragen sei. Diese streng ablaufsteuernde Sichtweise führte zum Entwurf von Softwaresystemen, die den Anwendern entweder keine Möglichkeiten zur Individualisierung der Arbeitsabläufe ließen oder deren Umfang durch die Antizipation aller möglichen Handlungsabläufe riesig anwuchs. An der Unfähigkeit zur Lösung dieser Probleme scheiterte die strukturierte Programmierung.

---

[8] Der Begriff wurde ursprünglich spöttelnd auf Wagners Musik angewendet.[Büc91, S.170]
[9] vgl. [LdB93, S.702]
[10] vgl. [Gry95, S.36]

S1

Lösungs-struktur orientiert sich an Problem-struktur

Die objektorientierte Programmierung behält nun die Vorteile der strukturierten Programmierung (z.B. Versuch der Widerspieglung des Anwendungssystem im Softwaresystem, genaue Schnittstellenspezifikationen[11]) in geläuterter Form bei, erlaubt es aber, Objekten der Anwendungswelt Objekte im Programm gegenüberzustellen. Dies bedeutet vor allem im Bürobereich, in dem Gegenstände statt Formeln im Vordergrund stehen, einen großen Vorteil. Fanden sich Bestandteile einer Ablage früher in vielen verschiedenen Prozeduren wieder, so können diese nun gebündelt in einem Objekt dargestellt werden. Auf diese Weise können die Entwickler leichter den Überblick über das System behalten und dem Anwender kann es in einer ihm bekannten Form präsentiert werden.

Objektorientierte Programmierung ermöglicht es den Entwicklern also, ein Analysergebnis, dessen Bestandteile Objekte des Anwendungsbereichs sind, bruchlos in Objekte eines Softwaresystems zu überführen - aber eine Garantie für eine gute Umsetzung ist das alleine nicht. Wenn z.B. die vom Entwickler in der Analyse ermittelten Objekte nicht mit denen übereinstimmen, die die Anwender für wichtig halten oder wenn die Anwender bei der Beschreibung ihrer Arbeit einige Objekte schlichtweg vergessen, weil ihnen deren Verwendung wegen der großen Häufigkeit gar nicht mehr bewußt wird, dann kann auch eine sehr gute Synthese diese Analysefehler nicht mehr wettmachen und das resultierende System ist entweder mangelhaft oder sogar völlig unbrauchbar. Erst durch die Verwendung eines Leitbilds, das für die Anwender verständlich und für die Entwickler, die im Anwendungsgebiet meistens nicht zu Hause sind, orientierend ist,[12] kann eine sinnvolle Analyse erfolgen. Das Leitbild soll dabei helfen, in der Anwendungswelt gemachte Beobachtungen besser einordnen zu können und ein Softwaresystem so zu gestalten, daß alle Komponenten -durch Ausrichtung am Leitbild- ein einheitliches Ganzes und keine Ansammlung von Fragmenten ergeben.

Das Leitbild zur WAM ist der Arbeitsplatz für qualifizierte und kooperative menschliche Tätigkeit wie z.B. eine Werkbank oder ein Schreibtisch. Das Leitbild wird durch passende Metaphern (Werkzeuge, Materialien, Automaten, Umgebung), die sowohl eine fachliche als auch eine softwaretechnische Interpretation haben, bildhaft konkretisiert. Die Analyse des Anwendungssystems und die Synthese des zukünftigen Softwaresystems werden also gleichermaßen unterstützt. Gemäß der Verwendung durch die Anwender werden Gegenstände des Anwendungssystems als Materialien, Werkzeuge oder Automaten innerhalb einer Umgebung betrachtet und finden dementsprechend Eingang in das Softwaresystem. Die Umsetzung der Metaphern bei der Systemsynthese wird durch Entwurfsmuster (Werkzeug-Material-Koppelung, Werkzeug-Kombination, Funktions-Interaktions-Trennung, Beobachtermechanismus, dynamische Subwerkzeug-Anbindung, Materialbehälter, Materialverwalter, Werkzeugverwalter, Ereignisverwalter)[13] erleichtert. Die Entwurfsmuster haben einen eher technischen Charakter und werden, da die technischen Aspekte des Entwurfs nicht Gegenstand dieser Arbeit sind, im Folgenden nicht weiter behandelt.

Als **Materialien** werden diejenigen Gegenstände eingestuft, die durch Handlungen des Anwenders zur Erreichung seines momentanen Arbeitsziels verändert oder betrachtet werden. Werkzeuge werden hingegen vom Anwender benutzt, um andere Gegenstände -nämlich die Materialien- zu verändern oder zu betrachten. **Werk-**

---

[11] vgl. [LdB93, S.702]
[12] vgl. [A38, S.24]
[13] vgl. [RZ95, S.19f]

zeuge sind die einzigen Bestandteile des Systems, von dem -durch Handhabung durch die Anwender- Änderungen ausgehen können. Während Materialen meistens aus der Anwendungswelt übernommen werden können, so ist dies bei Werkzeugen komplizierter, da das konkrete stoffliche Werkzeug oftmals zu speziell (Anspitzer) oder zu universell (Hände) ist, als daß es sinnvoll übertragen werden könnte. Deshalb orientieren sich die Softwarewerkzeuge an den dem Werkzeuggebrauch zugrundeliegenden Tätigkeiten und klammern sich nicht an einem bestimmten Gegenstand fest. Für ein Adressensucher-Softwarewerkzeug ist es z.B. unerheblich, ob Adressen im Anwendungssystem per *Hand* aus einem Adreßheft oder einer losen Zettelsammlung, per *Rechner* aus einer Datenbank oder per *Telefon* über einen Auskunftdienst ermittelt worden sind, denn die Tätigkeit ist immer die gleiche: zu einem Namen wird eine Adresse gesucht.

Der Gefahr einer Schwarz-Weiß-Einteilung in entweder Werkzeuge oder Materialien entgehen die Metaphern dadurch, daß sie Werkzeuge und Materialien nicht als absolute sondern als relative Kategorien einführen. Ein Gegenstand kann also in einem Zusammenhang als Werkzeug und in einem anderen als Material dienen. Zum Beispiel kann ein Hammer bis zum Ende seines Fertigungsprozesses als Material der Fertigungsmaschine angesehen werden, obwohl er später als Werkzeug zum Einschlagen von Nägeln dient.

Automaten arbeiten wie auch Werkzeuge auf Materialien. Sie nehmen den Anwendern auf deren Wunsch lästige Routineaufgaben ab, die keiner besonderen Qualifikation bedürfen und überprüfbare Ergebnisse liefern, die die Anwender in ihrer Arbeit unterstützen.[14] Automaten arbeiten eine formalisierte Routine gemäß ihrer Einstellung ohne äußere Einflußnahme bis zum Ende ab. Mögliche Einsatzgebiete für Automaten sind unter anderem Kopieren, Sortieren und Durchführen von zeitgesteuerten Operationen.

Die Kopplung zwischen Werkzeugen und Materialien sowie zwischen Automaten und Materialien wird durch Aspekte ausgedrückt. Da zumeist nicht nur genau ein Werkzeug oder Automat mit nur genau einem Material zusammen verwendet werden kann, wird die Beziehung zwischen diesen beiden Gegenstandstypen durch einen oder mehrere Aspekte abstrahiert. Ein Standardlocher kann z.B. nicht nur eingegangene Bestellformulare im DIN A4-Format lochen, sondern ist auch für Papier in allen anderen Funktionen und Formaten verwendbar; auch Klarsichtfolien und sogar 5¼-Zoll-Disketten können mit ihm gelocht werden: Der Locher kann alles lochen, was „lochbar“ ist. Um nun Werkzeuge und Automaten einerseits und Materialien andererseits effektiv zu entkoppeln, werden die gegenseitigen Beziehungen auf einen Sammlung von Aspekten abgebildet, unter denen die Werkzeuge und Automaten die Materialien sehen, bzw. unter denen die Materialien bearbeitet werden können. Ein Bestellformular besäße z.B. die Aspekte „registrierbar“, „kontrollierbar“, „auswertbar“, „kopierbar“, „weiterleitbar“ und „lochbar“. Ein Locher könnte alle Materialien bearbeiten, die „lochbar“ sind.

Die **Arbeitsumgebung** stellt den Ort dar, an dem *ein* Anwender seine Tätigkeiten ausüben kann und grenzt diesen gegenüber den Arbeitsumgebungen anderer Anwender ab. Alle ihm zur Verfügung stehenden Materialien, Werkzeuge und Automaten sind in der Umgebung vorhanden.[15]

---

[14] vgl. [Gry95, S.219]
[15] vgl. [Gry95, S.152]

Vorgehensweise

Das Leitbild und die Metaphern liefern Entwicklern und Anwendern einen gemeinsamen begrifflichen Rahmen, in dem sie ihre Vorstellungen vom bisherigen und zukünftigen System zum Ausdruck bringen können, aber sie machen keine Aussagen über die Art und Weise, in der die Analyse und die Synthese durchgeführt werden sollen. Soll die Analyse „auf einen Schlag“ durchgeführt, dann entworfen und implementiert und schließlich das fertige System den Anwendern vorgestellt werden? Sollen die bisherigen Arbeitsabläufe durch Interviews, Videoaufnahmen, Fragebögen oder Studium der Dienstvorschriften ermittelt werden? Mit diesen Fragen beschäftigt sich die Vorgehensweise zur WAM-Methode.

Die WAM-Methode versucht, Fehlentwicklungen, die aus einem mangelnden Verständnis zwischen Anwendern und Entwicklern hervorgehen können, durch intensive Kommunikation zwischen diesen beiden Gruppen von vornherein zu vermeiden. Wenn die Entwickler die Anwender nicht verstehen, dann können sie keine gute Analyse durchführen, und wenn die Anwender das Computer-Kauderwelsch der Entwickler nicht verstehen, dann können sie die Syntheseergebnisse nicht richtig bewerten und bemerken die Fehlentwicklungen erst dann, wenn das fertige Softwaresystem vor ihnen steht. Die WAM-Methode basiert daher auf den folgenden zwei Prinzipien, um eine effektive Sofwareentwicklung zu gewährleisten:

1. Die Entwickler müssen die Begriffe und Abläufe des Anwendungsgebiets ausreichend gut verstanden haben, bevor sie mit der Synthese eines Softwaresystems beginnen.
2. Die Anwender müssen in jeder Phase der Synthese in einer für sie verständlichen Weise über deren Verlauf unterrichtet werden, so daß sie Fehlentwicklungen, z.B. aufgrund zuvor nicht erkannter Analysefehler oder plötzlicher Änderungen im Anwendungsbereich, sofort erkennen und zusammen mit den Entwicklern beheben können.

Entwicklungsdokumente. Die Grundlage für die Kommunikation zwischen Anwendern und Entwicklern sind die Entwicklungsdokumente. In ihnen müssen alle Analyse- und Syntheseergebnisse in einer solchen Form festgehalten werden, daß sie als Grundlage für eine gleichberechtigte Diskussion dienen können, d.h. sowohl Anwender als auch Entwickler müssen sie vollständig verstehen können.

A1 + A3
ausr. viele Elemente + Verbindungen

Das **Glossar** dient als Basis aller Kommunikation und enthält ein Verzeichnis der relevanten Fachbegriffe der Anwendungswelt und der Entwicklerwelt sowie der begrifflichen Beziehungen. Das Glossar wird von den Entwicklern nach Angaben der Anwender verfaßt und wird den Anwendern solange zur Korrektur vorgelegt, bis ein gegenseitiges Einverständnis bezüglich der Begriffsbildung besteht. Begriffliche Veränderungen auf Anwender- oder Entwicklerseite finden sofort Eingang in das Glossar.

A1 + A3
ausr. viele Elemente + Verbindungen

Die **Szenarios** werden von den Entwicklern erstellt und beschreiben in ihrer Gesamtheit alle für wichtig befundenen Arbeitsabläufe der Anwendungswelt in der Anwendungssprache und werden auf der Basis von Interviews mit den Anwendern erstellt. Sie enthalten Verweise auf benachbarte Dokumente und bei Bedarf eine Liste der verwendeten Arbeitsgegenstände. Wie auch das Glossar werden sie solange den Anwendern zur Korrektur vorgelegt, bis die Beschreibungen aller relevanten Arbeitsgegenstände und -abläufe so abgefaßt sind, daß die Anwender sie wiedererkennen und die Entwickler sie verstehen können.

Die **Systemvisionen** lehnen sich strukturell eng an die Szenarios an und beschreiben imaginäre Arbeitsbläufe des zuküftigen Softwaresystems in der Anwendungssprache, wobei hier erstmals auch Fachbegriffe der Entwicklerwelt („Eingabefeld“, „Rollbalken“) verwendet werden. Eine Systemvision bezieht sich anfänglich direkt auf ein bestimmtes Szenario, so daß der alte Handlungsablauf und seine Einbettung zunächst beibehalten wird und lediglich einige der verwendeten Arbeitsgegenstände durch Softwarewerkzeuge oder -materialien ersetzt werden. In Systemvisionen können die Entwickler auch eigene Vorstellungen oder Verbesserungensvorschläge einbringen, die dann den Anwendern zur Diskussion vorgestellt werden. Die Systemvisionen dienen vorrangig als Entscheidungshilfe für die Entwickler bei der Gestaltung des Softwaresystems. Da nur textuell und graphisch beschriebene Systeme für Anwender oft schwer bewertbar sind, sollten den Systemvisionen möglichst rasch Prototypen folgen.

S1
Lösungsstruktur orientiert sich an Problemstruktur

Prototypen sind ablauffähige Teilsynthesen des zukünftigen Systems auf Basis von Systemvisionen und dienen wie diese als Entscheidungshilfe für die weitere Entwicklung des Systems. Ein Prototyp wird solange verändert, bis er seine Aufgabe zur allgemeinen Zufriedenheit erfüllt. Basierend auf den Erfahrungen mit dem Prototyp können weitere Systemvisionen und Prototypen in Angriff genommen werden.

S1
Lösungsstruktur orientiert sich an Problemstruktur

Im weiteren Verlauf dieser Arbeit werden Glossar, Szenarios und Systemvisionen entsprechend ihres grundlegenden Charakters zur Kategorie der Text- bzw. textuellen Dokumente zusammengefaßt. Prototypen werden als Softwaredokumente bezeichnet.

Das besondere an der WAM-Methode sind nicht die Dokumente selbst sondern die Art, Weise und Reihenfolge, in der sie erstellt werden:

Art und Weise. Dadurch, daß die Szenarios auf der Grundlage von Interviews mit den Anwendern erstellt werden und solange diskutiert werden, bis Anwender ihre Arbeitssituation richtig beschrieben sehen, ist sichergestellt, daß das Analyseergebnis auch tatsächlich mit dem Anwendungssystem korrespondiert und daß die Analyse keine Objekte enthält, die die Entwickler aus ihrer Sicht in die Anwendungswelt hineininterpretiert haben. Diese Strukturtreue bleibt auch bei der Erstellung der Systemvisionen gewahrt, die sich gestaltlich eng an die Szenarios anschmiegen und anfangs lediglich Objekten des Anwendungssystems Objekte im antizipierten Softwaresystem gegenüberstellen.

A2 + S2
keine willkürlichen Elemente

Σ
alle Merkmale einer hochwertigen Methode (S. 10)

Der Entwicklungsprozeß befindet sich zu keiner Zeit in einer bestimmten Phase im Sinne eines Phasenmodells, aus der es -auch bei zyklischer Wiederholung im Inneren- nur den Weg nach vorne in die nächste Phase gibt. Zwar folgen auch bei WAM Syntheseschritte meistens auf Analyseschritte,[16] jedoch wiederholt sich dieser Vorgang viele Male, so daß das zukünftige System nicht auf einen Schlag nach einmaliger Analyse erstellt wird, sondern in vielen kleinen Schritten aus weiteren Analysen bzw. Kritik an den bisherigen Teilsynthesen hervorgeht. Durch diese Kontrolle am Ende jedes kleineren Schritts sind große Fehlentwicklungen nahezu ausgeschlossen.

[16] vgl. [Gry95, S. 113]

Beweisskizze zur Lockerung der Beschränkungen von WAM

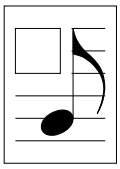
Nachdem WAM in diesem Kapitel vorgestellt worden ist, können mit der Auswertung des Entwicklungsprozesses in Kapitel 3.3.1 Betrachtungen über ihre Anwendbarkeit angestellt werden. Dazu wird jede Komponente der WAM-Beschreibung „Methode für qualifizierte und kooperative menschliche Tätigkeit“ -also „Methode“, „qualifiziert“, „kooperativ“ und „menschliche Tätigkeit“- als Beschränkung angesehen.

In einem ersten Schritt wird gezeigt, daß auch die Leitmotivanalysetätigkeit und die darauf abgestimmte Leitmotivanalyse-WAM innerhalb der Beschränkungen bleiben und somit WAM nicht nur eine Methode zur Unterstützung von Sachbearbeitertätigkeit ist.

Danach werden in einem zweiten Schritt die Beschränkungen selbst in Frage gestellt und gezeigt, daß WAMs Anwendungsgebiet nicht durch die Trennlinie zwischen qualifizierter und unqualifizierter, sondern zwischen selbstbestimmter und extern gesteuerter Tätigkeit begrenzt wird. Außerdem wird gezeigt, daß WAMs Charakter weder der einer reinen Methode noch der eines reinen Methodenrahmens, sondern vielmehr der eines Kerns einer adaptierbaren Methode ist.

Im in Kapitel 3.2 geschilderten Entwicklungsprozeß sind die sich auf WAM beziehenden Ereignisse mit dem Methoden-Symbol hervorgehoben.

3.1.2 Dokumente



Den Entwicklungsdokumenten kommt als Kommunikationsgrundlage im Entwicklungsprozeß nach WAM eine besondere Bedeutung zu. Was in ihnen nicht festgehalten werden kann, geht unweigerlich verloren. Um solche dokumentbedingten Verluste zu verhindern, müssen der Inhalt und die Struktur der Dokumente möglichst gut und schnell den aktuellen Zustand des Projekts widerspiegeln können. Im Laufe der Entwicklung des Leitmotivanalysewerkzeugs hat sich herausgestellt, daß die Anpassung des Dokumentinhalts und der Dokumentstruktur im Kleinen wie im Großen an die speziellen Bedürfnisse des Anwendungsgebiets zwar *prinzipiell* leicht möglich ist, daß aber deren *praktische* Umsetzung in konkrete Software- und vor allem Textdokumente so gut wie überhaupt nicht unterstützt wird. Während für die Implementation Programmierumgebungen und Klassenbibliotheken zur Verfügung stehen, so sucht man ein Pendant seitens der Dokumentverwaltung bisher vergeblich.[17] Textdokumente können nicht einfach erstellt, geändert, verflochten und mit Softwaredokumenten in Verbindung gebracht werden. Ihre Struktur kann nicht übersichtlich dargestellt und untersucht werden. Diese fehlende Entwicklungsumgebung für WAM ist nach meiner Ansicht -neben der bisher selbstauferlegten Beschränkung auf Sachbearbeitertätigkeiten- der größte Hemmschuh für die weitere Verbreitung von WAM: Ein Projektteam auf der Suche nach einer geeigneten Methode muß schon sehr von WAM überzeugt sein, bei der zusätzlich zum eigentlichen Produkt auch noch große Teile der Entwicklungsumgebung selbst erstellt werden müssen.

[17] Dies stellen auch Gryczan[Gry95, S. 218f] und Züllighoven[GZ95, S.37 und S.58] fest.

Zur Behebung dieses Mangels habe ich im Leitmotivanalyse-Projekt die rudimentäre Grundlage eines Hypertextdokumentationssystems erstellt, das meiner Meinung nach mit verhältnismäßig geringem Aufwand zu einer allgemeinen WAM-Entwicklungsumgebung ausgebaut werden könnte, so daß die These

Eine geeignete Entwicklungsumgebung für WAM fehlt, ist aber in greifbare Nähe gerückt

bestätigt werden kann. Die Beschreibung dieser WAM-Umgebung ist, neben der Darstellung der zugrundeliegenden Anpassung der Textdokumente an das Anwendungsgebiet, der Gegenstand dieses Kapitels.

Der Entwicklungsprozeß der Dokumente () und der zu deren Bearbeitung ersonnen Entwicklungsumgebung () werden schrittweise parallel geschildert. In diesem Kapitel wird der Zustand beschrieben, wie er sich vor Beginn des Leitmotivanalyse-Projekts darstellte: Entwicklungsdokumente gemäß *„Objektorientierte Anwendungsentwicklung“*[GKZ93] und eine nichtexistente Umgebung. In Kapitel 3.2 wird hervorgehoben, wie sich Erkenntnisse des Anwendungsgebiets zunächst in inhaltlichen und dann auch strukturellen Änderungen der Dokumente auswirken. In Kapitel 3.3.2 werden diese Änderungen zusammengefaßt: die Dokumentanpassungsbedingungen werden herausgearbeitet und eine ideale Umgebung wird anhand der beobachteten Mängel spezifiziert, realisiert und bewertet.

Ausgangszustand

Dokumente. Die vier in [GKZ93] beschriebenen Basisdokumenttypen der WAM-Methode sind Glossar, Szenario, Systemvision und Prototyp.[18] Sie dienten als Ausgangspunkt für die Dokumente im Leitmotivanalyse-Projekt und werden hier vorgestellt, um die im Projektverlauf durchgeführten Anpassungen zu illustrieren. Die folgende Beschreibung der WAM-Basisdokumente hat ihren Gegenpart in der Beschreibung der speziellen Leitmotivanalyse-Dokumente in Kapitel 3.3.2.

[18] Die Basisdokumenttypen schließen die im Rahmen einzelner Projekte (Gebos [GZ95], Leitmotivanalyse (Kapitel 3.3.2)) eingeführten Untertypen und Verfeinerungen nicht ein.

Die WAM-Basisdokumente

Das **Glossar** enthält

- eine Liste von Fachbegriffen. Über die Reihenfolge und die Gestalt der Begriffe werden keine Aussagen gemacht, obwohl von alphabetischer Ordnung ausgegangen werden kann.

Ein **Szenario** enthält

- eine Liste der verwendeten Werkzeuge und Materialien (die wiederum alle im Glossar aufgeführt sind),
- eine kurze Beschreibung des Kontexts, in dem der Arbeitsvorgang stattfindet (auch als Verweise auf anderen Szenarios),
- eine möglichst epische Beschreibung des Arbeitsvorgangs in der Sprache der Anwendungswelt,
- Verweise auf benachbarte Dokumente (Glossar, angrenzende Szenarios und auf dem Szenario aufbauende Systemvisionen).

Eine **Systemvision** enthält

- den Namen des zugrundeliegenden Szenarios mit Begründung,
- eine Beschreibung der verwendeten Operationen von Softwarewerkzeugen im antizipierten Arbeitsvorgang,
- Verweise auf benachbarte Systemvisionen oder Werkzeuge bezüglich des Arbeitsablaufs,
- eine Bildschirmskizze des Softwarewerkzeugs,
- eine szenarioartige Beschreibung des neuen Arbeitsablaufs,
- eine Beschreibung der Verfahrensvorschriften und/oder Algorithmen, nach/mit denen das Softwarewerkzeug arbeiten soll.

Ein **Prototyp** besteht aus

- einer Auswahl von unter bestimmten Aspekten ausimplementierten Softwarewerkzeugen und -materialien nach Maßgabe ausgewählter Systemvisionen.

Bis auf das Glossar, über das in [GKZ93] kaum Aussagen gemacht werden, besitzen alle Dokumenttypen einen oder mehrere Verweise auf andere Dokumenttypen bzw. auf andere Exemplare des gleichen Typs. Das Vernetzungsschema läßt sich wie folgt darstellen:


A3
ausr. viele Verbindungen

Die Kenntnis der konkreten Vernetzung zwischen den Dokumenten, die eine Untermenge der Menge aller möglichen Vernetzungen bildet, ist ein wichtiges Hilfsmittel um einen Überblick über die Struktur des Anwendungs- bzw. des Softwaresystems zu erhalten.

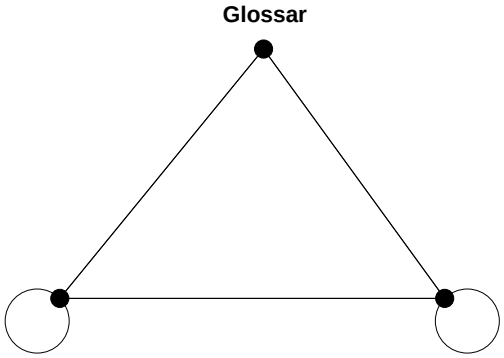


Abb. 3.1.2-1 Dokumentvernetzungsschema

Szenarios und Systemvisionen besitzen Verweise auf das Glossar und andere Dokumente ihres eigenen Typs. Systemvisionen können sich darüberhinaus auch noch auf Szenarios beziehen.

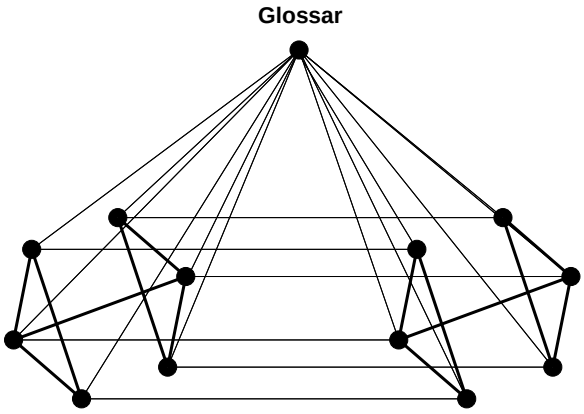


Abb. 3.1.2-2 Beispiel für eine konkrete Dokumentvernetzung

Verweis auf ein Dokument desselben Typs

Verweis auf ein Dokument anderen Typs

In diesem Fall orientieren sich die Systemvisionen strukturell an den Szenarios.

Entwicklungsumgebung. Budde und Züllighoven[19] sowie Gryczan[20] arbeiten die Softwarefabrik und die Softwarewerkstatt als zwei Pole der Gestaltung einer Softwareentwicklungsumgebung heraus. Das Leitbild der Fabrik basiert auf der ablaufsteuernden Sicht, die die am Entwicklungsprozeß beteiligten Menschen als austauschbar, niedrig qualifiziert und potentielle Fehlerquellen ansieht. Eine konsequente Verfolgung dieses Leitbilds führt zu integrierten CASE[21]-Tools, wie sie vor allem im Datenbankbereich Anwendung finden. Der Softwareentwickler wird auf einem von der Umgebung fest vorgegebenen Pfad Phase für Phase von der Analyse des Anwendungsbereichs zum fertigen Softwareprodukt geführt. Die Bedürfnisse der Entwickler werden von den Entwicklern der CASE-Tools antizipiert und festgelegt.

Das Leitbild der Werkstatt orientiert sich an einer Gruppe qualifizierter Handwerker, die ein bestimmtes Ziel vor Augen hat und dementsprechend Werkzeuge und Materialien in einer ihr sinnvoll erscheinenden Weise koordiniert einsetzt, um an dieses Ziel zu gelangen. Mit geeigneter Software kann der Entwicklungsprozeß bestenfalls unterstützt werden, ohne daß jedoch ein Zwang von ihr ausgeht, eine bestimmte Vorgehensreihenfolge zu wählen.

WAM hat von den Softwareentwicklern das gleiche Bild, das es auch von den Anwendern hat: qualifizierte Menschen, die in kooperativer Arbeit eine Aufgabe bearbeiten wollen. Da die Werkstatt eine hervorragende Ausprägung dieses Bildes ist, wird klar, daß das Leitbild der Fabrik, das sich in CASE-Tools manifestiert, kein Vorbild für eine Entwicklungsumgebung für WAM sein kann.

Bisher gibt es keine auf die WAM-Textdokumente zugeschnittenen Softwarehilfsmittel und demzufolge auch keine werkstattähnliche Umgebung von aufeinander abgestimmten Werkzeugen. Um eine solche

[19] vgl. [BZ90, S.216ff]
[20] vgl. [Gry95, S.23]
[21] **C**omputer **A**ided **S**oftware **E**ngineering

unterstützende Umgebung zu schaffen, müssen zunächst die Tätigkeiten der Entwickler im Rahmen des Entwicklungsprozesses nach WAM ermittelt und auf ihre Unterstützbarkeit hin untersucht werden. Diese Tätigkeiten sind die Erstellung, Vorlage und Modifikation der Text- sowie der Softwaredokumente:

Die textuellen Dokumente Glossar, Szenarios und Systemvisionen werden entweder von Hand auf Papier, mit Schreibmaschine oder mit Hilfe einer Textverarbeitung und einem Graphikprogramm erstellt und liegen bei Gesprächen zwischen den am Entwicklungsprozeß beteiligten Gruppen komplett in Papierform vor. Verweise auf andere Dokumente werden durch Heraussuchen der betreffenden Dokumente verfolgt. Nach Änderungen eines oder mehrerer Dokumente werden alle im Projekt vorhandenen Exemplare zugleich auf den neuen Stand gebracht, um die Konsistenz zu wahren.

Die Softwaredokumente Prototypen werden mit den üblichen Programmierwerkzeugen Editor, Compiler, Linker und Debugger in einer objektorientierten Sprache (z.B. Smalltalk, Eiffel, C++) erstellt. Für die Implementation in C++ stehen gut entwickelte Softwarebibliotheken zur Verfügung, mit deren Hilfe Programme mit WAM-spezifischer Architektur[22] und X-Window[23]-gemäßer Oberfläche wesentlich beschleunigt erstellt werden können. Außerdem steht ein Ressourcenbrowser zur Verfügung, mit dem die Darstellung von bereits vom Programmierer festgelegten Oberflächenelementen vereinfacht eingestellt werden kann.

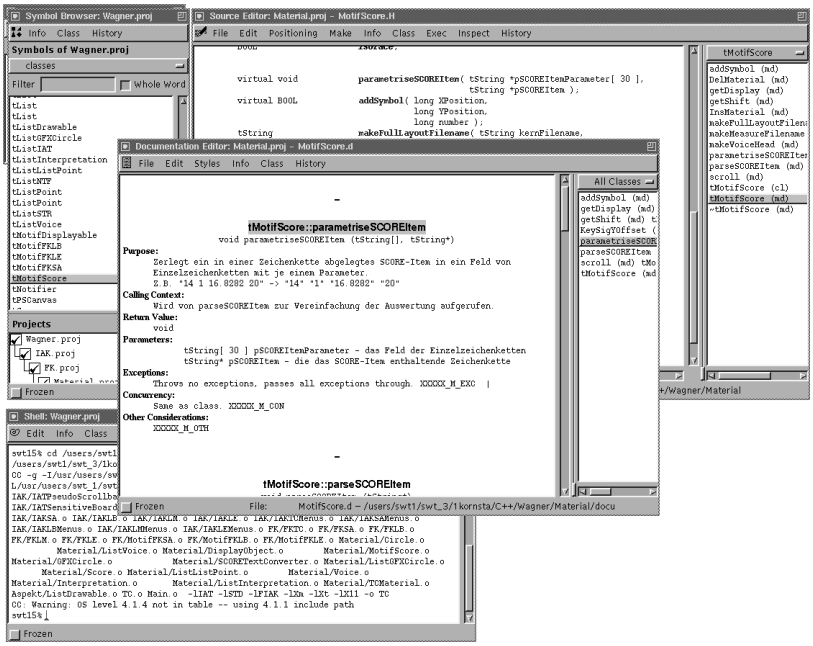


Abb. 3.1.2-3 Die Programmierumgebung *Sniff*
Im Hintergrund ein Symbolbrowser (oben links), ein Editor (oben rechts) und eine Shell (unten links). Im Vordergrund ein Dokumentationseditor.

Falls in C++ oder Eiffel[24] implementiert wird, bietet sich die Verwendung der integrierten Programmierumgebung *Sniff*[25] an. Über Editor, Debugger, Klassen- und Klassenhierarchiebrowser hinaus bietet *Sniff* auch eine

---

[22] Wie bereits in der Einleitung zu Kapitel 3.1.1 erwähnt, sind die technischen Aspekte der WAM nicht Gegenstand dieser Arbeit. Einen guten konzeptionellen Überblick über dieses Thema bietet [RZ95]. Die implementatorischen Details finden sich in [Rie93a].

[23] Das X Window System ist ein nichtproprietäres Fenstersystem, daß sich im Workstationbereich zusehends als Standard etabliert. Für das X Window System entwickelte Programme lassen sich auf jedem Rechner, auf dem es installiert ist, unabhängig von dessen Bauart darstellen. Nach WAM entwickelte Programme können u.a. mit der in [Rie93b] beschriebenen Bibliothek mit einer X-Oberfläche versehen werden.

[24] vgl. [Wil95]

Dokumentationserstellungshilfe an, mit der Klassen, Methoden, Variablen und andere Systembestandteile dokumentiert werden können.

Die weitere Diskussion basiert auf der Annahme, daß in C++ mit Unterstützung von *Sniff* implementiert wird.

Die Schwachpunkte der bestehenden Entwicklungsdokumente und der Entwicklungsumgebung

Die Schwachpunkte der bestehenden Entwicklungsdokumente und der Entwicklungsumgebung lassen sich nicht gleich gut erkennen. Während die Mängel der Dokumente sich erst bei der Arbeit mit ihnen zeigen -und dementsprechend erst in Kapitel 3.3.2 behandelt werden-, so treten die der Entwicklungsumgebung bereits in der obigen Beschreibung ihrer Eigenschaften zu Tage. Sie lassen sich auf zwei Grundprobleme zurückführen: Die der hochgradigen Vernetzung nicht gerechtwerdenden Repräsentation der Entwicklungsdokumente auf Papier und den Bruch beim Übergang von textuellen Systemvisionen zu ablauffähigen Prototypen.

Vernetzung. Die papiergebundene Darstellung der Entwicklungsdokumente läßt eine Behandlung der Verweise zu, die bestenfalls an die eines Lexikons heranreicht. Begriffe, zu denen eine weitere Erklärung existiert, werden durch eine besondere Schriftart oder Symbole (z.B. $\rightarrow$) hervorgehoben. Der Leser kann einem Verweis zwar folgen, muß dessen Ausgangsposition aber (durch Lesezeichen, Finger, etc.) markieren, falls er dorthin zurückkehren möchte. Die zahlreichen, stark verwobenen Dokumente eines Projekts können schon bald nicht mehr übersichtlich durchblättert werden, da nach kurzer Zeit die Lesezeichen nicht mehr verwaltet werden können, die Finger nicht mehr ausreichen oder der Tisch mit Dokumenten übersät ist. Die Suche muß also trotz weiteren Klärungsbedarfs vorzeitig abgebrochen werden, die Lesezeichen entfernt und die Suche an neuer Stelle fortgesetzt werden. Das konkrete Medium Papier steht der idealerweise uneingeschränkten Verfolgung der Verweise im Weg und behindert somit eine optimale Arbeit nach WAM.

Eine Übersicht über die Vernetzung der Dokumente, wie sie in Abb. 3.1.2-2 dargestellt ist, kann nur in aufwendiger Handarbeit erstellt werden. Dazu müssen alle Dokumente auf ihre Verweise hin durchsucht und das Ergebnis graphisch festgehalten werden. Diese an sich einfache aber dennoch zeitraubende Such- und Notieraufgabe verschlingt einen einen nicht unbeträchtlichen Anteil der Entwicklungszeit. Das Medium Papier behindert hier zwar nicht die Arbeit, es unterstützt sie aber auf keinerlei Weise. Es steht vielmehr als ein die Sicht beeinträchtigender Schirm zwischen den Entwicklern und der Struktur der Dokumente: Die Struktur ist in den Verweisdaten zwar enthalten, aber auf viele verschiedene Seiten verteilt, so daß sie in ihrer Gesamtheit nicht erkennbar ist.[26]

Bruch zwischen Text- und Softwaredokumenten. Einer der Hauptvorteile von WAM ist die methodisch bruchlose Verbindung von Analyse zur Synthese, bei der Elemente des Anwendungssystems in wiedererkennbarer Form auf das Softwaresystem abgebildet werden. Diese Bruchlosigkeit der *Methode* findet sich bei der *praktischen Durchführung* des Übergangs von Systemvisionen zu einem Prototypen nicht wieder. Hier treffen

---

[25] vgl. [Tak95]

[26] Analogien zur Darstellung der Dokumentvernetzung weist die Darstellung von Klassenbeziehungen auf. Hierzu gibt es in vielen Programmierumgebungen spezielle Klassenhierarchiebrowser.

die Welt der Textdokumente und die Welt der Softwaredokumente aufeinander, die durch eine Kluft getrennt sind, die weder in die eine noch in die andere Richtung leicht zu überschreiten ist. Ab der Erstellung des ersten Prototyps setzt eine „doppelte Buchführung“ ein, da dessen Beschreibung parallel in WAM-Textdokumenten und *Sniff*-Dokumentationen an die aktuelle Situation angepaßt werden muß. Außerdem können Elemente aus *Sniff*-Prototypbeschreibungen nicht auf einfache Weise in neue WAM-Systemvisionen einfließen und umgekehrt. In die eine Richtung müssen Bestandteile der Systemvisionen innerhalb des *Sniff*-Dokumentationssystems komplett neu erstellt werden und in die andere Richtung müssen mit dem *Sniff*-Dokumentationssystem hinzugefügte Details von Hand in neue Systemvisionen eingefügt werden.

Zwar werden sowohl in den WAM- als auch in den *Sniff*-Dokumenten Aspekte des zukünftigen Systems beschrieben, jedoch in beiden Dokumentationssystemen unter abweichenden Gesichtspunkten. Dies hat inhaltliche und strukturelle Konsequenzen.

- **Inhaltlich.** Die WAM-Systemvisionen richten sich an die Entwickler und dienen zur Konkretisierung von Entwurfsentscheidungen. Dazu wird ein sinnvoll erscheinender Ausschnitt des zukünftigen Systems (ein Ablauf, Werkzeug, Automat, usw.) beschrieben. Die *Sniff*-Dokumente richten sich an Entwickler und Programmierer und beziehen sich auf programmiersprachliche Systembestandteile (Klassen, Methoden, Aufzählungstypen, usw.). Ein direkter Bezug zwischen einem Ablauf und z.B. einer Klassenbeschreibung wird sich nur in den allerseltensten Fällen herstellen lassen. Diese generelle semantische Inkompatibilität ergibt sich allerdings hauptsächlich hinsichtlich des Basisdokumenttyps Systemvision. Im Gebos-Projekt hat sich neben den anwendungsorientierten Systemvisionen (Überblicksvision, Handhabungsvision, fachliche Ablaufvision) auch eine Gruppe technischer Systemvisionen herausgebildet (Einbettungsvision, Werkzeugvision, Automatvision, Materialvision, technische Ablaufvision),[27] die in Teilen große inhaltliche Überlappungen mit den technischen *Sniff*-Dokumenten aufweisen. Eine Verschmelzung bzw. teilweise Übernahme von technischen Inhalten zwischen diesen beiden Dokumenttypen wäre durchaus denkbar.[28]

- **Strukturell.** WAM-Systemvisionen sind (ggf. mit Graphiken versehene) Textdokumente, die für menschliche Augen bestimmt und dementsprechend formatiert sind. Die Dokumentstruktur wird ganz nach Belieben der am Entwicklungsprozeß beteiligten Personen z.B. durch bestimmte Einrückungen, Unterstreichungen, Fettdruck, Schriftarten oder -größen hervorgehoben. Eine Einheitlichkeit ist zwar wünschenswert aber nicht notwendig. Die *Sniff*-Dokumente müssen dementgegen für die Entwickler *und Sniff* interpretierbar sein und sind demzufolge stark strukturiert und haben ein fest definiertes Erscheinungsbild, daß ähnlich einem Formular nur ausgefüllt, nicht aber strukturell modifiziert werden kann. Der Entwickler hat des weiteren außer der Hervorhebung bestimmter Textabschnitte durch Schrägdruck keinerlei Einfluß auf die Gestaltung der einzelnen Einträge und muß mit einer Standardschriftart vorlieb nehmen. Graphiken können nicht eingebunden werden.[29]

Selbst wenn es also zu Überlappungen zwischen Teilen von WAM- und *Sniff*-Dokumenten kommt, können deren Inhalte nicht ausgetauscht oder gemeinsam genutzt werden, weil die WAM-Dokumente über keine ausreichende Strukturierung im Sinne einer festen Syntax verfügen, wie sie die *Sniff*-Dokumente aufweisen.

---

[27] vgl. [GZ95, S.82]
[28] vgl. [GZ95, S.37]
[29] vgl. [Tak95]

Skizze einer idealen Entwicklungsumgebung für WAM

Eine ideale Entwicklungsumgebung für WAM muß die beiden Hauptprobleme des Ist-Zustandes beheben:

- Die Repräsentation der Dokumente muß deren stark verwobener Struktur Rechnung tragen und deren Durchdringung mühelos machen.
- Der Übergang von Systemvisionen zur Dokumentation von Prototypen und zurück muß so leicht wie möglich sein.

Dies würde ein einheitliches Hypertextdokumentationssystem mit semantischer Komponente leisten, in dem sowohl die Textdokumente Glossar, Szenarios und Systemvisionen als auch die *Sniff*-Dokumentationen abgefaßt werden könnten. Diese einheitlichen Dokumente würden das Rückgrat des gesamten Entwicklungsprozesses bilden.

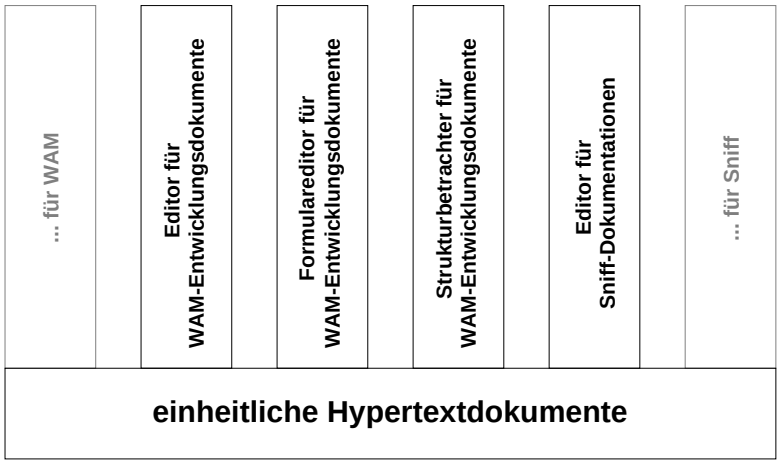


Abb. 3.1.2-4 Schematische Darstellung des einheitlichen Dokumentationssystems

Alle existierenden und zukünftigen Werkzeuge, die auf WAM- und *Sniff*-Seite mit Dokumenten arbeiten, setzen auf einen einheitlichen Dokumentstandard auf.

Im Inneren bestünden die Dokumente aus einer beliebigen Reihenfolge semantisch identifizierbarer Rubriken (z.B. „Kontextbeschreibung“, „Zweck“, „Basisklasse“). Die Systemstruktur wäre durch die Hypertextverweise festgelegt.

Mit einem solchen System könnten die Hauptprobleme des Ist-Zustands gelöst werden:

- Verweise könnten über Hyperlinks direkt und beliebig weit verfolgt werden (mit einem Hypertextbetrachter).
- Die Systemstruktur könnte automatisch aus den Dokumenten herausgefiltert und graphisch dargestellt werden (durch Analyse der Verweisinformation).
- Teilweise inhaltliche Übernahmen zwischen WAM- und *Sniff*-Dokumenten wären möglich bzw. bei einem einheitlichen System völlig unnötig.

Zusätzliche Vorteile wären:

- Die Dokumente wären überall immer auf dem aktuellen Stand (durch zentrale Speicherung und Betrachtung über Datennetze).
- Die Dokumente wären zu jedem Zeitpunkt des Entwicklungsprozesses, insbesondere bei Gesprächen, für alle Beteiligten einsehbar (durch zentrale Speicherung und Betrachtung über Datennetze).

Die die Basis des Systems bildenden Hypertextdokumente wären mit der Hypertextspezifikationssprache HTML[30] realisierbar.

Kapitel 3.2 enthält eine Schilderung des Entwicklungsprozesses, in dessen Verlauf eine Ausdifferenzierung der einfachen WAM-Dokumenttypen Glossar, Szenario und Systemvision hin zu speziellen Leitmotivanalyse-WAM-Dokumenten stattfand und in dem Teile der oben skizzierte Umgebung bereits realisiert wurden. Beide Prozesse sind mit dem Dokument-Symbol hervorgehoben. In Kapitel 3.3.2 befinden sich die erweiterten Dokumenttypen und Richtlinien, die sich bei deren Anpassung an das Anwendungsgebiet als nützlich erwiesen haben, sowie eine vollständige Spezifikation der idealen Entwicklungsumgebung und deren fertiggestellter Teile.

3.1.3 Anwendungsgebiet

Das Leitmotivanalysewerkzeug wurde aus der Taufe gehoben, um die Analyse von Richard Wagners *„Der Ring des Nibelungen“* zu erleichtern. Während der konkrete *Arbeitsvorgang* in den Entwicklungsdokumenten (Anhang B) präzise beschrieben wird, wird in diesem Kapitel der *Arbeitsgegenstand* - die nach der Leitmotivtechnik komponierte Operntetralogie *„Der Ring des Nibelungen“* - und die von ihm ausgehende Faszination für Musikwissenschaftler beschrieben. Nach einer Schilderung der bisherigen Ergebnisse der Leitmotivanalyse und der Hervorhebung von deren Schwachstellen werden die Möglichkeiten zu deren Überwindung durch eine Rechnerunterstützung skizziert und der Entwicklungsprozeß des Leitmotivanalysewerkzeugs in Kapitel 3.2 verfolgt. In Kapitel 3.3.3 wird der bisherige Stand des Werkzeugs sowie die in ihm enthaltenen Verbesserungen, die über die reine Rechnerunterstützung hinausgehen, beschrieben.

Entwicklung der Oper bis zu Wagner

Eine Oper ist ein musikalisches Bühnenwerk mit Darstellung einer Handlung durch Gesang und Instrumentalmusik. Sie entstand um 1600 in Italien als ein Versuch zur Erneuerung der griechischen Tragödie, deren Handlung in Gesängen vorgetragen wurde. Zunächst wurde auf der Grundlage des antiken Stoffs von einem Librettisten ein in Versform gefaßtes Textbuch (*Libretto*) geschrieben, das dann in einem zweiten Arbeitsschritt von einem Komponisten vertont wurde. Da es nicht möglich erschien, die gesamte Dauer der Tragödie gleichbleibend ausdrucksstark mit Musik zu unterlegen, wie dies bei einem kurzen Lied problemlos

[30] Eine kurze Beschreibung von HTML erfolgt in Kapitel 3.3.2.

möglich ist, wurde eine Trennung in Arie und Rezitativ vorgenommen, die sich beständig abwechseln. Das Rezitativ ist ein Sprechgesang, in dem die eigentliche dramatische Handlung -die Oberflächenstruktur- in Dialogen vorgetragen wird. Es wird musikalisch nur spärlich und ohne Verfolgung einer Melodie begleitet. Die Arie (oder bei mehreren Stimmen auch Ensembles wie Duett, Terzett, etc. oder aber Chor) ist ein melodischer Gesang, in dem die psychologische Tiefenstruktur der Handelnden ausgedrückt wird. Sie wird vom gesamten Orchester wirkungsvoll ausgestaltet und untermalt.[31] So unterschiedlich wie die ausgedrückten Empfindungen sind auch die Melodien der verschiedenen Arien einer Oper, so daß es innerhalb eines Werks meistens wenig melodische Verbindungen zwischen den einzelnen Arien gibt.

Die Arie-Rezitativ-Form der Oper blieb bis in die Mitte des 19. Jahrhunderts bestehen, obwohl sich bei den behandelten Stoffen und der musikalischen Gestaltung von Rezitativ und Arie Änderungen ergaben. Zu der als *opera seria* (ernste Oper) bezeichneten Oper, die weitestgehend auf antike Stoffe zurückgriff, trat in der zweiten Hälfte des 18. Jahrhunderts die *opera buffa* (komische Oper), die als Textgrundlage aktuelle Lustspiele aufgriff. Zwischen diesen beiden Polen entwickelten sich viele -auch oft national geprägte- Mischgattungen (z.B. mit ausgedehnten Ballettszenen in Frankreich), die jedoch alle am Arie-Rezitativ-Schema festhielten. Das Rezitativ wandelte sich vom ursprünglichen *Secco*-Rezitativ (trockenes R.), das nur mit wenigen Akkorden eines Tasteninstruments begleitet wurde, zum *Accompagnato*-Rezitativ (begleitetes R.), das nun auch stellenweise melodiös war und dementsprechend vom Orchester begleitet wurde. Trotzdem bestand weiterhin eine klare Trennung zu den Arien. Die Arien standen meistens noch immer musikalisch in keiner Beziehung zueinander, obwohl einige Komponisten gewisse Motive einer Arie in anderen wiederholten, um die dramatische Situation, die sie in der vorigen Arie betonten, erneut ins Gedächtnis zu rufen.

In der Aufführungspraxis trat das die Oper ursprüngliche bestimmende Drama zusehends in den Hintergrund, während die Arien und deren künstlerische Ausführung mehr und mehr in den Mittelpunkt des Interesses der Opernbesucher rückte. Arien wurden von Komponisten berühmten Sängern oft „auf den Leib“ geschrieben, so daß deren Virtuosentum maximal betont werden konnte. Das Rezitativ geriet immer mehr zum unvermeidlichen Beiwerk zwischen den Arien.[32]

Weiterentwicklung der Oper durch Wagner

Wagner beklagt in seinen zahlreichen theoretischen Schriften die von ihm beobachtete Entwicklung der Oper, in der „ein Mittel des Ausdruckes (die Musik) zum Zwecke [und] der Zweck des Ausdruckes (das Drama) … zum Mittel“ gemacht werde.[33] Diesen Mißstand gedenkt er durch einen neuen Operntypus zu beheben, in dem das Drama erneut (wie zu Beginn der Operngeschichte) die Grundlage für die Musik sein soll. Im Gegensatz jedoch zur klassischen *opera seria* soll es keine Trennung zwischen gefühlsbetonter Arie und verstandesbetontem Rezitativ geben. Da die handelnden Personen des Dramas über das ganze Werk hinweg von ihren Hoffnungen, Ideen und Erinnerungen angetrieben werden, sollen diese Beweggründe auch während der gesamten Spieldauer der Oper -und nicht nur in isolierten Arien- musikalisch sichtbar gemacht werden. Zur Umsetzung seiner Idee orientiert Wagner sich am symphonischen Stil, bei dem sich ein Netzwerk aus Mo-

S1 *Lösungsstruktur orientiert sich an Problemstruktur*

A1 + A2 *viele + kleine Elemente*

[31] vgl. [Dal86a, S.210]
[32] vgl. [Pah87, S.11ff]
[33] vgl. [Wag83, S.18f]

tiven über Sätze oder ganze Satzzyklen zieht. Während die Motive einer Symphonie rein musikalischen Ursprungs sind und sich deren Entwicklung an festen Regeln orientiert, ordnet Wagner den Motiven bestimmte Elemente der dramatischen Handlung zu (Personen, Gegenstände, Ideen, etc.) und entwickelt sie entsprechend dem dramatischen Fortgang der Ereignisse.[34] Die Musik dieser Oper neuen Typs untermalt also permanent die sich auf der Bühne abspielende Handlung entweder durch Bestätigung (das Motiv im Orchester entspricht dem Gesagten) oder Kontrastierung (das Motiv im Orchester drückt aus, was die Handelnden in Wirklichkeit antreibt, obwohl sie es verschweigen). Auch die Motive selbst, die bald Leitmotive genannt wurden, obwohl Wagner selbst stets nur von „Grundthemen" sprach,[35] stehen nicht beziehungslos nebeneinander, sondern gehen ebenfalls durch Variation (verwandeter Sinn) oder Kontrastierung (entgegengesetzter Sinn) aus bereits bekannten Motiven hervor.[36] Dieses Netz von kurzen, miteinander verwobenen Motiven, die sowohl mit der Handlung als auch untereinander in Beziehung stehen, und das von Thomas Mann als „Beziehungszauber"[37] bezeichnet wurde, soll dem Zuschauer ein vollständiges Gefühlsverständnis von der Handlung des Dramas geben.

A3 *kausale Verbindungen*

S2 *kurze, präzise Elemente*

Wagner -und später auch andere Komponisten- setzte seine Vorstellungen in allen seinen seit der 1851 erfolgten Formulierung seiner Idee geschriebenen Opern um, wobei er stets sowohl das Libretto entwarf als auch die Musik komponierte. Die so entstandene Werke sind *„Tristan und Isolde"* (1857 Komposition begonnen, 1865 uraufgeführt), *„Die Meistersinger von Nürnberg"* (1862, 1868), die Operntetralogie *„Der Ring des Nibelungen"* (bestehend aus *„Das Rheingold"*, *„Die Walküre"*, *„Siegfried"* und *„Götterdämmerung"*) (1853, 1876) und *„Parsifal"* (1877, 1882).

Σ *alle Merkmale einer hochwertigen Methode (S. 10)*

Die Leitmotivtechnik in der Musikwissenschaft

Schon bald nach der Aufführungen der Wagnerschen Opern wurde erkannt, daß die Leitmotive eine zentrale Rolle zum Verständnis derselben darstellen. Schon ein Jahr nach der Uraufführung des *„Rings"* wurde 1877 *ein „Thematischer Leitfaden durch die Musik zu Richard Wagner's Festspiel 'Der Ring des Nibelungen'"*[38] herausgegeben, der viele Leitmotive aufzeigte und mit Namen versah. Bis heute ist es jedoch umstritten, ob bzw. inwieweit die Leitmotive zur rein musikalischen Struktur der Werke beitragen. Aufgrund der vollkommen neuen Opernform wurde Wagner von Musikkritikern zunächst vorgeworfen, seine Werke seien formlos und er selbst sei ein Dilettant,[39] denn sein neuer Stil paßte weder in das Schema einer hergebrachten Oper noch in das einer Symphonie. Für die große Struktur des „Rings" wurden Deutungsversuche im Sinne einer viersätzigen Symphonie (*„Das Rheingold"*: Allegro, *„Die Walküre"*: Adagio appasionato, *„Siegfried"*: Scherzo mit Trio, *„Götterdämmerung"*: Finale mit Rückbeziehung)[40] oder auch nur eines einzigen Symphoniesatzes (*„Das Rheingold"* als Exposition, *„Die Walküre"* und *„Siegfried"* als Durchführung, *„Götterdämmerung"* als freie Reprise)[41] unternommen. Es dauerte bis 1924, bis Alfred Lorenz in *„Das Geheimnis der Form bei Richard Wagner. I. Band: Der musikalsiche Aufbau des Bühnenfestspiels 'Der Ring des Nibelungen'"*[Lor24]

---

[34] vgl. [Dal86b, S.75]
[35] vgl. [Rie86, S.631]
[36] vgl. [Dal86b, S.75]
[37] siehe [Man63, S.144] und vgl. [Flo83, S.8]
[38] vgl. [Bau88, S.591]
[39] vgl. [Dal86a, S.217f]
[40] vgl. [XXX]
[41] vgl. [XXX]

einen umfassenden Versuch zur Abwehr des Formlosigkeitsvorwurfs unternimmt. Lorenz faßt Leitmotivgruppen zu Perioden zusammen, deren Abfolge den strengen Formanforderungen zwar genügt, muß dazu allerdings teilweise aberwitzige Periodenlängen von nur 14 bis zu 840 Takten festlegen.[42] Carl Dalhaus widerlegt Lorenz 1969 in *„Gibt es ein Geheimnis der Form bei Richard Wagner?"*[Dal70] zwar von Grund auf, indem er aufzeigt, daß dieser den *„Ring"* gewaltsam in das ihm bekannte Formschema preßt, jedoch bleibt auch er eine Erklärung für die Form schuldig, so daß das Formproblem bis heute ungelöst ist.

Die Schwachpunkte der leitmotivischen Analyse

Da die Rolle, die die Leitmotive bei der Formgebung von Wagners Opern spielen, Hauptgegenstand der Untersuchungen ist, wäre eine vollständige tabellarische Aufstellung aller Leitmotive sowie eine Partitur, in der das Vorkommen aller noch so engmaschig verwobenen Leitmotive vermerkt sind, eine notwendige Grundlage. Weder das eine noch das andere existiert. Die vorhandenen Leitmotivtabellen sind unvollständig sowohl bezüglich der Anzahl der enthaltenen Leitmotive als auch bezüglich deren Gestaltenvielfalt im Laufe der Werke und beziehen sich stets auf die Klavierauszüge und nicht die vollständigen Partituren. Erschöpfend annotierte Partituren sind nicht vorhanden; lediglich isolierte Szenen wurden ausgiebig untersucht. Der Grund für diese Schwachstellen ist nicht in mangelndem Interesse sondern im gewaltige Umfang der zu analysierenden Opern zu suchen.[43] Er zwingt bei rein manueller Durchführung der Analysen zu so starken quantitativen und qualitativen Einschränkungen, daß ein detailliertes Gesamtbild bis heute nicht vorliegt, obwohl das Vorgehen zur Erlangung eines solchen prinzipiell klar ist: Die Partituren müßten lückenlos auf Leitmotive und deren Variationen hin durchsucht werden. Diese müßten dann sowohl in einer Leitmotivtabelle als auch in der Partitur vermerkt werden. Ausgehend von einer solchen Aufstellung könnten dann Aussagen über die musikalische Form von *„Der Ring des Nibelungen"* und anderer Opern besser gemacht werden als je zuvor.

Skizze einer Rechnerunterstützung für die Leitmotivanalyse

Das Aufspüren von Leitmotiven in der Partitur sowie das Treffen von Aussagen über die musikalische Struktur einer Oper sind zweifelsohne rein menschliche, nicht automatisierbare Aufgaben, da sie musisches Empfinden vorraussetzen. Der sehr zeitaufwendige Vorgang des Niederschreibens und der Markierung von Leitmotiven könnte jedoch von einem Rechner unterstützt werden, wenn die Opernpartituren in rechnerlesbarer Form vorlägen. Die Leitmotive könnten dann nach Maßgabe des Musikwissenschaftlers aus den Partiturdaten exzerpiert und automatisch in eine Leitmotivtabelle eingetragen werden. Ein Rechner könnte auch bei der Verwaltung der Leitmotivtabellen nützlich sein, die bei einer vollständiger Markierung aller Leitmotive und ihren Variationen bald gewaltig anwachsen würden. Der Rechner würde also nicht benutzt werden, um selbständig anhand irgendwelcher Suchalgoritmen die Partitur nach Leitmotiven zu durchsuchen (was er aufgrund deren unscharfen Erscheinungsbilds auch gar nicht zuverlässig könnte), sondern er würde die Musikwissenschaftler lediglich durch die Übernahme von aufwendigen Routineaufgaben unterstützen.

---

[42] vgl. [Bau88, S.259]
[43] Die Schott'sche Partiturausgabe vom Beginn dieses Jahrhunderts[Wag1,Wag2, Wag3 und Wag4] umfaßt 4357 Seiten.

Da die Partituren zu *„Das Rheingold“* und *„Die Walküre“* innerhalb der nächsten Jahre aufgrund der Umstellung von manuellem auf elektronischen Notensatz bei Musikverlagen (z.B. Schott (Mainz)) tatsächlich in rechnerlesbarer Form vorliegen werden, steht der Verwirklichung des oben skizzierten Softwarewerkzeugs zur Unterstützung der leitmotivischen Anlayse nichts mehr im Wege.

Kapitel 3.2 schildert den Prozeß der Entwicklung des Werkzeugs, in dem der Wissenserwerb über den Analysevorgang und dessen softwaretechnische Umsetzung mit dem Anwendungsgebiet-Symbol ♪ hervorgehoben sind. Kapitel 3.3.3 beschreibt dann zusammenhängend das vollständige Werkzeug und die Verbesserungen des herkömmlichen Verfahrens, die sich bereits jetzt ergeben haben. In Kapitel 4 wird auf die aufwendigen technischen Vorarbeiten eingegangen, die die Realsierung des Werkzeugs überhaupt erst ermöglicht haben.

3.2 Entwicklungsprozeß

Dieses Kapitel zeigt die Verzahnung zwischen den Entwicklungen der Methode WAM, der Entwicklungsdokumente und des Produkts Leitmotivanalysewerkzeug auf. Im Laufe der standardmäßigen Durchführung des Softwareerstellungsprozesses nach WAM werden ausgehend von der zunehmenden Durchdringung des Anwendungsgebiets nicht nur der Inhalt der Entwicklungsdokumente sondern auch deren Struktur und sogar die Methode selbst beeinflußt, so daß sie sich mit der Zeit immer enger an das spezielle musikwissenschaftliche Anwendungsgebiet anschmiegen. Während in Kapitel 3.1 und Kapitel 3.3 Methode, Dokumente und Anwendungsgebiet getrennt erörtert werden, so liegt hier der Schwerpunkt auf der gegenseitigen Induzierung von Weiterentwicklungen. Es dient gleichermaßen als empirische Grundlage für die in Kapitel 3.1 aufgestellten und in Kapitel 3.3 bestätigten Thesen.


S1
Lösungsstruktur orientiert sich an Problemstruktur für ♪→📄→⚠

In der folgenden episodischen Schilderung ist jedem Entwicklungsschritt ein Symbol (Methode ⚠, Dokumente 📄 und Anwendungsgebiet ♪) zugeordnet, um auf einen Blick kenntlich zu machen, welchem Bereich er zuzuordnenen ist. Um die Verzahnung zwischen der Entwicklung in allen Bereichen zu verdeutlichen, sind logische Abhängigkeiten durch Pfeile hervorgehoben.

Zunächst werden die Randbedingungen und das Wissen des Entwicklers zu Beginn des Prozesses getrennt für jeden Bereich kurz geschildert und dann die eigentliche Entwicklung in fünf -ausschließlich zur besseren Orientierung eingeführten- Zyklen beschrieben. Jeder Zyklus beginnt mit einem Gespräch zwischen Anwender und Entwickler und zeigt dann die daraus erwachsenen Änderungen in den Bereichen Methode, Dokumente und Anwendungsgebiet.

Randbedingungen

(1) Die aktiv am Entwicklungsprozeß beteiligten Personen waren jeweils ein Anwender (zwei nacheinander) und ein Entwickler.

(2) Der erste Anwender hatte keine Rechnererfahrung.

(3) Der Entwickler hatte bereits einige eigene Vorstellungen vom zukünftigen System.

(4) Die zu untersuchende Tätigkeit der *vollständigen* Leitmotivanalyse wurde von niemandem jemals ausgeübt.

Anfangswissen

⚠ Die WAM-Methode war dem Entwickler in der in *„Objektorientierte Anwendungsentwicklung“*[GKZ93] beschriebenen Form bekannt. Zugleich bestand die Absicht, die Methode nicht buchstabentreu sondern sinngemäß anzuwenden, um eine Adaption an eine musikwissenschaftliche Aufgabenstellung zu ermöglichen.

📄 Die Entwicklungsdokumente sollten gemäß der in *„Objektorientierte Anwendungsentwicklung“*[GKZ93] geschilderten Form auf Papier realisiert werden. Zu Beginn der Entwicklung bestand ein rudimentäres Glossar, das aus einem ersten Gespräch mit dem Anwender über den Begriff des Leitmotivs hervorgegangen war.

♪ Der Entwickler war interessierter Laie ohne Notenlesefähigkeit und konnte zwar Leitmotive aus Opernaufzeichnungen heraushören, diese aber nicht in der Partitur identifizieren oder ihre musikalischen Bestandteile bestimmen. Zur Vorbereitung wurde das Buch *„Allgemeine Musiklehre“*[Gra82] gelesen.

Als eigener Verbesserungsvorschlag bestand die Idee, Leitmotiven Farben zuzuordnen, um diese nach erfolgter Identifizierung in der Partitur „auf einen Blick“ erkennbar zu machen und ihre Struktur leichter mit der dramatischen Struktur des Texts vergleichen zu können.

1.Zyklus

(1)Papierektypus[44]-Methode. Im Vorgespräch hatte sich herausgestellt, daß vollständige Leitmotivtabellen seit Beginn der Beschäftigung mit Leitmotiven nicht mehr erstellt worden waren. Da eine solche Tabelle aber im Rahmen einer kompletten Analyse als notwendig erschien, mußte dieser Erstellungsprozeß rekonstruiert werden. Es stand fest, daß die Tabelle sämtliche Vorkommen aller Leitmotive als „Leitmotivnotizen" enthalten mußte, aber ihre genauen Komponenten konnten nicht sofort identifiziert werden.

In Anlehnung an das WAM-Vorgehen der Entwicklung in einer Vielzahl kleiner Analyse- und Syntheseschritte mit Diskussion zwischen Anwender und Entwickler wurde gemäß einer initialen Spezifikation des Anwenders ein „Leitmotivnotizzettel" erstellt, der dann vom Anwender zur tatsächlichen Analysearbeit während eines Interviews eingesetzt werden sollte. Durch Aufzeigen der dabei auftretenden Mängel sollte dieser „Papierektypus" solange verbessert werden, bis er den Anforderungen entsprach.

↓

(2)Interview. Das nach dem Vorgespräch erstellte Glossar wurde vom Anwender als korrekt bewertet. Der Anwender führte unter Verwendung der Leitmotivnotizzettel eine Analyse einer rein instrumentalen Opernstelle durch und erklärte dem Entwickler dabei sein Vorgehen. Die Leitmotivnotizzettel erwiesen sich teilweise als nicht umfangreich genug.

↓

(3)Entwicklungsdokumenterstellung. Anhand der schriftlichen Interviewaufzeichnungen und aus dem Gedächtnis heraus wurde das Glossar erweitert, ein Szenario „Leitmotivanalyse" erstellt und die Leitmotivzettel verbessert.

↓

(4)Entwicklungsdokumentverbesserung. Um die Verweise aus dem Szenario in das Glossar einfacher und übersichtlicher verfolgen zu können, wurden die Dokumente als Hypertexte erstellt. Zur Umsetzung diente die „Hypertext Markup Language"(HTML). Die Dokumente können mit geeigneten Hypertextbetrachtungsprogrammen (Mosaic, Netscape) gelesen werden.

↓

(5)Dokumentlayoutverbesserungen. Um Verweise in Hypertextdokumenten besser unterscheiden zu können, werden Verweise auf Glossareinträge unterstrichen und Verweise auf andere Szenarios *kursiv und unterstrichen* dargestellt.

↓

(6)Interne Dokumentstrukturverbesserung. Um den in Szenarios geschilderten Arbeitsvorgang besser einordnen zu können, werden in der Rubrik „Kontextbeschreibung" Sinn und Zweck des Vorgangs erwähnt. Die Rubrik „Verwendete Werkzeuge und Materialen" wurde in „Verwendete Arbeitsgegenstände" umbenannt, um keine 1:1-Übertragbarkeit auf Softwarewerkzeuge und -materialien zu suggerieren.

↓

(7)Entwicklungsdokumentdifferenzierung. Bei genauer Betrachtung ließen sich im Szenario „Leitmotivanalyse" drei Untertätigkeiten identifizieren. Das Szenario wurde aufgeteilt in „Leitmotivanalyse", „Leitmotivsuche" und „Leitmotiv notieren".

---

[44] gr.: Nachbildung, Abbild, Ggs.: Prototyp. [Dud90, S.209]

2.Zyklus

(1)Interview. Die dem Anwender vorgelegten Entwicklungsdokumente wurden nicht beanstandet. Der Anwender führte unter Verwendung der verbesserten Leitmotivnotizzettel eine Analyse einer Opernstelle mit Text durch und erklärte dem Entwickler dabei erneut sein Vorgehen. Die Leitmotivnotizzettel erwiesen sich in einem Punkt als nicht ausreichend.

↓

(2)Entwicklungsdokumentfortschreibung. Anhand der schriftlichen Interviewaufzeichnungen und aus dem Gedächtnis heraus wurde das Glossar erweitert, das Szenario „Leitmotiv markieren“ aus dem Szenario „Leitmotiv notieren“ ausgegliedert und die Leitmotivnotizzettel verbessert.

↓

(3)Interne Dokumentstrukturverbesserung. Um das angewachsene Glossar besser lesbar zu machen, wurde eine lexikonartige Eintragsstruktur eingeführt: Das Stichwort wird gefolgt von einer in einem Satz abgefaßten Kurzbeschreibung, gefolgt von einer Liste der Eigenschaften bzw. der Besonderheiten.

↓

(4)Dokumentlayoutverbesserung. Um das Glossar besser lesbar zu machen, wurde der Stichwortname fett hervorgehoben.

3.Zyklus

(1)Anwenderwechsel. Der ursprüngliche Anwender verfügte über keine Erfahrungen im Umgang mit Rechnern und hätte somit keine Systemvisionen beurteilen können. Für die Beurteilung der Systemvisionen und der Prototypen wurde daher ein Anwender mit ähnlichen Fach- und zusätzlichen Rechnerkenntnissen gewonnen. Ihm sollten zunächst die bestehenden Entwicklungsdokumente vorgelegt werden. Der ursprüngliche Anwender sollte erst dann wieder hinzugezogen werden, wenn ein für ihn wiedererkennbarer und benutzbarer Prototyp vorliegen würde.

↓

(2)Interview. Der Anwender forderte eine fachliche Präzisierung der Begriffe in den ihm vorgelegten Entwicklungsdokumenten. Darüberhinaus regte er eine Erweiterung des Produktrahmens auf ein Leitmotivlexikon an, in dem potentiell alle Leitmotive aus allen Opern, die nach der Leitmotivtechnik komponiert wurden, aufgenommen werden könnten.

↓

(3)Entwicklungsdokumentfortschreibung. Anhand der schriftlichen Interviewaufzeichnungen wurden das Glossar und die Szenarios verbessert. Basierend auf den Szenarios wurden im 1:1-Verhältnis unter dem gleichen Namen Systemvisionen entworfen. Zusätzlich wurde eine Systemvision „Leitmotiv betrachten“ eingeführt. Kurzbeschreibungen der zukünftigen Softwarewerkzeuge wurden in das Glossar eingefügt. Die Idee des Entwicklers zur kanonischen Partiturdarstellung wurde als Option in eine Systemvision aufgenommen.

↓

(4)Mangel in der Entwicklungsumgebung festgestellt. Beim Erstellen der Systemvisionen wurde klar, daß ein guter Hypertexteditor fehlt. Alle Systemvisionen mußten in zeitaufwendiger Handarbeit mit einem normalen Texteditor in HTML erstellt werden. Des weiteren wäre es vorteilhaft, wenn leere (an die fachliche Aufgabe angepaßte) Dokumente bereitstünden, die dann nur noch ausgefüllt werden müßten.

↓

(5)Interne Dokumentstrukturverbesserung. Um Systemvisionen besser entkoppeln zu können, werden in der Systemvisionrubrik „Verwendete Softwarewerkzeuge“ nur noch die direkt in der Systemvision verwendeten Werkzeuge und nicht auch die in Untersystemvisionen verwendeten Werkzeuge erwähnt.

4. Zyklus

(1)♪Interview. Die Systemvisionen wurden dem Anwender vorgelegt und von diesem kritisiert. Zu allen Systemvisionen wurden kleine Verbesserungen vorgeschlagen. Die Entwickleridee zur kanonischen Partiturdarstellung wurde als brauchbar und sinnvoll eingestuft.

↓

(2)♪Entwicklungsdokumentfortschreibung. Anhand der in den Dokumenten vorgenommenen schriftlichen Interviewaufzeichnungen wurden das Glossar und die Systemvisionen verbessert.

↓

(3)Dokumentstrukturerweiterung. Aufgrund der starken Unterschiede zwischen eher ablauforientierten Systemvisionen (z.B. „Leitmotivanalyse“) und eher werkzeugorientierten Systemvisionen (z.B. „Leitmotiv notieren“) wurden die Systemvisionuntertypen „Ablaufvision“, „Werkzeugvision“ und „Übersichtvision“ eingeführt. Die Ablaufvisionen basieren auf den existierenden Systemvisionen und beschreiben einen Handlungsablauf im zukünftigen Softwaresystem, ohne dabei jedoch auf Werkzeugdetails wie Aussehen und Handhabung einzugehen. Die Werkzeugvisionen beschränken sich auf jeweils ein Werkzeug, das in den alten Systemvisionen erwähnt wurde. Sie beschreiben genau ein Werkzeug mitsamt Aussehen und Handhabung, gehen aber nicht weiter auf den Arbeitsvorgang ein, in den die Werkzeugbenutzung eingebettet ist. Die Übersichtvisionen geben einen Überblick über die Arbeit der Anwender mit dem zukünftigen Softwaresystem und gleichen strukturell einer Ablaufvisionn, wobei jedoch die „Kontextbeschreibung“ sehr ausführlich ist und die „Beschreibung des neuen Arbeitsvorgangs“ ggf. soweit in die Tiefe geht, bis ein guter Gesamteindruck besteht.

↓

(4)♪Entwicklungsdokumentdifferenzierung. Die bestehenden Systemvisionen wurden gemäß der weiteren Untertypisierung in Ablauf-, Werkzeug und Übersichtvisionen unterteilt. Die Ablaufvisionen gingen unter Beibehaltung des Namens und Entfernung der werkzeugspezifischen Teile aus den bestehenden Systemvisionen hervor. Als Werkzeugvisionen kamen hinzu „Partituranalysewerkzeug“ (aus „Leitmotiv suchen“), „Leitmotivmarkierer“ (aus „Leitmotiv markieren“), „Leitmotivnotizeditor“ (aus „Leitmotiv notieren“) und „Leitmotivbetrachter“ (aus „Leitmotiv betrachten“). Die Ablaufvisionn „Leitmotiv betrachten“ stellte sich im Trennungsprozeß als reine Untermenge von „Leitmotiv suchen“ heraus und wurde entfernt. Die Ablaufvision „Leitmotivanalyse“ wurde zu einer Übersichtvision erweitert.

↓

(5)Interne Dokumentstrukturverbesserung. Bei Ablaufvisionen ist die Rubrik „Verwendete Verfahrensvorschriften/Algorithmen“ sinnlos und wurde entfernt. Die Bildschirmskizze enthält nunmehr keine konkreten Werkzeugbilder, sondern nur noch eine schematische Werkzeugkonfigurartion der Werkzeuge, die zur Erledigung der Aufgabe notwendig sind, bzw. ihr unmittelbar vorausgehen. Bei Werkzeugvisionen entfielen die Rubriken „Verwendete Operationen von Softwarewerkzeugen“ und „Zugrundeliegende Szenarios“.Hinzu kamen die Rubriken „Ablaufvisionen, in denen das Werkzeug verwendet wird“ und „Mögliche Operationen“. Die Rubrik „Kontextbeschreibung“ weicht der Rubrik „Kurzbeschreibung“ und „Beschreibung des neuen Arbeitsvorgangs“ „Detaillierte Beschreibung".

Die Systemvisionrubrik „Weitere Ideen“ wurde hinzugenommen, um momentan noch verfrühte aber vom Entwickler als sinnvoll erachtete eigene Ideen zu einem späteren Zeitpunkt in den Entwicklungsprozeß einbringen zu können.

5. Zyklus

(1)♪Interview. Die verbesserten Systemvisionen wurden dem Anwender vorgelegt und von diesem kritisiert. Zu einer Systemvision wurden eine kleine Verbesserungen vorgeschlagen.

↓

(2)♪Entwicklungsdokumentfortschreibung. Anhand der in den Dokumenten vorgenommenen schriftlichen Interviewaufzeichnungen wurde eine Systemvision verbessert. Basierend auf den Systemvisionen wurde ein Handhabungsprototyp erstellt. Die Handhabung ist mit der in den Systemvisionen beschriebenen identisch; lediglich das komplexe Partituranalysewerkzeug wurde etwas eingeschränkt; jedes Werkzeug erhielt ein „Hilfe"-Menü, in dem die dem Werkzeug zugrundeliegende Systemvision hypertextgerecht betrachtet werden kann. Ein kleiner Werkzeugkoordinator, der den leeren Schreibtisch des Anwenders symbolisiert, wurde hinzugefügt.

↓

(3)Dokumentstrukturerweiterung. Um die Differenzen zwischen dem ausführbaren Handhabungsprototyp und den idealen Systemvisionen festzuhalten, wurde das Textdokument „Prototyp" hinzugefügt. Es zeigt zu jedem Softwarewerkzeug sein Erscheinungsbild im Prototyp und listet seinen Funktionsumfang sowie ggf. dessen Abweichung von dem der zugrundeliegenden Systemvision auf. Falls das Dokument auf dem gleichen Rechnersystem wie der Handhabungsprototyp vorliegt, kann dieser direkt vom Dokument aus gestartet werden.

↓

(4)Mangel in der Entwicklungsumgebung festgestellt. Beim Erstellen des Prototyps wurde klar, daß ein guter Benutzungsoberflächengestalter (GUI-Builder) für WAM-Softwaresysteme fehlt. Die Benutzungsoberflächen für alle Werkzeuge mußten in zeitaufwendiger Handarbeit mit einem normalen Texteditor in C++ erstellt werden. Nur die Anordnung der einmal angelegten Oberflächenelemente wird bisher unterstützt.

3.3 Endzustand

3.3.1 Methode

Nach der in Kapitel 3.1.1 erfolgten Vorstellung der WAM-Methode und deren Einsatz in Kapitel 3.2 wird in diesem Kapitel darauf aufbauend der Beweis für die Thesen „***WAM ist eine Universalmethode für qualifizierte und kooperative menschliche Tätigkeit“*** und darüberhinaus für „***WAM ist ein adaptierbarer Methodenkern für selbstbestimmte menschliche Tätigkeit“*** ausgeführt. Dazu werden die vier Beschränkungen der WAM „Methode“, „qualifiziert“, „kooperativ“ und „menschliche Tätigkeit“ in jeweils zwei Schritten untersucht. Im ersten Schritt(**1**) wird überprüft, ob die dieser Arbeit zugrundeliegende musikwissenschaftliche Anwendung im Rahmen dieser Beschränkungen liegen, obwohl sie über reine Sachbearbeitertätigkeit, des bisherigen Anwendungsfelds der Gebos-WAM, hinausgeht. Danach wird in einem zweiten Schritt(**2**) die Berechtigung der Schranke selbst in Frage gestellt und je nach Ergebnis eine Konkretisierung oder Lockerung erwogen.

Zunächst werden die vier Schranken einzeln bezüglich der beiden Thesen geprüft und dann eine abschließende Bewertung vorgenommen.

Beschränkung „Methode“

1 **„WAM, wie sie zur Erstellung des Leitmotivanalysewerkzeugs eingesetzt wurde, ist eine Methode.“** WAM macht klare und lückenlose Aussagen zur Art und Weise, in der ein Softwaresystem erstellt werden soll, und ist somit eine Methode. Im Rahmen der Entwicklung mußte jedoch eine Erweiterung zur Rekonstruktion von nicht mehr ausgeübten Tätigkeiten mit Hilfe von Ektypen vorgenommen werden. Dieses Verfahren ist in keinem Werk über WAM beschrieben, so daß das Leitmotivanalysewerkzeug nicht exakt nach *der* WAM laut [GKZ93] oder [GZ95] entwickelt worden ist, sondern lediglich nach einer *WAM-ähnlichen* Methode. Wie dieses Einordnungsproblem gelöst werden kann, zeigt der sich anschließende Abschnitt.

2 **„WAM ist der Kern einer adaptierbaren Methode.“** Das Einordnungsproblem von WAM-ähnlichen Methoden stellt sich nicht nur bezüglich des in dieser Arbeit beschriebenen Entwicklungsprozesses. Auch im Rahmen der Erstellung des Gebos-Systems und bei einem Projekt im Labor-Bereich wurde die in [GKZ93] beschriebene WAM, die eine vereinfachte Form der Gebos-WAM darstellt, den besonderen Gegebenheiten des jeweiligen Anwendungsgebiets angepaßt. Obwohl die Gebos-WAM, da sie den Ursprung der WAM darstellt, oft synonym mit WAM verwendet wird, ist sie nicht *die* WAM. Der Antwort auf die Frage „Was ist WAM eigentlich?“ bzw. „welche WAM ist *die* WAM?“ soll hier nachgegangen werden.

Die in [GKZ93] und [Gry95] genannten Prinzipien, die hinter WAM stecken, sind das Leitbild und die Metaphern mit ihrer bruchlosen Abbildung von Teilen eines Anwendungsgebiets auf ein Softwaresystem sowie die Ermittlung der damit verbundenen Abläufe und Gegenstände (bezüglich des alten als auch des neuen Systems) durch intensive zyklische Kommunikation zwischen Entwicklern und Anwendern auf Basis von in Anwendersprache verfaßten Dokumenten. Die Gebos-WAM ist jedoch nur *eine* mögliche Ausprägung dieser Prinzipien. Auch andere Ausprägungen haben eine ähnliche Gestalt, obwohl sie nicht in allen Teilen mit ihr deckungs-

gleich sind. Beispielsweise kommt die Leitmotivanalyse-WAM dieser Arbeit aufgrund ihres Zweipersonenstückcharakters ohne die organisatorischen Bestandteile der Gebos-WAM aus. Da sich alle Nicht-Gebos-WAMs in allen *entscheidenden* Teilen -nämlich den zugrundeliegenden Prinzipien- mit der Gebos-WAM überlappen, ist ihre Klassifizierung als „Nicht-(Gebos-)WAMs" eine nichtgutzuheißende Härte. Es sind der WAM-Idee folgende Methoden, die durch Abwandlungen und Erweiterung der Gebos-WAM den speziellen Anforderungen des jeweiligen Anwendungsgebiets angepaßt worden sind, ohne dabei aber den WAM-Prinzipien untreu geworden zu sein.

Damit ist die Klärung der Fragen „was ist WAM eigentlich?" und „welche WAM ist *die* WAM?" möglich: Der Kern von WAM sind die WAM-Prinzipien und die Entwicklungsdokumentgrundtypen. Alle weiteren Eigenschaften sind das Resultat von Anpassungen an spezielle Anwendungsgebiete und somit „Bindestrich-WAMs" wie Gebos-WAM, Leitmotivanalyse-WAM und Labor-WAM. WAM selbst ist somit keine fertige Methode, die unmodifiziert für jedes Anwendungsgebiet eingesetzt werden kann, sondern vielmehr ein *adaptierbarer Methodenkern*, der angibt, wie er einem Anwendungsgebiet angepaßt werden kann. Dieser adaptierbare Methodenkern enthält sowohl einige konkrete Verfahrensschritte als auch eine Beschreibung der ihnen zugrundeliegenden Prinzipien, so daß für Situationen, die im Methodenkern nicht berücksichtigt sind, in Übereinstimmung mit den Prinzipien eigene Verfahrensschritte entwickelt werden können.[45]

Um diese Deutung der WAM zu illustrieren, wird im Folgenden der „adaptierbare Methodenkern WAM" zusammen mit seinen fachspezifischen Ausprägungen Gebos-WAM und Leitmotivanalyse-WAM vorgestellt.

---

[45] Gryczan stuft WAM als Methodenrahmen ein, beschreibt im weiteren Verlauf WAM aber wie eine Methode. Dazu zunächst seine Definitionen:

> „Ein *Methodenrahmen* verkörpert (1) eine grundlegende Sichtweise der Softwareentwicklung, bezieht sich auf (2) eine Klasse von Anwendungsbereichen und gibt (3) Richtlinien vor für die Auswahl von Methoden, Werkzeugen und Organisationsformen."[Gry95, S.2]
>
> „Eine *Methode* stellt einen Satz von Werkzeugen und Darstellungsmitteln zur Verfügung. Dazu beinhaltet sie eine Vorgehensweise, die den Einsatz der Werkzeuge und Darstellungsmittel als Menge von Regeln anleitet."[Gry95, S.2]

Dieser, mit „Werkzeugen" und „Darstellungshilfsmitteln" sehr auf die Mittel zur Umsetzung abstellenden Methoden-Definition möchte ich eine allgemeinere Definition aus einem Lexikon hinzufügen:

> „Methode [griech. méthodos ›Weg‹, ›Gang einer Untersuchung‹, eigtl. ›Weg zu etwas hin‹], …, ein nach Gegenstand und Ziel planmäßiges (methodisches) Verfahren, …"[Bro91, Bd. 14, S. 532]

Ein Plan ist gemäß Gryczan wiederum

> „ein Hilfsmittel ... das in einer Situation vom handelnden Subjekt ... verwendet [wird], um punktuelle Handlungsanweisungen zu geben."[Gry95, S.225]

Der Gegensatz zwischen Methodenrahmen und Methode stellt sich demnach wie folgt dar:

- Ein Methodenrahmen macht Aussagen über Methoden, ohne selbst konkrete Handlungsanweisungenzu geben.
- Eine Methode gibt konkrete Handlungsanweisungen, ohne über sich selbst zu reflektieren.

Der „adaptierbare Methodenkern" ist meiner Ansicht nach ein geeigneter Ausweg aus den begrifflichen Unsicherheiten um WAM, da er quer zu den Begriffsebenen „Methodenrahmen" und „Methode" liegt: Er gibt konkrete Handlungsanweisungen in einigen Fällen (Methoden-Anteil) und sagt, in welchem Sinne Erweiterungen und Anpassungen vorgenommen werden sollten (Methodenrahmen-Anteil).

Der adaptierbare Methodenkern WAM

1. WAM-Leitbild, -Metaphern und -Entwurfsmuster. Das Leitbild der Entwicklung ist[46] der Arbeitsplatz für selbstbestimmte menschliche Tätigkeit.

Das bestehende Anwendungssystem mit seinen Handlungsabläufen, Materialien, Werkzeugen, Automaten und Umgebungen wird entsprechend den gleichnamigen Metaphern bruchlos in ein Softwaresystem mit Softwarematerialien, -werkzeugen, -automaten und -umgebungen überführt.

Bei der Gestaltung des Softwaresystems kommen die Entwurfsmuster Werkzeug-Material-Koppelung, Werkzeug-Kombination, Funktions-Interaktions-Trennung, Beobachtermechanismus, dynamische Subwerkzeug-Anbindung, Materialbehälter, Materialverwalter, Werkzeugverwalter und Ereignisverwalter zum Einsatz.

S1
Lösungsstruktur orientiert sich an Problemstruktur

2. Vorgehensweise. Die Ermittlung der Bestandteile und Abläufe des bestehenden Anwendungssystems sowie des zukünftigen Softwaresystems erfolgt durch intensive zyklische Kommunikation zwischen Entwicklern und Anwendern auf Basis von zahlreichen, kleinen[47], in Anwendersprache verfaßten Dokumenten. Analyse (**A**) und Synthese (**S**) sind keine getrennten Phasen sondern können sich gegenseitig durchdringen, wobei lediglich zu Beginn einer Entwicklung stets ein Analysezyklus und zum (vorläufigen) Schluß stets ein Synthesezyklus stehen muß (in BNF: $\mathbf{A^+(SA^*)^+S^+}$). Analysezyklen werden solange durchgeführt, bis die Anwender ihr Arbeitsgebiet richtig in den Entwicklungsdokumenten wiedergegeben sehen, und bestehen aus:

A2 + S2
viele kleine + präzis formulierte Elemente

- Interview mit den Anwendern mit Aufzeichnungen durch die Entwickler
- Abfassung oder Veränderung von Entwicklungsdokumenten in Anwendersprache durch die Entwickler
- Vorlage der Entwicklungsdokumente bei den Anwendern mit Kritik durch die Anwender und deren Aufzeichnung durch die Entwickler

Synthesezyklen werden solange durchgeführt, bis zwischen Anwendern und Entwicklern ein Einverständnis über das in den Entwicklungsdokumenten wiedergegeben zukünftige Softwaresystem existiert und bestehen aus:

- Entwurf von das zukünftige System beschreibenden Entwicklungsdokumenten durch die Entwickler in für die Anwender verständlicher Weise
- Vorlage der Entwicklungsdokumente bei den Anwendern mit Kritik durch die Anwender und deren Aufzeichnung durch die Entwickler
- Abfassung oder Veränderung der Entwicklungsdokumente in für die Anwender verständlicher Weise durch die Entwickler

In Analysezyklen werden die Entwicklungsdokumente Glossar und Szenarios, in Synthesezyklen Glossar, Systemvisionen und Prototypen modifiziert.

Ein **Glossar** ist eine alphabetische Aufstellung der relevanten Begriffe des bestehenden und des zukünftigen Systems.

A1 + A3
ausreich. viele + kausal verknüpfte Elemente

Ein **Szenario** beschreibt einen Auschnitt des bestehenden Anwendungssystem. Die Summe aller Szenarios deckt als wesentlich erachteten Teile des zu analysierenden Systems ab.

Eine **Systemvision** beschreibt einen Ausschnitt des zukünftigen Softwaresystems. Die Summe aller Systemvisionen deckt einen den Szenarios entsprechenden Teil des zukünftigen Systems ab.

Ein **Prototyp** ist eine ablauffähige Teilbeschreibung des in den Systemvisonen festgehaltenen zukünftigen Softwaresystems, aus denen es in Synthesezyklen langsam hervorgeht. In einem begleitenden Textdokument[48]wird festgehalten, welche Systemvisionen dem Prototyp zugrundeliegen und welche implementationsbedingten Veränderungen sich gegenüber den Systemvisionen ergeben.

Gebos-WAM

Die Gebos-Methode wird ausführlich in [GZ95] beschrieben. Spezielle Ausprägungen sind:

[46] im Vorgriff auf das Ergebnis dieses Kapitels
[47] im Vorgriff auf ein Teilergebnis von Kapitel 3.2.2: Bei Verwendung der dort entwickelten Anpassungsrichtlinien ergeben sich Beschreibungen komplexer Systeme, die eher aus vielen kleinen als aus wenigen großen Entwicklungsdokumenten bestehen.
[48] im Vorgriff auf das Ergebnis von Kapitel 3.3.2

- Die **Bildung von Teams**, um die Kontinuität der Projektkultur bei ausgedehnten Projektlaufzeiten zu sichern,[49]
- Die **Bildung von Teams in Form von Arbeitskreisen** aus Anwendern und Entwicklern, um für Analyse- und Synthesezyklen einen konstanten personellen Rahmen zu etablieren,[50]
- Die **Bildung von Teams in Form von Gruppen** aus Entwicklern, um Entwicklungen in verschiedenen Projektsparten oder sogar mehreren parallel stattfindenden Projekten zu koordinieren,[51]
- **Zusätzliche Autor-Kritiker-Zyklen** zwischen Teams, um zu gewährleisten, daß alle Entwicklungen sich an den WAM-Prinzipien orientieren,[52]
- Die **Revisionsfähigkeit ausgewählter Dokumente**, um den Anforderungen der DV-Revision in großen Organisationen zu genügen,[53]
- Die Bildung von **Untertypen zu Entwicklungsdokumenten**, um die verschiedenen Aspekte des bestehenden und des zukünftigen Anwendungssystems zu verdeutlichen (Überblicksszenarios, Aufgabenszeanrios, Handlungsstudien; Gesamtvisionen (Überblicksvisionen, Einbettungsvisionen), Ablaufvisionen (fachliche AVen, technische AVen, Handhabungsvisionen) und Komponentenvisionen (Werkzeugvisionen, Automatenvisionen, Materialvisionen)),[54]
- Die Verwendung von **Referenzlinien und Projektstadien**, um komplexe Projekte anhand qualitativer und zeitlicher Kriterien planen zu können.[55]

Leitmotivanalyse-WAM

Die Leitmotivanalyse-WAM hat sich im Laufe des in dieser Arbeit beschriebenen Entwicklungsprozesses herausgebildet. Diese Beschreibung stellt lediglich eine Momentaufnahme dar, da sich die Methode vermutlich im zukünftigen Projektverlauf -wie auch die Gebos-WAM- dem Anwendungsgebiet noch weiter anpassen wird. Zunächst werden die auf den Randbedingungen beruhenden speziellen Ausprägungen aus Kapitel 3.1.3 und 3.2 zusammengetragen. Danach werden sie analog zur Gebos-WAM verallgemeinert dargestellt.

Randbedingung 1. Die aktiv am Entwicklungsprozeß beteiligten Personen waren jeweils ein Anwender (zwei nacheinander) und ein Entwickler.

⇒Der Entwickler war Interviewer, Moderator und Protokollant in einer Person.

Dies ist ein Sonderfall der normalen WAM und zog keine methodischen Änderungen nach sich.

Randbedingung 2. Der erste Anwender hatte keine Rechnererfahrung.

⇒Zur Bewertung der Systemvisionen und Prototypen war ein Anwenderwechsel notwendig.*(3,1)*[56]

---

[49] vgl. [GZ95, S.12]
[50] vgl. [GZ95, S.30]
[51] vgl. [GZ95, S.15]
[52] vgl. [GZ95, S.19]
[53] vgl. [GZ95, S.23]
[54] vgl. [GZ95, S.49ff]
[55] vgl. [GZ95, S.41ff]

Dies ist eine ungünstige Konstellation bei der Entwicklung nach WAM, aber ohne Anwenderwechsel hätte das Projekt ganz auf qualifizierte Anwenderkritik verzichten müssen.

Randbedingung 3. Der Entwickler hatte bereits einige eigene Vorstellungen vom zukünftigen System.

⇒Die interne Dokumentstruktur der Systemvisionen wurde im Rahmen der normalen WAM um die Rubrik „Weitere Ideen“ erweitert.*(4,5)* Die eigenen Vorstellungen wurden dem Anwender nicht aufgezwungen, sondern in den Systemvisionen als Option vorgeschlagen.*(3,4), (4,1)*

Eine Dokumentstrukturerweiterung ist für WAM in [GKZ93] (von 1993) nicht vorgesehen, wird aber für die Gebos-WAM in [GZ95, S.50] (von 1995) ausdrücklich erlaubt.

Randbedingung 4. Der zu unterstützende Prozeß der Erstellung *vollständiger* Leitmotivanalysen und Leitmotivtabellen war bei Projektbeginn nicht bekannt. Wirklich vollständige Leitmotivtabellen wurden wegen des riesigen Umfangs des *„Ring des Nibelungen“* nie erstellt. Im 19. Jahrhundert wurde lediglich versucht, Tabellen zusammenzustellen, in denen jedes Leitmotiv wenigstens einmal vorkam.*(Kapitel 3.1.3)*

⇒Um die Tätigkeit eines Leitmotivanalysators unterstützen zu können, mußte also zunächst die Tätigkeit selbst rekonstruiert werden. Während das Markieren von Leitmotiven in der Partitur auch heute noch ausgeübt wird und somit mit den normalen WAM-Prozeduren ermittelt werden konnte, wird das Erstellen von vollständigen Leitmotivtabellen nicht mehr durchgeführt bzw. wurde nie durchgeführt. Es bestand aber eine vage Vorstellung beim Anwender, wie diese Tätigkeit ausgeübt werden könnte und wie deren Ergebnis aussehen sollte. Die existierenden Leitmotivtabellen waren nur bedingt als Vorbilder geeignet, da sie nicht für die musikwissenschaftliche Analyse sondern als Notenvorlagen zum Nachspielen der Leitmotive gedacht waren.

Um die vage Vorstellung des Anwenders im wahrsten Sinne des Wortes zu konkretisieren, sollte er die Arbeitsmittel nennen, die er für eine in seinem Sinne ideale Durchführung einer Analyse benötigen würde. Neben den leicht beschaffbaren Gegenständen (Partitur, Bleistift, unvollständige Leitmotivtabelle aus dem 19. Jahrhundert, Tonaufnahme des *„Ring“* und Abspielgerät) wurde hier ein Leitmotivnotizzettel genannt, in den alle in der Partitur gefundenen Leitmotive so vollständig wie möglich eingetragen werden können.*(1,1)* Die Summe dieser Notizen würde dann, bei noch festzulegender Formatierung, eine wirklich vollständige Leitmotivtabbelle darstellen.

Dieser historische Arbeitsgegenstand mußte nachgebildet werden, um die Tätigkeit der vollständigen Leitmotivanalyse zu rekonstruieren und mit den Mitteln der bestehenden WAM analysieren zu können. Da es nicht zu erwarten war, daß der Anwender einen Gegenstand, mit dem er noch nie gearbeitet hatte, auf Anhieb vollständig und in allen Details spezifizieren konnte, wurde vom Entwickler in Anlehnung an die WAM-Prinzipien ein enger Autor-Kritiker-Zyklus gewählt, um den nachgebildeten (ektypischen) Arbeitsgegenstand zu entwerfen, zu bewerten und zu verbessern. Der erste Entwurf wurde nach den Angaben des Anwenders erstellt. Er nannte alle Bestandteile eines Leitmotivnotizzettels, die ihm notwendig erschienen.*(Anfangswissen,Dokumente)* Ein solcher Leitmotivnotizzettel wurde dann vom Entwickler angefertigt.*(1,1)* Um die Brauchbarkeit dieses nachgebildeten Gegenstands beurteilen zu können, wurde der Anwender gebeten, mit dessen Hilfe die Tätigkeit der Leitmotivanalyse auszuüben. Dazu wurden möglichst unterschiedliche Partiturstellen ausgewählt (zunächst

[56] 3.Zyklus, 1.Entwicklunsgschritt

eine rein instrumentale Ouvertüre, dann eine dramatische Handlung). Bereits beim ersten Einsatz im Rahmen eines normalen Interviews wurden die prinzipielle Eignung des Zettels aber auch erhebliche Mängel festgestellt und benannt. Die Analysearbeit wurde fortgesetzt, indem der Inhalt fehlender Rubriken auf den Rand oder an anderen Stellen auf dem Zettel vermerkt wurde.*(1,2)*

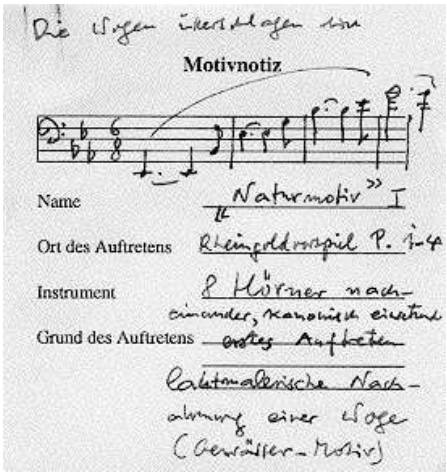


Abb. 3.3.1-1 Leitmotivnotizzettel aus dem 1. Zyklus

Der Zettel ist zwar prinzipiell benutzbar, aber einige Informationen passen räumlich nicht in ihre Rubrik (Partiturexzerpt, Grund des Auftretens) oder könnten noch weiter differenziert werden (Ort des Auftretens, Instrument).

Die erkannten Mängel wurden vom Entwickler in einer zweiten Version des Leitmotivnotizzettels behoben und der Zettel dem Anwender im nächsten Interview erneut als Arbeitsmaterial zur Verfügung gestellt.*(1,3)* Dabei trat nur noch eine weitere Unzulänglichkeit zu Tage,*(2,1)* die nach dem Interview vom Entwickler abgestellt wurde.*(2,2)*

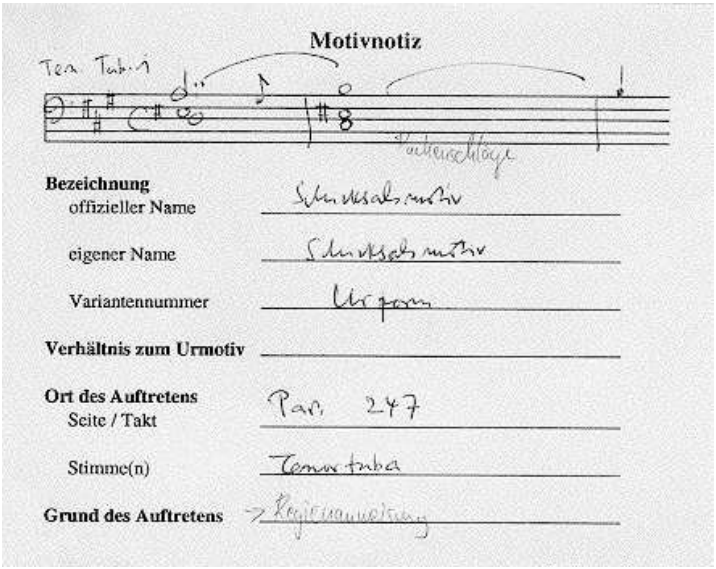


Abb. 3.3.1-2 Leitmotivnotizzettel aus dem 2. Zyklus

Die Änderungswünsche, die sich aus der Arbeit mit dem Zettel im 1.Zyklus ergeben haben, wurden umgesetzt. Lediglich die die Harmonisierung und die rhythmische Komponente ausführenden Instrumente können noch nicht notiert werden.

In drei Analysezyklen (erste Spezifikation im Vorgespräch, Einsatz während des ersten und zweiten Interviews) wurde also das nicht mehr existente Arbeitsmittel „Leitmotivnotizzettel“ gemäß den Angaben des Anwenders nachgebildet. Dadurch konnte die Tätigkeit der *vollständigen* Leitmotivanalyse, die am *„Ring“* noch nie ausgeübt worden war, (re)konstruiert werden. Diese Tätigkeit konnte dann wiederum in normalen WAM-Analysezyklen, die in die Rekonstruktionszyklen integriert waren, untersucht werden.

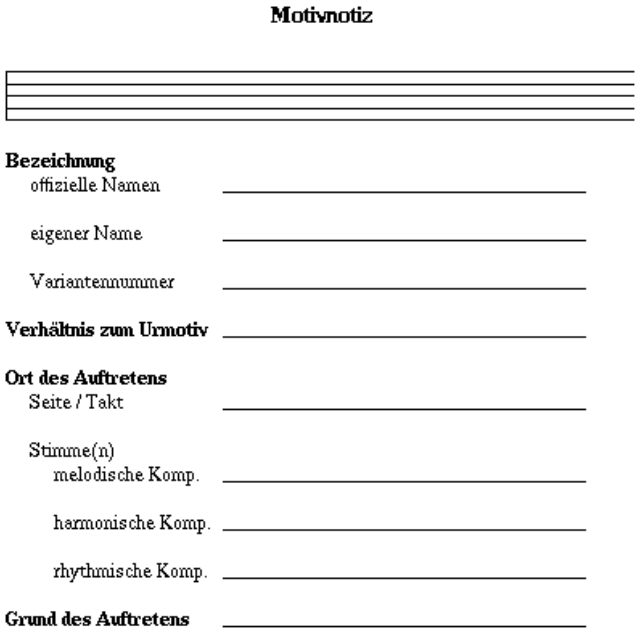


Abb. 3.3.1-3 Leitmotivnotizzettel aus dem 3. Zyklus
Die Änderungswünsche, die sich aus der Arbeit mit dem Zettel im 2. Zyklus ergeben haben, wurden umgesetzt. Seitdem mußten keine zusätzlichen Erweiterungen vorgenommen werden.

Das Verfahren mit Ektypen könnte allgemein einsetzbar sein, um nicht (mehr) ausgeübte Tätigkeiten zu rekonstruieren, wenn zwei Bedingungen erfüllt sind:

- Der Anwender muß mindestens eine **vage Vorstellung** von der zu rekonstruierdenden Tätigkeit und dem nachzubildenden Arbeitsgegenstand haben. Ansonsten kann er den Ektypen nicht einsetzen und kritisieren.
- Die Nachbildung muß mit **vertretbarem Aufwand** durchführbar sein, um den Ektypus schnell erstellen und verändern zu können.

Zusammenfassend ist die spezielle Ausprägung der Leitmotivanalyse-WAM:

- Der Einsatz von **Ektypen**, um nicht (mehr) ausgeübte Tätigkeiten zu rekonstruieren, sofern die Anwender wenigstens eine vage Vorstellung von ihm haben und die Nachbildung mit vertretbarem Aufwand durchgeführt werden kann. Die Ektypus-Autor-Kritiker-Zyklen fallen mit den normalen Analysezyklen zusammen.

Beschränkung „qualifizierte Tätigkeit“

1 **„Leitmotivanalyse ist eine qualifizierte Tätigkeit.“** Die fachlich richtige Durchführung einer aussagekräftigen Leitmotivanalyse kann nur von einem Musikwissenschaftler vorgenommen werden. Um Musik-

wissenschaftler zu werden, muß eine Person ein Universitätsstudium, das als eine vergleichsweise hohe Qualifikation anzusehen ist, erfolgreich abschließen. Durch den erfolgreichen Verlauf dieser Arbeit ist somit bewiesen, daß WAM auch auf Tätigkeiten, die eine höhere Qualifikation als die eines Sachbearbeiters erfordern, angewendet werden kann.

2 „WAM ist sowohl auf qualifizierte als auch auf unqualifizierte Tätigkeiten anwendbar." Die Beschränkung der WAM auf qualifizierte Tätigkeiten kann sowohl vor dem Hintergrund des Gegensatzes zwischen Unterstützung und Ablaufsteuerung als auch vor dem Hintergrund des Gegensatzes zwischen hoher und niedriger Qualifikation gesehen werden. Beide Interpretationen werden hier untersucht.

Unterstützung qualifizierter Tätigkeit contra Ablaufsteuerung. Gryczan arbeitet diesen Gegensatz in [Gry95] sehr deutlich heraus. Er legt dar, daß im Rahmen der strukturierten Programmierung das für Rechenanlagen gedachte Eingabe-Verarbeitung-Ausgabe-Prinzip unzulässigerweise auf die Menschen ausgedehnt wurde, die mit Rechnern arbeiten. Daher wurde davon ausgegangen, daß es einen optimalen Weg zur Lösung jedes Problems gibt und folglich genau dieser eine Weg den Anwendern von einem Programm vorgegeben werden sollte. Die Anwender wurden zu Dateneingebern für dieses Programm degradiert, ohne dabei eigene Arbeitsweisen beibehalten zu können. Diese Programme scheiterten an ihrer Inflexibilität, sich menschlichen Arbeitsweisen anpassen zu lassen.

Im Gegensatz zur ablaufsteuernden Sichtweise verzichtet die unterstützende Sichtweise darauf, den Anwendern, die nicht als Datentypisten sondern als qualifizierte Fachleute auf ihrem Gebiet angesehen werden, eine bestimmte Arbeitsreihenfolge aufzuoktroyieren. Sie stellt den Anwendern einen Satz von Arbeitsgegenständen zur Verfügung, mit dem sie ihre Aufgaben in eigener Regie durchführen können.

Im Sinne dieser Interpretation des Qualitätsbegriffs ist WAM aufgrund ihres Werkstatt-ähnlichen Leitbilds nur für die Erstellung von Anwendungssysteme für qualifizierte „Handwerker" und nicht für „Fließbandarbeiter" gedacht, denen ihr Arbeitsrhythmus von einem Automaten in Form eines Bandes aufgezwängt wird. Die Beschränkung bliebe also bestehen.

Unterstützung hochqualifizierter Tätigkeit contra Unterstützung niedrig qualifizierter Tätigkeit. Wenn jedoch feststeht, daß Anwender in ihrer Tätigkeit lediglich unterstützt werden sollen und ihnen das Ruder nicht aus der Hand genommen werden soll, dann erscheint eine Trennung entlang des Qualifikationsgrades der Anwender zunehmend unbedeutend. Bereits durch Gryczans und Züllighovens [GZ95] und nun durch diese Arbeit ist klar geworden, daß mit WAM Wissenschaftler ebenso unterstützt werden können wie Kreditsachbearbeiter. Gryczan deutet in [Gry95, S.219] sogar an, daß auch die Tätigkeiten von Menschen auf „Funktionsarbeitsplätzen" unterstützt werden könnten. Dies sind z.B. Datentypisten, die telefonisch eingehende Bestellungen aufnehmen. Diese große Spanne - vom hochqualifiztierten Wissenschaftler zum nahezu unqualifizierten Datentypisten - zeigt, daß der Qualifikationsgrad der Anwender keine Rolle spielt, solange sie selbstbestimmt arbeiten können. Für eine Systementwicklung nach WAM ist ausschließlich relevant, daß die zu unterstützende Tätigkeit von Menschen ausgeübt wird, denn nur Menschen können in den für WAM fundamentalen Autor-Kritiker-Zyklen partizipieren.

Im Sinne dieser Interpretation des Qualitätsbegriffs ist WAM also sowohl für hoch- als auch auch für niedrig qualifizierte Tätigkeiten geeignet, so daß die Schranke „Qualifikation" entfallen könnte.

Um die beiden unterschiedlichen interpretationsabhängigen Ergebnisse bezüglich der Schranke „Qualifikation“ miteinander in Einklang zu bringen, sollte die Schrankenbezeichnung so geändert werden, daß der Charakter der tatsächlichen Beschränkung zum Vorschein kommt: „Selbstbestimmung“: WAM kann zur Unterstützung hoch- und niedrig qualifizierter Tätigkeiten eingesetzt werden, solange die Anwender selbstbestimmt arbeiten können.

Beschränkung „kooperative Tätigkeit“

1 **„Leitmotivanalyse ist eine Einzelplatzarbeit.“** Die Leitmotivanalyse wird von nur einem Musikwissenschaftler zur Zeit durchgeführt. Die Arbeitsunterlagen können lediglich an einen Kollegen, der auch alleine arbeiten wird, weitergegeben werden. Das Anwendungsgebiet liegt also innerhalb der WAM-Beschränkungen.

2 **„WAM ist für kooperative Tätigkeiten einsetzbar.“** Gryczan geht in [Gry95] ausführlich auf die Möglichkeiten der Unterstützung von kooperativer Tätigkeit mit Hilfe von WAM ein und kommt zu dem Schluß, daß mindestens alle serialisierbaren kooperativen Tätigkeiten mit WAM unterstützt werden können. Gleichzeitige Bearbeitung von Materialien durch mehrere Anwender sei nach wie vor ein offenes Problem.

Die Beschränkung auf kooperative Tätigkeiten bleibt also unangetastet. Natürlich kann auch die Untermenge der Einzeltätigkeiten unterstützt werden.

Beschränkung „menschliche Tätigkeit“

1 **„Leitmotivanalyse ist eine menschliche Tätigkeit.“** Die Leitmotivanalyse wird und kann nur von menschlichen Musikwissenschaftlern durchgeführt werden, da sie musisches Empfinden vorraussetzt.*(Kapitel 3.1.3)* Der Entwicklungsprozeß des Leitmotivanalysewerkzeugs liegt also in dem von der WAM vorgegebenen Rahmen.

2 **„WAM bleibt auf menschliche Tätigkeiten beschränkt.“** Ein elementarer Bestandteil der WAM ist die Kommunikation zwischen Anwendern und Entwicklern. Nur wenn beide Gruppen aus Menschen bestehen, ist eine gegenseitige Verständigung möglich. Demzufolge bleibt die Beschränkung „menschlich“ bestehen.

Zusammenfassung

1 **„WAM ist eine Universalmethode für qualifizierte und kooperative menschliche Tätigkeit.“** In diesem Kapitel wurde durch Einzelüberprüfung der Eigenschaften des Anwendungsgebietes Leitmotivanalyse an den von WAM vorgegebenen Schranken bewiesen, daß mit ihr so unterschiedliche qualifizierte Tätigkeiten wie die eines Sachbearbeiters als auch die eines analysierenden Musikwissenschaftlers unterstützt werden können. Aufgrund der beträchtlichen Unterschiede dieser beiden Anwendungsgebiete liegt der Schluß nahe, daß es sich bei WAM um eine Universalmethode für alle Anwendungssysteme im Rahmen der Schranken „qualifizierte und kooperative menschliche Tätigkeit“ handelt.

Dieser Beweis steht und fällt mit der Definition des Methodenbegriffs. Versteht man darunter ein festes Verfahren, das unmodifiziert angewendet werden muß, dann ist WAM keine Methode. Erlaubt man Anpassungen im Geiste der Methode, dann ist WAM eine Methode. Und mehr:

2 „WAM ist ein adaptierbarer Methodenkern zur Unterstützung selbstbestimmter menschlicher Tätigkeit." Bei der Entwicklung des Leitmotivanalysewerkzeugs ist deutlich geworden, daß WAM ihren in [GKZ93], [Gry95], [RZ95] und [GZ95] beschriebenen Rahmen sprengt. WAM ist nicht (nur) *eine* mit der Gebos-WAM deckungsgleiche Methode, sondern vielmehr der Kern einer Familie von Methoden, die sich um die WAM-Prinzipien ranken. Zwar existiert neben der Gebos-WAM bisher nur die Leitmotivanalyse-WAM als spezielle Anpassung an ein Anwendungsgebiet, aber die Allgemeinheit der WAM-Prinzipien läßt vermuten, daß noch viele weitere Familienmitglieder hinzukommen werden.

Da wahrscheinlich nicht alle neuen WAM-Ausprägungen hundertprozentiges Neuland betreten werden, möchte ich anregen, WAM vollständig in Form einer Pattern-Language[57] niederzulegen. Jede neue Spezial-WAM könnte dann einfach in der Form „WAM-Kern plus Muster 4, 12 und 28" spezifiziert werden. Die Grundlage für eine solche Spezifikation könnte [RZ95] sein. Die dort aufgeführten Metaphern und Entwurfsmuster müßten um zwischenzeitlich hinzugekommene Metaphern („Automat") sowie bisher schon verwendete, aber nicht als Muster formulierte WAM-Elemente wie Verfahrensmuster (z.B. „Autor-Kritiker-Zyklus", „Entwicklunsgdokumente") und Organisationsmuster („Architekturgruppe") erweitert werden. Damit läge WAM in übersichtlicher Form vor und könnte noch leichter gesichtet und angepaßt werden.

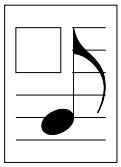
Des weiteren wurde die Bindung von WAM an qualifizierte Tätigkeiten relativiert. Ergänzend zur Annahme Gryczans, daß WAM auch zur Unterstützung niedrig qualifizierter Tätigkeiten eingesetzt werden könnte, wurde am Beispiel des analysierenden Musikwissenschaftlers gezeigt, daß WAM ebenso für sehr hochqualifizierte Tätigkeiten verwendbar ist. Somit tritt die Bedeutung der Qualifikation der Anwender zusehends hinter die Art und Weise zurück, in der sie ihre Aufgaben erledigen: selbstbestimmt.

Schließlich bleibt die Frage zu klären, wie WAM umschrieben werden sollte. Die kanonische Form müßte lauten: „Kern einer adaptierbaren Methode zur Entwicklung von Softwaresystemen zur Unterstützung qualifizierter und unqualifizierter selbstbestimmter kooperativer und nichtkooperativer menschlicher Tätigkeit". Da dies sehr langatmig ist, sollten wenigstens die Satzbestandteile weggelassen werden, die zusammen mit ihrem Komplement Teil des Satzes sind: also „kooperativ" und „nichtkooperativ" sowie „qualifiziert" und „unqualifiziert". Es verbleibt: „Kern einer adaptierbaren Methode zur Entwicklung von Softwaresystemen zur Unterstützung selbstbestimmter menschlicher Tätigkeit".

---

[57] Der Begriff wurde von Christopher Alexander in seinen Werken „*The Timeless Way of Building*"[Ale79] und „*A Pattern Language*"[AIS79] geprägt. Er bezeichnet eine Sammlung von knappen - Muster genannten - Konstruktionsregeln in „wenn-dann"-Form. Zusätzlich werden zu jedem Muster Verweise auf verwandte Muster und ein Beispiel angegeben. Der Begriff „Pattern Language" hat sich etabliert, obwohl der Mustersammlung keine linguistischen Eigenschaften anhaften.

3.3.2 Dokumente



In Kapitel 3.1.2 wurden die Basisdokumenttypen und die existierende Entwicklungsumgebung für WAM beschrieben. Kapitel 3.2 zeigte episodisch, wie beide durch die konkrete Verwendung im Projekt verändert wurden. Hier werden diese Veränderungen und Veränderungswünsche zusammengefaßt und untersucht. Die entstandenen Dokumenttypen werden mit den ursprünglichen Basistypen verglichen. Daraus werden Richtlinien zur Anpassung von Dokumenttypen an ein Anwendungsgebiet destilliert. Die Mängel der existierenden Umgebung werden zur Grundlage einer Spezifikation einer idealen Entwicklungsumgebung für WAM umgeformt. Diese wird zunächst völlig frei von existierenden Hard- und Softwarebeschränkungen beschrieben und dann auf ihre Umsetzbarkeit hin überprüft. Da einige Teile bereits realisiert worden sind, wird schließlich die These

negative Problembeschreibung

Eine geeignete Entwicklungsumgebung für WAM fehlt, ist aber in greifbare Nähe gerückt

bestätigt.

Dokumente und Umgebung werden erneut parallel behandelt. Zunächst werden die jeweiligen Ergebnisse aus dem Entwicklungsprozeß verdichtet, dann die Schlüsse daraus gezogen und im Falle der Entwicklungsumgebung deren Umsetzbarkeit geprüft.

Ergebnisse des Entwicklungsprozesses

Dokumente. Wie in jedem Entwicklungsprozeß nach WAM wurden auch im Leitmotivanalyse-Projekt die Dokumenttypen den speziellen Anforderungen des Anwendungsgebiets angepaßt. Dabei entstanden auch – unter gleichem oder neuem Namen – von Gebos-WAM abweichende Typen. Dies geschah nicht auf einen Schlag zu Projektbeginn, sondern die Veränderungen schälten sich langsam heraus, als die Struktur der Basisdokumenttypen die Gegebenheiten des Entwicklungsprozesses nicht mehr verzerrungsfrei widerspiegeln konnten. Diese Konflikte und die daraus entstandenen angepaßten Dokumenttypen (mit *kursiv hervorgehobenen* Veränderungen gegenüber den Basistypen) sind hier zusammengefaßt. Im nächsten Abschnitt werden aus den Gemeinsamkeiten Anpassungsrichtlinien abgeleitet.

Strukturkonflikte

- Dem **Glossar** mangelt es bei einer größeren Anzahl von Einträgen an Übersichtlichkeit.*(2,3)*[58]
- Bei **Szenarios** sagt die Kontextbeschreibung nichts über Sinn und Zweck des Arbeitsvorangs aus.*(1,6)* Einem Begriff in der Ablaufbeschreibung kann nicht angesehen werden, ob er im Glossar aufgeführt ist oder nicht.*(1,5)* Die Rubrik „Verwendete Werkzeuge und Materialien“ suggeriert eine direkte Übertragbarkeit von Werkzeugen und Materialien der Anwendungswelt in Softwarewerkzeuge und -materialien.*(1,6)*
- Der einfache Dokumentyp **Systemvisionen** ist nicht gut geeignet um sowohl dynamische als auch statische Aspekte des zukünftigen Softwaresystems zu beschreiben.*(4,3)* Besonders bei der statischen Beschreibung

[58] 2.Zyklus, 3.Entwicklungsschritt

eines Werkzeugs sind viele Rubriken schlecht oder gar nicht brauchbar („Verwendete Softwarewerkzeuge“, „Zugrundeliegende Szenarios“, „Kontextbeschreibung“, „Beschreibung des neuen Arbeitsvorgangs“) während Rubriken zum Beschreiben der Werkzeugoperationen ganz fehlen.*(4,5)*

Die Entwicklungsdokumenttypen der Leitmotivanalyse-WAM

Das **Glossar** enthält

- eine Liste von Fachbegriffen *in alphabetischer Reihenfolge. Zu jedem Begriff gibt es eine aus einem Satz bestehende Kurzbeschreibung, an die sich die ausführliche Beschreibung anpaßt. Softwarewerkzeuge besitzen anstelle der ausführlichen Beschreibung einen Verweis auf ihre Werkzeugvision.*

Zur besseren Übersicht erscheint die Begriffsbezeichnung eines Eintrags fett.

Ein **Szenario** enthält

- eine Liste der verwendeten *Arbeitsgegenstände,*
- eine kurze Beschreibung des Kontexts, in dem der Arbeitsvorgang stattfindet *und aus der Sinn und Zweck des Vorgangs ersichtlich sind,*
- eine möglichst epische Beschreibung des Arbeitsvorgangs in der Sprache der Anwendungswelt,
- Verweise auf benachbarte Dokumente (Glossar, angrenzende Szenarios und auf dem Szenario aufbauende Systemvisionen).

Systemvisionen *gibt es in der Ausprägung einer Ablaufvision, Werkzeugvision oder einer Übersichtsvision. Eine Ablaufvision beschreibt die dynamischen, eine Werkzeugvision die statischen Aspekte des zukünftigen Softwaresystems. Eine Übesichtsvision beschreibt die Arbeit eines Teils des zukünftigen Systems im Großen. Ablaufvisionen und Übersichtsvisionen sind strukturell identisch, wobei bei der letzteren die Kontextbeschreibung sehr ausführlich ist.*

Eine **Ablaufvision** oder **Übersichtsvision** enthält

- den Namen des zugrundeliegenden Szenarios mit Begründung,
- eine Beschreibung der verwendeten Operationen von Softwarewerkzeugen im antizipierten Arbeitsvorgang,
- Verweise auf benachbarte Systemvisionen oder Werkzeuge bezüglich des Arbeitsablaufs,
- eine Bildschirmskizze des Softwarewerkzeugs, *aus der die Anordnung, nicht aber die exakte Gestalt der im neuen Arbeitsablauf verwendeten Softwarewerkzeuge hervorgeht,*
- *eine Beschreibung des Kontexts, in dem der antizipierte Arbeitsvorgang stattfindet,*
- eine szenarioartige Beschreibung der neuen Arbeitsabläufe,
- Verweise auf benachbarte Dokumente (Glossar, zugrundeliegende Szenarios und benachbarte Systemvisionen).

Eine **Werkzeugvision** enthält

- *die Namen der Ablaufvisionen, in denen mit dem Werkzeug gearbeitet wird,*
- *eine Liste der möglichen Operationen,*
- eine *detaillierte* Bildschirmskizze des Softwarewerkzeugs,
- *eine Kurzbeschreibung,*
- eine *detaillierte* Beschreibung, *in der auf alle in „Möglichen Operationen" aufgezählten Operationen ausführlich eingegangen wird,*
- eine Beschreibung der Verfahrensvorschriften und/oder Algorithmen, nach/mit denen das Softwarewerkzeug arbeiten soll,
- *weitere Ideen der Entwickler für die Gestaltung des Werkzeugs,*
- Verweise auf benachbarte Dokumente (Glossar, zugrundeliegende Szenarios und benachbarte Systemvisionen).

Ein **Prototyp** besteht aus

- einer Auswahl von unter bestimmten Aspekten ausimplementierten Softwarewerkzeugen und -materialien nach Maßgabe ausgewählter Systemvisionen,
- *einem Prototypbegleitdokument, das die folgenden Elemente enthält:*
  - *die zugrundeliegenden Systemvisionen,*
  - *eine Liste aller Werkzeuge mit ihrem Erscheinungsbild im Prototyp und den jeweiligen implementationsbedingten Abweichungen von den idealen Werkzeugvisonen,*
  - *eine Möglichkeit zum Starten des ablauffähigen Prototyps,*
  - *Verweise auf benachbarte Dokumente (Glossar, zugrundeliegende Szenarios und benachbarte Systemvisionen).*

Alle Verweise auf Glossareinträge und andere Dokumente sind in allen Dokumenttypen optisch hervorgehoben.

- Zu **Prototypen** fehlen entsprechende Textdokumente, in denen auf die Abweichungen von dem in den Systemvisionen entworfenen Ideal und andere implementationsbedingte Besonderheiten hingewiesen wird.*(5,3)*

Entwicklungsumgebung. Die bereits in Kapitel 3.1.2 identifizierten Schwachpunkte der existierenden Entwicklungsumgebung wurden gleich zu Beginn des Entwicklungsprozesses durch die Umsetzung der Dokumente als Hypertexte entschärft.*(1,4)* Die verbleibenden und neu aufgetauchten Probleme werden hier zusammengetragen. Auf ihrer Grundlage wird im nächsten Abschnitt eine ideale Entwicklungsumgebung entworfen. Zugleich wird anhand dieser Aufzählung klar, daß der erste Teil der These „***Eine geeignete Entwicklungsumgebung für WAM fehlt****, ist aber in greifbare Nähe gerückt"* den Tatsachen entspricht.

- Die starke verweisbedingte **Vernetzung** der Dokumente kann auf Papier nicht angemessen realisiert werden. Verweise sind nicht auf einen Blick erkennbar und können durch Blättern nur bedingt verfolgt werden.*(Kapitel 3.1.2)*
- Die **Erstellung von Hypertexten in HTML** wird mit einem Texteditor bald sehr unübersichtlich.*(3,4)*
- Der **Übergang von Textdokumenten zu Softwaredokumenten** gestaltet sich als sehr aufwendig, da WAM- und *Sniff*-Dokumente inhaltlich und strukturell stark voneinander abweichen.*(Kapitel 3.1.2)*
- Die **Erstellung der graphischen Benutzungsoberfläche** der Softwarewerkzeuge muß zeitraubend und unübersichtlich von Hand in C++ erfolgen.*(5,4)*

Folgerungen aus den Ergebnissen des Entwicklungsprozesses

Dokumente. Die im letzten Abschnitt vorgestellten konkreten Leitmotivanalyse-WAM-Dokumenttypen sind, wie auch die Gebos-WAM-Typen, nicht auf andere Projekte mit abweichenden Anwendungsgebieten übertragbar. Übertragbar sind aber die Beobachtungen über die Art und Weise, in der Veränderungsbedarf sich bemerkbar macht und in der solche Inhalt-Form Konflikte gelöst werden können. Diese werden zunächst aus der Menge der durchgeführten Veränderungen herausgefiltert und dann zu Anpassungsrichtlinien in Form eines Musters zusammengefaßt.

Sowohl bei der strukturellen Anpassung des Dokumente (z.B. des Glossarinhalts oder der Anzahl der Szenarios) an das Anwendungsgebiet als auch bei der sich eventuell daraus ergebenden strukturellen Anpassung der Dokumenttypen selbst (z.B. der Bildung neuer Dokumenttypen wie Ablauf- und Werkzeugvisionen) hat es sich im Laufe des Entwicklungsprozesses gezeigt, daß Änderungen stets vom Dokument*inhalt* ausgegangen sind.(*(1,3), (1,6), (1,7), (2,2), (2,3), (3,3), (3,5), (4,3), (4,4), (4,5), (5,3)*) Willkürliche strukturelle Änderungen, z.B. die Schaffung der Systemvision „Leitmotiv betrachten"*(3,3)* ohne wirkliche Grundlage in den Szenarios waren nicht von langer Dauer.*(4,4)* Änderungen (z.B. die Aufspaltung eines großen Szenarios in zwei kleinere oder die Streichnung einer bestimmten Systemvisionrubrik) erwiesen sich erst dann als dauerhaft,

wenn sie aus einem klaren Inhalt-Form-Konflikt hervorgegangen waren,(z.B. *(1,7), (2,3), (4,3)*) der wiederum erst dann sichtbar wurde, wenn eine größere Anzahl von Dokumenten eines Typs vorhanden war.

Alle Inhalt-Form-Konflikte eines Typs konnten in gleicher Weise gelöst werden, so daß es naheliegt, die Probleme zusammen mit ihren Lösungen als allgemeine Anpassungsrichtlinien zu bezeichnen. Mit ihrer Hilfe sollte es in jedem Projekt möglich sein, aus den Basisdokumenttypen Schritt für Schritt anwendungsspezifische Spezialdokumente zu entwicklen. Falls WAM tatsächlich als Pattern Language spezifiziert werden sollte, dann wären diese Richtlinien (in „Wenn-Dann-Form“) ein weiterer Kandidat zur Ergänzung der bereits in Kapitel 3.3.1 vorgeschlagenen Muster.

Anpassungsrichtlinien zur Dokumentstrukturierung

Eine **Dokumentstrukturerweiterung** ist sinnvoll, wenn sich innerhalb einer Beschreibung (einem Glossareintrag, in einem Szenario oder einer Systemvision) mehrere Abschnitte isolieren lassen, die untereinander nur minimal (z.B. nur durch einmalige Erwähnung am Absatzanfang oder -ende) in Verbindung stehen. Die Struktur sollte um soviele weitere Exemplare desselben Typs (Glossareintrag, Szenario, Systemvision) erweitert werden, wie isolierte Abschnitte vorhanden sind. Der Inhalt der alten Beschreibung kann oft zur Kontextbeschreibung der neuen Exemplare werden. Dieser Erweiterungsbedarf wird offenbar

- durch direkte Betrachtung
- durch Verweise auf das Dokument, die sich eigentlich nur auf einen ganz speziellen Abschnitt statt auf das ganze Dokument beziehen.

Eine **Dokumentstrukturvereinfachung** ist sinnvoll, wenn sich die Inhalte mehrerer Beschreibungen (Glossareinträge, Teile eines Szenarios oder einer Systemvision) stark überlappen oder identisch sind. Die Beschreibungen sollten entweder zu einer einheitlichen Beschreibung zusammengefaßt oder entlang einer anderen Grenze getrennt werden. Dieser Zusammenlegungsbedarf zeigt sich

- durch direkte Betrachtung
- durch die Schwierigkeiten bei der Angabe eines eindeutigen Verweises in anderen Dokumenten, da alle in Frage kommenden Beschreibungen gleich gut geeignet erscheinen.

Eine **Erweiterung der inneren Dokumenttypenstruktur** ist ratsam, wenn ein wichtiger Aspekt der Beschreibungen dauerhaft nicht einer einzelnen Rubrik zugeordnet werden kann, sondern in allen Dokumenten des Typs als Anhängsel in (ggf. verschiedenen) anderen Rubriken auftaucht. Eine geeignete Rubrik sollte hinzugefügt, bzw. bei Häufungen der Anhängsel in einer bestimmten Rubrik, begrifflich erweitert werden.

Eine **Vereinfachung der inneren Dokumenttypenstruktur** ist ratsam, wenn bestimmte Rubriken über längere Zeiträume hinweg nicht genutzt werden. Sie sollten entfernt werden.

Die **Untertypisierung der Dokumenttypen** ist angezeigt, wenn sich verschiedene inhaltliche Schwerpunkte innerhalb der Dokumente eines Typs ergeben (z.B. eher dynamische und eher statische Systemvisionen). Nach einer Untersuchung der Unterschiede sollten soviele Untertypen des alten Typs gebildet werden, wie klar isolierbare Schwerpunkte vorhanden sind, wobei die neuen Untertypen den semantischen Rahmen des alten Typs vollständig abdecken sollten. Im Zweifelsfall sollten eher weniger Untertypen gebildet werden und im Verlaufe der weiteren Arbeit mit diesen beobachtet werden, ob eine weitere Untertypisierung eine Grundlage hat oder nicht. In jedem Fall sollten die alten Obertypen nicht weiter verwendet werden, da es sonst zu unerwünschten großflächigen Überlappungen käme.

Entwicklungsumgebung. Schon während des Entwicklungsprozesses wurden die WAM-Textdokumente als Hypertexte erstellt und damit zwei Hauptprobleme der bisherigen Umgebung gelöst. Damit wurde der Grundstein zu einem kompletten Hypertextdokumentationssystem gelegt. Es wird in diesem Abschnitt in seiner Idealform beschrieben, in der es auch die restlichen bisher erkannten Probleme behebt. Im nächsten Abschnitt

werden die bisherige Teilrealisierung und die Möglichkeiten zur vollständigen Umsetzung des Ideals untersucht.

Das Rückgrat des **Hypertextdokumentationssystems**[59] bilden die einheitlichen Dokumente, die das Projekt vom ersten Szenario bis zum Pilotsystem und darüber hinaus begleiten. Dabei sind nicht die Dokumenttypen selbst fixiert, was beim bei WAM ständig stattfindenden Adaptiontsprozeß auch gar nicht möglich wäre, sondern die die Art und Weise, wie die Dokumentinhalte (Rubriken, Bilder, Verweise, etc.) kodiert sind. Auf diesem Dokumentformat setzen dann die Werkzeuge auf, die die Entwickler bei ihrer Arbeit unterstützen: WAM-seitig Hypertextdokumentbetrachter, Dokumenteditoren, Dokumenttypeditoren und Dokumentstrukturbetrachter, *Sniff*[60]-seitig Klassen-, Methoden- und sonstige Komponentendokumentationen.

Das Hypertextdokumentationssystem löst alle im Entwicklungsprozeß beobachteten Probleme:

- Die Vernetzung der Dokumente wird durch Hypertexte adäquat ausgedrückt.
- Für HTML als Hypertextrealisierungssprache stehen zahlreiche Editoren und Betrachter zur Verfügung.
- Der Bruch zwischen WAM- und *Sniff*-Dokumenten entfällt entweder, weil zwei technische Dokumente zu einem zusammengelegt werden können oder weil beide von WAM- und *Sniff*- aus betrachtet werden können.

Neben dem Dokumentationssystem fehlt einzig noch ein **GUI-Builder für WAM**, um einfach und schnell Benutzungsoberflächen für Prototypen anhand der Bildschirmskizzen in den Systemvisionen erstellen zu können. Mit ihm können die Entwickler sich auf die Handhabung und die Funktionalität der Prototypen konzentrieren. Durch ihn wird die Übergangszeit von Systemvisionen zu einem ersten Prototyp wesentlich verkürzt, da der Schwerpunkt der anfänglichen Entwicklung erfahrungsgemäß auf der Gestaltung der Oberfläche liegt.

Realisierungsmöglichkeiten einer idealen Umgebung

Ein ideales Hypertextdokumentationssystem mit GUI-Builder ist keine Zukunftsmusik, sondern könnte mit vertretbarem Aufwand realisiert werden. Zunächst werden die bereits existierenden Teile und dann die „Blaupause“ für ein komplettes System beschrieben. Die dazugehörige Machbarkeitseinschätzung ist zugleich ein Beleg für den zweiten Teil der These *„Eine geeignete Entwicklungsumgebung für WAM fehlt,* ***ist aber in greifbare Nähe gerückt“***.

Entwicklungsumgebung. Der Aufwand für die Umsetzung eines **einheitlichen Hypertextdokumentationssystems** für WAM und *Sniff* muß im Verhältnis zu dessen Nutzen stehen. Obwohl die Vorteile

[59] Abb. 3.1.2-4 enthält eine schematische Darstellung.
[60] Sniff dient hier nur als ein Platzhalter für eine beliebige Programmierumgebung. Auch das Dokumentationssubsystem jeder anderen Umgebung könnte auf das einheitliche Dokumentformat abgestimmt werden.

beträchtlich wären, sollte zunächst Bestehendes auf seine Brauchbarkeit hin untersucht werden, bevor eine vollständige Neuimplementation ins Auge gefaßt wird.

Da *Sniff* im Gegensatz zu den WAM-Dokumenten bereits ein festes Dokumentationsformat besitzt, wird zunächst vorgeführt, daß die verlockende Ausdehnung des *Sniff*-Formats auf WAM-Dokumente nicht sinnvoll ist. Daraufhin wird gezeigt, warum das im Leitmotivanalyse-Projekt bereits verwendete HTML sich als Dokumentformat empfiehlt.

***Sniff*-Dokumentationssystem.** In Sniff besteht die Möglichkeit, zu jeder Klasse und jedem ihrer Bestandteile eine Dokumentation zu erstellen. Je nach Typ (Klassendokumentation, Methodendokumentation, etc.) existiert ein bestimmtes Formular mit unabänderlichen Rubriken, die bei der initialen Erstellung bereits ausgefüllt sind, soweit die zugrundeliegende Klassen- oder Methodendefintion es erlaubt. Der Entwickler kann nun den Inhalt dieser Rubriken nach seinen Bedürfnissen mit Text füllen. Zur Formatierung der Einträge kann er die Einrükkung am linken Rand verändern und ausgewählte Begriffe durch Kursivdruck hervorheben.

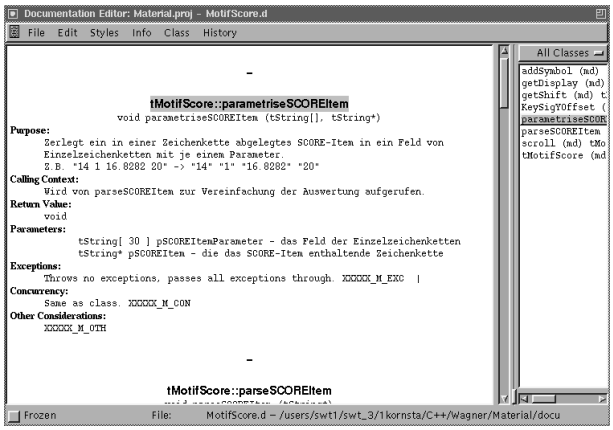


Abb. 3.3.2-1 *Sniff* Dokumentationseditor mit Methodendokumentation

```
> Member Function: tMotifScore::parametriseSCOREItem ;
  $ void parametriseSCOREItem (tString[], tString*)
> Purpose:
  | Zerlegt ein in einer Zeichenkette abgelegtes SCORE-
Item in ein Feld von Einzelzeichenketten mit je einem
Parameter.
  | Z.B. "14 1 16.8282 20" -> "14" "1" "16.8282" "20"
> Calling Context:
  | Wird von parseSCOREItem zur Vereinfachung der Auswer-
tung aufgerufen.
> Return Value:
  | void
> Parameters:
  = tString[ 30 ] pSCOREItemParameter - das Feld der
Einzelzeichenketten
  = tString* pSCOREItem - die das SCORE-Item enthaltende
Zeichenkette
> Exceptions:
  | Throws no exceptions, passes all exceptions through.
XXXXX_M_EXC  |
> Concurrency:
  | Same as class. XXXXX_M_CON
> Other Considerations:
  | XXXXX_M_OTH
```

Abb. 3.3.2-2 Methodendokumentation im *Sniff*-eigenen Format.

Das *Sniff*-Dokumentationsformat ist also für das ideale Hypertextdokumentationssystem nicht geeignet. Zwar lassen sich bestimmte semantisch festgelegte Rubriken darstellen, aber Bilder und Verweise auf andere Dokumente lassen sich nicht repräsentieren. Eine Erweiterung des *Sniff*-Formats um diese Elemente wäre möglich, wenn zugleich auch der Dokumentationsbetrachter so verändert würde, daß Bilder und Verweise dargestellt und letztere auch beliebig verfolgt werden könnten. Durch diese Erweiterungsarbeiten wäre der Vorteil, der sich aus einer direkten Verwendung des *Sniff*-Formats für die Dokumente des gesamten Entwicklungsprozesses ergeben hätte, allerdings beträchtlich geschmälert, zumal das *Sniff*-Format ohnehin nur über sehr beschränkte Formatierungsmittel verfügt und daher ausschließlich unter dem Gesichtspunkt des minimalen Änderungsaufwandes, nicht aber unter dem der minimalen Abstriche vom Idealsystem als Dokumentationssystemgrundlage in Frage käme. Günstiger wäre es in diesem Falle, gleich auf ein ausdrucksstärkeres Format umzusteigen und die Werkzeuge an dieses anzupassen.

HTML. Die **H**yper**T**ext **M**arkup **L**anguage ist eine Sprache zur Darstellung von Hypertexten. Sie basiert auf dem Meta-Standard für strukturierte Dokumente SGML[61] und erlaubt die Darstellung aller für das Idealsystem geforderten Elemente sowie vielfältige Formatierungsmöglichkeiten: Verweise auf andere Dokumente sowie auf bestimmte Dokumentabschnitte sind ebenso möglich wie die Integration von Bildern. In der noch in der Diskussion befindlichen Version 3.0 kann auch die Semantik von Dokumentabschnitten spezifiziert werden.[62] Zur Betrachtung von HTML-Dokumenten gibt es zahlreiche sogenannte Web-Browser, die vornehmlich zur Betrachtung des in HTML-Dokumenten abgefaßten WorldWideWebs[63] dienen. Diese Programme erlauben eine sehr komfortable Navigation durch Hypertexte mit beliebig vielen Rücksprüngen und anspruchsvoller graphischer Darstellung. Anstelle einer exakten Spezifikation dieser umfangreichen und zu den „Kontextfreien" gehörenden Sprache, wird hier die mögliche Spezifikation eines Szenarios und dessen graphische Darstellung in einem Web-Browser gezeigt.

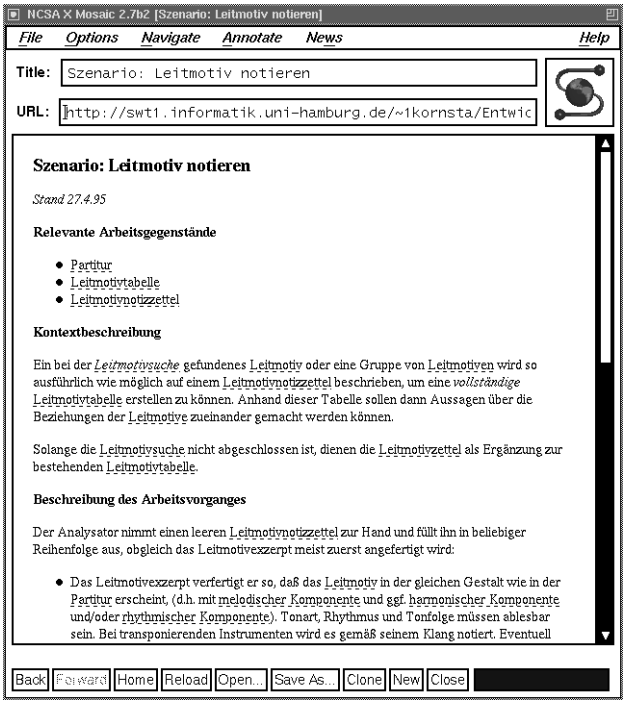


Abb. 3.3.2-3 Ein Szenario auf HTML-Basis im WebBrowser Mosaic

Verweise auf Glossareinträge, andere Szenarios oder Systemvisionen sind unterstrichen dargestellt.

```
<HTML>
<HEAD>
<TITLE>Szenario: Leitmotiv notieren</TITLE>
</HEAD>
<BODY>
<H1>Szenario: Leitmotiv notieren</H1>
<H2><I>Stand 27.4.95</I></H2>

<DIV CLASS=Relevante_Arbeitsgegenstaende>
<H3>Relevante Arbeitsgegenst&auml;nde</H3>
<UL>
  <LI><A HREF="glossar.html#Partitur">Partitur</A>
  <LI><A HREF="glossar.html#Leitmotivtabelle">Leitmotivtabelle</A>
  <LI><A HREF="glossar.html#Leitmotivnotiz">Leitmotivnotizzettel</A>
</UL>
</DIV>

<DIV CLASS=Kontextbeschreibung>
<H3>Kontextbeschreibung</H3>
Ein bei der <I><A HREF="szenario2.html">Leitmotivsuche</A></I> gefundenes <A HREF
= "glossar.html#Leitmotiv">Leitmotiv</A> oder eine Gruppe von <A HREF =
"glossar.html#Leitmotiv">Leitmotiven</A> wird so ausf&uuml;hrlich wie
m&ouml;glich auf einem <A HREF =
"glossar.html#Leitmotivnotiz">Leitmotivnotizzettel</A> beschrieben, um eine
<I>vollst&auml;ndige</I> <A HREF =
"glossar.html#Leitmotivtabelle">Leitmotivtabelle</A> erstellen zu k&ouml;nnen.
Anhand dieser Tabelle sollen dann Aussagen &uuml;ber die Beziehungen der <A HREF
= "glossar.html#Leitmotiv">Leitmotive</A> zueinander gemacht werden
k&ouml;nnen.<P>
Solange die <A HREF="szenario2.html">Leitmotivsuche</A> nicht abgeschlossen ist,
dienen die <A HREF = "glossar.html#Leitmotivnotiz">Leitmotivzettel</A> als
Erg&auml;nzung zur bestehenden <A HREF =
"glossar.html#Leitmotivtabelle">Leitmotivtabelle</A>.
</DIV>

<DIV CLASS=Arbeitsvorgangsbeschreibung>
<H3>Beschreibung des Arbeitsvorganges</H3>
Der Analysator nimmt einen leeren <A HREF =
"glossar.html#Leitmotivnotiz">Leitmotivnotizzettel</A> zur Hand und f&uuml;llt
ihn in beliebiger Reihenfolge aus, obgleich das Leitmotivexzerpt meist zuerst
angefertigt wird:<P>
<UL>
  <LI> Das Leitmotivexzerpt verfertigt er so, da&szlig; das <A HREF =
```

Abb. 3.3.2-4 Szenario im HTML-Format

Die logische Gliederung des Dokuments durch `DIV`-*tags* ist durch Fettdruck hervorgehoben.

HTML ist schon alleine durch seine vollkommene Abdeckung aller Bedürfnisse des Idealsystems die erste Wahl für dessen praktische Umsetzung. Hinzu kommt noch die große Planungssicherheit, die dadurch entsteht, daß HTML sich im starken Einsatz im WorldWideWeb befindet und somit nicht von heute auf morgen mitsamt den dazugehörigen Betrachtungswerkzeugen verschwinden wird. Ebenfalls durch die intensive Verwendung bedingt gibt es zusehends mehr Werkzeuge, die die komfortable Erstellung von HTML-Dokumenten ermöglichen. Aus all diesen Faktoren ergibt sich, daß auf HTML basierende Entwicklungsdokumente mit verhältnismäßig geringem Aufwand realisierbar wären.

---

[61] **S**tandard **G**eneralized **M**arkup **L**anguage, ISO Standard Nr. 8879:1986
[62] vgl. [Rag95]

Als positiver Nebeneffekt ergäbe sich noch die Möglichkeit für alle am Entwicklungsprozeß beteiligten Gruppen, die Dokumente mit Web-Browsern über große Entfernungen von überall her - zum Beispiel zur Gesprächsvorbereitung - zu betrachten. Durch eine zentrale Speicherung der Dokumente entfielen auch die Probleme, die sich bei der Verwaltung von verschiedenen über das ganze Projekt verteilten Versionen der Entwicklungsdokumente ergeben.

Wie gezeigt wurde, ist ein einheitliches Dokumentationssystem aufgrund der Unzulänglichkeiten des *Sniff*-Dokumentationsformats kurzfristig nicht machbar. Lediglich der WAM-spezifische Teil könnte schon jetzt verwirklicht werden. Mittelfristig ergäbe sich aber die Möglichkeit einer vollständigen Realisierung, wenn es gelänge, die Entwickler von *Sniff* von den Vorteilen einer Umstellung vom bisherigen Format auf HTML zu überzeugen.

Realisierung des WAM-Teils. Alle Exemplare der WAM-Entwicklungsdokumente wurden erstmals vollständig in HTML erstellt und konnten, wie z.B. in Abb. 3.3.2-3 zu sehen ist, mit Web-Browsern hypertextgerecht betrachtet werden: Verweise auf andere Dokumente werden durch Unterstreichung hervorgehoben und können beliebig weit verfolgt werden. So sind z.B. alle im Glossar enthaltenen Begriffe in allen Szenarios markiert und können über einen einfachen Mausklick nachgeschlagen werden. Von dort aus kann beliebig weit weitergesucht und schließlich zum Ausgangpunkt der Suche zurückgekehrt werden. Ein Ausdruck der kompletten Entwicklungsdokumente befindet sich im Anhang.

Damit sind aber noch nicht alle WAM-seitigen Systemkomponenten verwirklicht. Die fehlenden Bestandteile sollten aber in kürzester Zeit ergänzt werden können:

- Die **Festlegung der Semantik der einzelnen Dokumentrubriken** ist notwendig, um Informationen zwischen WAM- und *Sniff*-Dokumenten austauschen zu können. Semantische festgelegte Unterabschnitte können zwar erst in der noch in der Diskussion befindlichen HTML-Version 3.0 dargestellt werden, aber da bereits heutige Web-Browser und andere HTML-Interpreter unbekannte *tags*[64] ignorieren, können diese 3.0-*tags* schon heute strukturierend verwendet werden. Dies sieht für die Rubrik „Kontextbeschreibung“ aus Abb. 3.3.2-4 wie folgt aus:

```
<DIV CLASS=Relevante_Arbeitsgegenstaende>
<H3>Relevante Arbeitsgegenst&auml;nde</H3>
<UL>
  <LI><A HREF="glossar.html#Partitur">Partitur</A>
  <LI><A HREF="glossar.html#Leitmotivtabelle">Leitmotivtabelle</A>
  <LI><A HREF="glossar.html#Leitmotivnotiz">Leitmotivnotizzettel</A>
</UL>
</DIV>
```

- Die **graphische Darstellung der Dokumentstruktur** gewährt einen guten Überblick über die Beziehungen zwischen den Entwicklungsdokumenten. Sie könnte automatisch aus den Verweisinformationen der Doku-

---

[63] Eine Einführung in die Welt des Internets, zu denen diese Begriffe gehören, würde den Rahmen dieser Arbeit sprengen. Eine gute Einführung findet sich in [Ram95].

[64] *tags* ist der HTML-Jargon für im Text befindliche Formatierungs- und Strukturierungsinformationen. Sie werden stets mit spitzen Klammern (<...>) eingefaßt.

mente[65] herausgefiltert und graphisch aufbereitet werden. Durch Anklicken eines Dokumentsymbols könnte man direkt zum entsprechenden Dokument gelangen. Zur Verwirklichung kann z.B. auf Ulfert Weiss' IAT-SymbolNet zurückgegriffen werden.[66]

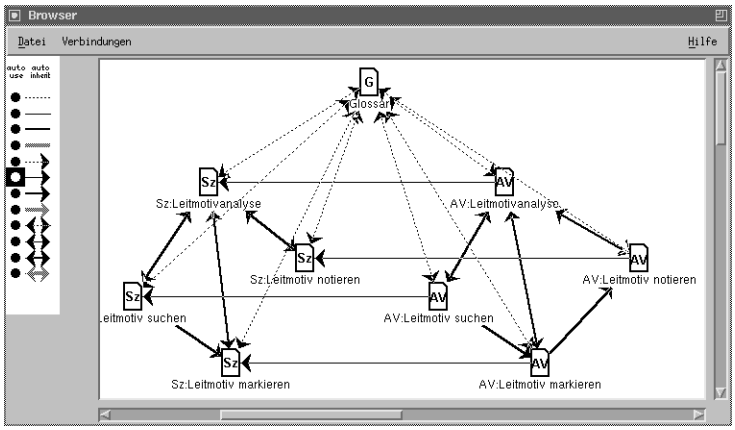


Abb. 3.2.2-5 Der von Ulfert Weiss entwickelte Interaktionstyp zur Darstellung von vernetzten Dokumenten.
Dargestellt werden die Szenarios und Ablaufvisionen zu den Leitmotivanalysewerkzeugen. Die Anordnung der Dokumente orientiert sich am Ideal aus den Abb. 3.1.2-1 und 3.1.2-2.

- Die **Unterstützung der Hypertexterstellung** ist wünschenswert, um die Dokumente nicht mit einem normalen Texteditor direkt in HTML erstellen zu müssen. Da der sichtbare Dokumenttext und die strukturgebenden *tags* sich gegenseitig durchdringen (siehe Abb. 3.3.2-4), ist die Dokumentbearbeitung mit einem normalen Editor sehr unübersichtlich. Zwar gibt es mittlerweile einige spezielle HTML-Editoren, aber keiner der von mir getesteten Editoren und HTML-Erstellungshilfen (*tkHTML*, *asWedit*, *HTML Editor*, *MS-Word Internet Assistant* und *gt_html.dot*) verfügt über die Eigenschaften, die ich mir nach langer Arbeit mit reinen Texteditoren wünsche:
  - Vollständiges Editieren in WYSIWYG[67] ohne Beschneidung der HTML-Ausdrucksmöglichkeiten,
  - Ausgabe übersichtlich formatierter HTML-Dokumente,
  - Verweise auf existierende Dokumente oder Begriffe können über Menüs eingefügt werden, ohne den Verweis vollständig in HTML zu spezifizieren.

[65] Das HTML-*tag* für Verweise ist `<A HREF="Dokumentname">Verweistext</A>`.
[66] vgl. [Wei94]
[67] Akronym für „**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et“: Bei der Texterstellung erscheint der Text so, wie er auch nachher auf Papier oder im Web-Browser erscheint.

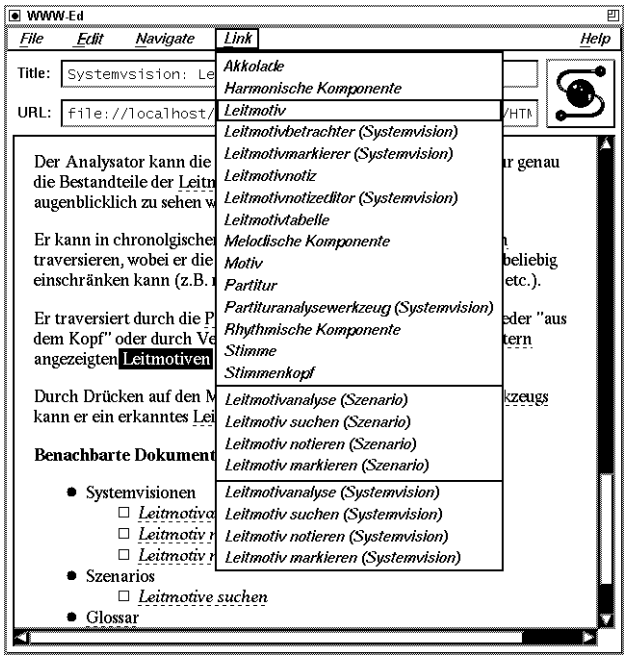


Abb. 3.3.2-6 Der ideale Editor für die Erstellung von WAM-Entwicklungsdokumenten in HTML

Das Dokument erscheint so, wie später im WebBrowser und Verweise auf bereits existierende Dokumente können über ein Menü eingefügt werden.

Aufgrund des regen Interesses, sich - mit in HTML erstellten Hypertexten - im WorldWideWeb zu präsentieren, besteht ein zunehmend größerer Bedarf an komfortablen HTML-Editoren, so daß in der nächsten Zeit mit einer Entwicklung in Richtung eines für das Idealsystem geeigneten Editors gerechnet werden kann.

Realisierung des *Sniff*-Teils. Ein einheitliches Hypertextdokumentationssystem auf HTML-Basis ist nur erreichbar, wenn - zusätzlich zu den im letzen Abschnitt beschriebenen Maßnahmen - *Sniff* sich bei der Repräsentation seiner Dokumentationen HTMLs bedient und deren Darstellung in einem dementsprechend erweiterten Dokumentationsbetrachter vornimmt. Zur besseren Integration der nach WAM (oder anderen Methoden) erstellten Entwicklungsdokumente sollte nicht nur ein minimales HTML, das zur Grobdarstellung einer herkömmlichen *Sniff*-Dokumentation ausreicht, sondern das vollständige HTML implementiert werden. Somit bestünde eine klar definierte Schnittstelle zu anderen Dokumentationssystemen. Der implementatorische Aufwand dürfte hierbei geringer einzuschätzen sein als die Überzeugungsarbeit, die bei den *Sniff*-Entwicklern zu leisten sein wird. Als Argumente können angeführt werden:

- Das bisherige Dokumentationsformat ist unflexibel und erlaubt keine Erstellung von anspruchsvoll gestalteten Dokumenationen auf dem Bildschirm oder auf Papier: Die Rubriken sind fest in Anzahl, Reihenfolge und Sprache; die Formatierung läßt sich nicht einem vorhandenen Projektstil anpassen.

  Mit HTML könnte die graphische Gestaltung (Sprache, Reihenfolge, Formatierungen) vollkommen den Entwicklern überlassen werden. Lediglich die semantischen Rubrikkennzeichnungen (`<DIV CLASS=...>`) wären einzuhalten.

- HTML könnte von *Sniff* genausogut wie das bisherige Format nach *Sniff*-relevanten Bezeichnern durchsucht werden: Der bisherige Schlüssel zu Beginn einer Dokumentation (z.B. „`$ class ...`“) ist der einzige Teil des Dokuments, der von *Sniff* gelesen wird.

  In HTML könnte diese Information z.B. im Titel (`<TITLE>...</TITLE>`) auftauchen.

- *Sniff* könnte von einer reinen Programmierumgebung (d.h. nur Unterstützung der Implementation) zum Teil *jeder* vollständigen Entwicklungsumgebung (d.h. Unterstützung von der Analyse bis zur Implementation) auf HTML-Basis werden: Durch einen offenen Dokumentstandard könnte jede Entwicklungsumgebung ihre Dokumente kontrolliert und verlustfrei an *Sniff* übergeben, ohne bei Implementationsbeginn mit der Dokumentation bei Null beginnen zu müssen.

  Mit Hilfe einer kleinen Übersetzungstabelle, in der die semantischen einander entsprechenden Rubriknamen aufgeführt sind, könnten *Sniff* und die jeweilige Entwicklungsumgebung Dokumente austauschen, bzw. interpretieren.

Die Umstellung wäre also verlustfrei, böte einen Qualitätsgewinn für die Dokumentation und eröffnete *Sniff* eine Perspektive in Richtung einer vollständigen Entwicklungsumgebung.

Nun fehlt nur noch der **GUI-Builder für WAM**. Er könnte auf Basis des bestehenden Ressourcenbrowsers erstellt werden. Der Ressourcenbrowsers dient dazu, bereits von den Entwicklern festgelegte Oberflächenelemente (Knöpfe, Schalter, Eingabefelder, Menüs, etc.) so zu arrangieren, daß sich dem Anwender eine Oberfläche darbietet, die der in der Systemvision enthaltenen Bildschirmskizze möglichst nahekommt. Dieser Browser müßte dahingehend erweitert werden, daß nicht nur *bestehende* Oberflächenelemente angeordnet sondern auch *neue* Elemente angelegt werden können. Aus diesen Elementen könnten dann die für die Oberflächengestaltung zuständigen Teile des Programmsystems ohne Zutun der Entwickler generiert werden.[68]

Eine solche Erweiterung des Resourcenbrowsers müßte mit vertretbarem Aufwand durchzuführen sein, da die Oberflächenelementverwaltung bereits vollständig vorhanden ist und lediglich um eine Erzeugungssequenz, in der der Name und die logische Verknüpfung mit den bisherigen Oberflächenelementen angegeben werden müssen, erweitert werden müßte. Die Erzeugung der Programmteile könnte dann analog zum Schreiben der erstellten Ressourcen erfolgen.

[68] Bei der Erstellung des Prototyps zum Leitmotivanalysewerkzeug mußten die Konstrukturen, Destruktoren und Initialisierungssequenzen der 109 Oberflächenelemente des Leitmotivbrowsers noch unter erheblichem Zeitaufwand von Hand erstellt werden.

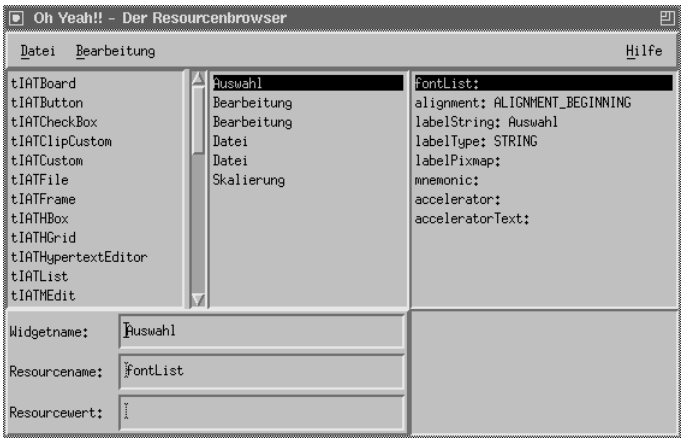


Abb. 3.2.2-7 Der von Dirk Riehle entwickelte Ressourcenbrowser.

Die meisten Ressourcen zu allen Interaktionstypen eines Programms können zur Laufzeit modifiziert werden. Die linke Liste enthält alle verfügbaren Interaktionstypenarten. In der mittleren Liste erscheinen die Namen der tatsächlich verwendeten Interaktionstypen einer Art. Ganz rechts werden die Ressourcen angezeigt, die mit Hilfe des Ressourcenbrowsers geändert werden können. Nach Eingabe eines Wertes im untersten Textfeld wird der ausgewählte IAT prompt in abgewandelter Form dargestellt.

Zusammenfassung

In diesem Kapitel wurden die Dokumente und die die Arbeit mit ihnen unterstützende Entwicklungsumgebung untersucht.

Die WAM-Dokumente wurden im Laufe des Leitmotivanalyse-Projekts sukzessive inhaltlich und strukturell an die speziellen Bedürfnisse dieses Entwicklungsprozesses angepaßt. Dabei stellte sich heraus, daß Änderungen der Dokumentstruktur stets vom Dokumentinhalt ausgingen. Die Umstände, unter denen Strukturänderungen vorgenommen worden sind, wurden untersucht und daraus allgemeinen Anpassungsrichtlinien destilliert.

Die Entwicklunsgumgebung für WAM war bei Arbeitsbeginn nur im technischen Bereich existent. Die Erstellung und Verwaltung der Dokumente wurde überhaupt nicht unterstützt. Basierend auf den im Projektverlauf festgestellten Mängeln des bisherigen Zustands wurde eine ideale Umgebung für WAM spezifiziert und in Teilen auch bereits verwirklicht. Diese Umgebung unterstützt den gesamten Entwicklungsprozeß. Sie fußt auf einem einheitlichen Hypertextformat, in dem sowohl WAM- als auch *Sniff*-Dokumente repräsentiert werden können, und enthält einen GUI-Builder für WAM.

Unübersichliches Blättern in WAM-Dokumenten und die Übergangsschwierigkeiten von WAM- zu *Sniff*-Dokumenten entfallen damit. Nachdem gezeigt wurde, daß das *Sniff*-Dokumentationsformat nicht ohne größere Änderungen als Grundlage für dieses einheitliche Hypertextformat dienen kann, wurde HTML gewählt. Es wurde gezeigt, daß die ausschließlich auf WAM bezogenen Werkzeuge des Entwicklungsprozesses binnen kurzer Zeit erstellt werden könnten. Schließlich wurden Argumente geliefert, die für eine Umstellung des bisherigen *Sniff*-Dokumentationsformats auf HTML sprechen. Damit wären wesentliche Verbesserungen für WAM und *Sniff* verbunden: Für WAM alleine könnten die Dokumente einfacher erstellt und gesichtet werden. Für *Sniff* alleine ergäbe sich die Möglichkeit, von einer reinen Programmierumgebung zum die Programmierung unterstützenden Teil jeder offenen Entwicklungsumgebung zu werden, die ihre Dokumente in HTML

spezifiziert. Mit einer Kombination aus WAM und *Sniff* könnten Softwaresysteme schließlich bruchlos mit WAM und *Sniff* entwickelt werden.

Der GUI-Builder für WAM entlastet die Entwickler von der aufwendigen manuellen Programmierung der Benutzungsoberfläche und ermöglicht eine noch schnellere Erstellung von ersten Prototypen. Der GUI-Builder könnte mit vertretbarem Aufwand als Erweiterung des bestehenden Ressourcenbrowsers verwirklicht werden.

Damit steht fest, daß die anfänglich formulierte These

Eine geeignete Entwicklungsumgebung für WAM fehlt, ist aber in greifbare Nähe gerückt

zutreffend ist.

3.3.3 Anwendungsgebiet

Nachdem in Kapitel 3.1.3 die Besonderheiten der Opern Wagners und die Schwierigkeiten, die sich bei deren Analyse ergeben, beschrieben wurden und in Kapitel 3.2 die Entwicklung eines diese Schwierigkeiten lindernden Leitmotivanalysewerkzeugs geschildert wurde, so werden in diesem Kapitel die anwendungsspezifischen Vorteile beleuchtet, die sich bereits durch den Entwicklungsprozeß ergeben haben.

Anfangs wird das prototypische Leitmotivanalysewerkzeug kurz beschrieben und dann auf die über die reine Unterstützung hinausgehenden Verbesserungen der Analyse durch das Werkzeug eingegangen. Eine ausführliche Beschreibung des aus Anwendersicht Werkzeugs befindet sich in den Systemvisionen in Anhang B. Die softwaretechnische Architektur ist in Kapitel 4 beschrieben.

Im Rahmen der bisherigen Leitmotivanalyse werden als Arbeitsmaterialien die Partitur, Tonaufnahmen der Oper, Leitmotivtabellen, eigene Leitmotivnotizzettel sowie Bleistift und Radiergummi verwendet. Als Tätigkeiten lassen sich isolieren Leitmotivsuche in der Partitur durch Hören oder Vergleichen mit Leitmotivtabellen, Markierung eines gefundenen Leitmotivs in der Partitur und Notieren eines gefundenen Leitmotivs auf einem Leitmotivnotizzettel. Diese Leitmotivnotizzettel dienen als Ergänzung zu den Leitmotivtabellen bei der Suche nach weiteren Leitmotiven.

Dieser Vorgang wurde so in eine Gruppe von Leitmotivanalysewerkzeugen umgesetzt, daß die Arbeitsabläufe unverändert durchgeführt werden können. Die Werkzeuge orientieren sich dabei an den Tätigkeiten der manuellen Leitmotivanalyse. Es sind die Leitmotivsuche in der Partitur (dargestellt in einem Partituranalysewerkzeug) durch Hören (Funktion des Partituranalysewerkzeugs) oder Vergleichen mit Leitmotivtabellen (dargestellt in Leitmotivbetrachtern), Markierung eines gefundenen Leitmotivs in der Partitur (mit einem Leitmotivmarkierer) und Notieren eines gefundenen Leitmotivs (in einem Leitmotivnotizeditor). Diese Leitmotivnotizen dienen als Ergänzung zu den Leitmotivtabellen (in den Leitmotivbetrachtern) bei der Suche nach weiteren Leitmotiven.

Die Vorteile dieser Werkzeuge sind die Entlastung des Musikwissenschaftlers von den Tätigkeiten der manuellen Anfertigung von Partiturexzerpten für die Leitmotivnotizen und der Verwaltung dieser Notizen: Da die Partitur in elektronischer Form vorliegt, kann die vom Musikwissenschaftler markierte Partiturstelle automatisch exzerpiert und zusammen mit dem Ort des Auftretens in die Leitmotivnotiz eingetragen werden. Einmal mit einer Leitmotivnotiz in eine Leitmotivtabelle eingetragene Daten können schnell und einfach mit Leitmotivbetrachtern in diesen wiedergefunden und mit Partiturstellen verglichen werden.

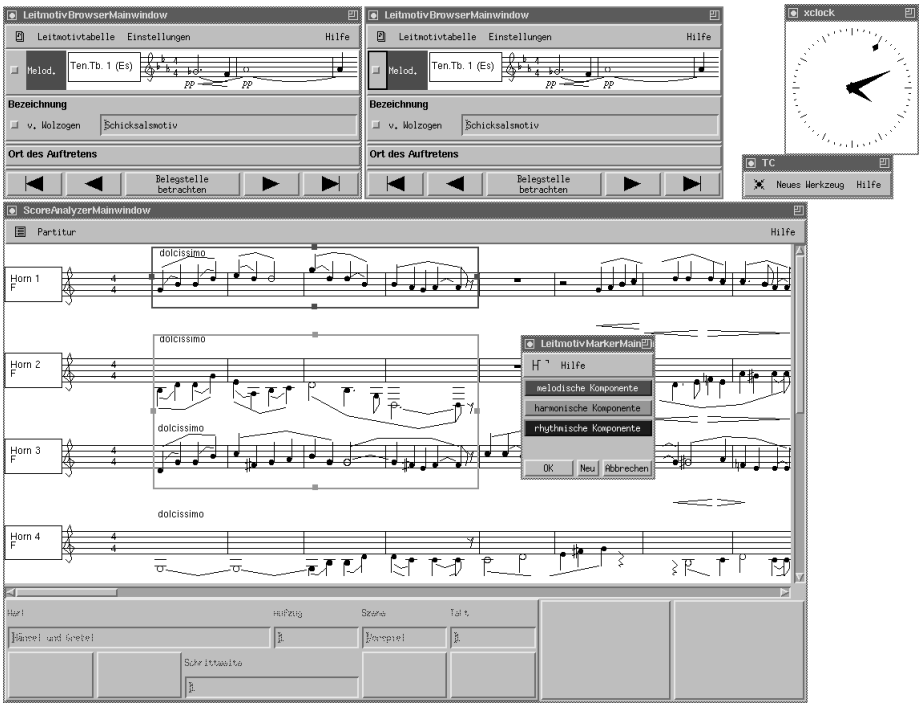


Abb. 3.3.3-1 Die Leitmotivanalysewerkzeuge während der Markierung eines Leitmotivs in der Partitur

Oben links und in der Mitte je ein Leitmotivbetrachter, in denen der Inhalt von Leitmotivtabellen dargestellt wird. Oben rechts unter der Uhr der Werkzeugkoordinator, mit dessen Hilfe neue Werkzeuge aufgenommen oder benutzte Werkzeuge abgelegt werden können. Unten ein Partituranalysewerkzeug, in dem der Ausschnitt einer Partitur mit Markierungen zu sehen ist. Darin im rechten, mittleren Teil ein Leitmotivmarkierer, mit dem zu markierende Komponenten ausgewählt werden können. Ein Leitmotivnotizeditor wird erscheinen, sobald die Markierung abgeschlossen sein wird.

Die über diese reine Unterstützung hinausgehende Verbesserung des Analysevorgangs ist die Einführung der kanonischen Partiturdarstellung. Zum besseren Verständnis erfolgt zuvor eine Beschreibung des klassischen Verfahrens.

Die klassische Partiturdarstellung orientiert sich vornehmlich an den Bedürfnissen eines Dirigenten. (Die Musiker und Sänger erhalten nur Partiturauszüge, in denen allein ihre Partien enthalten sind.) Alle Orchsterstimmen (Instrumente und Sänger) werden vollständig aber mit minimalem Platzbedarf dargestellt. Pausierende Stimmen werden nicht gedruckt. Wenn z.B in einem aus zwanzig Stimmen bestehende Orchester eine Stimme ein Solo hat, so werden die anderen 19 Stimmen nicht gedruckt und auf einer Partiturseite findet die maximal mögliche Anzahl von Takten der Solostimme Platz. Dieses Verfahren reicht auch zur musikwissenschaftlichen Anlayse vollkommen aus, ist jedoch erschwerend bei der Gewinnung einer Übersicht über die Instrumentation

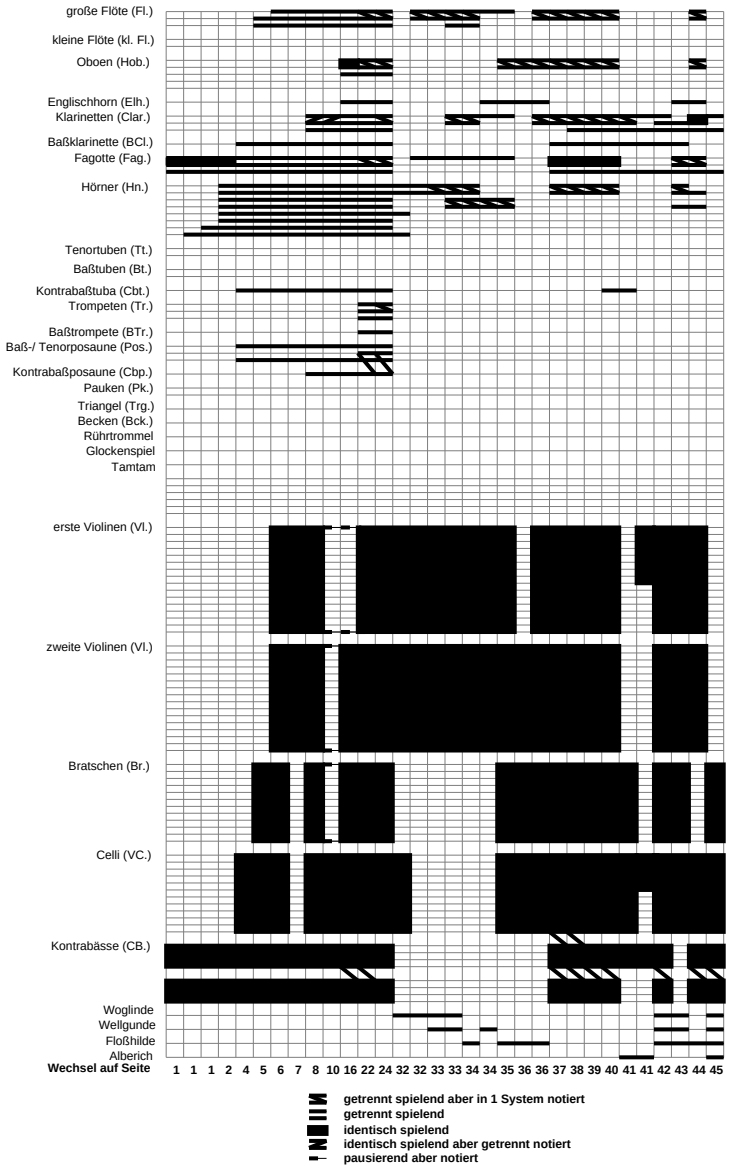


Abb. 3.3.3-2 Instrumentationsskizze des Anfangs des *Rheingold*-Vorspiels.

Die Anzahl der gleichzeitig aktiven Stimmen schwankt von 5 Stimmen auf der ersten Partiturseite (Fagott 1, 2, und 3, Kontrabässe 1 und 2) bis zu 37 Stimmen auf der 13. Partiturseite (große Flöte 1-3, Oboen 1-3, Englischhorn, Klarinetten 1-3, Baßklarinette, Fagott 1-3, Horn 1-8, Kontrabaßtuba, Trompete 1-3, Baßtrompete, Posaune 1-3, Kontrabaßposaune, erste und zweite Violinen, Bratschen, Violincelli, Kontrabässe 1 und 2).

über mehrere Seiten hinweg, da die Instrumente je nach Anzahl der gerade pausierenden bzw. aktiven Stimmen stets an einer anderen Stelle auf der Partiturseite auftauchen.

Die kanonische Partiturdarstellung orientiert sich an diesem Übersichtlichkeitskriterium des Musikwissenschaftlers auf Kosten des Raumsparkriteriums der klassischen Methode. Alle Stimmen werden, gleich ob pausierend oder nicht, permanent angezeigt, so daß stets ein Überblick über die Instrumentation im Großen gewonnen werden kann.

Zusammenfassung

Die Leitmotivanalyse konnte durch die Erstellung des Leitmotivanalysewerkzeugs für den Musikwissenschaftler erleichtert werden. Das Werkzeug unterstützt ihn bei den zeitaufwendigen Aufgaben der Partiturexzerptanfertigung sowie der Suche in Leitmotivtabellen und ermöglicht es ihm somit, sich auf die eigentlichen Aufgaben der vorbereitenden Leitmotivanalyse - das Auffinden und Markieren von Leitmotiven - zu konzentrieren. Dadurch werden umfangreichere Leitmotivanalysen in Zukunft in wesentlich kürzerer Zeit erstellt werden können als bisher.

4 WAM als Hilfsmittel zur Schaffung einer Umgebung für die rechnergestützte allgemeine musikwissenschaftliche Analyse

Das Interesse der Musikwissenschaft an einer Umgebung zur Durchführung rechnergestützter Analysen ist groß. Jedoch befinden sich die existierenden Programme entweder in der Nähe des einen oder des anderen der beiden folgenden Pole, ohne daß ein Brückenschlag bisher geglückt wäre: Entweder stellen sie nur eine passive hypermediale Visualisierung einer bereits existierenden Analyse dar[69] oder sie ermöglichen komplexe eigene Analysen nur mit stark abstrahierten Daten und Programmen, deren fachlicher Bezug für den informatisch nicht Vorgebildeten nicht mehr zu erkennen ist[70].

Dieser Brückenschlag darf für den Spezialfall der im letzten Kapitel beschriebenen Leitmotivanalysewerkzeuge als geglückt angesehen werden: Der Anwender kann in einer elektronischen Version seiner gewohnten Umgebung arbeiten und trotzdem die zeitsparenden, komplexen Operationen der Rechnerunterstützung in Anspruch nehmen. Aus dieser Tatsache heraus ergibt sich zweierlei. Zum ersten kann die These

> ***Mit den in dieser Arbeit verwendeten Werkzeugen und Methoden ist eine Umgebung für allgemeine musikwissenschaftliche Analysen in greifbare Nähe gerückt***

abgeleitet und bestätigt werden. Zweitens wird gezeigt, daß die von mir zum Brückenschlag geschaffene WAM-gemäße Architektur zur Kapselung des Subsystems *Humdrum* auch zur Kapselung anderer bestehender Subsysteme angewendet werden kann und somit eine Alternative zum bisher zu diesem Zweck verwendeten Automaten darstellt.

Da all diese Erkenntnisse auf der Entwicklung eines musikwissenschaftlichen Anwendungssystems fußen, wird zunächst eine Charakterisierung der aktuellen Situation der Musikwissenschaft und der ihr zur Verfügung stehenden Arten von Rechnerunterstützung mitsamt den ihnen innewohnenden Schwachpunkten vorgenommen. Danach werden die beiden Hauptprobleme (Bildung untereinander inkompatibler Datenbestände und mangelhafte fachliche Einbettung der Analyseprogramme) und ihre Lösung in zwei Unterkapiteln genauer ausgeführt. In Kapitel 4.1 wird das von mir entwickelte Programmpaket *scr2hmd* beschrieben, mit dem die zahlreich vorhandenen reinen Notensatzdaten (*SCORE*-Format) in für die Analyse geeignete Form (*Humdrum*-Format) gebracht werden können und somit die Insularität bisheriger Datenbestände gelindert wird. In Kapitel 4.2 wird die technische Seite der fachlichen Kapselung *Humdrum*s durch die Leitmotivanalysewerkzeuge gezeigt. Während in Kapitel 3 die fachlichen Aspekte im Vordergrund stehen, geht es hier darum, wie das bestehende *Humdrum*-System mit minimalem Aufwand aber effektiv mit dem WAM-Teil des Systems gekoppelt werden konnte und wie dieses Verfahren verallgemeinert werden kann.

---

[69] Z.B. William Renwicks *CD-Brahms*[RW92] oder Seiferts Mac-Programm
[70] Z.B. David Hurons *Humdrum*, das in Kapitel 4.1 ausührlich beschrieben wird.

Die musikwissenschaftliche Analyse

Die Musikwissenschaft klassischer Prägung ist eine geisteswissenschaftliche Disziplin, die Inhalt, Form und Wirkung musikalischer Werke untersucht. Ihre Ergebnisse werden durch Interpretation musikalischen, biographischen und historischen Materials gewonnen, wobei vor allem im tonalen musikalischen Bereich auf zahlreiche Formlehren zurückgegriffen werden kann.

Dem entgegen steht die vor allem im angelsächsischen Bereich verbreitete „music theory". Sie beruft sich auf ein im naturwissenschaftlichen Sinne „objektives" Theoriegerüst, anhand dessen frei von jeglicher Interpretation festgestellt werden kann, ob eine Kompositionen bestimmten Kriterien genügt oder nicht. Um diese in Kontinentaleuropa weniger bekannte Forschungsrichtung kurz vorzustellen, seien ihre wesentlichen Schwerpunkte hier aufgezählt:

- Die Erforschung tonaler Zusammenhänge, hierarchisch gegliederter Strukturen und Stimmenführungsphänomnen in Fortsetzung der Arbeit Schenkers durch Meyer, Narmour, Lerdahl und Jackendorff.[71]
- Die Deduktion von Musikwerken aus deren thematischen Grundgestalten durch Epstein in Forsetzung der Arbeit Schönbergs und Rétis.[72]
- Die Untersuchung des Verhältnisses der Stärke musikalischer Elemente zueinander durch Berry.[73]
- Die Analyse atonaler Musik mit Hilfe mengentheoretischer Methoden durch Forte.[74]

Der Hauptverdienst der „music theory" besteht in ihrem neuen Blickwinkel, der die Vernetzung der verschiedenen musikalischen Elemente stärker in den Vordergrund stellt als in der klassischen Musikwissenschaft. Dies kann allerdings nicht über ihre Schwächen hinwegtäuschen. Die Aufbereitung des musikalischen „Rohmaterials" erfolgt nicht wirklich objektiv sondern bereits im Hinblick auf die jeweilige Theorie. Zudem wird der musikalischen Hermeneutik kein Raum gelassen, so daß keine Aussagen über Sinn und Bedeutung eines Werks sowie dessen historischen und biographischen Kontext gemacht werden können.[75]

Trotz der unterschiedlichen Herangehensweisen beider Forschungsrichtungen, sind für beide Unterstützung durch Rechner möglich.

Dies gilt zunächst nicht für den interpretativen Teil der klassische Musikwissenschaftt, da sie musisches Empfinden voraussetzt: Kein Rechner ist in der Lage zu beurteilen, ob der dritte Satz von Beethovens sechster Symphonie tatsächlich an ein „Lustiges Zusammensein der Landleute" erinnert oder ob das Zwischenspiel vor dem vierten Akt von Bizets „*Carmen*" wirklich nichts von der bevorstehenden Eifersuchtstat Don Josés erahnen läßt.

Die der interpretativen Arbeit zugrundeliegende quantitative Analyse bietet jedoch durchaus Ansatzpunkte für eine Unterstützung durch Rechner, da sie sich auf das Suchen, Vergleichen und Zählen quantifizierbarer Größen beschränkt. Ein Rechner könnte diese Tätigkeiten bei geeigneter Aufbereitung der Musikdaten und ausge-

[71] vgl. [Fus95, S.3]
[72] vgl. [Fus95, S.3]
[73] vgl. [Fus95, S.79]
[74] vgl. [Fus95, S.58]
[75] vgl. [Fus95, S.70ff.]

stattet mit adäquaten Werkzeugen ähnlich gut versehen wie ein Assistent mit rudimentärem Musikwissen. Es ist reine Fleißarbeit, herauszufinden, in welchen Werken Bachs das B-A-C-H-Motiv vorkommt, ob die Tempi sich bei Mozart im Laufe der Zeit verlangsamt haben oder wie häufig der Anfang der *„Marseillaise“* in Tschaikowskys *„1812 Ouvertüre“* erklingt.

Bei der „music theory“ ist das Unterstützungspotential noch wesentlich höher, da viele ihrer Theorien algorithmisierbar sind und somit Musikwerke direkt durch Rechner untersuchbar sind. Zum Beispiel können im Rahmen einer Analyse nach Fortes Mengentheorie im Anschluß an die manuell erfolgte Segmentierung eines Stücks alle weiteren Aufbereitungs- (Tongruppierung, Sortierung, Rotation, Transponierung, Normalisierung)[76] und Analyseschritte (Ähnlichkeit, Zusammengehörigkeit)[77] automatisch durchgeführt und deren Ergebnisse tabellarisch aufbereitet werden.

Existierende Rechnerunterstützung

Die eben aufgezählten Einsatzmöglichkeiten für Rechner sind bisher kaum verwirklicht worden. Dies liegt nicht nur an den fehlenden Programmen sondern vor allem auch an dem Mangel analysierbarer Daten. Sofern musikalische Werke nicht in elektronischer Form vorliegen, können sie auch keiner rechnergestützten Analyse zugeführt werden.

Zwar gibt es eine Vielzahl von Musikdatenformaten mit teilweise großen Datenbeständen, doch nur ein geringer Anteil von ihnen ist für allgemeine Analysen geeignet. Während für eine Studie über die Wandlung der Interpretation eines Werks durch einen Dirigenten mehrere akustische Aufzeichnung des von ihm dirigierten Werks vorhanden sein müssen, so läßt sich die Einhaltung der Regeln des Kontrapunktes am besten in einer Partitur nachvollziehen. Aufgrund dieser Vielschichtigkeit musikalischer Informationen hat sich *das* Musikdatenformat und eine dementsprechende Analyseumgebung bisher nicht herausgebildet.[78]

Nach ihrem Zweck lassen sich vier Domänen von Musikdatenformaten und darauf arbeitenden Werkzeugen unterscheiden. Die Zielrichtungen dieser Hauptgruppen sind tonbandgleiche akustische Wiedergabe („Akustik“), Wiedergabe mittels elektronisch ansteuerbarer Instrumente („Instrumentansteuerung“), Wiedergabe durch Menschen („Notensatz“) und musikwissenschaftliche Analyse („Analyse“).[79] Bevor die aus diesen Formatverschiedenheiten resultierenden Probleme erörtert werden, werden zunächst die einzelnen Domänen kurz bezüglich ihrer Möglichkeiten charakterisiert. Außerdem wird festgestellt, welche für eine Analyse relevanten Daten ihnen entnommen werden können.

Akustik. Bei tonbandgleicher akustischer Wiedergabe geht es darum, Musik, Stimmen und Geräusche aller Art möglichst naturgetreu über Lautsprecher auszugeben. Die Daten werden entweder durch Digitalisierung einer

---

[76] vgl. [Fus95, S.60f.]
[77] vgl. [Fus95, S.64ff.]
[78] Einen Versuch zur Schaffung eines universellen Musikdatenformats stellt SMDL (**S**tandard **M**usic **D**escription **L**anguage, ISO Draft International Standard 10743) dar. SMDL soll eine SGML-Anwendung werden und Informationen über die Tonfolge, deren graphischer Darstellung, die klanglichen Eigenschaften der Instrumente und analytische Daten enthalten können.[Slo93] Es ist in der musikwissenschaftlichen Gemeinde noch umstritten, ob eine so umfassende Beschreibung überhaupt möglich ist. Ein Ausschnitt dieser Diskussion ist in [Smo95], [New95] und [Smo96] festgehalten.
[79] Für einen andere mögliche Klassifizierung vgl. [HMSW93]

Klangquelle gewonnen oder gemäß anderer Musikdaten (s.u.) synthetisch generiert. Die Datenstruktur besteht aus einer Zeitreihe von Klangwellenbeschreibungen. Mehrstimmige Klangkörper werden nicht aufgespalten sondern als eine Einheit angesehen. Die Daten können in einer Art digitalem Tonstudio bearbeitet werden, um z.B. die Lautstärke, die Geschwindgkeit, die Tonhöhe oder die Reihenfolge der Daten zu verändern. Informationen über die den Klängen zugrundeliegende logische Struktur wie z.B. in einer Partitur lassen sich aus den Daten nicht gewinnen, da bisher kein Verfahren existiert, den Gesamtklang in stimmenbezogene Einzelklänge zu unterteilen.[80] Daher lassen sich außer rein akustischen Analysen keine weiteren musikwissenschaftlichen Untersuchungen durchführen. Datenformate, die zu dieser Gruppe gehören sind WAV, AU und SND. Daten sind in Form von Tonträgern in großem Umfang vorhanden.

Instrumentansteuerung. Daten, die zur Wiedergabe durch elektronisch ansteuerbare Instrumente dienen, können dadurch hörbar gemacht werden, daß sie einem Synthesizer übermittelt werden, der diese dann interpretiert und entweder selbst akustisch ausgibt oder als tonbandgleiche akustische Daten (s.o) abspeichert. Der Synthesizer kann dabei sowohl ein eigenständiges Instrument als auch ein rechnerinterner Tongenerator sein. Daten entstehen durch die Aufzeichnung des Spiels eines menschlichen Interpreten oder werden aus anderen Musikdaten (s.u) generiert. Die Datenstruktur besteht aus mehreren nach Stimmen getrennten Spuren. Diese Spuren enthalten Informationen über die Interpretation des Stücks in Form von Höhe, Länge und eventuell Anschlagstärke der zu spielenden Töne. Auch Informationen über die Klangeigenschaften der einzelnen Stimmen können enthalten sein. Bearbeitungsmöglichkeiten bestehen in der Neuzuordnung von Stimmen und Instrumenten, der Veränderung der Tonhöhe und der Reihenfolge sowie der Transponierung. Da die Daten zur Ansteuerung von Geräten gedacht sind, ist ihre Codierung meistens nicht für menschliche Leser geeignet. Die Informationen über die Tonhöhen sind nur in beschränktem Umfang zur musikwissenschaftlichen Analyse geeignet, da aus ihr nicht ersichtlich ist, welcher von mehreren enharmonischen Tönen gemeint ist (z.B. cisis = d = eses). Auch die Informationen über die Tondauer sind bei Herkunft der Daten aus einer menschlicher Interpretation oft stark abweichend von der exakten mathematischen Tondauer. Notensatzinformationen wie Notenschlüssel und Tonartversetzungszeichen lassen sich aus den Daten nicht gewinnen. Vertreter dieser Gruppe sind MIDI[81] und die Music-N-Familie (Music-11, Csound und cmusic).[82] Daten sind vorhanden, jedoch oft von geringer Qualität.

Notensatz. Die Wiedergabe durch Menschen erfordert eine vollständige Aufzeichnung der Partiturinformation in graphischer Form. Die Daten können entweder manuell gesetzt oder durch Korrektur eingescannter gedruckter Partituren gewonnen werden. Automatischer optimaler Notensatz ist bisher nicht möglich.[83] Die Datenstruktur enthält Informationen über die seitenweise Anordnung der die Partitur ausmachenden graphischen Symbole. Logische Zusatzinformationen wie Tonhöhe, -dauer und Zuordnung zu einer Stimme können, müssen aber nicht enthalten sein, da sie für das Druckbild unerheblich sind. Die Bearbeitungsmöglichkeiten beschränken sich auf die Neuanordnung der graphischen Symbole auf einer Seite z.B. zur Verbesserung der Lesbarkeit.

---

[80] vgl. [HMSW93, S.34]
[81] Eine Kurzbeschreibung des **M**usical **I**nstruments **D**igital **I**nterface wird in [HMSW93, S.34f] gegeben.
[82] vgl. [HMSW93, S.36]
[83] vgl. [Byr94, S.19f]

Je mehr logische Informationen direkt aus den Daten gelesen werden können, desto brauchbarer sind sie für die musikwissenschaftliche Analyse. Um die vollständige Partiturinformation zu erhalten, müssen die Daten gemäß der graphischen Position ihrer Symbole interpretiert werden. Notensatzformate sind DARMS[84], *SCORE*[85] und NIFF[86]. Qualitativ hochwertige Daten sind vorhanden aber oft urheberrechtlich geschützt.

Analyse. Daten zur musikwissenschaftlichen Analyse enthalten idealerweise Elemente aus allen anderen bisher genannten Domänen, um möglichst alle Aspekte eines Werks abzudecken. Daten können manuell erstellt werden oder durch Interpretation aus Daten zur Ansteuerung von Synthesizern oder Notensatzdaten herausgelesen werden. Die Datenstruktur muß Auskunft über die komplette logische Struktur der Partitur wie isolierte Stimmen, enharmonisch eindeutige Tonhöhen und mathematisch exakte Tonlängen geben. Bearbeitungsmöglichkeiten sind alle Untersuchungen quantitativer Art wie Suchen nach bestimmten Tonfolgen, Manipulationen wie z.B. Transponierung und Umkehrung und Markierung von ausgewählten Passagen. Für die Analyse geeignete Formate sind BR numeric codes[87], *Humdrum-Kern*[88] und SMDL[89]. Daten sind kaum vorhanden.

Jede Art von Rechnerunterstützung steht vor dem Problem, sich entweder auf ein kleines Teilgebiet von musikalischen Aspekten zu beschränken oder die benötigten vollständigen Daten selbst zu generieren bzw. durch möglichst geschickte Extraktion aus anderen Daten zu gewinnen. Für das Leitmotivanalysewerkzeug habe ich den letzten Weg gewählt, da eine Beschränkung ebensowenig in Frage kam wie die manuelle oder optische Erfassung der sehr umfangreichen Opern. Das zu diesem Zweck entwickelte *scr2hmd* konvertiert Notensatzdaten im *SCORE*-Format in Analysedaten in zwei *Humdrum*-Formaten und wird im folgenden Kapitel 4.1 vorgestellt.

Nachdem mit *scr2hmd* das Problem der Inkompatibilität der bestehenden Daten abgeschwächt worden ist, wird in Kapitel 4.2 gezeigt, wie -die nun mit Daten versorgte- mächtige aber unhandliche *Humdrum*-Analyseumgebung mit Hilfe von WAM gekapselt werden kann. Auf diese Weise entsteht ein effektives Werkzeug, dessen Benutzung für Musikwissenschaftler wie selbstverständlich erscheint und dessen Architektur auch auf andere Analyseaufgaben übertragbar ist.

---

[84] vgl. [HMSW93, S.35]
[85] vgl. [Smi94]
[86] vgl. [MRD 10:2] und [Gra96]
[87] vgl. [HIW93]
[88] vgl. [Hur94]
[89] vgl. [Slo93] und [A44]

4.1 Die zugrundeliegende Analyseumgebung aus *SCORE*, *Humdrum* und *scr2hmd*

Bevor im nächsten Kapitel die fachliche Kapselung der Analyseprogrammsammlung *Humdrum* durch WAM-Methoden beschrieben werden kann, muß hier zunächst die aus *SCORE*, *Humdrum* und *scr2hmd* bestehende „Roh-Analyseumgebung“ vorgestellt werden. In *SCORE* liegen die beim Musikverlag Schott gesetzten Wagnerpartituren vor. Aus diesen Notensatzdaten werden mit dem von mir im Rahmen dieser Arbeit entwickelten Konvertierungsprogrammpaket *scr2hmd* die für die Analyse geeigneten *Humdrum*-Daten erzeugt. Mit *Humdrum* können die Daten schließlich über eine UNIX-Schnittstelle auf komplexe Weise manipuliert und analysiert werden. Es wird gezeigt, daß *Humdrum* und *SCORE* zusammen in der Lage sind, den gesamten Bedarf an musikwissenschaftlichen Anlaysedaten und -werkzeugen abzudecken und daß durch geschickte Interpretation der Notensatzdaten nahezu vollständige Analysedaten erzeugt werden können.

Dazu werden zunächst die beiden Eckpfeiler der Konvertierung *Humdrum* und *SCORE* einzeln behandelt und dann hinsichtlich ihrer Komplementarität untersucht. Danach werden der Leistungsumfang und die Arbeitsweise von *scr2hmd* beschrieben und kritisiert.

Humdrum

Humdrum ist eine Sammlung von Musikdatenformaten und darauf arbeitenden UNIX-artigen Werkzeugen. Durch Kombination mehrerer dieser Werkzeuge lassen sich nahezu alle denkbaren quantitativen musikwissenschaftlichen Analysen (teilweise auch aus dem Gebiet der „music theory“) durchführen.[90] Es gibt 36 Datenformate, die die meisten interessanten Aspekte von musikalischen Werken auszudrücken in der Lage sind. Mit den 59 Werkzeugen können die Daten zwischen den verschieden Formaten (inklusive MIDI) konvertiert sowie untersucht und manipuliert werden.

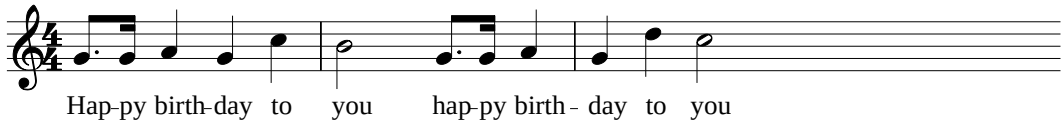


Abb 4.1-1 Der Anfang von *„Happy Birthday“* dient in diesem Kapitel als Beispiel für die Darstellungsmöglichkeiten mit *Humdrum*-Datenformaten.

Datenformate

Alle *Humdrum*-Datenformate[91] sind nach dem gleichen Schema aufgebaut und bedienen sich ausschließlich des ASCII-Zeichenvorrates. Sequentiell aufeinanderfolgende Ereignisse befinden sich in aufeinanderfolgenden Zeilen während parallel auftretende Ereignisse in einer Zeile, getrennt durch Tabulatoren, angeordnet sind. Direkt untereinanderstehende Ereignisse gehören logisch zusammen und bilden eine Sequenz, die im *Humdrum*-Sprachgebrauch spine genannt wird. Jede Sequenz kann drei Arten von Informationen enthalten: Inter-

[90] vgl. [Hur94, S.7]

[91] In dieser Arbeit werden alle *Humdrum*-Formate wie in [Hur94] mit führendem `**` geschrieben. Lediglich die auch außerhalb diese Kapitels verwendeten Formate `**kern` und `**layout` werden zur besseren Lesbarkeit *Kern* und *Layout* geschrieben.

```
!! Mildred and Patti Hill, composers
!! `Happy Birthday´
**pitch       **recip   **text
!re A4=440Hz  !         !(lyrics)
G4            8.        Hap-
G4            16        -py
A4            4         birth-
G4            4         -day
C5            4         to
B4            2         you ;
G4            8.        hap-
G4            16        -py
A4            4         birth-
G4            4         -day
D5            4         to
C5            2         you ;
*-            *-        *-
```

Abb 4.1-2 *„Happy Birthday“* als Sammlung von *Humdrum*-Dateien.

Der zeitliche Verlauf ist von oben nach unten. ****pitch** gibt Auskunft über die Tonhöhe (amerikanischer Tonbuchstabe und Oktavennummer). ****recip** gibt die Tondauer in der Form 1/x an (**16** entspricht einer 16tel Note). ****text** enthält den Text zu jeder Note.

pretationen, die die Deutung der Daten bestimmen und die mit einem ***** beginnen, Kommentare, die mit einem **!** beginnen und schließlich die eigentlichen Daten.[92] Das Lied *„Happy Birthday“* könnte wie in Abb. 4.1-2 in *Humdrum* repräsentiert werden..

```
!! Mildred and Patti Hill, composers
!! `Happy Birthday´
**kern        **text
*clefG2       *         Violinschlüssel
*k[]          *         keine Tonartversetzungszeichen
*M4/4         *         4/4-Takt
*C:           *         C-Dur
!re A4=440Hz  !(lyrics)
=1            =1        1. Takt
8.g/L         Hap-
16g/Jk        -py
4a/           birth-
4g/           -day
4cc\          to
=2            =2        2. Takt
2b\           you ;
8.g/L         hap-
16g/Jk        -py
4a/           birth-
=3            =3        3. Takt
4g/           -day
4dd\          to
2cc\          you ;
*-            *-
```

Abb 4.1-3 *„Happy Birthday“* als *Kern*- und ****text**-Dateien.

Kern vereinigt in sich ****recip** und ****pitch** (wobei die Oktave durch die Anzahl der Tonbuchstaben kodiert ist) und enthält zusätzlich rudimentäre Notensatzinformationen (z.B. **/** Hals nach oben, **** Hals nach unten, **L** Querbalkenanfang, **J** Querbalkenende).[93]

---

[92] vgl. [Hur94, S.8ff]
[93] Das Beispiel stammt aus [Hur94, S.13]

Die Datenformate sind größtenteils Spezialformate, die nur einen oder wenige Aspekte von Musik repräsentieren, und auf denen neben den allgemeinen manipulativen Werkzeugen nur wenige spezialisierte Analysewerkzeuge arbeiten. **`**pitch`** kann z.B. nur Tonhöhen, Pausen, Taktstriche, Bindebögen, Phrasen und Abweichungen vom reinen Ton abbilden,[94] während **`**recip`** sogar nur Tondauern und Taktstriche abbilden kann.[95] Das einzige Allzweckformat für die klassische Partiturdarstellung heißt *Kern* und enthält Informationen über Tonhöhen, Tondauern, Bindungen, Pausen, Artikulationen, Phrasierung, Takstriche und rudimentären Notensatz.

Werkzeuge

Die Werkzeuge lassen sich in zwei Gruppen unterteilen. Die eine Gruppe arbeitet auf allen *Humdrum*-Datenformaten und verändert meistens nur die interne Struktur nicht aber das Format der Daten. Hierzu gehören Werkzeuge zum Aufteilen, Extrahieren und Zusammenfügen von Dateien sowie zur Suche von Mustern. Die zweite Gruppe verarbeitet nur bestimmte Datenformate und nimmt entweder formatspezifische Analysen oder Konvertierungen in andere Formate vor. Werkzeuge dieser Gruppe können Daten von *Kern* nach **`**pitch`** konvertieren, die harmonische Stellung von Akkorden feststellen und transponieren. Der eigentliche Nutzen von *Humdrum* besteht nicht in den vielen einzelnen Werkzeugen sondern erschließt sich erst durch deren Kombination. Durch die Verkettung vieler kleiner Verarbeitungsschritte lassen sich komplexe quantitative Analysen durchführen.[96] Dies soll am Beispiel einer Mustersuche im „*Happy Birthday*“-Fragment verdeutlicht werden. Das Muster könnte bei der Kürze des untersuchten Fragments natürlich von jedem Musikwissenschaftler „auf einen Blick“ erkannt werden, so daß der Aufwand der Auswahl und Verkettung von geeigneten Werkzeugen und Formaten als nicht gerechtfertigt erscheinen mag. Jedoch sollte bedacht werden, daß mit der gleichen Befehlskette auch eine komplette Opernpartitur in kürzester Zeit durchsucht werden könnte, wozu ein Mensch mehrere Stunden oder Tage benötigen würde.

Wir möchten herausbekommen, ob und an welchen Stellen die melodische Folge, die zu „*Happy Birthday*“ gesungen wird (g4, g4, a4, g4), in den Daten auftaucht. Da diese Melodie auch in anderen Transponierungen auftreten kann, suchen wir ganz allgemein nach einer Tonfolge, deren zweiter Ton identisch zum ersten(+-0), deren dritter Ton zwei Halbtöne höher(+2) und deren vierter Ton wieder zwei Halbtöne niedriger ist(-2). Wir suchen also nach jeder Tonfolge, die in Halbtonschritten ausgedrückt das Profil 0 +2 -2 hat. Um die Suche durchführen zu können, müssen wir „*Happy Birthday*“ zunächst in eine Halbtondarstellung konvertieren und dann die Entfernung zwischen jeweils zwei Folgehalbtönen berechnen. Wenn die Daten in diesem Format sind, können wir unsere Halbtonfolge suchen und als Anmerkung den Originaldaten hinzufügen. Die entsprechende *Humdrum*-Befehlsfolge sähe wie folgt aus, wobei **`happy.krn`** das „*Happy Birthday*“ in **`**kern`** und **`happy.pat`** das Suchmuster 0 +2 -2 enthält.

```
extract -i'**kern' happy.krn | semits | xdelta -s = | patt -t Happy-Muster -s = -f happy.pat | extract -i'**patt' | assemble happy.krn
```

[94] vgl. [Hur94, S.131]
[95] vgl. [Hur94, S.135]
[96] vgl. [Hur94, S.8]

```
!! Mildred and Patti Hill, composers
!! `Happy Birthday´
**patt              **kern              **text
*                   *clefG2             *
*                   *k[]                *
*                   *M4/4               *
*                   *C:                 *
!                   !re A4=440Hz        !(lyrics)
.                   =1                  =1
Happy-Muster        8.g/L               Hap-
.                   16g/Jk              -py
.                   4a/                 birth-
.                   4g/                 -day
.                   4cc\                to
.                   =2                  =2
.                   2b\                 you ;
Happy-Muster        8.g/L               hap-
.                   16g/Jk              -py
.                   4a/                 birth-
.                   =3                  =3
.                   4g/                 -day
.                   4dd\                to
.                   2cc\                you ;
*-                  *-                  *-
```

Abb. 4.1-4 „Happy Birthday“ mit automatisch markiertem „Happy-Muster“ (g g a g).

Die **`**patt-`** Sequenz wurde durch den aufeinanderfolgenden Einsatz von 6 Humdrum-Werkzeugen generiert.

Beurteilung

*Humdrum*s Stärken sind die vielen Datenformate, deren klare Struktur und die zahlreichen kleinen Werkzeuge die auf den Formaten arbeiten. Durch geschickte Kombination von Werkzeugen läßt sich fast jede quantitative musikwissenschaftliche Analyse durchführen. Durch die Konvertierungsmöglichkeit in Instrumentansteuerungsdaten (MIDI) sind die Daten-Domänen Analyse, Instrumentansteuerung und Akustik (indirekt über Instrumentansteuerung) abgedeckt. In den Datenformaten kommt die logische Struktur der Daten durch deren Benennung („a4“) und deren Anordnung in Sequenzen gut zum Ausdruck.

*Humdrum*s Schwächen sind der Mangel an zur Bearbeitung zur Verfügung stehenden Daten, die nur rudimentären Repräsentationsmöglichkeiten von Notensatzinformationen und die Voraussetzung, daß der Musikwissenschaftler UNIX beherrschen muß. Der Datenmangel liegt größtenteils in der Neuigkeit *Humdrum*s begründet und wird durch *scr2hmd* und kontinuierlich im Aufbau befindliche Bestände gelindert. Alleine das Fehlen von Notensatzinformationen hindert *Humdrum* daran, *das* zentrale Musidatenformat zu werden, denn obwohl die anderen drei Daten-Domänen erschlossen werden können, so ist ein für Publikationszwecke befriedigender Notensatz auf Basis der vorhandenen Datenformate nicht möglich. Zuletzt ist die Einbettung in die rein textorientierte UNIX-Umgebung zwar Grundlage der Vielseitigkeit *Humdrum*s, aber sie schränkt den potentiellen Benutzerkreis sehr stark ein.

Da *Humdrum*s Stärken enorm und seine Schwächen nicht struktueller Natur sind und durch Weiterentwicklung behoben werden können, ist es nach meiner Ansicht[97] die vielseitigste und zukunftsträchtigste aller verfügbaren Werkzeugsammlungen zur rechnergestützten Analyse.

SCORE

„*SCORE*“ ist die Bezeichnung für ein Notensatzprogramm und die von diesem erzeugten Formate. Das Programm ermöglicht den uneingeschränkten elektronischen Notensatz komplexer Partituren in professioneller Qualität. Die Partituren können textuell oder mit der Maus eingegeben werden und anschließend so auf einer Partiturseite positioniert werden, daß ein Musiker sie leicht lesen kann und zugleich ein ästhetischer Gesamteindruck entsteht. Auf der Grundlage einer eingegebenen Partitur können Auszüge für ausgewählte Orchesterstimmen oder Druckvorlagen erstellt werden.[98]

Datenformate

SCORE erzeugt Dateien in zwei Datenformaten, mit denen die komplette Partiturinformation repräsentiert werden kann. Beide Formate enthalten eine vollständige Liste der sich auf einer Partiturseite befindenden graphischen Symbole in der Reihenfolge ihrer Eingabe. Das „*SCORE*“ genannte Format ist ein platzsparendes Binärformat, das „*PMX*“ genannte Format ein längeres aber dafür besser lesbares ASCII-Format.[99]

Die Daten sind nur schwach strukturiert. Mehrere Seiten einer Partitur werden als getrennte Einheiten angesehen und als separate Dateien verwaltet. Abgesehen von der Option, die Zusammengehörigkeit der einzelnen Seiten durch fortlaufende Numerierung (z.B. **`"PART001.SCO"`**, **`"PART002.SCO"`**, …) auszudrücken, stehen die Einzelseiten beziehungslos nebeneinander. Innerhalb einer Seite ist jedes Symbol einem bestimmten System zugeordnet, wobei die Systeme vom unteren Seitenrand her beginnend und nach oben hin fortlaufend numeriert sind. Unabhängig von dieser Zuordnung kann jedes Symbol an beliebiger Stelle in den Daten auftauchen, so daß Blöcke, die die Symbole jeweils eines Systems enthalten, zwar möglich, nicht aber vorgeschrieben sind. Weitere Unterteilungen innerhalb einer Seite, z.B. Akkoladen, Überschriften und Taktzahlen, können nur graphisch, nicht aber logisch repräsentiert werden.

Für jedes Symbol gibt es sowohl im *SCORE*- als auch im *PMX*-Format einen Datensatz, der aus einer Liste von Parametern besteht, die das Symbol eindeutig bestimmen und seine Position auf der Partiturseite festlegen. Je nach Symboltyp gibt es eine unterschiedliche Parameterstruktur, jedoch enthalten die ersten vier Parameter (P1-P4) stets die gleichen Informationen:[100]

- P1 gibt an, um welche Art von Symbol es sich handelt.
- P2 enthält die Nummer des Systems, zu dem das Symbol gehört.
- P3 bestimmt die horizontale Position des Symbols, wobei eine Einheit bei Standardruckgröße ungefähr einem Millimeter entspricht. Der Nullpunkt befindet sich am linken Rand der Seite.

---

[97] *Humdrum* ist erst 1994 veröffentlicht worden, so daß Belegstellen noch nicht existieren.
[98] vgl. [Smi94, S.1]
[99] vgl. [Smi94, S.1]
[100] vgl. [Smi94, S.3]

- P4 legt die vertikale Position des Symbols in Schritten zu je der halben Entfernung zwischen zwei Systemlinien fest, wobei der Nullpunkt sich drei Schritte unterhalb der untersten Systemlinie befindet. Die Zählrichtung ist von unten nach oben.

Als Beispiel für eine Codierung wird in Abbildung 4.1-5 der Anfang von *„Happy Birthday“* in *PMX* gezeigt. Die einzelnen Symbole sind zur besseren Lesbarkeit von oben nach unten entsprechend ihrer horizontalen Position angeordnet. Dies soll aber nicht darüber hinwegtäuschen, daß Daten, wenn sie vom Verleger kommen, in willkürlicher Reihenfolge vorliegen können.

P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**	**P13**	**P14**
8	1	0	0	0	200								
3	1	1.19962											
18	1	6.79786	0	4	4								
16	1	11.3164	-4	1	1	0	0	0	0	0	6	8.57363	_00Hap
1	1	12.7772	5	10	0	0.75	0	10					
6	1	12.7772	5	5	21.4345	11	0	0	11	0	21.4345		
4	1	19.7636	-3.2	-3.2	18.4868	1	1.28571	-1.68584					
16	1	21.3901	-4	1	1	0	0	0	0	0	5	5.14624	_00py
1	1	21.4345	5	10	0	0.25	0	0					
16	1	28.0363	-4	1	1	0	0	0	0	0	8	9.72124	_00birth
1	1	29.394	6	10	0	1							
4	1	37.38	-3.2	-3.2	36.1977	1	1.28571	-1.36817					
16	1	39.2576	-4	1	1	0	0	0	0	0	6	7.43117	_00day
1	1	40.0048	5	10	0	1							
16	1	50.1641	-4	1	1	0	0	0	0	0	5	4.00377	_00to
1	1	50.1685	8	20	0	1							
14	1	59.5625	1										
16	1	61.7115	-4	1	1	0	0	0	0	0	6	7.71936	_00you
1	1	62.6584	7	20	1	2							
16	1	80.7828	-4	1	1	0	0	0	0	0	6	7.43117	_00hap
1	1	81.5809	5	10	0	0.75	0	10					
6	1	81.5809	5	5	89.7584	11	0	0	11	0	89.7584		
4	1	87.8851	-3.2	-3.2	86.6794	1	1.28571	-1.44					
16	1	89.714	-4	1	1	0	0	0	0	0	5	5.14624	_00py
1	1	89.7584	5	10	0	0.25	0	0					
16	1	96.3602	-4	1	1	0	0	0	0	0	8	9.72124	_00birth
1	1	97.651	6	10	0	1							
4	1	106.911	-3.2	-3.2	107.533	1	1.28571	0.131822					
14	1	107.773	1										
16	1	110.363	-4	1	1	0	0	0	0	0	6	7.43117	_00day
1	1	111.11	5	10	0	1							
1	1	121.274	9	20	0	1							
16	1	121.269	-4	1	1	0	0	0	0	0	5	4.00377	_00to
16	1	129.793	-4	1	1	0	0	0	0	0	7	9.15001	_00you;
1	1	131	8	20	1	2							

Abb. 4.1-5 *„Happy Birthday“* im *PMX* -Format.

In *PMX* wird jedes Symbol durch bis zu 14 Paramater (**P1**-**P14**) repräsentiert. Ein Parameter steht meistens – aber nicht immer – für ein bestimmte Eigenschaft. Der Symboltyp und die für ihn spezifische Interpretation wird durch **P1** bestimmt. Z.B. steht 1 für eine Note, 3 für einen Schlüssel, 4 für einen Strich, 8 für ein System, 14 für einen Taktstrich und 16 für einen Text.

Die Parameter legen fast ausschließlich die graphische Gestalt und die Anordnung eines Symbols auf einer Partiturseite fest und enthalten bis auf drei Ausnahmen keine logischen Informationen. Die drei Ausnahmen sind allesamt optional und nur dann notwendig, wenn Auszüge gemacht oder automatische Verbesserungen des Layouts vorgenommen werden sollen. Es sind die Folgenden:

- Jedem System kann gemäß der in ihm notierten Orchesterstimme(n) eine eindeutige Instrumentnummer zugeordnet werden, anhand derer später Auszüge erstellt werden können. Dies ist notwendig, da Orchesterstimmen nicht an gleichbleibenden vertikalen Positionen notiert werden müssen. Tritt zum Beispiel eine hohe Stimme hinzu, so verschieben sich traditionsgemäß alle tiefern Stimmen um einen Platz nach unten.

Ohne die Instrumentnummer könnte das Programm nicht feststellen, welche Stimme in welchem System notiert ist.[101]

- Die mathematische Notendauer weicht von der graphischen Notendauer ab. Dies kann bei Abbreviaturen oder anomalen Teilungen (Triole, Quintole, etc.) der Fall sein.[102]
- Die mathematische Pausendauer weicht von der graphischen Pausendauer ab. Dieser Fall tritt ein, wenn eine ganze Pause einen ganzen Takt Pause bezeichnet, dessen Länge aber nicht 4/4 oder eine gleichwertige Dauer beträgt.[103]

Beurteilung

*SCORE*s Stärke ist die Fähigkeit, eine vollständige Partitur in professioneller Notensatzqualität repräsentieren zu können und die Tatsache, daß es von namhaften Musikverlagen verwendet wird (Peters, Schirmer, Schott).[104]Die Vollständigkeit wird dadurch erreicht, daß alle Symbole, die sich auf einer Partiturseite befinden können und aus denen sie besteht, mit *SCORE* darstellbar sind. Die Verwendung durch die Verlage bedeutet, daß in *SCORE* große musikalische Werke in einer Form elektronisch vorliegen, die der gedruckten ebenbürtig ist. Dies kann von keinem anderen Musikdatenformat gesagt werden.

*SCORE*s Schwächen aus Sicht der rechnergestützten musikwissenschaftlichen Analyse sind sein Strukturmangel und das fast vollständige Fehlen von Daten aus den Nicht-Notensatz-Domänen. Die vorgeschriebene Struktur geht nur soweit, daß eine Partiturseite aus mehreren Symbolen besteht, die einem System zugeordnet werden müssen. Alles weitere (Seitenzusammengehörigkeit und -reihenfolge, Zuordnung von Systemen zu festen Orchesterstimmen und logische Noten- und Pausenlängen) ist optional. Alle optionalen Daten sind schlimmstenfalls nicht vorhanden und müssen zu Analysezwecken aus den vorhandenen Daten rekonstruiert werden.

Der direkte analytische Nutzen von *SCORE* ist vernachlässigbar. Indirekt jedoch ist es eine wertvolle Datenquelle zur Gewinnung vollständiger, qualitativ hochwertiger Analysedaten.

Beurteilung *Humdrums* und *SCORE*s

SCORE ergänzt *Humdrum* ideal, da es genau die Domäne „Notensatz“ hervorragend abdeckt, die bei *Humdrum* am schwächsten ausgeprägt ist, und selbst keine konkurrierende „Analyse"-Struktur aufweist. Durch

[101] vgl. [Smi94, S.8]
[102] vgl. [Smi94, S.4]
[103] vgl. [Smi94, S.5]
[104] vgl. [Hol90, Acknowledgements]]

eine Konvertierung von *SCORE*-Daten nach *Humdrum* können alle domänenbedingten Mängel von *Humdrum* beseitigt werden. *Humdrum* kann dann über hochwertige und zahlreiche strukturierte Daten aus den Bereichen „Analyse“, „Notensatz“, „Instrumentansteuerung“ und „Akustik“, die mit vielseitigen und mächtigen Analysewerkzeugen untersucht werden können. Damit könnte es *das* Musikdatenformat und Grundlage für allgemeinste musikwissenschaftliche Analysen werden.

Diese Konvertierung leistet *scr2hmd*.

scr2hmd

Scr2hmd ist ein Programmpaket zur automatischen Konvertierung von Notensatzdaten im *SCORE*-Format in Analysedaten in zwei *Humdrum*-Formate. Mit seiner Hilfe können vollständige Partituren in wenigen Stunden und ohne qualitative Einbußen der Analyse zur Verfügung gestellt werden.

Im Folgenden wird zunächst die Qualität der Umwandlung anhand einer Charakterisierung der Merkmale der Daten in den beiden *Humdrum*-Ausgabe-Formaten *Kern* und dem neu hinzugekommenen *Layout* beschrieben. Daraufhin wird ein Überblick über den Konvertierungsprozeß und die an ihm beteiligten Programme gegeben und schließlich die Arbeitsweise des zentralen Programms *pmx2hmd* erläutert. Die technischen Daten der Programme und deren Aufrufe aus UNIX heraus befinden sich im Anhang.

Ausgabedaten

scr2hmd erzeugt im Regelfall für jede in der Partitur notierte Orchesterstimme genau eine *Kern*- und genau eine *Layout*-Datei, die die in ihnen repräsentierte Orchesterstimme vom ersten bis zum letzten Takt des Werks enthalten. *Kern* und *Layout* zusammen bilden die Partiturinformationen vollständig ab. *Kern* enthält die logischen Analysedaten ohne jegliche Informationen über deren graphische Gestaltung, *Layout* enthält die Daten für die graphische Gestaltung (inklusive Gesangstexte) ohne irgendeine Information über die logische Bedeutung der Symbole. Für die normale rechnergestützte quantitative musikwissenschaftliche Analyse in einer textorientierten UNIX-Umgebung sind nur die *Kern*-Daten von Interesse. Sie enthalten alle analytisch wertvollen Daten. Die *Layout*-Daten werden nur dann benötigt, wenn die Partitur graphisch dargestellt werden soll. Dies ist bei der Kapselung von *Humdrum* durch eine graphische Benutzungsoberfläche, in der der Anlysator die Partitur tatsächlich auf dem Bildschirm sehen kann, sowie bei der Gewinnung von Notensatzdaten aus den *Humdrum*-Daten der Fall.

Die von *scr2hmd* erzeugten *Kern*- und *Layout*-Dateien enthalten normalerweise genau eine Sequenz, die die Daten für genau eine Orchesterstimme vom ersten bis zum letzten Takt des Werkes enthalten. Als eine Orchesterstimme gilt dabei jede Stimme, die am linken Rand eines Systems der Partitur als individuelle Stimme notiert wird. Zwei Stimmen, die in einem System notiert werden gelten als zwei Stimmen, z.B. Fagott 1 und Fagott 2. Eine Instrumentengruppe, die nicht weiter aufgegliedert wird, gilt ebenfalls als eine Stimme, z.B. Violoncello (statt Violoncello 1, Violoncello 2, ..., Violoncello 12).

Zusätzlich sollte vom Benutzer eine Partiturdatei angelegt werden, aus der hervorgeht, welche *Kern*- (und damit auch *Layout*-Dateien) zu einem Werk gehören.

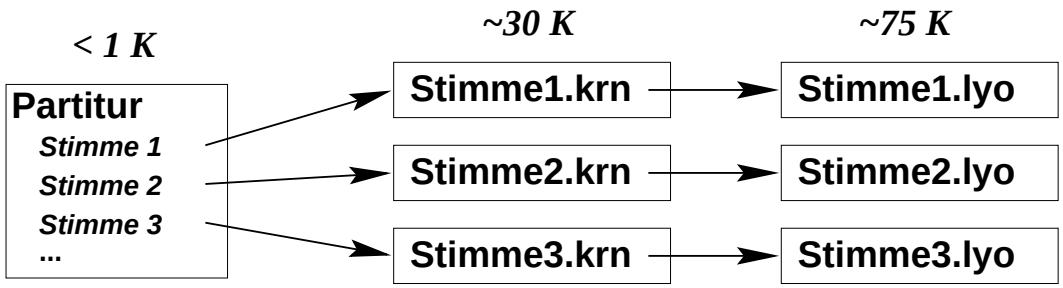


Abb. 4.1-6 Dateistruktur nach der Konvertierung mit *scr2hmd*.

Die Partiturdatei enthält eine Liste aller enthaltener Stimmen in Form von Verweisen auf logische *Kern*-Dateien. Die *Kern*-Dateien enthalten wiederum Verweise auf zur Darstellung benötigte *Layout*-Dateien. Minimale, durchschnittliche und maximale Dateigrößen (in Kilobyte) für alle Dateitypen sind im Beispiel *„Hänsel und Gretel“* (Gesamtgröße 4910 Kilobyte):

	Minimum	**Durchschnitt**	**Maximum**
Partitur	1	1	1
Kern	1 (Becken)	50	200 (Violine I)
Layout	1 (Becken)	50	200 (Violine I)

Ein Sonderfall der Sequenzstruktur tritt auf, wenn die Stimme im Verlauf der Partitur vorübergehend getrennt wird (z.B. Violoncelli divisi). Die *Kern*-Datei enthält dann für die Dauer der Teilung zwei oder mehrere Sequenzen, zur *Layout*-Datei kommen weitere *Layout*-Dateien hinzu.

Im Folgenden werden die Eigenschaften der konvertierten Daten im Detail beschrieben. Sofern keine weiteren Angaben gemacht werden, stimmen Art und Umfang der Codierung mit der in *„The Humdrum Toolkit Reference Manual“*[Hur94] überein.

***Kern*.** Die Möglichkeiten des Formats werden fast vollkommen ausgeschöpft. Auf die rudimentäre graphische Gestaltung (**`/\LJKk`**) wird bewußt zugunsten der *Layout*-Daten verzichtet. Bindungen zwischen Tönen unterschiedlicher Höhe (**`()`**) werden nicht abgebildet. Probennummern oder ähnliche logische Markierungen werden nicht berücksichtigt. Repräsentiert werden:

- Tonhöhen (**`abcdefgABCDEFG-#n`**)
- Tondauern (**`0123456789.q`**) samt -auch über Partiturseitengrenzen hinausgehenden- Bindungen (**`[_]`**).
- Abbreviaturen und Artikulationszeichen, soweit *Kern* über diese verfügt (**`MmS$TtWwROz´"`~^:I`**).
- Pausen (**`r`**)
- Numerierte Taktstriche (**`=1234567890`**). Um jeden Takt eindeutig identifizieren zu können, werden alle Takte des Werks durchlaufend numeriert.
- Instrumentname (**`*I`**)
- Transponierung (**`*ITr`**)
- Notenschlüssel (**`*clef`**)
- Tonartversetzungszeichen (**`*k`**)

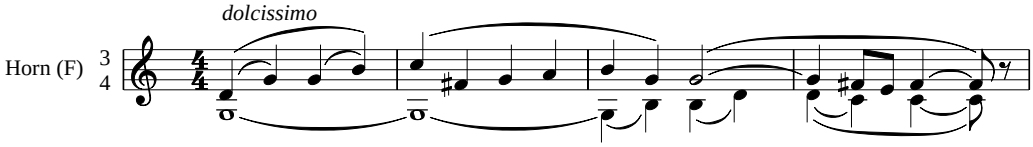
- Takt (**`*M`**)
- Verweis auf die zugehörige *Layout*-Datei (**`!layout:`**)

Layout. *Layout* ist ein im Standardumfang von *Humdrum* nicht enthaltenes Format. Es enthält taktweise die Informationen zur Darstellung desselben in *SCORE*-Parametern. Diese Art der Codierung ist nicht im Sinne von *Humdrum*, das ein Datum (und nicht mehrere durch Leerzeichen getrennte Parameter) pro Symbol vorschlägt,[105] und dient in dieser Form nur als momentane Arbeitsgrundlage. Da mehrere Symbole synchron zu einer Note oder Pause notiert werden können, besteht innerhalb von Takten kein Verhältnis der Gleichzeitigkeit zwischen *Kern*- und *Layout*-Daten; lediglich die Taktstriche sind synchron.

Eine verfeinerte Version von *Layout* könnte *SCORE*-Parameter z.B. mit Doppelpunkten trennen, so daß *Humdrum* sie als eine Einheit auffassen würde. Dadurch könnten mehrere Symbole in einer Zeile dargestellt und eine Gleichzeitigkeit zu den *Kern*-Daten hergestellt werden.

Die *Layout*-Daten werden nahezu unmodifiziert aus den *SCORE*-Daten übernommen. Nur die Parameter, die sich auf die horizontale Position beziehen[106] werden so umgerechnet, daß sie sich auf den Beginn des Taktes und nicht mehr auf den linken Seitenrand beziehen.

[105] vgl. [Smi94, S.13]
[106] P3 sowie P9 bei Pausen, P11, P12, P14 und P15 bei Querbalken und P6 bei Linien.



**kern	**layout
*Icor	*SCORE
*ITrd4c7	*Icor
=1	=1
!layout:…	!file: hg001.spmx
*clefG2	!staff:14
*M4/4	1 14 13.3602 2 10 0 1 0 0 0 0
4G	16 14 13.6785 17 1 1 0 0 0 0 0 13 13.2094 dolcissimo
4c	5 14 14.4385 10.5 14.5 32.4518 2.20067 -1
4c	5 14 15.0251 6 8 19.5451 1.326 -1
4e	1 14 19.5083 5 10 0 1 0 0 0 0
=2	1 14 25.654 5 10 0 1 0 0 0 0
4f	5 14 27.0518 8 10 32.0118 1.348 -1
4B	1 14 31.7996 7 10 0 1 0 0 0 0
4c	14 14 37.9542 1 0
4d	=2
=3	1 14 1.9203 8 10 0 1 -0.5 0 0 0
4e	5 14 3.6176 13.8 12.8 35.1243 2.47533 -1
4c	1 14 8.0801 4 12 0 1 0 0 0 0
[2c	1 14 14.2258 5 10 0 1 0 0 0 0
=4	1 14 20.3715 6 10 0 1 0 0 0 0
4c]	14 14 26.5261 1 0
8B	=3
8A	1 14 1.9202 7 10 0 1 -1
[4B	1 14 8.0657 5 10 0 1 0 0 0 0
8B]	5 14 14.1048 6 6 30.3965 1.5146 -2
8r	1 14 14.2155 5 10 1 2 0 0 0 0
	5 14 15.6648 10.5 10.5 53.0385 2.76867 -1 2
	14 14 28.5875 1 0
	=4
	1 14 1.92 5 10 0 1 0 0 0 0
	1 14 8.082 4 12 0 0 0.5
	1 14 12.196 3 10 0 0 0.5
	5 14 16.171 5.5 5.5 25.339 1.15842 -2
	1 14 16.31 4 10 0 1 0 0 0 0
	1 14 24.519 4 10 0 0 0.5 0 1 0 0
	2 14 28.633 0 1 0 0.5 0 -1
	14 14 32.742 1 0

**kern	**layout
*Icor	*SCORE
*ITrd4c7	*Icor
=1	=1
!layout:…	!file: hg001.spmx
*clefG2	!staff:14
*M4/4	1 14 13.3602 -2 20 2 4 -1.5 0 0 0
[1C	5 14 13.3602 -2.5 -2.5 39.8745 -2.02
=2	16 14 13.6785 17 1 1 0 0 0 0 0 13 13.26
1C_	14 14 37.9542 1 0
=3	=2
4C]	1 14 1.9203 -2 20 2 4 -1.5 0 0 0
4E	5 14 1.9203 -2.5 -2.5 28.4463 -2.026
4E	14 14 26.5261 1 0
4G	=3
=4	5 14 1.7315 -4 -2 6.5448 -1.34067 -1
4G	1 14 1.9202 -2 20 0 1 -1.5 0 0 0
4F	1 14 8.0657 0 20 0 1 -1.5 0 0 0
[4F	5 14 14.1982 -3 -1 19.0115 -1.34067
8F]	1 14 14.2155 0 20 0 1 -1.5 0 0 0
8r	1 14 20.3695 2 20 0 1 -1.5 0 0 0
	14 14 28.5875 1 0
	=4
	5 14 1.891 -3 -4 23.131 -2.062 -1 2
	1 14 1.92 2 20 0 1 -1.5 0 0 0
	5 14 2.037 -0.5 -1.5 6.557 -1.11914
	1 14 8.082 1 20 0 1 -1.5 0 0 0
	1 14 16.325 1 20 0 1 -1.5 0 0 0
	5 14 16.704 -0.5 -0.5 23.143 -1.221
	1 14 24.519 1 20 0 0.5 -1 1 0 0
	2 14 28.633 0 1 0 0.5 0 -1
	14 14 32.742 1 0

Abb. 4.1-7 Die ersten vier Takte der Hörner Nummer 3 und 4 aus Engelbert Humperdincks *„Hänsel und Gretel“* im Notenbild und nach der Konvertierung durch *scr2hmd*.

In der ersten und dritten Spalte die *Kern*-Dateien, in der zweiten und vierten Spalte die dazugehörigen *Layout*-Dateien.

Da die Daten mit Ausnahme der horizontalen Positionen nicht weiter verändert werden, werden Orchesterstimmen, die in der Partitur gemeinsam mit einer anderen in einem System notiert werden, unter Umständen nicht optimal dargestellt. Dies betrifft vor allem die Halsrichtung. Eine automatische Korrektur ist aber aufgrund der Komplexität von Notensatzproblemen nicht möglich.[107]

Repräsentiert werden:

- Symbole in *SCORE*-Parameterdarstellung (**`1234567890.`**)
- Instrumentname (**`*I`**)
- Numerierte Taktstriche (**`=1234567890`**)
- Verweise auf die ursprüngliche *SCORE*-Datei (**`!file:`**)
- Verweise auf die Systemnummer innerhalb der ursprünglichen *SCORE*-Datei nach *SCORE*-Zählweise. (**`!staff:`**)

[107] Siehe Fußnote auf Seite 68

scr2hmd Zusammenfassung

Scr2hmd wandelt *SCORE*-Daten in *Humdrum*-Daten um. Je Orchesterstimme, auch wenn sie teilweise mit einer anderen gemeinsam in einem System notiert wird, wird eine *Kern*- und mindestens eine *Layout*-Datei angelegt. Die *Kern*-Datei enthält die logischen Partiturinformationen über Tonhöhen und -dauern, Abbreviaturen, Artikulationen, Pausen, Taktnummern, Instrumentname, Transponierung, Notenschlüssel, Tonartversetzungszeichen und Takt. Die *Layout*-Dateien enthalten die Notensatzinformationen in der gleichen Vollständigkeit wie *SCORE*. Gesangstexte und logische Abschnittsmarkierungen wie Probennummern werden bisher nicht verarbeitet, könnten aber in Zukunft ohne weiteres berücksichtigt werden.

Die Probleme, die sich bei möglichen Erweiterungen von *scr2hmd* stellen, sind kleiner als die, die bereits bei der Trennung von in einem System notierten Orchesterstimmen überwunden worden sind. Die Lösung dieses und anderer Interpretationsprobleme steht im Mittelpunkt des Rests dieses Kapitels.

Der Konvertierungsprozeß

Die *SCORE*-Daten liegen in unsortierter, schwach strukturierter Form und ohne logischen Informationsgehalt in vielen Partiturseitendateien vor. Um sie in stimmenorientierte *Humdrum*-Daten konvertieren zu können, sind die folgenden Schritte notwendig:

1. Die zu einer Orchesterstimme gehörigen Symbole eines Systems müssen durch Interpretation im Kontext mit Notenschlüssel, Tonartversetzungszeichen und Transpositionsinformation als Noten, Pausen, Taktstriche, etc. gedeutet werden.
2. Vorher muß ermittelt werden, welche Symbole eines Systems welcher Orchesterstimme zugeordnet sind.
3. Davor muß festgestellt werden, welche der über alle Partiturseiten verstreuten Systeme zu einer Orchesterstimme gehören und ob diese durchgängig notiert sind oder nicht.
4. Ganz zu Beginn müssen die Partiturseitendateien intern so geordnet werden, daß eine algorithmische Verarbeitung überhaupt möglich ist.
5. Ggf. müssen die Daten in einem weiteren Vorbereitungsschritt erst noch aus dem Binärformat in das für Verarbeitungszwecke günstigere ASCII-Format umgewandelt werden.

Aus Benutzersicht führt *scr2hmd* alle diese Schritte auf einmal durch. Intern werden die Arbeiten jedoch auf mehrere kleine Programme verteilt:

- *scr2pmx* wandelt die *SCORE*-Binärdaten in *SCORE*-ASCII-Daten (*PMX*-Dateien) um.
- *pmx2spmx* sortiert den Inhalt der *PMX*-Dateien und entfernt doppelte Datensätze.
- *staffscan* untersucht die Systemstruktur der *PMX*-Dateien und hält die Ergebnisse in Form der zu einer Orchesterstimme gehörenden Systeme, deren ursprüngliche Transpositionen und der Struktur der Takte fest.
- *pmx2hmd* liest anhand der von *staffscan* ermittelten Steuerinformationen die *PMX*-Dateien, trennt mehrere in einem System notierte Orchesterstimmen und interpretiert schließlich die einzelnen Symbole. Die Ergebnisse werden in *Kern*- und *Layout*-Dateien geschrieben.

Die Besonderheiten und die nicht-trivialen Algorithmen der einzelnen Programme werden im Folgenden programmweise erläutert. Zusätzlich wird auf noch nicht beseitigte Fehler und mögliche Verbesserungen eingegangen.

scr2pmx wandelt *SCORE*-Binärdaten, die im Regelfalle zum Abspeichern von Partiturseiten verwendet werden, in zur Verarbeitung besser geeignete *SCORE*-ASCII-Daten (*PMX*) um. Die erzeugten *PMX*-Dateien weichen in zwei Aspekten von den *PMX*-Dateien ab, die *SCORE* mit dem Kommando **`pmx`** erzeugt:

- Alle Gleitkommazahlen werden mit minimaler Nachkommastellenanzahl geschrieben, um Platz zu sparen.
- Texte (Code 16) werden in einer Zeile und mit allen Parametern geschrieben.

Algorithmus: Die Umwandlung erfolgt gemäß dem Algorithmus aus [Smi94, S.2].

Fehler: Vom *SCORE*-Programm aus können komplexe Partiturseiten in zwei Teilen (oberer und unterer Teil) abgespeichert werden. Die unteren Hälften solcher getrennt abgespeicherter Partiturseiten werden nicht richtig umgewandelt. Der Algorithmus aus [Smi94] gibt leider keine Auskunft über die Behandlung dieser Art der Aufteilung.

pmx2spmx bringt die *PMX*-Dateien in eine solche Form, daß sie in geordneter Weise systemweise von links nach rechts gelesen werden können, ohne dabei auf doppelte Symbole zu stoßen.

Algorithmus: Mehrere Symbole mit identischer horizontaler Position werden mit dem UNIX-Tool sort in der Reihenfolge Noten, Pausen, Bindebögen sortiert. Symbole gleichen Typs werden nach ihrer vertikalen Position geordnet.

staffscan untersucht die *PMX*-Dateien auf ihre Systemstruktur. Es stellt fest, welche Instrumente in der Partitur vorkommen, in welchen Systemen sie auf welcher Seite notiert, wie sie anfänglich transponiert und wieviel Takte in jeder Akkolade vorhanden sind.

Algorithmus:

- **Instrumente.** Als Instrumentnamen werden alle Texte links eines Systems angesehen. Falls ein unbekannter Instrumentname gefunden wird, muß der Benutzer diesen einer Zuordnungstabelle hinzufügen.
- **Systemzuordnung.** Gemäß der Instrumentnamen am linken Systemrand wird jedem System eine oder mehrere Orchesterstimmen zugewiesen. Mehrere oder logisch nicht richtig zugeordnete Instrumentnamen werden anhand ihrer vertikalen Position dem richtigen System zugeordnet.
- **Transponierung.** Die eingeklammerten Teile der Instrumentnamen werden als Transponierung interpretiert.
- **Taktstruktur.** Die Taktstrichsymbole im untersten System einer Akkolade werden gezählt. Die Taktstruktur ist wichtig für die richtige Taktnumerierung von Orchesterstimmen, die zwischenzeitlich pausieren.

pmx2hmd liest die zu einer Orchesterstimme gehörenden Systeme, filtert die zur gewünschten Orchesterstimme gehörigen Symbole heraus und wandelt sie kontextsensitiv in *Kern*- und *Layout*-Daten um.

Algorithmus: Anhand der von staffscan erstellten Orchesterstimmenpfaddatei werden die zu einer Stimme gehörenden Systeme ermittelt und Takte, in denen sie pausiert, mit Leertakten aufgefüllt. Zur Stimmenseparierung bei mehreren Stimmen in einem System wird eine Listenstruktur erstellt, in der synchrone Noten, Pausen und Bindebögen, deren horizontale und vertikale Position, ggf. Halsrichtung und Dauer der Noten und textuelle Indikatoren für aktive Stimmen vermerkt werden. Durch Interpretation dieser Liste ist eine exakte Stimmenzuordnung möglich. Anhand dieser Informationen werden die zu einer Stimme gehörenden Symbole aus den *PMX*-Dateien gelesen und durch Interpretation in *Kern*- und *Layout*-Daten überführt. Abschließend werden bei Bedarf noch Transponierungen, Verbinden von partiturseitenübergreifenden Bindebögen und Aufteilungen von vorübergehend getrennt spielenden Instrumenten vorgenommen.

Fehler: Synchrone ganze Noten ohne „logische Halsrichtung“ (int(P5/10)) werden bei vertikaler Vertauschung (obere Note gehört zur unteren, untere Note gehört zur oberen Stimme) nicht der richtigen Orchesterstimme zugeordnet.

Mögliche Verbesserungen: Die *Kern*-Datei könnte mit logischen Markierungen (section labels) versehen werden.

Gesangstexte könnten in eine separate `**text`-Datei ausgegeben werden.

Mit *SCORE*, *Humdrum* und *scr2hmd* steht eine im technischen Sinne vollständige Analyseumgebung zur Verfügung: Analysedaten aus allen möglichen Datendomänen sind vorhanden und können nahezu beliebig untersucht und manipuliert werden. Damit diese Umgebung auch von Musikwissenschaftlern problemlos genutzt werden kann, wird im nächsten Kapitel die mit WAM realisierte fachliche Kapselung vorgestellt. Doch auch im ungekapselten Rohzustand ist die ***Umgebung für allgemeine musikwissenschaftliche Analysen*** schon ein wenig näher gerückt.

4.2 Fachliche Kapselung durch WAM

Im Falle einer absoluten Neuentwicklung mit WAM bleibt es den Entwicklern vollständig selbst überlassen, wie sie die fachlichen Werkzeuge, Materialien, Automaten und Umgebungen auf technischer Seite realisieren. Die vorhandenen Entwurfsmuster geben zwar Aufschluß über die dabei anzuwenden architektonischen Prinzipien, aber die Auswahl der konkreten Struktur ist Sache der Entwickler.

Bei der Entwicklung der Leitmotivanalysewerkzeuge liegt der Fall anders. Mit *Humdrum* und den in *Kern* und *Layout* vorliegenden Partiturdaten ist auf unterster technischer Ebene bereits eine sehr vielseitige und umfangreiche Sammlung von Daten und UNIX-artigen Werkzeugen gegeben. Auf dieses System nicht aufzubauen würde einen unnötig großen Implementationsaufwand bedeuten.

Die Lösung besteht in einer fachlichen Kapselung von *Humdrum*. Das bedeutet, daß sich die Leitmotivanalysewerkzeuge dem Anwender gemäß dem WAM-Leitbild als elektronische Version seines gewohnten fachlichen Arbeitsplatzes präsentieren und daß zugleich alle Aktionen des Anwenders letztendlich durch *Humdrum*-Operationen realisert werden. Die Architektur ist also zum Anwender hin die „normale“ WAM-Architektur und zu den Daten und Operationen hin die „normale“ *Humdrum*-Architektur; dazwischen findet die eigentliche Kopplung statt. Diese Kopplung ist der Gegenstand dieses Kapitels.

Zunächst wird die Kopplung bei den Leitmotivanalysewerkzeugen vorgestellt. Danach werden die Vor- und Nachteile dieser Lösung diskutiert und eine Ausweitung auf weitere musikwissenschaftliche Analyseaufgaben skizziert. Schließlich werden Überlegungen angestellt, inwiefern diese Art der Kapselung von bestehenden Systemen durch WAM generalisiert werden kann.

Kapselung bei den Leitmotivanalysewerkzeugen

Den fachlichen Leitmotivanalysewerkzeugen Partituranalysewerkzeug, Leitmotivmarkierer, Leitmotivnotizeditor und Leitmotivbetrachter und den Materialien Partitur und den aus Leitmotivnotizen bestehenden Leitmotivtabellen stehen die 39 UNIX-artigen *Humdrum*-Werkzeuge sowie die Partitur- und Leitmotivtabellendateien im *Kern*- und *Layout*-Format gegenüber. Bevor auf die eigentliche Kopplung zwischen WAM- und *Humdrum*-Teil eingegangen werden kann, muß zunächst erklärt werden, welche dieser WAM-Systembestandteile auf welche *Humdrum*-Systembestandteile abgebildet werden.

Sowohl bei WAM als auch bei *Humdrum* sind die Materialien bzw. Daten an sich für den Anwender nicht faßbar. Der Zugriff geschieht niemals direkt sondern immer mittels eines fachlichen (WAM) bzw. eines UNIX-artigen (*Humdrum*) Werkzeugs. Diesen Werkzeugen obliegt es, die jeweiligen Gegenstände des Handelns in geeigneter Form zu repräsentieren. Was immer der Anwender von diesen Gegenständen sieht: er sieht es nur durch die Brille, die sein Werkzeug ihm bietet. WAM-seitig sind dies das Partituranalysewerkzeug und der Leitmotivbetrachter, auf *Humdrum*-Seite z.B. die Konvertierer **`midi`** und **`semits`** oder die UNIX-Werkzeuge **`cat`** und **`more`**.

Da die Leitmotivanalysewerkzeuge soweit wie möglich auf *Humdrum* aufsetzen, ist aus den oben genannten Gründen ein direkter Zugriff auf die *Humdrum*-Daten vom WAM-Teil aus in den meisten Fällen gar nicht notwendig. Die *Humdrum*-Daten können fast ausschließlich direkt mit Hilfe der *Humdrum*-Werkzeuge bearbeitet werden.

Von diesem Vorgehen muß nur in einem -allerdings bedeutenden- Fall abgegangen werden: bei der graphischen Materialdarstellung. Da *Humdrum* dem Anwender nur eine rein textuelle Benutzungsschnittstelle bietet, kann für die graphische Repräsentation der Partitur und der Leitmotivtabellen nicht auf vorhandene *Humdrum*-Werkzeuge zurückgegriffen werden, so daß in diesem einen Punkt der WAM-Teil direkt auf die *Humdrum*-Daten zugreifen muß.

Anhand dieses einen Punktes wird die Kopplung von Material an Daten, die sich ansonsten auf das Bereitstellen des richtigen Dateinamens für *Humdrum*-Werkeuge beschränkt, im nächsten Abschnitt exemplarisch dargestellt. Danach wird die wesentlich unkompliziertere Kopplung der fachlichen mit den *Humdrum*-Werkzeugen beschrieben.

Materialien

In Kapitel 4.1 wurde die *Humdrum*-Datenstruktur der Partitur[108] ausführlich dargestellt. Die dort beschriebenen Dateien enthalten prinzipiell alle zur Visualisierung notwendigen Informationen, aber sie sind nicht immer in einer solchen Form, daß eine direkte Verwendung seitens des WAM-Teils möglich ist. Zum einen müssen die zur Interpretation der Noten notwendigen Kontextinformationen[109] erst aus den Daten aggregiert werden, zum anderen verbietet sich aufgrund des Umfangs der Daten eine vollständige Haltung im Hauptspeicher.

Die Kopplung muß also zweierlei leisten: Erstens müssen die notwendigen Kontextinformationen bereitgehalten werden, damit die Werkzeuge zügig auf das Material zugreifen können, ohne dieses erst langwierig generieren zu müssen. Zweitens muß eine interne Datenstruktur gefunden werden, die eine platzsparende Verwaltung der Materialien erlaubt.

Dies leisten die von mir eingesetzten Metamaterialien. Sie enthaltenen die Partiturinformationen nicht direkt sondern beschränken sich auf Informationen *über* die in *Humdrum*-Dateien abgelegten Materialinformationen. Der WAM-Teil geht mit diesen Metamaterialien so um, als ob es sich dabei um normale fachliche Materialien handeln würde. Von den Metamaterialien aus wird dann, sofern es mit einem Nachschlagen in den internen Kontextinformationen nicht getan ist, gezielt auf die externen *Humdrum*-Dateien zugegriffen.

Das Metamaterial Partitur besteht zunächst einmal aus den (gemäß Glossar selbstverständlichen) Komponenten „Name“ und den Verweisen auf die einzelnen Orchesterstimmen. Diese Orchesterstimmen ent-

[108] Als weiteres Material bestehen noch die Leitmotivtabellen. Da die Art der Kapselung jedoch weitgehend identisch zu der der Partitur ist, beschränke ich mich hier auf die Partitur.
[109] Kontextinformationen sind Notenschlüssel, Tonartversetzungszeichen, Transponierung des Instruments und Takt.

halten wiederum zunächst einmal nur ihren Namen und den Namen der *Kern*-Datei, in der ihre logischen Daten festgehalten sind.

Hinzu kommen die Kontext- und anderen aggregierten Informationen, die sich nicht punktuell aus den Dateien ablesen lassen. Beim Laden der Partitur werden alle enthaltenen *Humdrum*-Dateien komplett durchgekämmt, die erforderlichen Informationen generiert und in den entsprechenden Datenstrukturen abgelegt. Zu diesen Informationen gehören der Interpretationskontext, die graphischen Taktlängen und die graphische Partiturdarstellung.

Der **Interpretationskontext**, der die musikalische Interpretation eines Notenzeichens erst möglich macht, wird für jede Orchesterstimme separat verwaltet, da er zu keinem Zeitpunkt für das ganze Orchester identisch ist. Z.B. werden die Violinen zumeist im Violinschlüssel, die Bratschen im Altschlüssel und die Bässe im Baßschlüssel notiert. Da diese Kontextinformation jedoch auch nicht jeden Takt wechselt, enthält jede Orchsterstimme eine Interpretationsliste, die zu jedem Takt, in dem sich der Interpretationskontext ändert, einen Eintrag mit dem ab dort geltenden Kontext enthält. Wenn also z.B. der Anwender die Partitur ab Takt 189 betrachten möchte, dann kann das Metamaterial die dazugehörige Kontextinformation liefern, indem es in den internen Interpretationslisten aller Orchesterstimmen die ersten Einträge heraussucht, deren Taktzahl 189 oder kleiner ist, und diese Informationen an das darstellende Werkzeug weitergibt. Die externen *Humdrum*-Daten müssen nicht erneut gelesen werden.

Die **graphischen Taktlängen** sind im Gegensatz zu den Kontextinformationen für alle Orchesterstimmen identisch und können zentral von der Partitur verwaltet werden. Damit der Anwender die Partitur von einem ganz bestimmt Takt an betrachten kann, muß die horizontale Verschiebeung vom momentanen Betrachtungspunkt aus exakt berechnet werden können. Dies ist nicht durch die Multiplikation einer festen Taktlänge mit der Anzahl der zwischen der aktuellen und der Zieltakttatktzahl liegenden Takte machbar, da alle Takte unterschiedliche graphische Längen aufweisen können. So benötigen z.B. sechzehn Sechzehntelnoten in einem 4/4-Takt mehr Raum als eine ganze Note. Die Taktlängen werden wie die Interpretationskontexte beim Laden der Partitur einmalig aus den *Layout*-Dateien herausgelesen und dann in einem Feld abgelegt, das die graphische Anfangsposition jedes Taktes enthält. Wenn z.B. der Anwender momentan den 32. Takt betrachtet und anschließend den 145. Takt betrachten möchte, so kann das Metamaterial die notwendige horizontale Verschiebung aus der Differenz der entsprechenden Feldinhalte sofort berechnen. Dadurch kann ein aufwendiges „Durchhangeln“ durch die entsprechenden *Humdrum*-Dateien vermieden werden.

Die **graphische Partiturdarstellung** ist zu komplex, als daß sie vollständig im Hauptspeicher gehalten werden könnte. Alleine die externen *Layout*-Daten nehmen mit XXXX KB bei *„Hänsel und Gretel“* beträchtlichen Raum ein und die dementsprechende interne Datenstruktur mit allen graphischen Details und den zu deren Darstellung notwendigen Methoden liegt mindestens um den Faktor 10 darüber. Daher wird stets nur der Bereich zwischen dem ersten und dem letzten im Darstellungswerkzeug sichtbare Takt aus den *Layout*-Daten gelesen, durch Interpretation in eine Sammlung von unter Motif darstellbaren graphischen Elementen (`tGFXDrawables`) überführt und diese in Listenform dem Betrachtungswerkzeug zur Verfügung gestellt. Da die strikte Verfolgung dieser Politik bei jedem horizontalen Verschieben des sichtbaren Ausschnitts zu Zugrif-

fen auf die externen Dateien führt, wird momentan der umliegende Bereich bis zu 50[110] Takten ebenfalls in graphische Elemente umgewandelt, so daß sie bei Bedarf ohne Verzögerung ins Blickfeld gerückt werden können.

Prinzipiell ist es aber so, daß die graphische Partiturdarstellung direkt auf die externen *Humdrum*-Daten zugreift. Daher kann eine interne Datenstruktur entfallen und nur die Verweise auf die entsprechenden *Kern*-Dateien (die wiederum Verweise auf die dazugehörigen *Layout*-Dateien enthalten) werden den einzelnen Orchesterstimmen beigegeben. Wenn der Anwender die Partitur z.B. ab dem 73. Takt betrachten möchte, so prüft das Metamaterial zunächst, ob es die Anfrage aus den intern bereits aufbereiteten graphischen Partiturdaten befriedigen kann. Falls ja, dann werden die in der Liste von graphischen Elementen enthaltenen Partitursymbole anforderungsgemäß verschoben und dem Betrachtungswerkzeug zur Verfügung gestellt. Anderenfalls werden für alle Orchesterstimmen die *Kern*-Dateien ermittelt und aus ihnen die dazugehörigen *Layout*-Dateinamen (mit dem *Humdrum*-Werkzeug **yank**) extrahiert. Aus diesen wird wiederum (mit **yank**) der gewünschte Ausschnitt isoliert und dann Zeichen für Zeichen in eine interne graphische Repräsentation umgewandelt. Diese wird dann schließlich in Listenform dem Betrachtungswerkzeug zugänglich gemacht.

[110] Diese Zahl ist nicht fix und wird nach weiteren Versuchen und unter Berücksichtigung der jeweiligen Prozessorleistung weiter erhöht oder vermindert werden. Ähnlich wie bei Datenbanken geht es darum, das optimale „Working Set“ zu bestimmen [ST73], auf das man möglichst oft zurückgreifen will, ohne auf externen Datenangewiesen zu sein.

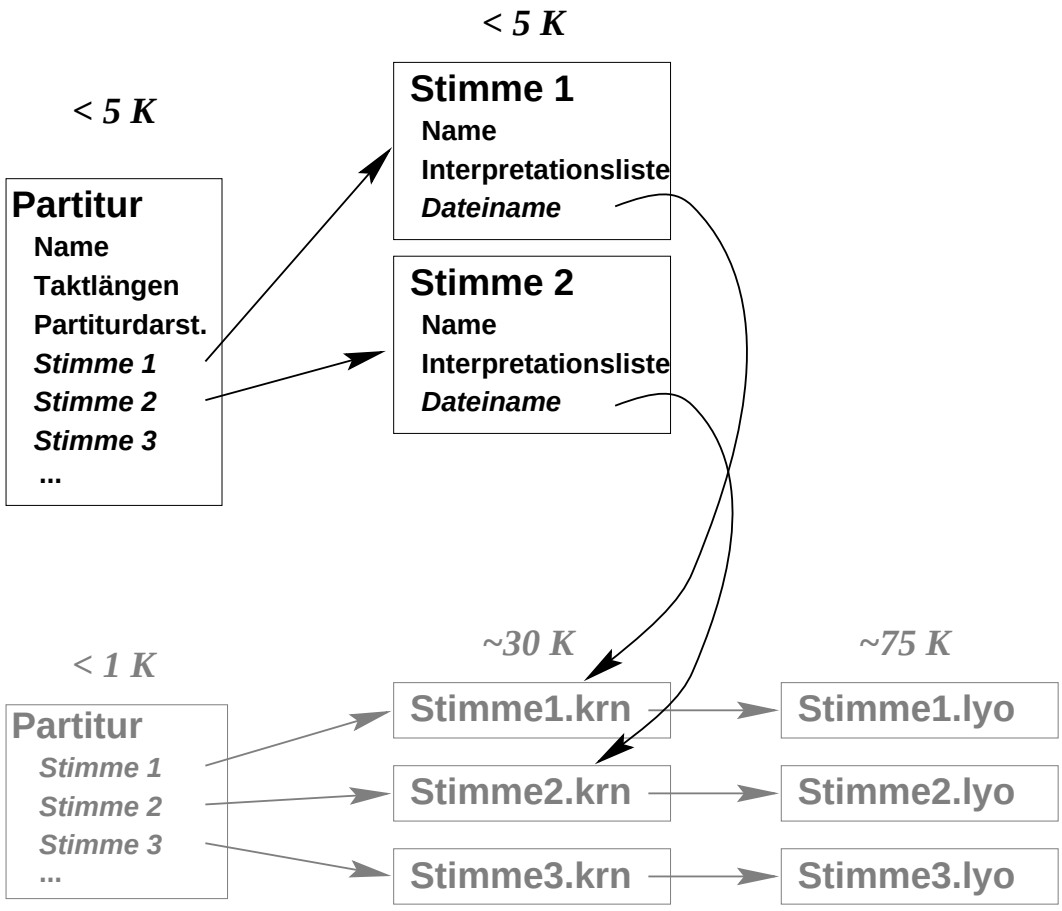


Abb. 4.2-1 Die Datenstruktur des Metamaterials Partitur.

Die Massendaten werden zur Laufzeit nicht intern gespeichert sondern nur als Verweise auf die ihnen zugrundeliegenden externen Dateien geführt. Lediglich Metadaten (graphische Taktlängen, graphische Partiturdarstellung und Kontextinterpretationslisten) werden direkt im Hauptspeicher gehalten.

Die Partitur enthält neben ihrem Namen und Verweisen auf die in ihr notierten Orchesterstimmen nur ein Aufstellung aller Taktlängen und eine graphische Repräsentation des aktuellen Ausschnitts in Form einer Liste graphischer Elemente. Die Orchesterstimmen enthalten nur ihren Namen als direktes Datum. Die sie eigentlich ausmachenden musikalischen Informationen sind nur indirekt über den Dateinamen der jeweiligen *Kern*-Datei angegeben. Mit der Interpretationsliste läßt sich schnell bestimmen, in welchem Kontext eine gegebene Note zu interpretieren ist.

Die Metamaterialien koppeln also mit minimalem Aufwand die fachlichen Materialien, als welche sie sich den anderen WAM-Systembestandteilen ja darstellen, an *Humdrum*-Daten. Mit Hilfe einiger geschickt ausgewählter Metainformationen über die *Humdrum*-Daten ist es zugleich möglich, die direkten Zugriffe auf ein Minimum zu beschränken. Metamaterialien scheinen daher ein geeignetes Mittel für eine schlanke fachliche Kapselung bestehender Datenbestände zu sein.

Werkzeuge

Den mächtigen fachlichen Werkzeugen mit ihren Operationen steht eine Vielzahl von kleinen *Humdrum*-Werkzeugen gegenüber, mit denen alle Analyseanforderungen erfüllt werden können, so daß die Aufgabe der fachlichen Kapselung hier in der geeigneten Kombination der vorhandenen *Humdrum*-Werkzeuge besteht.

Die Operationen der Fachwerkzeuge zerfallen dabei in zwei Gruppen. Die eine Gruppe führt direkt zum Einsatz von *Humdrum*-Werkzeugen gemäß dem aktuellen Werkzeugzustand, die andere beeinflußt lediglich den Werkzeugzustand.[111]

Ein **direkter** Zugriff auf *Humdrum*-Werkzeuge erfolgt nur an zwei Stellen:

- Beim Abspielen der Partitur vom Partituranalysewerkzeug aus wird eine *Kern*-Repräsentation des Materials mit **`midi | smf`** in eine Form umgewandelt, die für einen internen oder externen Synthesizer verständlich ist.
- Bei der Beendung des Leitmotivmarkierungsvorgangs werden die ausgewählten Partiturabschnitte mit **`yank`** extrahiert und mit **`assemble`** zu einer vorläufigen Leitmotivnotizdatei zusammengefügt.

Die **indirekten** Zugriffe bewirken lediglich eine Änderung der Parametrisierung der *Humdrum*-Werkzeuge bei einem späteren direkten Zugriff.

- Bei der Leitmotivmarkierung werden die markierten Bereiche für jede Leitmotivkomponente zwischengespeichert.
- Bei der Erstellung einer Leitmotivnotiz enthalten die TextIATs die Informationen, die später der Leitmotivnotizdatei hinzugefügt werden.

Die Kapselung von *Humdrum* durch fachliche Werkzeuge und Materialien erfolgt bei den Leitmotivanalysewerkzeugen also über direkt und indirekt auf *Humdrum* aufbauende Werkzeuge und über Metamaterialien. Darüberhinaus sind keine weiteren Mechanismen vonnöten. Das so entstandene System stellt dem Anwender eine fachlich verständliche Analyseumgebung zur Verfügung und kommt gegenüber *Humdrum* mit einem minimalen zusätzlichen Verwaltungsaufwand aus.

Der Nachteil der von mir gewählten Lösung besteht in der Dezentralität der Zugriffe auf das *Humdrum*-Subsystem. Während dies zwar ein gutes Laufzeitverhalten durch eine minimale Anzahl von zusätzlichen Architekturebenen (mit den dementsprechenden zusätzlichen Methodenaufrufen) bedeutet, so könnte dies bei einem größeren Werkzeugsammlung zu unübersichtlichen Lösungen führen. In diesem Fall wäre ein zentraler Automat, der alle Zugriffe auf das *Humdrum*-System kapselt, die günstigere Lösung.

[111] Die Standardoperationen eines Werkzeugs (Material laden, speichern, anzeigen, Subwerkzeug öffnen oder schließen) werden in der FIAK-Bibliothek beschrieben bzw. sind schon im Abschnitt über die Materialien behandelt worden.

Da jedoch die Anzahl der direkten Zugriffe bisher sehr gering ist und aufgrund der großen Datenmengen ein schneller Zugriff einen kritischen Faktor bei der Benutzbarkeit darstellt, scheint mir die implementierte Lösung in diesem speziellen Fall weiterhin gut vertretbar zu sein.

Kapselung bei allgemeinen musikwissenschaftlichen Analysewerkzeugen

Nachdem sich die fachliche Kapselung von *Humdrum* bei den Leitmotivanalysewerkzeugen als erfolgreich herausgestellt hat, stellt sich die Frage, ob die angewendeten Verfahren auch auf andere musikwissenschaftliche Analyseaufgaben ausdehnbar sind. Die zwei denkbaren Alternativen, weitere Spezialwerkzeuge oder eine universelle *Humdrum*-Kapselung, werden im diesem Abschnitt spekulativ erörtert.

Die Anzahl der denkbaren **Spezialwerkzeuge** hängt alleine vom Einfallsreichtum der Musikwissenschaftler ab und hat daher als unbegrenzt zu gelten. Da all diese Möglichkeiten aus ersichtlichen Gründen nicht in dieser Arbeit untersucht werden können, sollen zunächst die Gemeinsamkeiten dieser möglichen Spezialwerkzeuge hervorgehoben und darauf aufbauend die Verallgemeinerbarkeit der Leitmotivanalysewerkzeugarchitektur geprüft werden.

Bereits aus der Bezeichnung *Spezial*werkzeug ist ersichtlich, das nicht die gesamte Palette der musikwissenschaftlichen Analysemethoden (und damit auch der *Humdrum*-Werkzeuge) benötigt wird. Es kann also ein szenarioartig umreißbarer Arbeitsvorgang ermittelt werden, bei dem sich der Anwender einer endlichen Anzahl von Werkzeugen und Materialien bedient. Da nicht von einer informatischen Schulung des Anwenders ausgegangen werden kann, und somit eine direkte Arbeit mit textuellen *Humdrum*-Werkzeugen ausscheidet, kann auch auf eine den fachlichen Ansprüchen genügende Visualisierung des Materials nicht verzichtet werden. Bei diesen Materialien kann es sich um alles handeln, was durch *Humdrum*-Datenstrukturen ausgedrückt werden kann: z.B. vollständige Partituren[112], abstrakte Themen- oder Frequenzmuster.

Diese Charakterisierung ist der der Leitmotivanalysewerkzeuge sehr ähnlich. Daher scheint eine Ausdehnung der dort verwendeten Architekturprinzipien (Kopplung über Metamaterialien und direkten *Humdrum*-Werkzeugzugriff) auf Spezialwerkzeuge auch möglich zu sein, wenn es gelingt, eine geeignete Visualisierung der jeweiligen Materialien zu implementieren. Da eine Implementation der sehr komplexen Notensatzinformation jedoch schon in dieser Arbeit realisiert wurde, müßten andere, weniger komplexe Arten musikalischen Materials eigentlich auch mit vertretbarem Aufwand darstellbar sein. Da ebenfalls nur wenige Werkzeuge verwendet werden, kann wahrscheinlich auch in diesen Fällen auf einen zentralen *Humdrum*-Automaten verzichtet werden.

Ein **Humdrum-Universalwerkzeug**, das die gesamte *Humdrum*-Funktionalität zur Verfügung stellt, kommt nur für solche Musikwissenschaftler in Frage, die bereits zuvor mit *Humdrum* gearbeitet haben und mit seiner

[112] Notensatzdaten können mit dem zusätzlichen *Layout*-Format dargestellt werden.

UNIX-artigen Arbeitsweise vertraut sind. Dies dürfte nur auf einen Bruchteil der musikwissenschaftlichen Gemeinde zutreffen. Für diesen Anwenderkreis stellt folglich nicht die mangelnde fachliche Kapselung das Problem dar, sondern die fehlende Visualisierung der untersuchten Materialien. Zu diesem Zweck müßte ein Universalwerkzeug sämtliche *Humdrum*-Datenformate geeignet graphisch darstellen und beliebig (d.h. mit allen zur Verfügung stehenden Werkzeugen) manipulieren können. Das bedeutet, daß im Gegensatz zu einem Spezialwerkzeug eine Beschränkung auf einige wenige *Humdrum*-Werkzeuge nicht in Frage käme.

Ein solches Universalwerkzeug könnte als eine Art Werkzeugkasten dargestellt werden, aus dem der Anwender die benötigten *Humdrum*-Werkzeuge entnehmen und zu den gewünschten Befehlsketten kombinieren kann. Das End- oder auch bereits die Zwischenergebnisse könnten dabei vorschauartig angezeigt werden. Außerdem könnten sinnlose Werkzeugkombinationen auf Wunsch von vornherein ausgeschlossen werden, indem die ungeeigneten Werkzeuge für ein Ankopplung an die bestehende Befehlskette einfach nicht zur Verfügung gestellt werden.

Bis auf die zentrale Materialdarstellung hätte ein solches Universalwerkzeug wenig mit Spezialwerkzeugen wie den Leitmotivanalysewerkzeugen gemein. Da auf sehr vielfältige Weise auf *Humdrum*-Werkzeuge und -Daten zugegriffen wird, käme der architektonischen Klarheit wegen für die Implementation wahrscheinlich nur ein zentraler *Humdrum*-Automat in Frage. Vermutlich ergäben sich daraus auch keine größeren Laufzeiteinbußen, sofern nicht überwiegend mit den komplexen Notensatzdaten gearbeitet wird.

Eine allgemeine Kapselung von *Humdrum* zur Durchführung musikwissenschaftlicher Analysen scheint vor allem dann auf die architektonischen Prinzipien der Leitmotivanalysewerkzeuge zurückgreifen zu können, wenn es sich um eine spezialisierte Untersuchungen handelt. In diesen Fällen könnte basierend auf den Ergebnissen dieser Arbeit eine Klassenbibliothek erstellt werden, aus der weitere Spezialwerkzeuge durch Anpassung und Erweiterung der Leitmotivanalysewerkzeuge erzeugt werden können. Wenn diese neuen Werkzeuge wieder in die Bibliothek eingingen, könnten dann in Zukunft Spezialanwendungen relativ zügig aus den bereits bestehenden Teilstücken zusammengestellt werden. Auf diese Weise könnten die Vorzüge *Humdrum*s von nahezu jedem Musikwissenschaftler genutzt werden, ohne ihn mit der fachfremden, UNIX-artigen Benutzungsschnittstelle zu konfrontieren.

Falls diese Schnittstelle aber doch vollständig genutzt werden soll, dann sollte das entsprechende *Humdrum*-Universalwerkzeug mit einem *Humdrum*-Automaten versehen werden, der alle Zugriffe zentral kapselt. Für den informatisch ungeübten Musikwissenschaftler ändert ein solches Werkzeug jedoch nichts an der Ausgangssituation, daß die abstrakte *Humdrum*-Schnittstelle zwischen ihm und der Durchführung seines gewohnten Arbeitsvorgangs steht.

Kapselung von allgemeinen, nicht WAM-gemäßen Subsystemen

In den letzten beiden Abschnitten wurde, ausgehend von speziellen musikwissenschaftlichen Fragestellungen und einer Ausrichtung auf *Humdrum*, gezeigt, daß es bei der fachlichen Kapselung von bestehenden Syste-

men Alternativen zum Einsatz von Automaten gibt. In diesem letzten Abschnitt werden Auswahlkriterien vorgestellt, anhand derer meiner Meinung nach entschieden werden sollte, ob Automaten oder aber Metamaterialien und direkte Subsystemzugrife eingesetzt werden sollten.

Es ist selbstverständlich, daß nicht alle Softwaresysteme in gleicher Weise dazu geeignet sind, mit Hilfe von WAM gekapselt zu werden. Ausgangspunkt muß stets ein ordnungsgemäßer Entwicklungsprozeß nach WAM mit den dazugehörigen fachlichen Dokumenten sein. Erst wenn sich dort konkrete Ansatzpunkte zur (Weiter-)Verwendung bestehender Systemteile ergeben, sollten die dementsprechenden technischen Dokumente auf eine Kapselung ausgerichtet werden. Es ist nicht sinnvoll, WAM lediglich als „technischen Steinbruch“ zu betrachten, wenn damit Systeme konserviert werden, die sich nicht an den fachlichen Bedürfnissen der Anwender orientieren.

Daher ist es auch klar, daß diese automatenlose Architekturalternative natürlich nur technische Automaten betrifft. *Fachliche* Automaten wie z.B. Taschenrechner oder Kopierer, die in fachlichen Entwicklungsdokumenten festgeschrieben sind, können selbstverständlich nicht ersetzt werden sein.

Die fachliche Kapselung eines bestehenden Softwaresystems mit Hilfe von **Automaten** liegt besonders dann nahe,

- wenn das Systeme eine hohe „Rüstzeit“ aufweist, d.h. für die Ausführung von Operationen aufwendig hoch- und heruntergefahren werden muß.
  Direktzugriffe würden hier zu unvertretbar hohen Ausführungszeiten führen.
  Ein solches System könnte z.B. eine relationale Datenbank sein, die erst eine längere Initialisierungsphase durchläuft, bevor sie Zugriffe über eine prozedurale Schnittstelle erlaubt.
- wenn das Systeme eine komplexe und breit genutzte Schnittstelle besitzt.
  Direktzugriffe oder die Verwendung von Metamaterialien wären hier zu heterogen und zu sehr über den WAM-Teil des Systems verstreut, als das noch von einer übersichtlichen Kopplung gesprochen werden könnte.
- wenn an das System keine hohen Durchsatzanforderungen gestellt werden.
  Da die Verwendung von Automaten eine weitere interne Schnittstelle bedeutet, führen sie zu geringen Laufzeiterhöhungen, die aber unkritisch sind, wenn die mit dem System ausgetauschten Datenmengen nicht besonders groß sind oder nicht unbedingt absolut verzögerungsfrei verfügbar sein müssen.
  Bei sporadischen Datenbankabfragen fällt z.B. eine halbe Sekunde zusätzlicher Antwortzeit nicht ins Gewicht.

Dementgegen bietet sich die Kapselung mit **Metamaterialien und Direktzugriffen** an,

- wenn das System eine sehr kurze „Rüstzeit“ hat, d.h. Operationen nahezu verzögerungsfrei umgesetzt werden können.
  Bei diesen Eigenschaften müßte kein aufwendiger Automat permanent im Hintergrund laufen, um zügige Zugriffe auf das System zu erlauben.

Ein solches System könnte eine Client/Server-Datenbank sein, bei der alle Befehle über kleine, parametrisierbare Clients abgewickelt werden können. Auch UNIX-artige Systeme aus Kleinstwerkzeugen kommen in Frage.

- wenn nur sehr wenige Funktionen des Systems tatsächlich benötigt werden.
  Ein allgemeiner, zentraler Automat mit breiter Schnittstelle ist überflüssig, wenn nur von sehr wenigen Stellen des WAM-Teils aus eine Handvoll von Operationen genutzt werden. In diesem Fall ist eine „direkte Verdrahtung“ der betreffenden Komponenten auf WAM- und Subsystem-Seite ökonomischer und führt aufgrund der geringen Nutzungsbreite im WAM-Teil noch nicht zu Unübersichtlichkeiten.
  Wenn z.B. stets nur der freie Platz in einem Verzeichnis oder auf einem ganzen Laufwerk überprüft werden soll, dann stellt die allgemeine Kapselung der dazu notwendigen Betriebssystemoperationen einen unverhältnismäßig hohen Aufwand dar, wenn es z.B. bei UNIX mit einem `shell("du -s" );` auch getan ist.

- wenn an das Systeme hohe Durchsatz- und Geschwindigkeitsanforderungen gestellt werden.
  Bei zeitkritischen Anwendungen sollte jede zusätzliche interne Schnittstelle oder Transformation vermieden werden. Metamaterialien und Direktzugriff sparen gegenüber Automaten eine interne Schnittstelle ein.
  Beispiele hierfür gibt es in der Prozeßdatenverarbeitung und bei Datenbanken mit großen, nicht-textuellen Datenbeständen.

Anhand dieser Kriterien wird ersichtlich, daß *Humdrum* zwar ein gutes, aber nicht das einzige Einsatzfeld für die automatenlose Architekturalternative darstellt. Die Metamaterialien und der Direktzugriff sind auch für eine ganze Reihe von speziellen Anwendungssystemen geeignet, für die ein Automat entweder unnötig aufwendig oder zeitlich nicht vertretbar ist.

Zusammenfassung

In diesem Kapitel wurden die technischen Aspekte der Leitmotivanalysewerkzeuge und der dabei verwendeten Kapselung *Humdrum*s behandelt. Zunächst wurden die Insularität der bisherigen musikalischen Datenformate und die fachlich ungenügenden Benutzungsschnittstellen der Analysewerkzeuge als Hauptprobleme der bestehenden Rechnerunterstützungen musikwissenschaftlichen Analyseumgebungen identifiziert und dann schrittweise gelöst.

In einem ersten Schritt wurde das Programmpaket *scr2hmd* entwickelt, mit dessen Hilfe der Analyse vormals unzugängliche Notensatzdaten zugeführt werden können.

Auf dieser Grundlage wurde in einem zweiten Schritt das für reine Musikwissenschaftler nahezu unverständliche Analysesystem *Humdrum* fachlich so gekapselt, daß es nun von Wagner-Experten wie selbstverständlich genutzt werden kann. Die technische Kopplung der WAM-Kapsel und des *Humdrum*-Kerns erfolgte dabei nicht nach dem Muster des technischen Automaten sondern mit Metamaterialien und Werkzeugdirektzugriff. Diese alternative aber WAM-konforme Architektur erwies sich als sehr ökonomisch und ist meiner Meinung nach

nicht nur für die Kapselung von *Humdrum* sondern auch für andere bestehende Systeme mit ähnlichen Merkmalen (wenige kleine schnell ausführbare Operationen mit hohen Durchsatzanforderungen) geeignet.

Des weiteren wurde festgestellt, das mit *SCORE*, *scr2hmd*, *Humdrum* und WAM sowohl weitere musikwissenschaftliche Spezialwerkzeuge als auch ein *Humdrum*universalwerkzeug erstellt werden könnten, so daß die These

> ***Mit den in dieser Arbeit verwendeten Werkzeugen und Methoden ist eine Umgebung für allgemeine musikwissenschaftliche Analysen in greifbare Nähe gerückt***

schließlich bestätigt werden kann.

5 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Erstellung eines Leitmotivanalysewerkzeugs für *„Der Ring des Nibelungen“*, um eine Grundlage für den Vergleich der architektonischen Prinzipien Wagners, Christopher Alexanders und WAMs zu schaffen. Dieses Ziel wurde bis zur Verwirklichung eines ersten Handhabungsprototyps erreicht.

Auf dem Wege dorthin haben sich für die rechnergestützte musikwissenschaftliche Analyse und für WAM zahlreiche Erkenntnisse und Verbesserungen ergeben.

Für WAM stellte sich heraus, daß die Spannweite der durch sie unterstützbaren Tätigkeiten von hochqualifizierter musikwissenschaftlicher Analyse bis zu niedrig qualifizierter Entgegenahme telefonischer Bestellungen reicht. Dabei wurde auch klar, daß WAM unter Anwendung der eigenen Prinzipien auf sich selbst leicht an ein fremdes Anwendungsgebiet angepaßt werden kann. Demzufolge wurden die bisherigen Grenzen WAMs („Methode für qualifizierte und kooperative menschliche Tätigkeit“) auf den angemessenen Raum „adaptierbarer Methodenkern für selbstbestimmte menschliche Tätigkeit“ erweitert.

Auf technischem Gebiet wurde ein bestehendes Subsystem (die UNIX-werkzeugartige Analyseumgebung *Humdrum*) erstmals nicht mit Hilfe eines Automaten sondern mit Metamaterialien und Werkzeugdirektzugriffen effektiv und schlank durch ein WAM-System fachlich gekapselt. Diese alternative Kopplungsmethode scheint sich besonders dann zu empfehlen, wenn nur von wenigen Stellen aus aber dafür schnell und umfangreich auf ein Subsystem zugegriffen werden soll.

Besonders deutlich trat das Fehlen einer den gesamten WAM-Entwicklungsprozeß begleitenden Entwicklungsumgebung hervor. Zu Beginn dieser Arbeit mußten alle Dokumente unübersichtlich auf Papier erstellt und dann bei Beginn der ersten Implementationen komplett neu erstellt werden. Zu diesem Zweck wurde ein vollständiges Hypertextdokumentationssystem auf Basis von HTML-Dokumenten mit Dokument- und Formulareditoren, Betrachtern und Programmierumgebung entworfen und teilweise realisiert. Mit dem vollendeten System könnten Entwickler in ihrer Arbeit vom ersten Glossareintrag bis zum lauffähigen System unterstützt werden.

Für die musikwissenschaftliche Analyse entstanden die prototypischen Leitmotivanalysewerkzeuge. Durch ihre Weiterentwicklung werden vollständige Leitmotivgesamtanalysen bald wesentlich zeitsparender erstellt werden können. Aus der gelungenen fachlichen Kapselung *Humdrum*s durch WAM-Methoden konnte außerdem gefolgert werden, daß dieses bisher für nicht UNIX-bewanderte Musikwissenschaftler unbenutzbare mächtige Analysesystem in Zukunft auch diesem Personenkreis zugänglich gemacht werden kann. Mit *scr2hmd* habe ich schließlich ein Programm geschaffen, mit dem höchstwertige Notensatzdaten in analysefähige Analysedaten konvertiert werden können und somit der Mangel an brauchbaren Analysematerial beträchtlich gelindert wird.

Nun ist der Weg für die Weiterentwicklung der prototypischen Leitmotivanalysewerkzeuge zum arbeitsfähigen Pilotsystem geebnet. Mit ihm wird vielleicht bald, über 100 Jahre nach der Komposition von *„Tristan und Isolde“*, *„Die Meistersinger von Nürnberg“*, *„Der Ring des Nibelungen“* und *„Parsifal“*, das Rätsel um die Form bei Wagner gelüftet werden können.

Ob und inwiefern diese Form mit den Entwurfsprinzipien Alexanders und WAMs korreliert, erfüllt mich mit Spannung.

Literaturverzeichnis

[Ale64] Christopher Alexander, *Notes on the Synthesis of Form* (Cambridge, Massachusetts: Harvard University Press, 1964)

[Ale79] Christopher Alexander, *The Timeless Way of Building* (New York: University Press, 1979)

[AIS79] Christopher Alexander, Sara Ishikawa und Murray Silverstein, *A Pattern Language* (New York: University Press, 1979)

[Bau88] Hans-Joachim Bauer, *Richard-Wagner-Lexikon* (Bergisch Gladbach: Lübbe, 1988)

[Bro91] F.A. Brockhaus, *Brockhaus-Enzyklopädie in 24 Bänden*, 18. Auflage (Mannheim: Brockhaus, 1991)

[BZ90] Reinhard Budde und Heinz Züllighoven, *Software-Werkzeuge in einer Programmierwerkstatt*, (München: Oldenbourg, 1990)

[Büc91] Georg Büchmann, *Geflügelte Worte*, 38. Auflage (Frankfurt: Ullstein, 1991)

[Byr94] Donald Byrd, „Music Notation Software and Intelligence“ in: *Computer Music Journal*, 18. Jahrgang, 1. Heft (Frühjahr 1994), S. 17-20

[Coo68] Deryck Cooke, *Introduction to „Der Ring des Nibelungen“* (London: Decca 1968)

[Dal70] Carl Dalhaus, „Gibt es ein Geheimnis der Form bei Richard Wagner?“ in: Carl Dalhaus, *Das Drama Richard Wagners als musikalisches Kunstwerk* (Regensburg: Bosse, 1970), S. 9-16

[Dal86a] Carl Dalhaus, „Die Musik“ in: Ulrich Müller und Peter Wapnewski, *Richard-Wagner-Handbuch* (Stuttgart: Kröner, 1986), S. 197-221

[Dal86b] Carl Dalhaus, „Wagners Stellung in der Musikgeschichte“ in: Ulrich Müller und Peter Wapnewski, *Richard-Wagner-Handbuch* (Stuttgart: Kröner, 1986), S. 60-85

[Dud90] Dudenredaktion, *Duden. Das Fremdwörterbuch*, 5. Auflage (Mannheim: Dudenverlag, 1990)

[Flo83] Constantin Floros, „Der ‘Beziehungszauber’ der Musik im ‘Ring des Nibelungen’ von Richard Wagner“, in: *Neue Zeitschrift für Musik*, 144. Jahrgang, 7./8. Heft (Juli/August 1983), S. 8-14

[Fus95] Hans-Ulrich Fuss, *Zur jüngeren Entwicklung der musikalischen Analyse in den USA* (zur Veröffentlichung vorgesehen)

[Gra82] Hermann Grabner, *Allgemeine Musiklehre*, 14. Auflage (Kassel: Bärenreiter, 1982)

[Gra96] Cindy Grande, *NIFF 6a. Notation Interchange File Format* (**`ftp://blackbox.cartah.washington.edu`**)

[Gry95] Guido Gryczan, *Situierte Koordination computergestützter qualifizierter Tätigkeit über Prozeßmuster*, Dissertation (Hamburg: 1995)

[GKZ93] Guido Gryczan, Klaus Kilberth und Heinz Züllighoven, *Objektorientierte Anwendungsentwicklung* (Braunschweig: Vieweg, 1993)

[GZ95] Guido Gryczan und Heinz Züllighoven, *Die Gebos-Methode* (zur Veröffentlichung vorgesehen)

[HMSW93] Mitch Harris, Eduardo Miranda, Alan Smaill, Geraint Wiggins, „A Framework for the Evaluation of Music Representation Systems“ in: *Computer Music Journal*, 17. Jahrgang, 3.Heft (Herbst 1993), S. 31-42

[Hol90] William Holab, *Using SCORE* (Half Moon Bay, California: Passport Design, 1990)

[Hur94] David B. Huron, *The Humdrum Toolkit Reference Manual* (Menlo Park, California: CCARH, 1994)

[LdB93] Lektorat des BI-Wissenschaftverlags, *Duden. Informatik*, 2. Auflage (Mannheim: Dudenverlag 1993)

[Lor24] Alfred Lorenz, *Das Geheimnis der Form bei Richard Wagner, 1. Band: Der musikalische Aufbau des Bühnenfestspiels „Der Ring des Nibelungen“* (Berlin: Max Hesse, 1924)

[Man63] Thomas Mann, *Richard Wagner und unsere Zeit* (Frankfurt: S. Fischer, 1963)

[New96] Steven Newcomb, „Reality Checking of SMDL“ in: Stephen Page, *Music-Research Digest*, 10. Jahrgang, 2. Ausgabe (**`ftp://cattell.psych.upenn.edu/pub/Music.Research/10.2`**)

[Pah87] Kurt Pahlen, *Oper der Welt*, 4. Auflage (Bindlach: Gondrom, 1987)

[Rag95] Dave Raggett, HyperText Markup Language Specification Version 3.0 (**`http://www.w3.org/hypertext/WWW/MarkUp/html3/CoverPage.html`**)

[Ram95] F. Ramm, *Recherchieren und Publizieren im World Wide Web* (Braunschweig: Vieweg, 1995)

[RW92] William Renwick und David Walker, „CD-Brahms: An Interactive Multimedia Program in Music Analysis“ in: *Computers in Music Research*, 4. Heft (Herbst 1992), S. 45-76

[Rie86] Jörg Riedlbauer, „Wagners ‘Leitmotive’ im ‘Ring des Nibelungen’ und ihre operndramaturgische Tradition“ in: *Österreichische Musikzeitschrift*, 41. Jahrgang, 12. Heft (Dezember 1986), S. 629-633

[Rie93a] Dirk Riehle, *Dokumentation zur FIAK v1.1 Bibliothek* (Hamburg: 1993)

[Rie93b] Dirk Riehle, *Dokumentation zur IATMotif v1.0 Bibliothek* (Hamburg: 1993)

[RZ95] Dirk Riehle und Heinz Züllighoven, „A Pattern Language for Tool Construction and Integration Based on the Tools and Materials Metaphor“ in: James O. Coplien und Douglas C. Schmidt, *Pattern Languages of Program Design* (Reading, Massachusetts: Addison-Wesley, 1995), S. 9-42

[Smo95] Stephen Smoliar, „Reality Checking of SMDL“ in: Stephen Page, *Music-Research Digest*, 9. Jahrgang, 36. Ausgabe (**`ftp://cattell.psych.upenn.edu/pub/Music.Research/9.36`**)

[Smo96] Stephen Smoliar, „information about SMDL“ in: Stephen Page, *Music-Research Digest*, 10. Jahrgang, 2. Ausgabe (**`ftp://cattell.psych.upenn.edu/pub/Music.Research/10.2`**)

[Slo93] Donald Sloan, „Aspects of Music Representation in HyTime/SMDL“ in: *Computer Music Journal*, 17. Jahrgang, 4. Heft (Winter 1993), S. 51-59

[ST73] D. R. Slutz und I. L. Traiger, *A Note on the Calculation on of Average Working Set Size* (Yorktown Heights: IBM Research Laboratory, 1973)

[Smi94] Leland Smith, *The SCORE Music Printing System* (Menlo Park, California: 1994)

[Tak95] TakeFive Software, *Sniff+2.1 User's Guide and Reference* (Salzburg: 1995)

[Wag1] Richard Wagner, *Das Rheingold* (Mainz: B. Schott's Söhne, o.J.)

[Wag2] Richard Wagner, *Die Walküre* (Mainz: B. Schott's Söhne, o.J.)

[Wag3] Richard Wagner, *Siegfried* (Mainz: B. Schott's Söhne, o.J.)

[Wag4] Richard Wagner, *Götterdämmerung* (Mainz: B. Schott's Söhne, o.J.)

[Wag83] Richard Wagner, „Oper und Drama" in: Dieter Borchmeyer, *Richard Wagner. Dichtungen und Schriften*, Band 7 (Frankfurt: Insel, 1983)

[Wei94] Ulfert Weiss, *Spezifikation und Implementation eines Interaktionstypen zur graphischen Darstellung von Dokumenten und Dokumentbeziehungen als Erweiterung einer objektorientierten Bibliothek unter Verwendung des Fenstersystems Motif*, Studienarbeit (Hamburg: 1994)

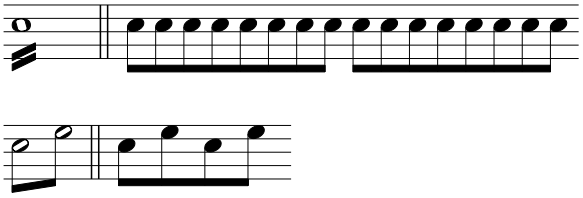
[Wil95] Jan Willamowius, *Ein Browser für die objektorientierte Programmiersprache Eiffel 3*, Studienarbeit (Hamburg: 1995)

Anhänge

A Glossar musikwissenschaftlicher Begriffe

Abbreviatur. Abkürzung in der Notenschrift aus Gründen der Raumersparnis oder besseren Übersicht.

1. Tonwiederholungen. Über längere Zeit zu wiederholende Noten oder zweinotige Folgen. Notiert mit unverbundenen (eine Note) bzw. verbundenen (zweinotige Folge) →Querbalken, deren Anzahl den Wert der einzelnen Noten angibt.



Bsp.: eine Note (oben), zweinotige Folge (unten)

2. Oktavzeichen zur Vermeidung vieler Hilfslinien. Bis zum Ende der Linie werden alle Noten um sieben →diatonische Tonstufen höher (8va) oder niedriger (8va bassa) gespielt.

Bsp.: 8va

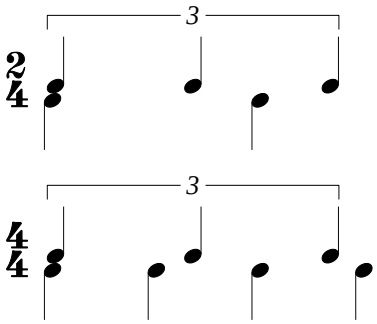
3. →Verziehrungen

Akkolade. Eigentlich Klammer, die zusammengehörige →Systeme in einer →Partitur verbindet. Auch Bezeichnung für diese Systeme als Ganzes.

Akkord. Gleichzeitiger geordneter Zusammenklang von mindestens drei Tönen verschiedener Tonhöhe. Zum Bereich der →Harmonik gehörend.

Akt, bei Wagner: **Aufzug.** Hauptabschnitt einer Oper. Weitere Unterteilung in →Szenen möglich.

Anomale Teilung. Im Bereich der anomalen Teilung werden Töne einer bestimmten Dauer gleichmäßig gedehnt oder verkürzt. Die Anzahl der Noten bestimmt den Namen der anomalen Teilung (Duole, Triole, Quintole, etc.) und wird an einer Klammer oder am Querbalken notiert.



Bsp.: verkürzende Triole (3x1/6 statt 2x1/4; oben), verlängerende Triole (3x1/3 statt 4x1/4; unten)

Anschlag. Art und Weise der Tongebung bei Tasteninstrumenten, bestimmt durch die Bewegung der Finger.

Artikulation. Verbindungs- oder Trennungsarten von Tönen, die aufgrund bestimmter Spielweisen miteinander verschmolzen (legato) oder aber durch winzige Pausen voneinander getrennt werden (staccato, portato, marcato). Durch →Bindebögen oder Artikulationszeichen direkt über oder unter den Noten notiert.

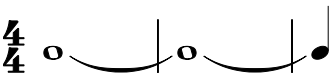
Bsp.: legato, staccato, portato, marcato

Auszug. Die volle Werklänge abdeckende Teilpartitur, in der nur eine oder einige wenige Orchesterstimmen notiert sind. Dient als Notenblattsammlung für die jeweiligen Musiker.

Bindebogen. Zwei oder mehr Noten umfassender Bogenstrich in der Partitur.

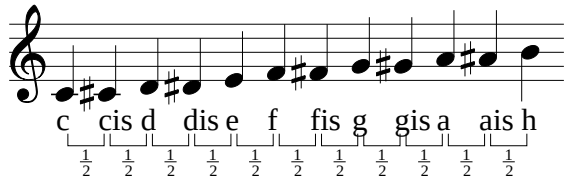
1. zwischen Noten verschiedener Höhe (Legatobogen): Töne sollen gebunden, d.h. ohne abzusetzen gespielt werden (→Artikulation).

2. zwischen Noten gleicher Höhe (Haltebogen): Zusammenfassung von Einzelnoten zu einem Ton, dessen Länge der Summe der der Einzeltonlängen entspricht.



Bsp.: Haltebogen für einen Ton mit 9/4-Dauer

Chromatische Tonleiter. Tonleiter vom Umfang einer →diatonischen Tonleiter, die aber im Gegensatz zu dieser ausschließlich in gleichgroßen Halbtonschritte voranschreitet (+½ +½ +½ +½ +½ +½ +½ +½ +½ +½ +½ (+½).



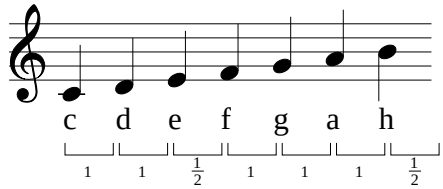
Bsp: Chromatische Tonleiter ab C

Diatonische Tonleiter. Tonleiter, die in Ganz- und Halbtonschritten von c bis h schreitet: c (Grundton) d e f g a h (c, Grundton der nächsthöheren Oktave).

Dur-Tonleiter: Die Abstände zwischen den Tönen sind vom Grundton ausgehend +1 +1 +½ +1 +1 +1 +½ (1=Ganzton, ½=→Halbton).

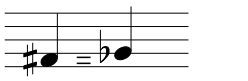
Moll-Tonleiter: Die Abstände zwischen den Tönen sind vom Grundton ausgehend +1 +½ +1 +1 +1 +1 +½.

Die Tonleitern werden nach ihren Grundtönen benannt (C-Dur, a-Moll).

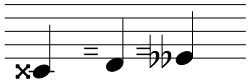


Bsp.: C-Dur-Tonleiter

Enharmonische Töne. Verschieden notierte Töne, die aber gleich klingen.



fis = ges



cisis= d = eses

Enharmonische Verwechslung. Vertauschung →enharmonischer Töne in der Notenschrift zur logischen Weiterführung von Harmonie oder Melodie.

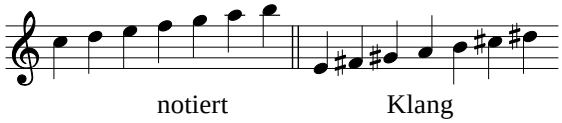
Halbton. Die kleinste Tonstufe des →diatonischen Tonsystems. Bei →diatonischen Dur-Tonleitern sind nur die Abstände zwischen dem 3. und 4. sowie dem 7. und 8. Ton Halbtonschritte. Bei →chromatischen Tonleitern sind alle Abstände Halbtonschritte. Halbtöne, die nicht einer diatonischen Tonleiter angehören, werden durch →Versetzungszeichen kenntlich gemacht.

Ein Ganztonschritt entspricht zwei Halbtonschritten.

Harmonik. Die Lehre von der Bedeutung des gleichzeitigen Zusammenklangs von Tönen verschiedener Höhe, die tonartlich zueinander in Beziehung stehen, und deren Verbindung. Neben →Melodik und →Rhythmik ein Hauptelement der Musik.

Instrument. Gerät zum Hervorbringen musikalisch brauchbarer Töne. In der musikalischen Praxis untergliedert in Streichinstrumente, Blasinstrumente und Schlaginstrumente.

Transponierende (Blas-)Instrumente werden in der Partitur anders notiert, als sie klingen, um dem Musiker das Spielen zu erleichtern oder um Hilfslinien zu sparen. Transponierende Instrumente der ersten Gruppe sind Englischhorn, Klarinette, Trompete und Waldhorn. Zur zweiten Gruppe gehören Piccoloflöte (klingt 1 Oktave höher als notiert), Kontrafagott und Kontrabaß (je 1 Oktave tiefer).



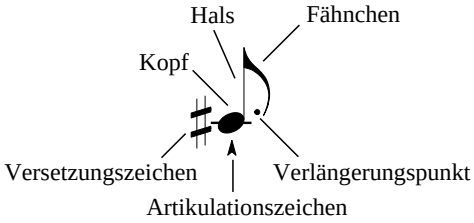
Bsp.: C-Dur-Tonleiter für Horn in E

Kontrapunkt. Kompositionstechnik, bei der zu einem Thema neue melodisch selbständige Stimmen erfunden und nach strengen Regeln fortgeführt werden.

Leitmotivtechnik. Das Verfahren, ein musikalisches Drama (→Oper) durch ein dichtes Gewebe leitmotivischer Beziehungen von innen heraus zusammenzuhalten.

Melodik. Die Lehre von der Beschaffenheit von Tonfolgen, die als selbständige charakteristische Gebilde auftreten. Neben →Harmonik und →Rhythmik ein Hauptelement der Musik.

Note. Zeichen zur schriftlichen Festlegung von →Tönen. Gibt die →Tondauer durch ihre Gestalt und die →Tonhöhe durch die Stellung des Notenkopfs im →System in Verbindung mit →Notenschlüssel und →Versetzungszeichen an. Besteht aus Notenkopf, Notenhals, und Fähnchen. Um eine Note herum können →Artikulationszeichen, →Versetzungszeichen und →Verlängerungspunkte angeordnet sein, die den Notenwert beeinflussen.



Notendauer. Die Notengrunddauern sind (wie bei →Pausen) die Potenzen von ½ von 0 aufwärts, also 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, etc. Die Notengrunddauer wird durch eine Kombination von Notenkopf, Notenhals und Fähnchen ausgedrückt:

Ganze Noten bestehen nur aus einen unausgefüllten Kopf,

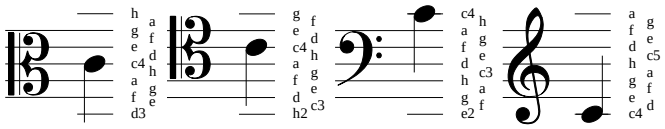
Halbe Noten besitzen zusätzlich einen Hals,

Viertel Noten sind wie halbe Noten, jedoch mit ausgefülltem Kopf,

Achtel, Sechszehntel-, Zweiundreissigstelnoten, etc. besitzen zusätzlich zu Viertelnoten ein Fähnchen je weitere Halbierung der Notendauer (Achtel: 1, Sechszehntel: 2, Zweiunddreisigstel: 3, etc.). Von den Grunddauern abweichende Tondauern können durch →Verlängerungspunkte oder →anomale Teilung erreicht werden.

Bsp.: Notengrunddauern von 1/1 bis 1/64

Notenschlüssel. Stilisierte Buchstaben, die je nach ihrer vertikalen Anordnung die Zuordnung von →diatonischen Tonwerten zu den Linien eines →Systems festlegen. Werden verwendet, um bei der Notation mit möglichst wenig Hilfslinien auszukommen.



C-Schlüssel als Altschl. *C-Schlüssel als Tenorschl.* *F-Schlüssel als Baßschl.* *G-Schlüssel als Violinschl.*

Oper. Musikalisches Bühnenwerk mit Darstellung einer Handlung durch Gesang und Instrumentalmusik. Je nach Operntyp steht das zugrundeliegende Drama, die Musik oder die Verschmelzung der beiden (Musikdrama) im Vordergrund. Besteht meistens aus →Ouvertüre und einem oder mehreren ggf. in →Szenen unterteilten →Akten.

Ouvertüre. Instrumentale Einleitung einer →Oper

Partitur. Notenschriftliche Aufzeichnung ein- oder mehrstimmiger Musik in der die einzelnen →Stimmen innerhalb von →Akkoladen so übereinander angeordnet sind, daß der Verlauf der Einzelstimmen, ihre Koordination und die Zusammenhänge abgelesen werden können.

Pause. Zeichen zur schriftlichen Festlegung der Dauer des Schweigens einer Orchesterstimme. Die Pausengrunddauern sind (wie bei →Noten) die Potenzen von ½ von 0 aufwärts, also 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, etc. und werden durch die Form des Pausenzeichens ausgedrückt.

Von den Grunddauern abweichende Pausendauern können durch →Verlängerungspunkte erreicht werden.

Bsp.: Pausengrunddauern von 1/1 bis 1/64

Phrase. Melodische Sinneinheit. Kann vom Komponisten unvollkommen durch einen →Bindebogen bezeichnet werden. Wegen Überlagerungen oft nicht eindeutig identifizierbar.

Probennummer. Zeitliche Markierung in einer →Partitur, an der das Einsetzen zu Probenzwecken sinnvoll erscheint.

Querbalken. Ein dicker Strich, der bei einer Gruppe von Achtel- oder kürzeren →Noten die Notenhälse miteinander verbindet. Die Anzahl der Querbalken entspricht der Anzahl der (nun nicht mehr gezeichneten) Fähnchen. Dient der Übersichtlichkeit, der Phrasierung (→Phrase) bei Singstimmen oder (bei ganzen, halben und Viertelnoten) als →Abbreviatur.

Rhythmik. Lehre von der Ordnung und Gliederung einer Tonfolge nach Zeitdauer und Gewicht der Einzeltöne. Neben →Melodik und →Harmonik ein Hauptelement der Musik.

Satz. In sich selbständiger Teil eines Instrumentalwerks, z.B. einer →Symphonie, Sonate oder Suite.

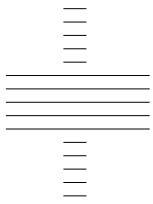
Stimme. Teil einer →Partitur, die ein Musiker (sowohl Sänger als auch Instrumentalist) auszuführen hat. Bei Streichern werden auch Gruppen von gleichartigen Instrumenten zu einer Stimme zusammengefaßt. Durch Voranstellung der Anweisung „divisi“ kann verlangt werden, daß Streicherstimmen zeitweise als getrennte Stimmen spielen sollen.

Einzeln oder zusammen mit einer anderen Stimme in einem →System notiert.

Symphonie. Orchesterwerk, bestehend aus einer Folge von meist drei oder vier →Sätzen in Sonatenform.

System. Gruppe von fünf übereinander angeordneten parallelen, waagerechten Linien, auf und zwischen denen die Noten eingetragen werden. Falls die fünf Linien zur Notation von besonders hohen oder tiefen Noten nicht ausreicht, werden kurze Hilfslinien oberhalb oder unterhalb des Systems hinzugefügt. Die Zuordnung von Tonhöhen zu bestimmten Linien und Zwischenräumen wird durch →Notenschlüssel und →Tonartversetzungszeichen festgelegt.

Eine oder zwei →Stimmen können in einem System notiert werden.



Bsp.: leeres System mit Hilfslinien

Szene. Unterabschnitt eines →Aktes einer →Oper, die durch das Auf- und Abtreten einer oder mehrerer Personen begrenzt ist.

Takt. Musikalisches Maß- und Bezugssystem, das die Betonungsabstufung und zeitliche Ordnung der Töne regelt. Die Taktart wird am Beginn eines Stückes durch einen Bruch angegeben.

Bsp.: Betonung im 4/4-Takt. Die Hauptbetonung liegt auf dem 1., die Nebenbetonung auf dem 3. Ton

Taktstrich. Senkrechter Strich durch ein →System oder eine ganze →Akkolade, der bei der nachfolgenden Note einen Schwerpunkt anzeigt und der die →Takte abteilt.

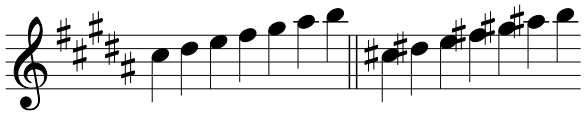
Tempo. Die Geschwindigkeit eines Musikstücks. Kann entweder mathematisch (88 Viertelnoten pro Minute) oder menschlich (largo, andante, allegro) angegeben werden. Erst durch eine Tempoangabe können die absoluten Zeitwerte der relativ spezifizierten →Notendauern bestimmt werden.

Ton. Die elementare Einheit musikalischen Materials. Hat (im Gegensatz zu einem Geräusch) eine eindeutig bestimmbare →Tonhöhe und →Tondauer.

Tonart. Die Zugehörigkeit eines Stücks zu einer Tonart geschieht durch Untersuchung der darin verwendeten Tonhöhen. Je nachdem, wie die Töne voranschreiten, gehört das Stück zu einer Dur- oder Molltonart (→diatonische Tonleiter).

Tonartversetzungszeichen. Direkt hinter dem →Notenschlüssel notierte →Versetzungszeichen, die für alle Noten bis zu den nächsten Tonartver-

setzungszeichen gelten und somit individuelle Versetzungszeichen vor den Noten überflüssig machen. Charakteristisch für →Tonarten.



Bsp.: H-Dur Tonleiter mit und ohne Tonartversetzungszeichen

Tondauer. Für jeden →Ton wird in der Notenschrift normalerweise genau eine →Note gesetzt, aus deren Gestalt seine Dauer hervorgeht. Dauert der Ton länger als einen →Takt, so werden mehrere Noten identischer Höhe durch →Bindebögen verbunden.

Die absolute Tondauer wird je nach →Tempo der Musik unterschiedlich in Noten umgesetzt. Ein Ton von 1 Sekunde Länge entspricht bei einem Tempo von 60 Viertelnoten pro Minute einer Viertelnote, während er bei einem Tempo von 80 Viertelnoten pro Minute einer punktierten Viertelnote (→Verlängerungspunkt) entspricht.

Tonhöhe. Durch die Frequenz festgelegte Eigenschaft eines →Tons. Je größer die Frequenz ist, desto höher wird der Ton wahrgenommen. Aus der unendlichen Anzahl möglicher Töne werden nur Töne, die in einem bestimmten Frequenzverhältnis zueinander stehen, in der Musik verwendet und mit Tonbuchstaben (C, D, E, F, G, A, H, C) bezeichnet. Die kleinste verwendete →Tonstufe ist der →Halbton. Aus der Festlegung des Tons A auf 440 kHz können alle anderen Tonhöhen absolut bestimmt werden.

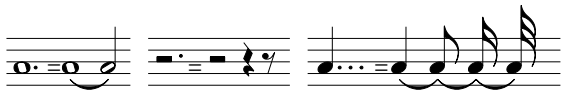
Die Tonhöhe wird in der Notenschrift durch die vertikale Anordnung einer →Note im →System ausgedrückt.

Tonleiter. Eine aufsteigende Folge von Tönen, die in einem einem festen Tonhöhenverhältnis (→Tonstufe) zueinander stehen: →chromatische und →diatonische Tonleiter.

Tonstufe. Tonhöhendistanz von einem Ton einer →Tonleiter zum nächsten. Kann entweder einen →Halbton oder einen Ganzton (= 2 Halbtöne) betragen.

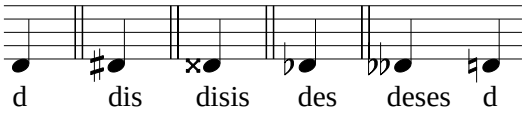
Transponierung. Das Verfahren, ein Musikstück unter Beibehaltung der ihm eigenen →Tonstufendistanzen in eine →Tonart mit einem anderen →Grundton (→diatonische Tonleiter) versetzen. Siehe auch transponierende →Instrumente.

Verlängerungspunkt. Zeichen zur Verlängerung der Dauer von →Noten und →Pausen um jeweils die Hälfte ihres bisherigen Wertes. Weitere Punkte verlängern um jeweils die Hälfte des vorangegangenen Punktes.



Bsp.: 3/2 =1/1+1/2 *7/8 =1/2+1/4+1/8* *15/32 =1/4+1/8+1/16+1/32*

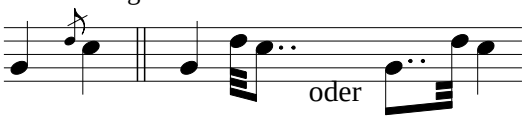
Versetzungszeichen. Zeichen zur Erhöhung oder Erniedrigung von Tönen um einen oder zwei Halbtönen, bzw. deren Aufhebung. Gelten bis zum Ende des →Taktes, in dem sie notiert sind, für alle Noten gleicher Höhe. Auch direkt hinter dem →Notenschlüssel als →Tonartversetzungszeichen notiert.



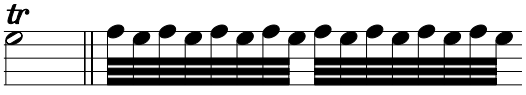
Bsp.: Die Modifikationsmöglichkeiten durch Voranstellen von Versetzungszeichen

Verziehrung. Ausschmückende zusätzliche Tonfigur, die nicht zur melodischen Substanz gehört. Meist durch besondere Zeichen oder kleine Noten angedeutet.

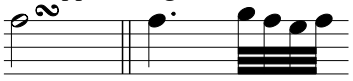
1. Vorschlag



2. Triller



3. Doppelschlag



Zwischenspiel. Verbindene Instrumentalmusik zwischen zwei →Akten einer →Oper.

B Entwicklungsdokumente

Alle textuellen Entwicklungsdokumente wurden in HTML abgefaßt und können am besten direkt unter **`http://swt1.informatik.uni-hamburg.de/~1kornsta/Entwicklungsdokumente/Uebersicht.html`** (Stand April 1996) eingesehen werden. Die hier abgedruckten Dokumente sollten lediglich als Ersatz dienen, falls gerade kein WebBrowser zur Hand ist.

Um die Orientierung zu Erleichtern, habe ich in den Übersichtsdokumenten alle Verweise um ihre Seitenzahl in diesem Anhang ergänzt.

Übersicht über die Entwicklungsdokumente

Glossar

Stand 2.5.95, revidiert 18.5.95

- **Akkolade**
  Eckige Klammer am linken Rand von Systemen einer Partiturseite, die angibt daß die in den umfaßten Systemen notierten Stimmen zur gleichen Zeit spielen. Teilakkoladen fassen Stimmengruppen innerhalb einer Akkolade zusammen.
- **Harmonische Komponente**
  Bestandteil von harmonisierten Leitmotiven.
  - Legt die Harmonisierung eines Leitmotivs fest, indem sie die melodische Komponente derart ergänzt, daß das Leitmotiv einer bestimmten Tonart angehört oder nicht angehört.
  - Wird von allen Instrumenten gespielt, die nicht die melodische Komponente und nicht die rhythmische Komponente spielen.
  - *Bsp.: erstes Walhall-Leitmotiv in WA II 4, Stimmen: 2.-8. Horn*

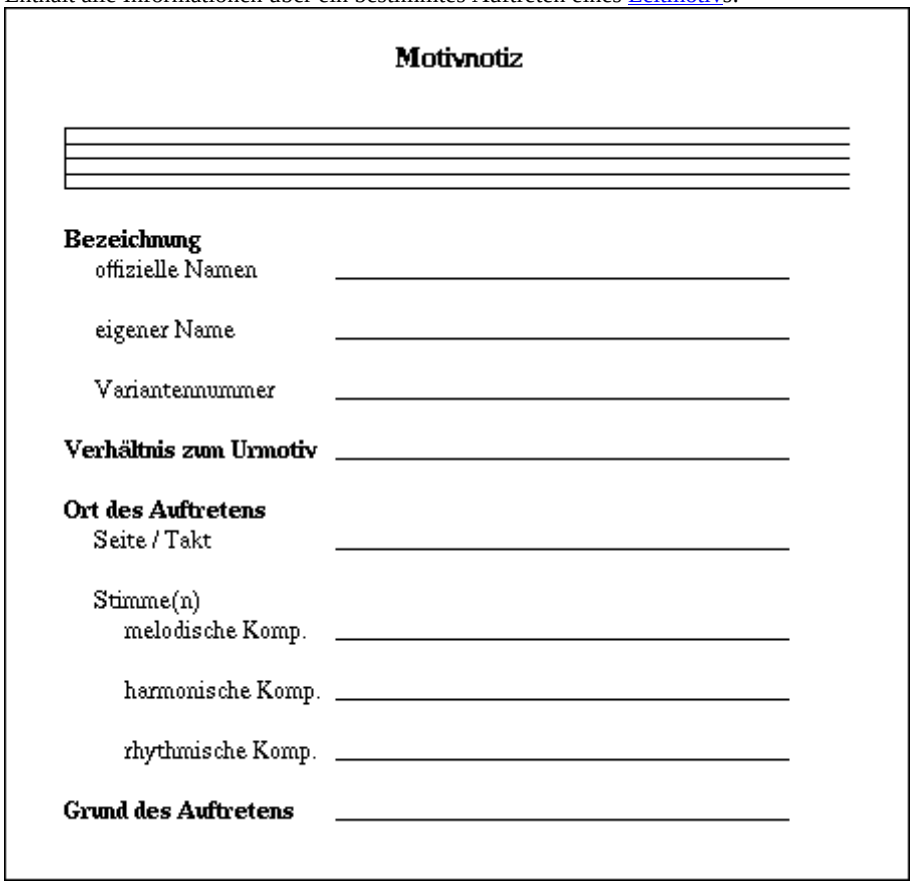
- **Leitmotiv**
  melodisch-semantische Einheit im Wagnerschen Sinne (d.h. zugeordnete Bedeutung und Wiedererkennbarkeit)
  - Struktur:
    - besteht aus melodischer Komponente und in manchen Fällen einer harmonischen Komponente und/oder einer rhythmischen Komponente. *Bsp:*
      - *Leitmotiv nur mit melodischer Komponente: erstes Natur-Leitmotiv **a** in RG Vorspiel*
      - *Leitmotiv mit harmonischer Komponente: erstes Walhall-Leitmotiv in WA II 4*
      - *Leitmotiv mit harmonischer Komponente und rhythmischer Komponente: erstes Schicksals-Leitmotiv in WA II 4*

    - melodisch (im klassischen Sinne = Folge von Tönen bestimmter Höhe) fest. Melodische Variation nur durch Transposition möglich.
    - harmonisch anfänglich fest (z.B. C-Dur). Kann im Kontext des Dramas verändert werden. Die Harmonisierung wird bestimmt durch die melodische Komponente und die harmonische Komponente.
    - rhythmisch fest. Rhythmische Variation nur durch gleichmäßige Augmentation oder Diminuation *aller* Töne möglich. Nur der erste und der letzte Ton kann in seiner Dauer leicht von diesem Muster abweichen, ohne daß es sich dabei um eine neue Variante handelt. *Bsp. hierfür: Natur-Leitmotiv im RG-Vorspiel, Takt 17+21 und in Takt 25.*
      *Bsp: Natur-Leitmotiv **a** ist anderes Leitmotiv als Natur-Leitmotiv **b** (keine Variation), da anderer Rhythmus.*
    - Leitmotive von größerer Länge ("Gesangsmotive") umfassen gesamte Gesangslänge bis zur ersten Wiederholung. *Bsp.: Rheintöchtergesang, Winterstürme-Leitmotiv, Wotans Scheidegruß)*

  - Bedeutung:
    - Leitmotiv ohne initiale Bedeutungszuweisung durch dramatisches Geschehen verkörpert "Ahnung" (im Wagnerschen Sinne), die beim ersten Erklingen im Kontext mit dramatischer Handlung (umfaßt auch Bühnenbild) ("Vergegenwärtigung") ausgedeutet wird. *Bsp.: Natur-Leitmotiv, Schicksals-Leitmotiv, Todesklage-Leitmotiv.*
    - Leitmotiv mit initialer Bedeutungszuweisung, das aber im späteren Verlauf nicht wieder aufgegriffen wird, ist lediglich eine Motiv.
    - Leitmotive können im späteren Verlauf ("Erinnerung") auch fragmentarisch auftreten.

- **Leitmotivbetrachter**
  Softwarewerkzeug zum Betrachten einer Leitmotivtabelle. Näheres in der Systemvision Leitmotivbetrachter.
- **Leitmotivmarkierer**
  Softwarewerkzeug zum Markieren eines Leitmotivs in der Partitur. Näheres in der Systemvision Leitmotivmarkiererer.

- **Leitmotivnotiz**
  Enthält alle Informationen über ein bestimmtes Auftreten eines Leitmotivs.



  - Leitmotivexzerpt
    - Vollständiges Leitmotiv (d.h. melodische Komponente und ggf. harmonische Komponente und/oder rhythmische Komponente) im Realklang (Nicht identisch mit Notation in der Partitur bei transponierenden Instrumenten.)
    - Abbreviaturen nicht ausschreiben. *Z.B. Vorschläge.*
    - im optimalen Schlüssel (d.h. mit der minimalen Anzahl von Hilfslinien)
  - Leitmotivbezeichnung
    - Namen
      - "offizielle" Namen gemäß existierenden Leitmotivtabellen. *Z.B. Natur-Leitmotiv a bei Windsperger.*
      - eigener Name. *Z.B. Gewässer-Leitmotiv.*
    - Variantennummer gemäß Chronologie des ersten Auftretens (als römische Zahl)
  - Verhältnis zum Urmotiv
  - Ort des Auftretens
    - Seitennummer der Partitur, besser: Taktnummer
    - Stimmen, in denen das Leitmotiv erklingt getrennt für melodische Komponente und ggf. harmonische Komponente und/oder rhythmische Komponente. Schreibweise: ***Stimme, in der Leitmotivbestandteil beginnt- Stimme, in der Leitmotivbestandteil endet.***
  - (vermuteter) Grund für Auftreten
    - dramatische Gründe
    - musikalische Gründe
    - ...

- **Leitmotivnotizeditor**
  Softwarewerkzeug zum Erstellen einer Leitmotivnotiz. Näheres in der Systemvision Leitmotivnotizeditor.
- **Leitmotivtabelle**
  NOCH ZU ERSTELLEN
  u.a.:
- Tabellarische Aufstellung aller in der Partitur vorkommenden Leitmotive und Leitmotivvarianten. Jeder Eintrag gibt Tonart, Rhythmus und Tonfolge sowie den Leitmotivnamen samt Variantenbezeichnung an.
  - bestehende Leitmotivtabellen (v. Wolzogen, Windsperger, übrige Sekundärliteratur)

        - unvollständig bezüglich Leitmotivvarianten
        - Notation gemäß Klavierauszug
        - Einträge oft länger als notwendig

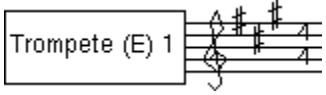
- **Melodische Komponente**
  Kernbestandteil eines Leitmotivs.
    - Stellt die charakteristische Melodie (= Tonfolge) eines Leitmotivs dar.
    - Bei einem harmonisierten Leitmotiv wird die melodische Komponente meistens von dem Instrument gespielt, das die höchsten Töne spielt. Eine Ausnahmen kann z.B. bei Leitmotiven auftreten, die auch Gesangsstimmen umfassen. Die anderen Instrumente spielen die harmonische Komponente. *Bsp.: erstes Walhall-Leitmotiv in WA II 4, 1. Horn*
    - Tritt im Laufe des Leitmotivs eine Stimme hinzu, die noch höher ist, als die bisher höchste Stimme, so wechselt die melodische Komponente auf die hinzugetretene Stimme über. Die Stimme, die vormals die melodische Komponente spielte, trägt nun auch zur harmonische Komponente bei.
      Bsp.: Naturmotiv in Takt 53 (S.5) des RG-Vorspiels. Wechselt in Takt 55 von Fagott 1 auf Flöte 3

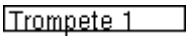
- **Motiv**
  melodische Einheit mit Bedeutungszuweisung, die im späteren Verlauf der Handlung jedoch nicht wieder aufgegriffen wird.
  Vermutete Bsp.: Grane in Götterdämmerung, Vorspiel, 2.Szene; Alberich in Rheingold, 1.Szene.
- **Partitur**
  Notenschriftliche Aufzeichnung ein- oder mehrstimmiger Musik, in der die einzelnen Stimmen innerhalb von Akkoladen so übereinander angeordnet und mit senkrecht durchlaufenden oder unterbrochenen Taktstrichen verbunden sind, daß der Verlauf der Einzelstimmen, ihre Koordination und die Zusammenhänge abgelesen werden können. Auf einer Partiturseite können mehrere Akkoladen notiert sein.
    - Die Stimmen innerhalb einer Akkolade sind ihrer Höhe nach von oben nach unten angeordnet; z.B. wie im Folgenden:
        - Holzbläser
            - Flöten
            - Oboen
            - Klarinetten
            - Fagotte
        - Blechbläser
            - Trompeten
            - Hörner
            - Posaunen
            - Tuben
        - Schlaginstrumente
            - Pauken
            - Triangel
            - Becken
            - ...
        - Harfen
        - Solostimmen
        - Chorstimmen
        - Streicher
            - Violinen
            - Bratschen
            - Violoncelli
            - Kontrabässe
        - Orgel
    - Zusammenfassung von Gruppen wird mittels durchgezogener Taktstriche und / oder Teilakkoladen verdeutlicht.

- **Partituranalysewerkzeug**
  Softwarewerkzeug zum Betrachten der Partitur sowie zum Suchen und Markieren von Leitmotiven. Näheres in der Systemvision Partituranalysewerkzeug.
- **Rhythmische Komponente**
  Bestandteil eines Leitmotivs, der rhythmisch selbständig bzw. losgelöst neben der melodischen Komponente und der harmonischen Komponente existiert. Bei einem rein rhythmischen Leitmotiv stellt die rhythmische Komponente den Kern des Leitmotivs dar.
    - Wird meist von einem Schlaginstrument gespielt.
    - *Bsp. für rhythmische Komponente: Paukenschläge zum Schicksals-Leitmotiv in WA II 4*

- *Bsp. für rhythmische Komponente als Leitmotivkern: Nibelungenmotiv in RG 2*

- **Stimme**
  Umfaßt Orchesterinstrumente und Singstimmen
- **Stimmenkopf**
  Graphischer Bestandteil einer im Partituranalysewerkzeug angezeigten Stimme, der Auskunft über die betreffende Stimme gibt und der die Ausführung stimmbezogener Operationen erlaubt.

  Eingeblendet: 

  Ausgeblendet: 

  - Gibt Auskunft über:
    - Name der Stimme
    - ggf. Instrumentennummer *Z.B. Fagott 1*
    - ggf. Transponierung der Stimme
    - für die folgenden Noten geltenden Notenschlüssel
    - für die folgenden Noten geltenden Versetzungszeichen
    - für die folgenden Noten geltende Metrik
  - Erlaubte Operationen
    - Doppelklick auf eingeblendeten Stimmenkopf führt zu Ausblendung
    - Doppelklick auf ausgeblendeten Stimmenkopf führt zu Einblendung
    - Einfacher Klick und Ziehen erlaubt horizontale Verschiebung der Stimme

Übersicht | Szenario-Übersicht | Systemvision-Übersicht

Szenarios

Typographische Konventionen in Szenarios

- Verweise auf andere Szenarios werden kursiv unterstrichen dargestellt (*Leitmotivanalyse*)
- Verweise auf Begriffe aus dem Glossar werden unterstrichen dargestellt (Leitmotiv)

Übersicht | Glossar | Systemvision–Übersicht

Szenario: Leitmotivanalyse

Stand 27.4.95, revidiert 18.5.95

Relevante Arbeitsgegenstände

- Partitur
- Leitmotivtabelle
- Leitmotivnotizzettel

Kontextbeschreibung

Die Musikwissenschaft untersucht unter anderem Inhalt und Form musikalischer Werke. Ein besonderes Interesse gilt dabei der Frage, inwieweit Inhalt und Form miteinander korrespondieren.

Richard Wagner verwendete erstmals - beginnend mit der Operntetralogie *Der Ring des Nibelungen* - eine Leitmotivtechnik, die u.a. Personen, Gegenständen und gedanklichen Konzepten der Handlung Leitmotive in der Musik zuordnet. Diese Leitmotive sind ein - wenn nicht sogar das wichtigste - strukturgebende Element des *Rings*.

Ziel der Leitmotivanalyse ist die vollständige Markierung aller Leitmotive in der Partitur sowie eine Aufstellung der Beziehungen der Leitmotive untereinander. Da die existierenden Leitmotivtabellen sehr unvollständig sind, wird hier zusätzlich zur sonst nur üblichen *Markierung* von Leitmotivem in der Partitur eine eigene Leitmotivtabelle auf der Grundlage von eigenen Leitmotivnotizen angelegt.

Beschreibung des Arbeitsvorganges *(unvollständig)*

Der Analysator *sucht* anhand von Leitmotivtabellen und eigenen Aufzeichnungen Leitmotive in der Partitur. Ein gefundenes Leitmotiv wird *markiert* und *ausführlich notiert*.

Andere Dokumente

- Szenarios
  - *Leitmotivsuche*
  - *Leitmotiv markieren*
  - *Leitmotivnotiz erstellen*
- Glossar

Szenario: Leitmotive suchen

Stand 27.4.95

Relevante Arbeitsgegenstände

- Partitur
- Leitmotivtabellen
- Leitmotivnotizzettel
- Aufnahme der in der Partitur notierten Oper und geeignetes Abspielgerät (wünschenswert)

Kontextbeschreibung

Im Rahmen der *Leitmotivanalyse* wird mithilfe von Leitmotivtabellen und eigenen Notizen die Partitur nach Leitmotiven durchsucht. Diese werden *markiert* und *ausführlich notiert*.

Beschreibung des Arbeitsvorganges

Wenn möglich hört sich der Anlaysator die zu untersuchende Stelle der Partitur an, indem er eine geeignete Aufnahme von ihr auf einem Wiedergabegerät abspielt, und liest während der Wiedergabe in der Partitur mit.

Daraufhin geht er in der Partitur zum Beginn der zu untersuchenden Stelle zurück. Zusätzlich nimmt er die Leitmotivtabellen und seine eigenen Leitmotivnotizen zur Hand. Er geht vom Anfang her beginnend durch die Partitur und sucht Leitmotive. Erkennen kann er ein Leitmotiv entweder anhand seiner Erinnerung an die Wiedergabe bzw. durch erneutes Hören der betreffenden Stelle oder anhand der Gestalt des Leitmotivs, die er von vorangegangenen Anlaysesitzungen her kennt.

Den Namen des Leitmotivs kann er dann durch Vergleich der Gestalten der Leitmotive einer der Leitmotivtabellen oder seinen eigenen Leitmotivnotizen entnehmen. Falls es sich um ein bisher undokumentiertes Leitmotiv handelt, so muß er selbst eine der dramatischen Handlung angemessene vorläufige Benennung vornehmen.

Wenn Name und Gestalt des Leitmotivs ermittelt sind (melodische Komponente und ggf. harmonische Komponente und/oder rhythmische Komponente) wird es *markiert*.

Zu jedem markierten Leitmotiv oder zu jeder markierten Gruppe gleichnamiger Leitmotive legt er eine Leitmotivnotiz an.

Andere Dokumente

- Szenarios
  - *Leitmotivanalyse*
  - *Leitmotiv markieren*
  - *Leitmotivnotiz erstellen*
- Glossar

Übersicht | Glossar | Szenario-Übersicht | Systemvision-Übersicht

Szenario: Leitmotiv notieren

Stand 27.4.95

Relevante Arbeitsgegenstände

- Partitur
- Leitmotivtabelle
- Leitmotivnotizzettel

Kontextbeschreibung

Ein bei der *Leitmotivsuche* gefundenes Leitmotiv oder eine Gruppe von Leitmotiven wird so ausführlich wie möglich auf einem Leitmotivnotizzettel beschrieben, um eine *vollständige* Leitmotivtabelle erstellen zu können. Anhand dieser Tabelle sollen dann Aussagen über die Beziehungen der Leitmotive zueinander gemacht werden können.

Solange die Leitmotivsuche nicht abgeschlossen ist, dienen die Leitmotivzettel als Ergänzung zur bestehenden Leitmotivtabelle.

Beschreibung des Arbeitsvorganges

Der Analysator nimmt einen leeren Leitmotivnotizzettel zur Hand und füllt ihn in beliebiger Reihenfolge aus, obgleich das Leitmotivexzerpt meist zuerst angefertigt wird:

- Das Leitmotivexzerpt verfertigt er so, daß das Leitmotiv in der gleichen Gestalt wie in der Partitur erscheint, (d.h. mit melodischer Komponente und ggf. harmonischer Komponente und/oder rhythmischer Komponente). Tonart, Rhythmus und Tonfolge müssen ablesbar sein. Bei transponierenden Instrumenten wird es gemäß seinem Klang notiert. Eventuell vorkommende Abbreviaturen (z.B. Vorschläge) werden beibehalten. Der Notenschlüssel wird so gewählt, daß nur eine minimale Anzahl von Hilfslinien verwendet werden muß.
- Er entnimmt den bestehenden Leitmotivtabellen die "offiziellen" Leitmotivnamen und notiert diese sowie ggf. einen eigenen Leitmotivnamen, sofern dieser eine semantisch besser passende Beschreibung darstellt.
- Er vergibt gemäß der Chronologie des Auftretens eine Variantennummer, die er als römische Zahl notiert.
- Er beschreibt das Verhältnis zur Urform des Leitmotivs, z.B. mit "verwandt mit" oder "fernere Verwandschaft mit".
- Er vermerkt die Position des Leitmotivs bezüglich der Zeit und der Stimme / den Stimmen, in denen das Leitmotiv erklingt. Die zeitliche Position gibt er falls möglich anhand der Taktnummer, sonst anhand der Partiturseitennummer an.
  Die Instrumente gibt er so exakt wie möglich getrennt für melodische Komponente und ggf. harmonischer Komponente und/oder rhythmische Komponente an (z.B. Fagott 1).
  Falls einer dieser Bestandteile das Instrument wechselt, so notiert er die Instrumente in chronologischer Reihenfolge und mit Bindestrichen getrennt.
- Er gibt den (vermuteten) Grund für das Auftreten des Leitmotivs an. Dabei kann es sich z.B. um einen dramaturgischen oder musikalischen Grund handeln.

Falls ein Leitmotiv in gleicher (d.h. mit höchstens am Anfang oder Ende in der Länge veränderten Tönen) Gestalt gruppiert vorkommt, kann auch die ganze Gruppe auf einem Leitmotivnotizzettel notiert werden. Dabei sollte die Art der Gruppierung angegeben werden. Ggf. abweichende Noten werden in Klammern dargestellt. Um eine Gruppe handelt es sich, wenn die Leitmotive sich überlappen oder unmittelbar aufeinander folgen.

Andere Dokumente

- Szenarios
  - *Leitmotivsuche*
- Glossar


Übersicht
Glossar
Szenario–Übersicht
Systemvision–Übersicht

Szenario: Leitmotiv markieren

Stand 27.4.95

Relevante Arbeitsgegenstände

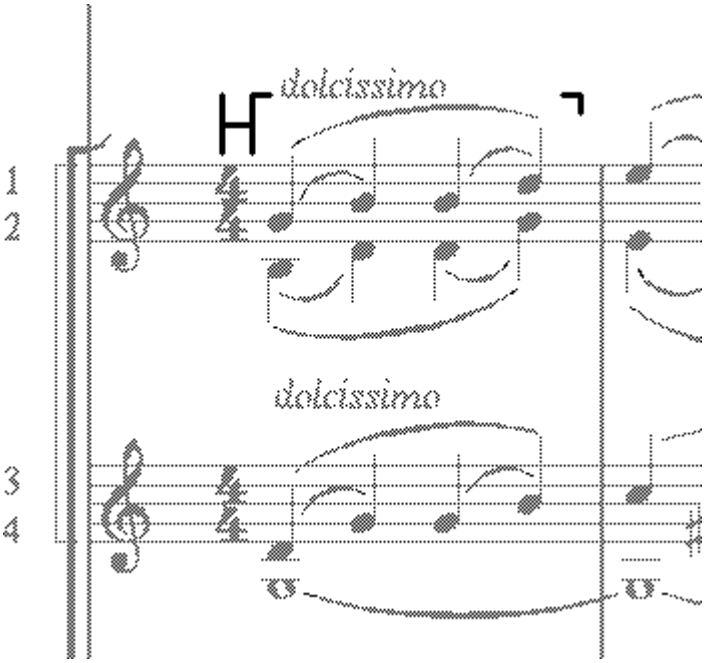
- Partitur
- Bleistift

Kontextbeschreibung

Ein bei der *Leitmotivsuche* gefundenes Leitmotiv wird in der Partitur markiert, um am Ende der *Leitmotivanalyse* die genaue Abfolge von Leitmotiven betrachten zu können.

Beschreibung des Arbeitsvorganges

Leitmotive werden gemäß den Konventionen der *zweiten Wiener Schule* z.B. mit einem Bleistift markiert.



Andere Dokumente

- Szenarios
  - *Leitmotivanalyse*
  - *Leitmotive suchen*
- Glossar

Übersicht | Glossar | Szenario-Übersicht | Systemvision-Übersicht

Systemvisionen

Typographische Konventionen in Systemvisionen

- Verweise auf Ablaufvisionen oder Szenarios werden kursiv unterstrichen dargestellt (*Leitmotivanalyse*)
- Verweise auf Begriffe aus dem Glossar oder Werkzeugvisionen werden unterstrichen dargestellt (Leitmotiv)

Übersicht | Glossar | Szenario-Übersicht

Systemvision: Leitmotivanalyse

Stand 18.5.95

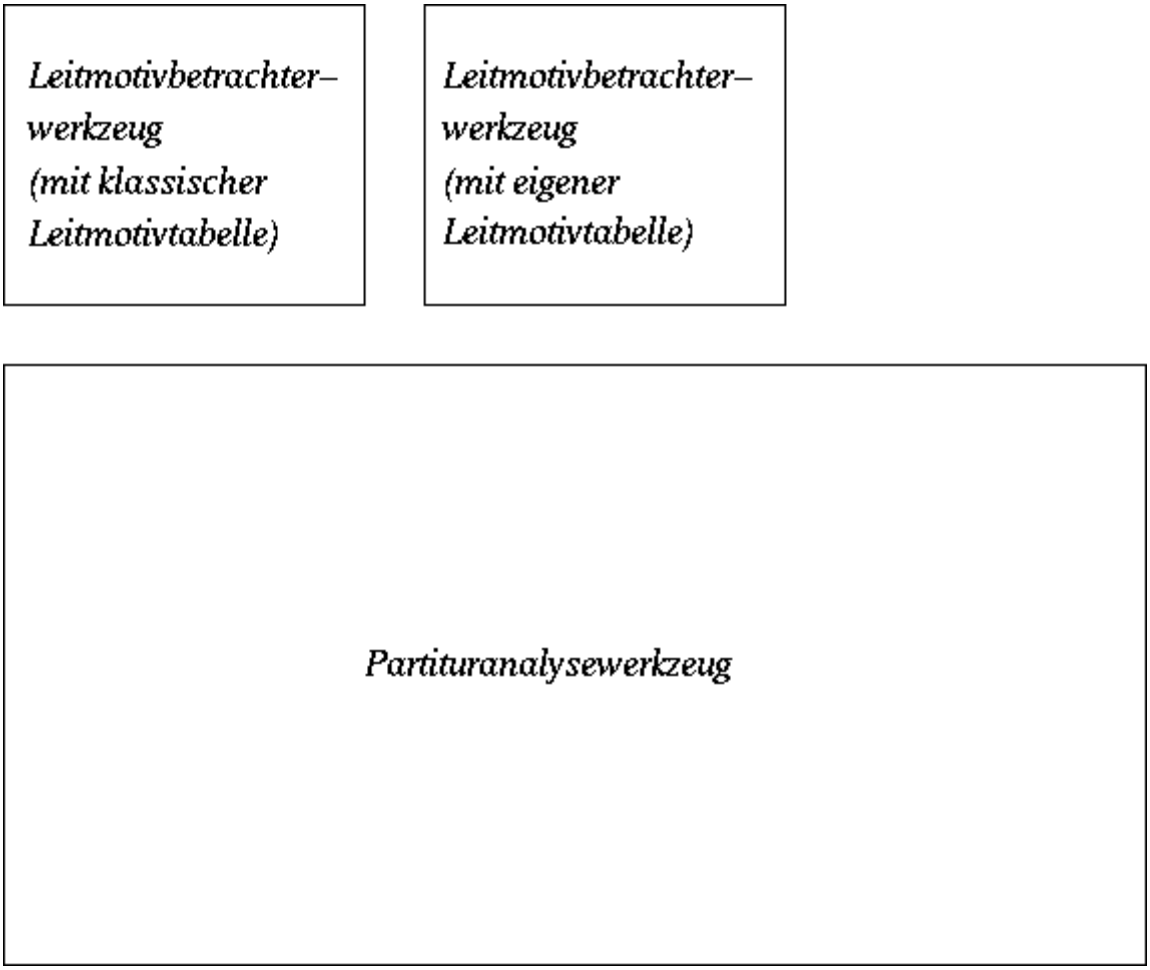
Zugrundeliegende Szenarios [mit Begründung]

- *Leitmotivanalyse*
    - Es wird mit den gleichen Materialen gearbeitet

Verwendete Operationen von Softwarewerkzeugen

- Partituranalysewerkzeug
    - Partitur auswählen
- Leitmotivbetrachter
    - Leitmotivtabelle auswählen

Bildschirmskizze



Kontextbeschreibung

Motivation

Die bisherigen Leitmotivanalysen decken aufgrund des sehr großen Umfangs des *Ring des Nibelungen* nur sehr wenige Szenen oder Szenenabschnitte der Tetralogie ab und untersuchen vornehmlich das "lokale" Verhältnis

von Inhalt (dramatischer Handlung) und Form (Leitmotivstruktur). Die Arbeit mit Partituren, Leitmotivtabellen und eigenen Leitmotivnotizzetteln ist aus folgenden Gründen sehr mühsam:

- Beim Vergleichen der Leitmotive ist eine hohe Genauigkeit notwendig
- Das Anlegen einer Leitmotivnotiz (insbesondere die Anfertigung des Leitmotivexzerpts) ist auf einem Leitmotivnotizzettel ist sehr zeitaufwendig
- Das Auffinden einer Leitmotivnotiz ist bei einer mehr als eine Szene umfassenden Analyse mit hohem Zeitaufwand verbunden

Aus diesen Gründen unterblieben bisher sehr zum Bedauern der musikwissenschaftlichen Gemeinde größere Leitmotivanalysen, die szenenübergreifende Aspekte behandeln wie z.B.:

- Leitmotivstruktur im Großen
- Leitmotiv und Instrumentation im Zeitverlauf
- Beziehung der Leitmotive untereinander

Um in Zukunft solche Analysen durchführen zu können, soll das in diesen Systemvisionen beschriebene Softwaresystem erstellt werden. Es soll dem analysierenden Musikwissenschaftler die zeitaufwendigen Aufgaben beim Anlegen und Wiederauffinden von Leitmotivnotizen soweit wie möglich abnehmen und ihm so eine Konzentration auf die eigentliche Analysetätigkeit erlauben. Durch diese Entlastung sollte dann auch das Untersuchen der Beziehung der Leitmotive untereinander möglich werden. Dies würde sich dann in zusätzlichen Szenarios (z.B. *Leitmotivbeziehungen ermitteln*) und dementsprechenden Systemvisionen niederschlagen.

Überblick über das zukünftige Softwaresystem

Die Materialien der bisherigen Leitmotivanalyse (Partituren, Leitmotivtabellen, eigene Leitmotivsammlung) sollen dabei auf korrespondierende Software-Materialien gleichen Namens abgebildet werden, wobei die Leitmotivnotizensammlung als Leitmotivtabelle im Entstehen angesehen wird.

Die Bearbeitung dieser Software-Materialien soll mit Software-Werkzeugen erfolgen, die in etwa den Teilhandlungen der bisherigen Leitmotivanalyse (Partitur betrachten, Leitmotiv markieren, Leitmotiv notieren, Leitmotiv in der Partitur mit Leitmotivnotiz vergleichen -> Partiturbetrachter, Leitmotivmarkierer, Leitmotivnotizeditor, Leitmotivbetrachter) entsprechen.

Das Zusammenspiel dieser Software-Werkzeuge im Rahmen typischer Arbeitssituationen (wie sie in den Szenarios beschrieben sind) wir im Folgenden in Ablaufvisionen geschildert, ohne dabei die Arbeitsweise der Software-Werkzeuge im Detail zu beschreiben. Letzteres geschieht in den zu diesem Zweck angelegten Werkzeugvisionen.

Der zukünftige Arbeitsablauf zum Erstellen einer Leitmotivanalyse samt Erstellung einer eigenen Leitmotivtabelle kann also wie folgt dargestellt werden:

Zusammenfassung

Der Analysator durchsucht also mithilfe des Partituranalysewerkzeugs und Leitmotivbetrachtern die Partitur nach Leitmotiven. Diese werden dann *markiert* und *ausführlich notiert*.

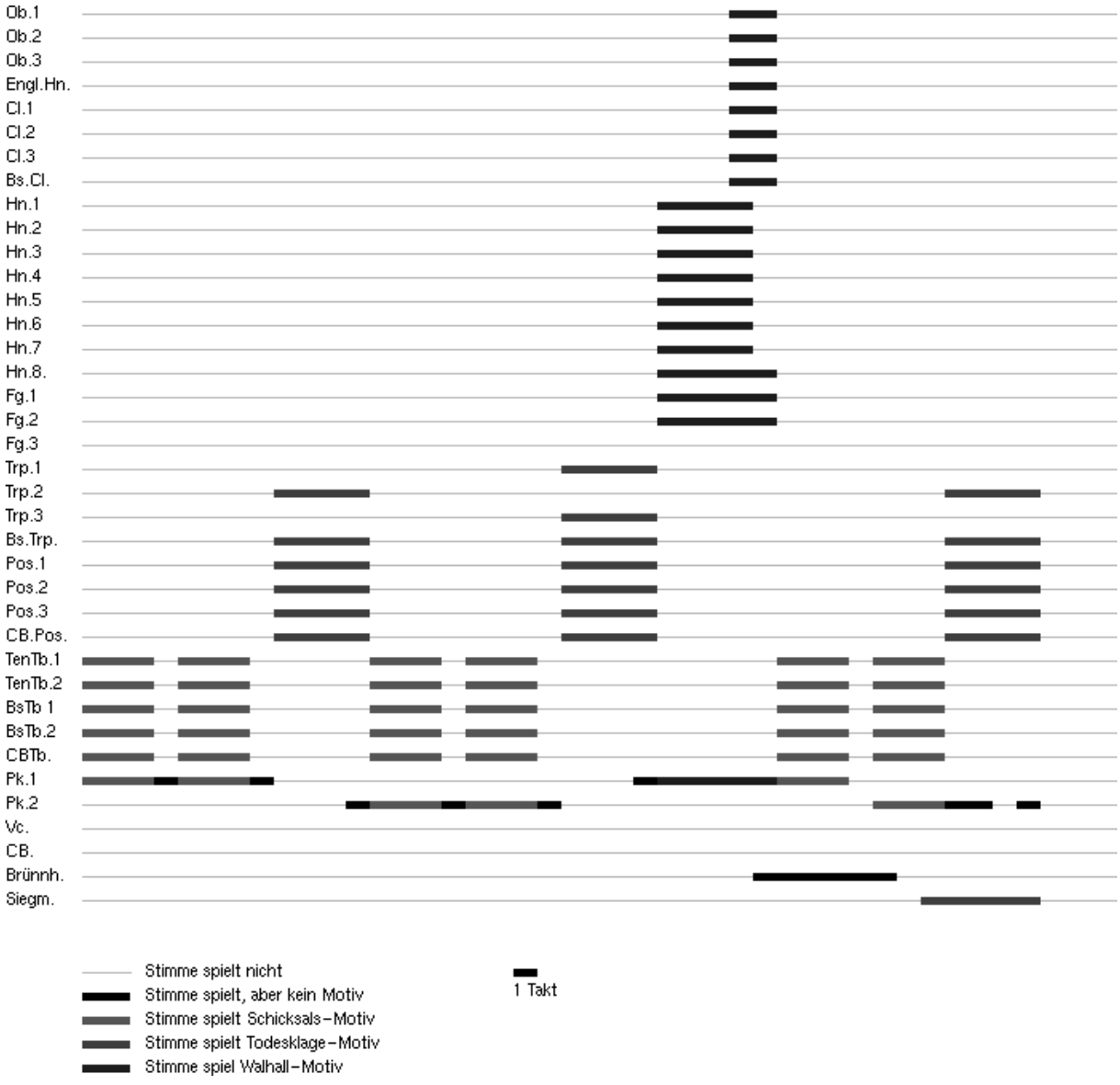
Beschreibung des neuen Arbeitsvorganges

Der Analysator wählt eine Partitur aus, die er mit dem Partituranalysewerkzeug analysieren möchte. Er startet ein (Leitmotivtabelle "klassisch") oder zwei (+ eigene Leitmotivtabelle) Leitmotivbetrachter und positioniert sie so auf dem Bildschirm, daß sie das Partituranalysewerkzeug nicht überdecken, damit er Leitmotive in der Partitur mit Leitmotiven in den Leitmotivtabellen vergleichen kann. Er kann nun die Partitur betrachten und Leitmotive suchen.

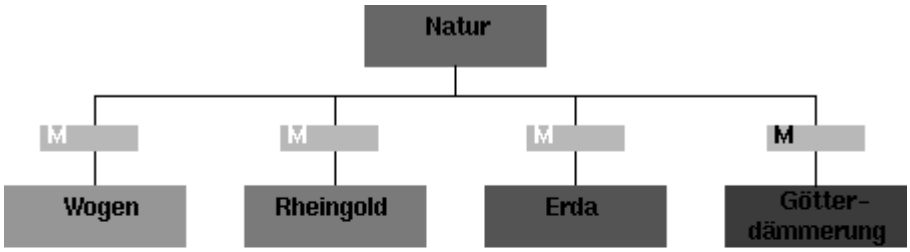
Weitere Ideen / Mögliche Erweiterungen

1. Eine Zuordnung von Leitmotiven und Farben erscheint hilfreich, um die Großstruktur der Leitmotive betrachten zu können. Dazu müßte beim *Leitmotivnotieren* zusätzlich eine Farbe ausgewählt werden, sofern es sich sich um ein neues Leitmotiv handelt (Variationen sollten die gleiche Farbe wie die Urform haben, um nur die semantische Komponente zu betonen). Das Partituranalysewerkzeug müsste einen Modus haben, der die alleinige Betrachtung der Leitmotivstruktur unter Vernachlässigung der Noten zuließe.

   Bsp.: Leitmotivstruktur des Beginns der 4.Szene des 2.Aufzuges aus *Die Walküre*



2. Ein Werkzeug zur Unterstützung der Untersuchung der Leitmotivbeziehungen könnte sich als hilfreich erweisen. Dabei könnte das Verhätnis anhand der eigenen Leitmotivtabelle (die ja bereits Aussagen über das Verhätnis eines Leitmotivs zum Urmotiv bzw. einem anderen Leitmotiv macht) musikwissenschaftlich aufgeschlüsselt und -falls möglich- graphisch dargestellt werden.
   Bsp.: Das Verhältnis einiger Leitmotive zum Natur-Motiv (bei Leitmotiv-Farbzuordnung):



Benachbarte Dokumente

- Systemvisionen
  - *Leitmotive suchen*
- Szenarios
  - *Leitmotivanalyse*
- Glossar

Übersicht | Glossar | Szenario-Übersicht
Systemvision-Übersicht

Systemvision: Leitmotiv suchen

Stand 18.5.95

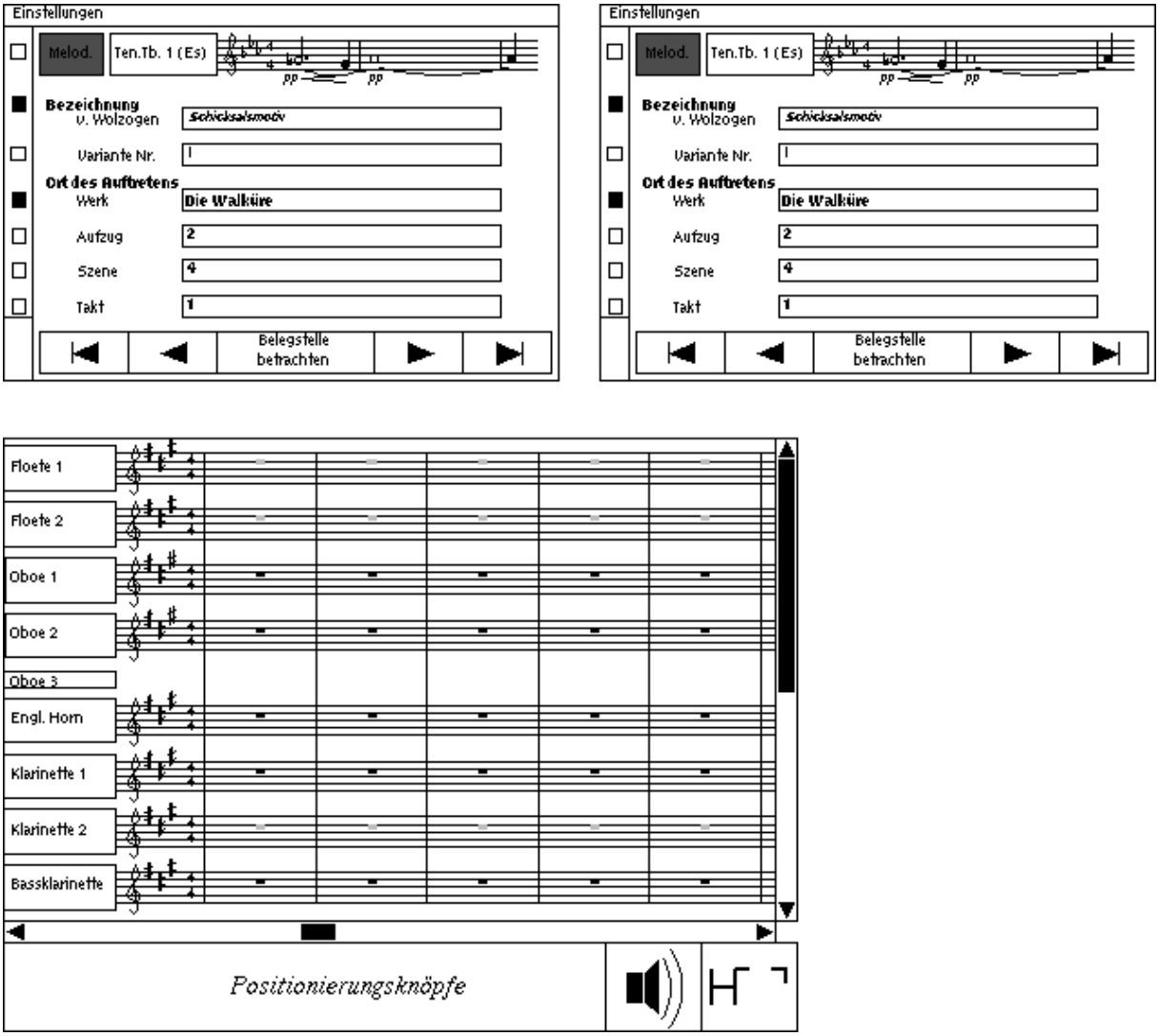
Zugrundeliegende Szenarios [mit Begründung]

- *Leitmotive suchen*
    - Es wird mit denselben Materialien gearbeitet

Verwendete Operationen von Softwarewerkzeugen

- Partituranalysewerkzeug
    - Positionieren
    - Partitur akustisch wiedergeben
    - Stimmen ein- und ausblenden
    - Stimmen vertikal neu anordnen
- Leitmotivbetrachter
    - Anzuzeigende Leitmotivnotizbestandteile auswählen
    - Für Traversierung relevante Leitmotivnotizbestandteile auswählen
    - Durch Leitmotivtabelle traversieren
    - Leitmotivbelegstelle in der Partitur betrachten

Bildschirmskizze



Kontextbeschreibung

Im Rahmen der *Leitmotivanalyse* wird mithilfe des Partituranalysewerkzeugs und Leitmotivbetrachtern die Partitur nach Leitmotiven durchsucht. Diese werden *markiert* und *ausführlich notiert*.

Beschreibung des neuen Arbeitsvorganges

Der Analysator stellt das Partituranalysewerkzeug so ein, daß die zu analysierende Partiturstelle sichtbar wird. Er kann ihm momentan uninteressant erscheinende Stimmen ausblenden und Stimmen vertikal neu anordnen, so daß ihm wichtig erscheinende Stimmen direkt nebeneinander dargestellt werden.

Auf Wunsch kann er eine Tonaufnahme der in der Partitur niedergeschriebenen Oper hören. Die Wiedergabe beginnt an der angezeigten Partiturstelle.

Der Analysator kann die Leitmotivbetrachter so einstellen, daß nur genau die Bestandteile der Leitmotivnotizen angezeigt werden, die er augenblicklich zu sehen wünscht.

Er kann in chronolgischer Reihenfolge durch die Leitmotivnotizen traversieren, wobei er die in Frage kommenden Leitmotivnotizen beliebig einschränken kann (z.B. nur Urmotive, nur in *Götterdämmerung*, etc.).

Er traversiert durch die Partitur und sucht Leitmotive, die er entweder "aus dem Kopf" oder durch Vergleich mit den in den Leitmotivbetrachtern angezeigten Leitmotiven erkennt.

Durch Drücken auf den Markierungsknopf des Partituranalysewerkzeugs kann er ein erkanntes Leitmotiv *markieren* und *notieren*.

Benachbarte Dokumente

- Systemvisionen
  - *Leitmotivanalyse*
  - *Leitmotiv notieren*
  - *Leitmotiv markieren*
- Szenarios
  - *Leitmotive suchen*
- Glossar

Übersicht | Glossar | Szenario–Übersicht | Systemvision–Übersicht

Systemvision: Leitmotiv notieren

Stand 18.5.95

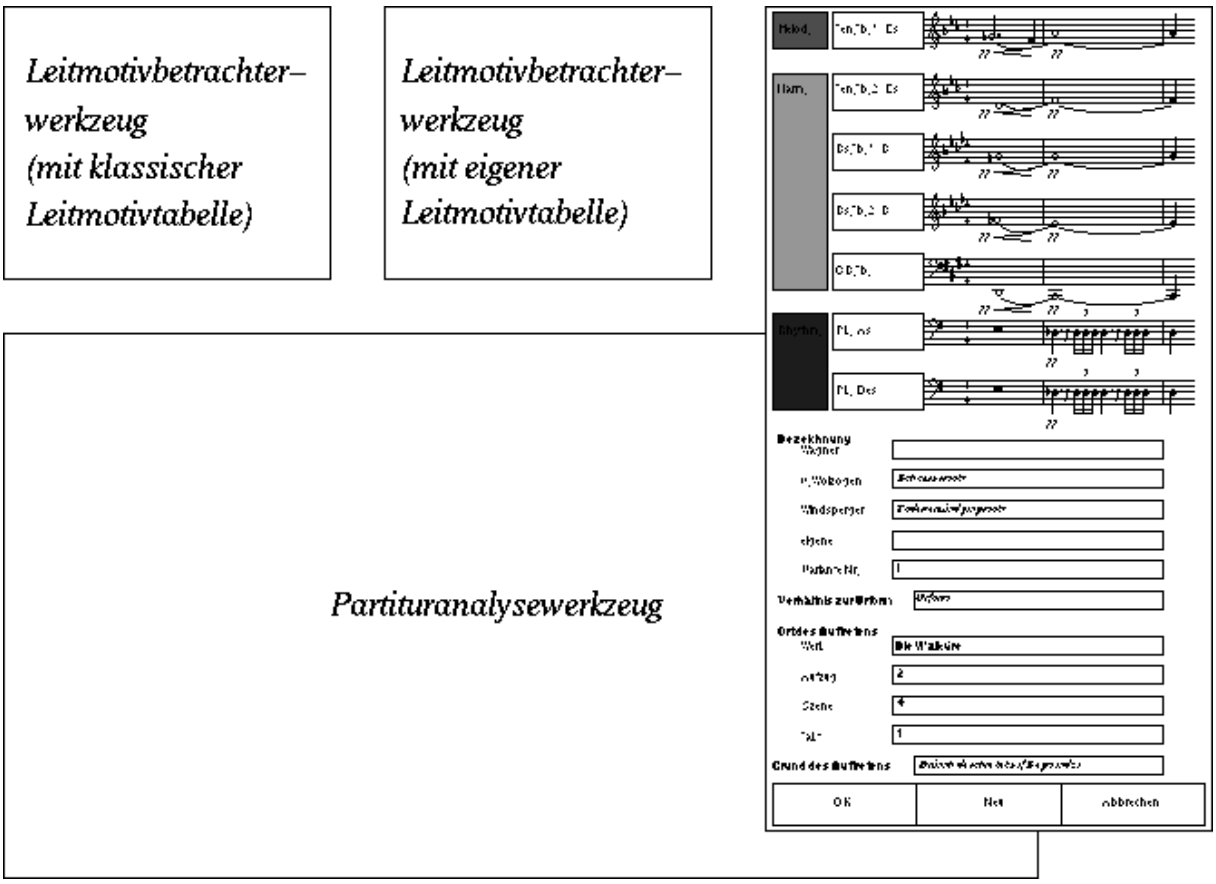
Zugrundeliegende Szenarios [mit Begründung]

- *Leitmotiv notieren*
    - Es wird mit denselben Materialien gearbeitet

Verwendete Operationen von Softwarewerkzeugen

- Leitmotivnotizeditor
    - Leitmotivnotiz ausfüllen
    - Leitmotivnotiz in eigene Leitmotivtabelle übernehmen
    - Leitmotivnotiz verwerfen

Bildschirmskizze



Kontextbeschreibung

Der Leitmotivnotizeditor erscheint automatisch nach der *Leitmotivmarkierung*. Im Anschluß kann die *Leitmotivsuche* fortgesetzt werden.

Ein notiertes Leitmotiv wird in der eigenen Leitmotivtabelle abgelegt und steht über die Leitmotivbetrachter sofort zur Verfügung.

Beschreibung des neuen Arbeitsvorganges

Eine Leitmotivnotiz wird vollständig in ein Formular eingetragen.

Dabei werden alle Bestandteile der Leitmotivnotiz, die sich direkt aus der vorhergegangenen Markierung ableiten lassen (Leitmotivexzerpt, Positionsangaben) automatisch eingetragen. Der Analysator kann sich auf die Eingabe der musikwissenschaftlich relevanten Bestandteile (Name, Verhältnis des Leitmotivs zur Urform, vermuteter Grund für das Auftreten des Leitmotivs) beschränken.

Nach dem Ausfüllen kann der Analysator die Leitmotivnotiz entweder in die eigenen Leitmotivtabelle übernehmen oder die Notiz verwerfen. Der Leitmotivnotizeditor verschwindet daraufhin und der Analysator kann mit der Leitmotivsuche fortfahren.

Benachbarte Dokumente

- Systemvisionen
  - *Leitmotive suchen*
  - *Leitmotiv markieren*
- Szenarios
  - *Leitmotiv notieren*
- Glossar

Systemvision: Leitmotiv markieren

Stand 18.5.95

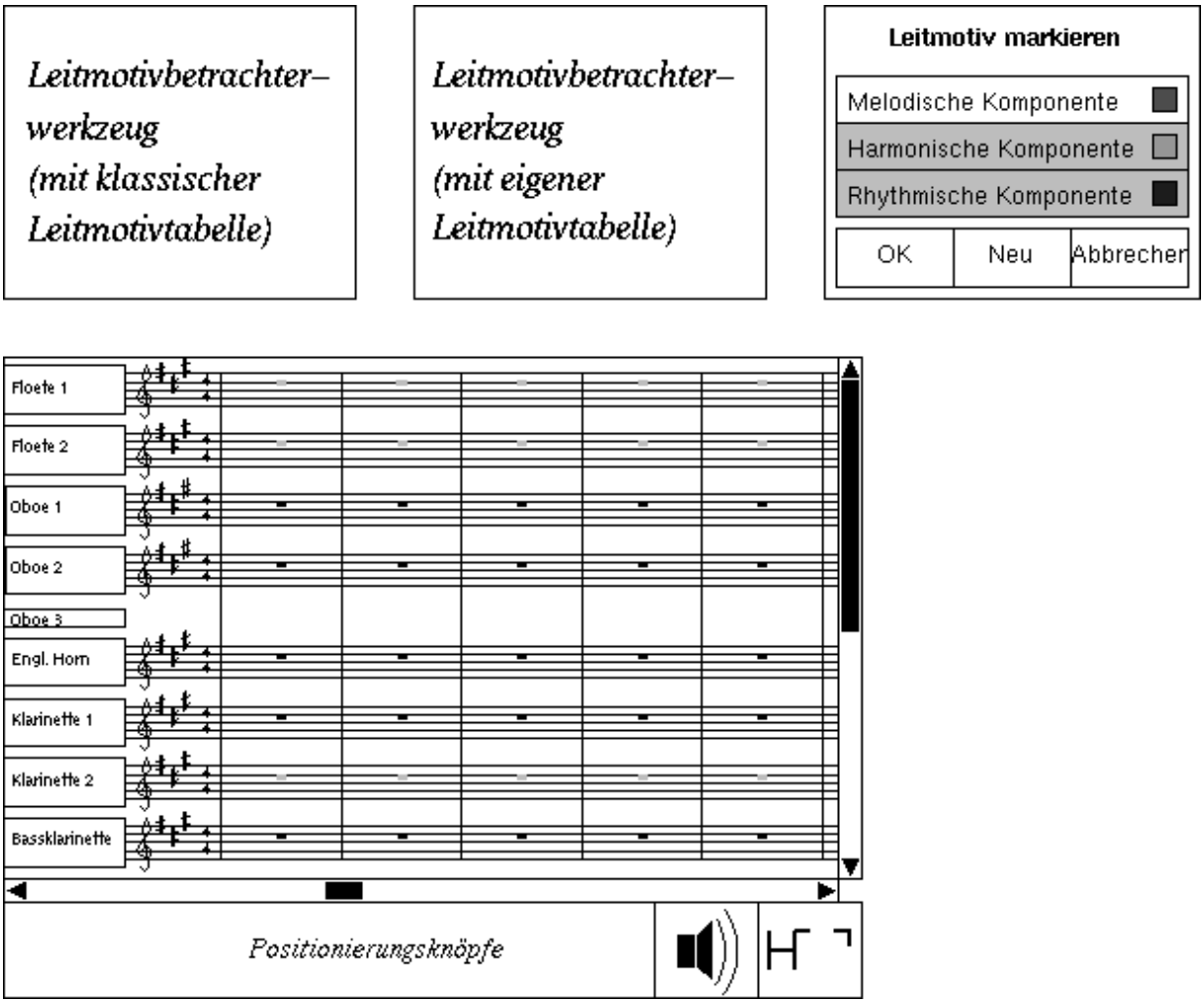
Zugrundeliegende Szenarios [mit Begründung]

- *Leitmotiv markieren*
    - Es wird mit den gleichen Materialen gearbeitet

Verwendete Operationen von Softwarewerkzeugen

- Partituranalysewerkzeug
    - Positionieren
    - Leitmotiv markieren
- Leitmotivmarkierer
    - Zu markierende Leitmotivkomponente auswählen
    - Leitmotivkomponente markieren
    - Markierung als Grundlage für eine Leitmotivnotiz verwenden
    - Markierung verwerfen

Bildschirmskizze



Kontextbeschreibung

Der Analysator positioniert mithilfe des Partituranalysewerkzeugs die Partitur so, daß das vollständige Leitmotiv sichtbar ist und drückt auf den Knopf "Markieren", woraufhin ein Leitmotivmarkierer erscheint.

Nach der Markierung wird automatisch ein Leitmotivnotizeditor aufgerufen.

Beschreibung des neuen Arbeitsvorganges

Der Analysator wählt zunächst aus, welchen der drei möglichen Leitmotivkomponenten er markieren will. Dann markiert er in der Partitur stimmenweise die ausgewählte Komponente. Die markierte Leitmotivkomponente wird entsprechend der jeweiligen Leitmotivkomponente farbig hervorgehoben:

- Melodische Komponente: rot
- Harmonsiche Komponente: grün
- Rhythmische Komponente: blau

Nachdem er alle Komponenten des Leitmotivs markiert hat, kann er dieses als Leitmotivnotiz in die eigene Leitmotivtabelle aufnehmen, oder aber die Markierung (z.B. weil er sich geirrt hat) verwerfen.

Benachbarte Dokumente

- Systemvisionen
  - *Leitmotive suchen*
  - *Leitmotiv notieren*
- Szenarios
  - *Leitmotiv markieren*
- Glossar

Systemvision: Partituranalysewerkzeug

Stand 18.5.95

Ablaufvisionen, in denen mit dem Softwarewerkzeug gearbeitet wird

1. *Leitmotivanalyse*
2. *Leitmotive suchen*
3. *Leitmotiv markieren*

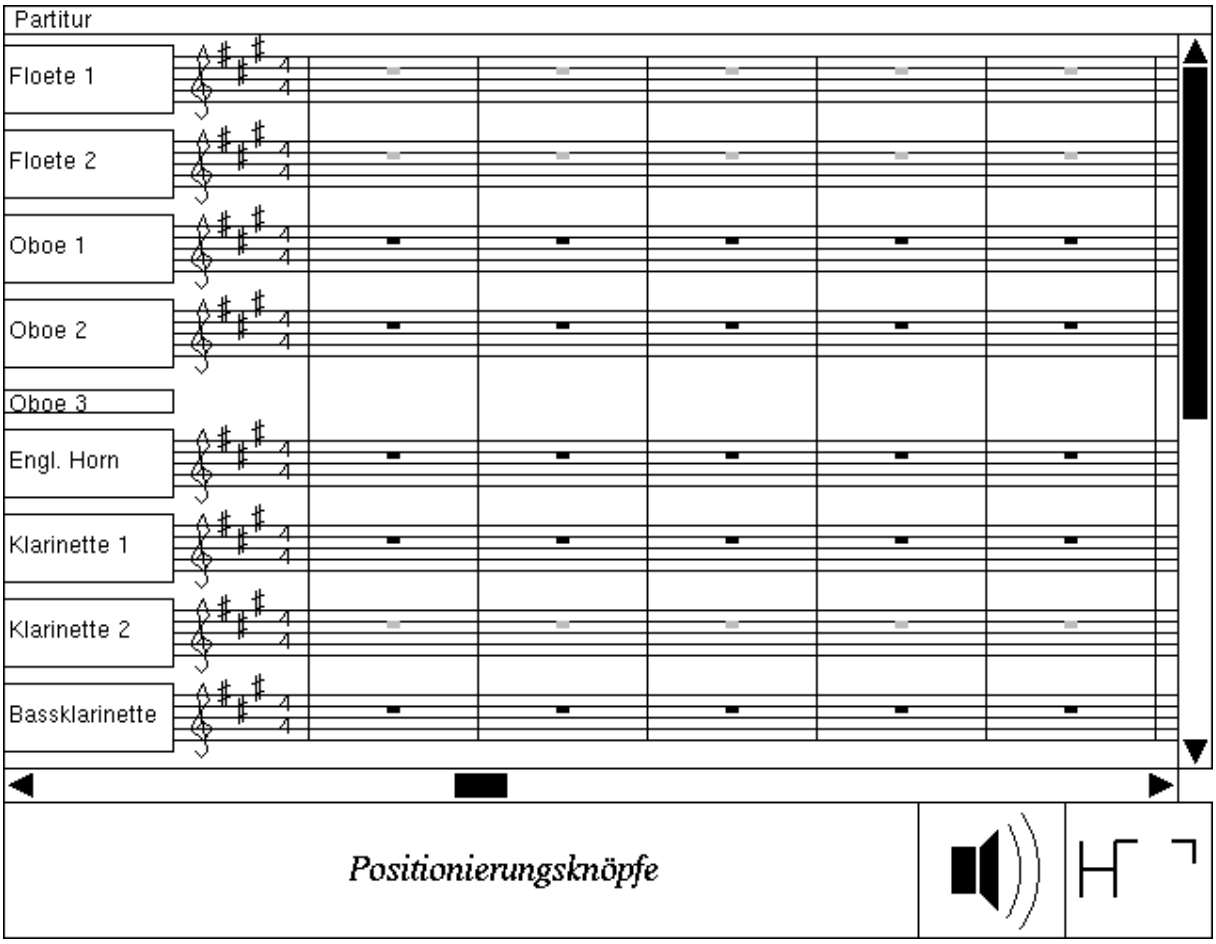
Mögliche Operationen

Nummer der Ablaufvisionen, in der Operation verwendet wird, in Klammern dahinter

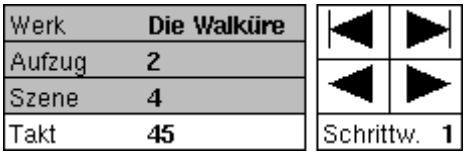
- Zu analysierende Partitur auswählen (1)
- Stimmen ein- und ausblenden (2)
- Stimmen vertikal neu anordnen (2)
- Partitur positionieren (2)
- Partitur akustisch wiedergeben (2)
- Leitmotivmarkierung einleiten (2)

Bildschirmskizze

Partituranalysewerkzeug



Positionierungsknöpfe (vertikale Anordnung)



Positionierungsknöpfe (horizontale Anordnung)

Kurzbeschreibung

Das Partituranalysewerkzeug dient zum Betrachten der Partitur. In Verbindung mit dem Leitmotivmarkierer können Leitmotive in der im Partituranalysewerkzeug angezeigten Partitur markiert werden.

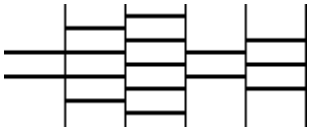
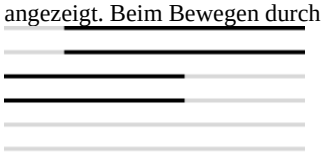
Detailierte Beschreibung

Zu analysierende Partitur auswählen

Zu Beginn der Arbeit mit dem Werkzeug muß festgelegt werden, welche Partitur analysiert werden soll. Dies geschieht über die Auswahl des Menüpunktes "Auswahl" aus dem Menü "Partitur" und kann jederzeit während der Arbeit wiederholt werden, wobei die bisherige Partitur samt der durchgeführten Markierungen abgespeichert wird.

Partiturdarstellung

Die bearbeitete Partitur wird im oberen Bereich des Werkzeuges ausschnittsweise angezeigt. Für ihre Darstellung kommen zwei Methoden in Frage:

-  Die Partitur erscheint wie in gedruckter Form. Pausierende Stimmen werden nicht angezeigt. Beim Bewegen durch die Partitur kommt es zu "Sprüngen" an den Grenzen von Partiturseiten.
-  Die Partitur erscheint in kanonischer Form. Alle Stimmen (auch die pausierenden) werden angezeigt. Jede Stimme kann dadurch über Partiturseitengrenzen hinaus kontinuierlich verfolgt werden, ohne daß es zu Sprüngen an den Grenzen von Seiten der zugrundeliegenden gedruckten Partitur kommt.

Die zweite Methode verspricht eine größere Übersichtlichkeit und sollte prototypisch implementiert werden.

Partitursichtmanipulation (Stimmen ein- und ausblenden sowie Stimmen vertikal neu anordnen)

Um bei der kanonischen Partiturdarstellung nicht zu viele pausierende Stimmen im angezeigten Partiturausschnitt zu sehen und um einen räumlich möglichst direkten Vergleich von Stimmen zu ermöglichen, stehen die folgenden Operationen zur Verfügung:

- Stimmen können (auch gruppenweise) durch Doppelklick auf den Stimmenkopf ein- und ausgeblendet werden. Gruppen können selektiert werden, indem mit der Maus ein Rechteck über den Stimmenköpfen aufgezogen wird. Ein Doppelklick hat folgende Wirkung:
  - eine oder mehrere der selektierten Stimmen sind eingeblendet: Alle selektierten Stimmen werden ausgeblendet
  - sonst: Alle selektierten Stimmen werden eingeblendet
    Ausgeblendete Stimmen erscheinen mit einem verkleinerten Stimmenkopf und ohne dazugehöriges System.
- Stimmen können (auch gruppenweise) durch Klicken auf den Stimmenkopf und Ziehen an eine neue Position vertikal neu angeordnet werden. Gruppen können selektiert werden, indem mit der Maus ein Rechteck über den Stimmenköpfen aufgezogen wird.

Partitur positionieren

Um jeden Teil der Partitur betrachten zu können, kann der sichtbare Partiturausschnitt mit Hilfe von Rollbalken und Positionierungsknöpfen verschoben werden.

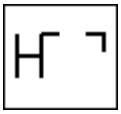
- Rollbalken
  Mit Rollbalken kann der sichtbare Partiturausschnitt relativ zur bisherigen Position verschoben werden (z.B. "ein paar Takte zurück")
  - mit dem Rollbalken am rechten Rand des sichtbaren Partiturausschnitts kann dieser vertikal verschoben werden. Sozusagen ein Blick auf den weiter oben oder unten liegenden Teil der Partiturseite
  - mit dem Rollbalken am unteren Rand des sichtbaren Partiturausschnitts kann dieser horizontal verschoben werden. Dies entspricht einem Vor- oder Zurückblättern in der Partitur
- Positionierungsknöpfe.
  Diese Knöpfe erlauben eine absolute zeitliche Positionierung des sichtbaren Partiturausschnitts. Die jeweilige Operation bezieht sich auf die hell hervorgehobene logische Einheit (Werk, Aufzug, Szene oder Takt). Die Schrittweite kann direkt in das Feld "Schrittweite" eingegeben werden. Alternativ können auch die Zahlen in den Feldern "Aufzug", "Szene" und "Takt" nach Anklicken über die Tastatur eingegeben werden.
  -  An den Anfang der Einheit
  -  An das Ende der Einheit
  -  ***Schrittweite*** Einheiten zurück
  -  ***Schrittweite*** Einheiten vor

Partitur akustisch wiedergeben

Durch Drückern auf den Knopf "Abspielen"  kann (beginnend bei der ausgewählten Stelle) eine Tonaufnahme der in der Partitur niedergeschriebenen Oper abgespielt werden. Die Partitur wird dabei bei Erreichen des Seitenrandes sprungweise so aktualisiert, daß die Darstellung mit der Tonwiedergabe korrespondiert. Manuelle Posisitionsänderungen werden bei der Wiedergabe berücksichtigt.

Die Wiedergabe kann durch erneutes Drücken auf den Knopf Wiedergabe  beendet werden. Die Partitur verharrt an dieser Stelle.

Leitmotivmarkierung einleiten

Um ein Leitmotiv in der Partitur zu markieren, muß zunächst der sichtbare Partiturauschnitt so positioniert werden, daß alle Leitmotivbestandteile sichtbar sind. Durch Drücken auf den Markierungsknopf 

wird der Leitmotivmarkierer sichbar, mit dessen Hilfe das Leitmotiv *markiert* werden kann. Der Markierungsknopf erscheint solange schattiert, bis der Markierungsvorgang abgeschlossen ist.

Verwendete Verfahrensvorschriften / Algorithmen

Weitere Ideen / Mögliche Erweiterungen

Für die Analyse der Leitmotivstruktur im Großen könnte es hilfreich sein, einen Modus zu haben, der die alleinige Betrachtung der Leitmotivstruktur unter Vernachlässigung der Noten zuließe.
Beispiel in der *Systemvision Leitmotivanalyse* in der gleichen Rubrik.

Benachbarte Dokumente

- Systemvisionen
  - *Leitmotiv suchen*
  - *Leitmotiv notieren*
  - *Leitmotiv markieren*
  - Leitmotivmarkierer
- Szenarios
  - *Leitmotive suchen*
- Glossar

Systemvision: Leitmotivnotizeditor

Stand 18.5.95

Ablaufvisionen, in denen mit dem Softwarewerkzeug gearbeitet wird

1. *Leitmotive notieren*

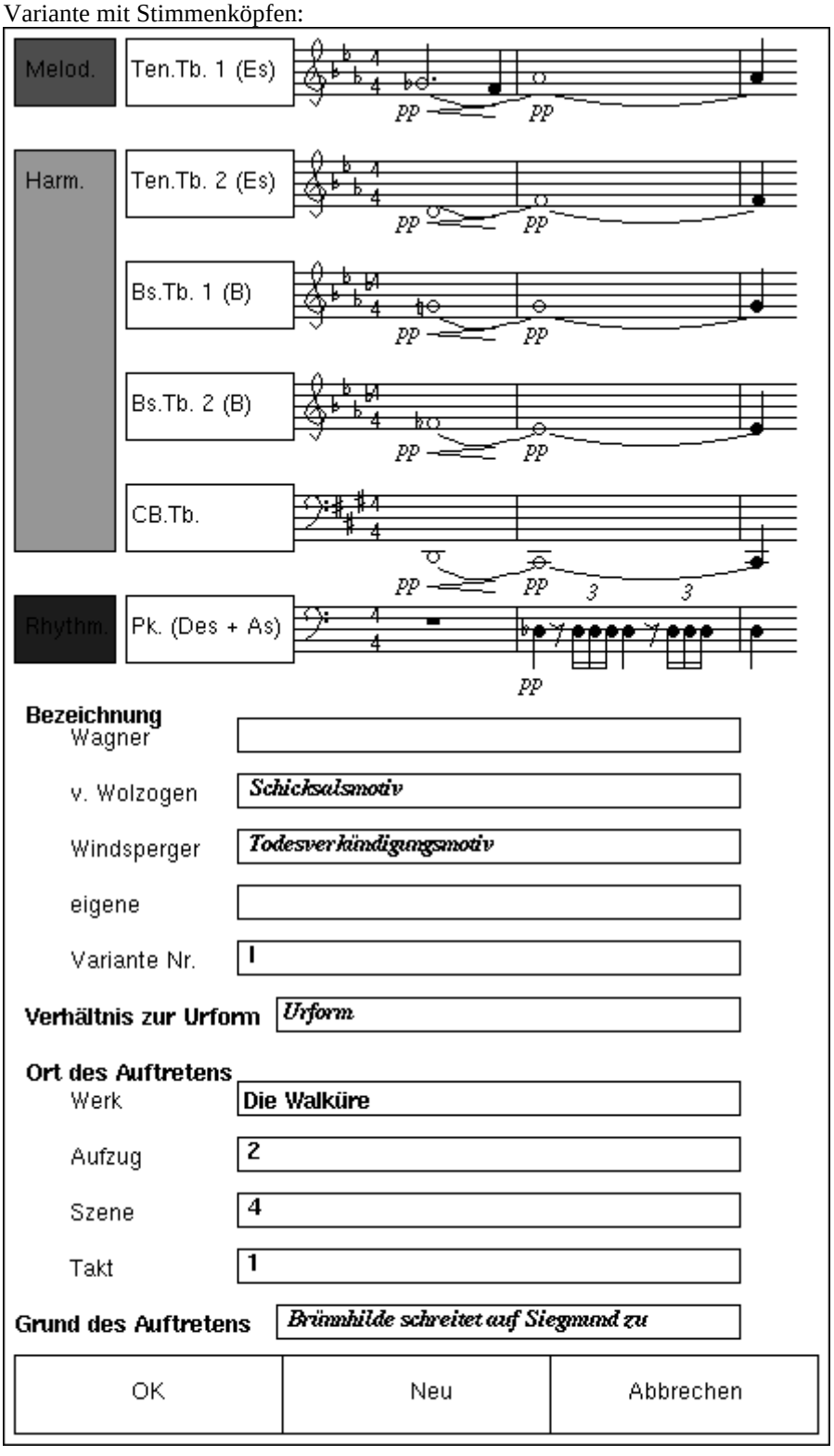
Mögliche Operationen

Nummer der Ablaufvisionen, in der Operation verwendet wird, in Klammern dahinter

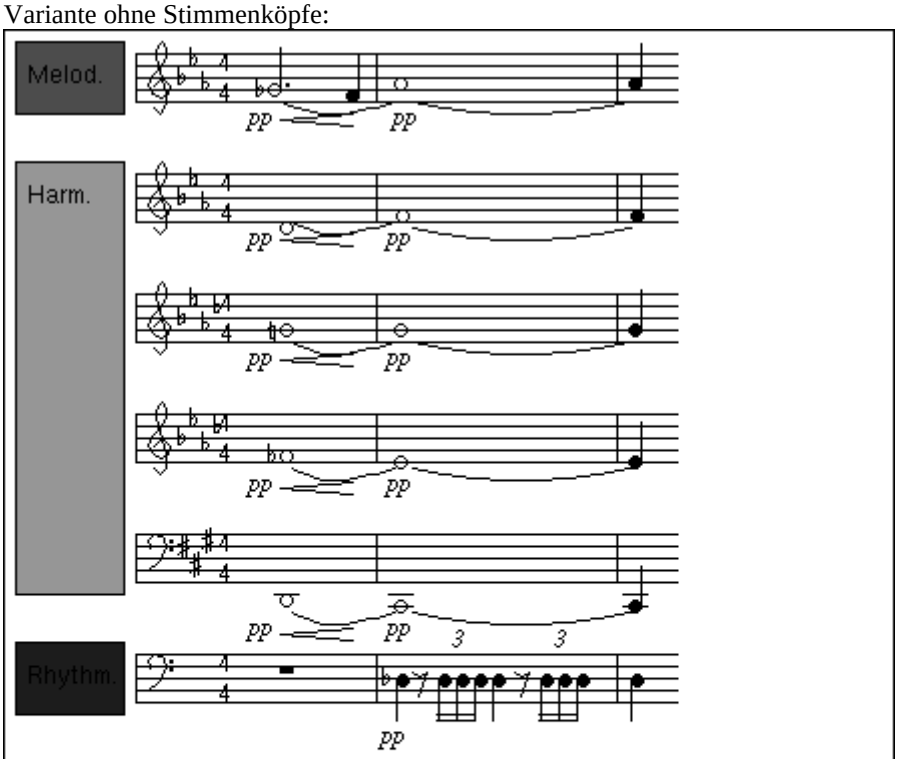
- Leitmotivnotizbestandteile eintragen (1)
- Notiervorgang beenden und Leitmotivnotiz in eigene Leitmotivtabelle übernehmen (1)
- Leitmotivnotiz verwerfen und auf Basis der Markierung neu erstellen (1)
- Leitmotivnotiz verwerfen und mit Leitmotivsuche fortfahren (1)

Bildschirmskizze

Variante mit Stimmenköpfen:



Variante ohne Stimmenköpfe:



Bezeichnung

Wagner	
v. Wolzogen	*Schicksalsmotiv*
Windsperger	*Todesverkündigungsmotiv*
eigene	
Variante Nr.	I

Verhältnis zur Urform *Urform*

Ort des Auftretens

Werk	**Die Walküre**
Aufzug	2
Szene	4
Takt	1

Stimmen

Melodische Komponente	**Ten.Tb. 1 (Es)**
Harmonische Komponente	**Ten.Tb. 2 (Es), Bs.Tb. 1 (B), Bs.**
Rhythmische Komponente	**Pk. (Des + As)**

Grund des Auftretens *Brünnhilde schreitet auf Siegmund zu*

OK | Neu | Abbrechen

Kurzbeschreibung

Der Leitmotivnotizeditor dient zur Erstellung einer Leitmotivnotiz. Eine fertige Leitmotivnotiz wird in die eigene Leitmotivtabelle übernommmen.

Detailierte Beschreibung

Leitmotivnotizbestandteile eintragen

Eine Leitmotivnotiz wird vollständig in ein Formular eingetragen.

Automatisch werden eingetragen:

- Das Leitmotivexzerpt
- Die Leitmotivposition
- Zusätzlich in der Variante ohne Stimmenköpfe: Die Stimmen, in denen die Leitmotivkomponenten auftreten

Automatisch vorgeschlagen werden:

- Die chronologische Variantennummer

Selbst einzutragen sind:

- Der Leitmotivname gemäß allen relevanten Quellen (Auswahl aus Liste oder eigener neuer Name)
- Das Verhätnis zur Urform des Leitmotivs
- Der vermutete Grund für das Auftreten

Notiervorgang beenden und Leitmotivnotiz in eigene Leitmotivtabelle übernehmen

Druck auf den Knopf "OK". Der Leitmotivnotizeditor verschwindet und mit der *Leitmotivsuche* kann fortgefahren werden.

Leitmotivnotiz verwerfen und auf Basis der Markierung neu erstellen

Druck auf den Knopf "Neu". Alle selbst vorgenommenen Eintragungen werden gelöscht.

Leitmotivnotiz verwerfen und mit Leitmotivsuche fortfahren

Druck auf den Knopf "Abbruch". Alle selbst vorgenommenen Eintragungen und die Markierung in der Partitur werden gelöscht, der Leitmotivnotizeditor verschwindet und mit der *Leitmotivsuche* kann fortgefahren werden.

Verwendete Verfahrensvorschriften / Algorithmen

Weitere Ideen / Mögliche Erweiterungen

Für die Analyse der Leitmotivstrukturkönnte es hilfreich sein, den Leitmotiven Farben zuzuordnen, um sie auf einen Blick in der Partitur erkennen zu können. Für die Farbe müßte die Leitmotivnotiz um den Eintrag "Farbe" erweitert werden.

Beispiel in der *Systemvision Leitmotivanalyse* in der gleichen Rubrik.

Benachbarte Dokumente

- Systemvisionen
  - *Leitmotive suchen*
  - *Leitmotiv notieren*
  - *Leitmotiv markieren*

- Szenarios
    - *Leitmotiv notieren*
- Glossar

Systemvision: Leitmotivmarkierer

Stand 18.5.95

Ablaufvisionen, in denen mit dem Softwarewerkzeug gearbeitet wird

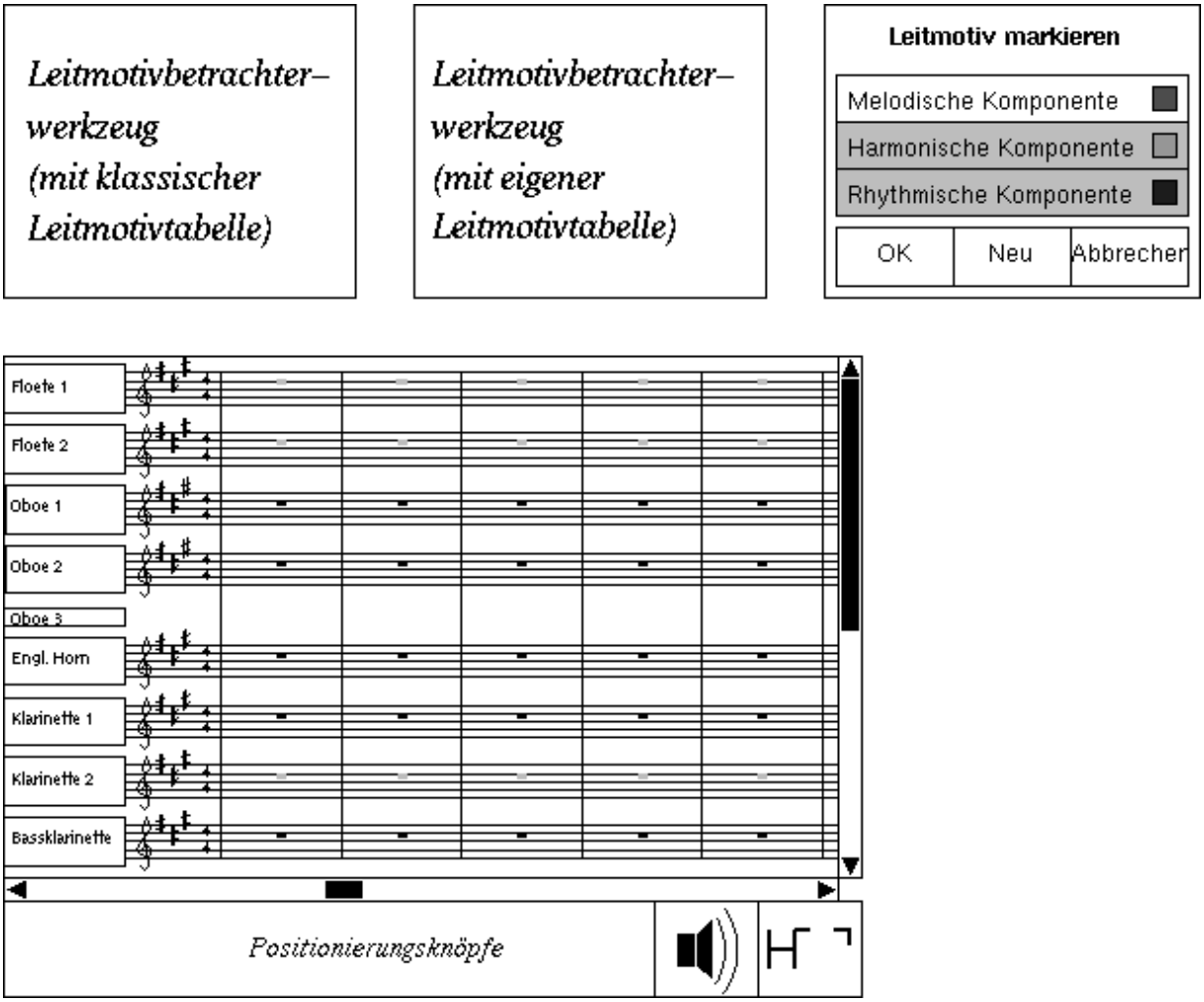
1. *Leitmotive markieren*

Mögliche Operationen

Nummer der Ablaufvisionen, in der Operation verwendet wird, in Klammern dahinter

- Zu markierende Leitmotivkomponente auswählen (1)
- Ausgewählte Leitmotivkomponente in der Partitur markieren (1)
- Markierungsvorgang beenden und Notiervorgang beginnen (1)
- Markierungsvorgang wiederholen (1)
- Markierung verwerfen und mit Leitmotivsuche fortfahren (1)

Bildschirmskizze



Kurzbeschreibung

Der Leitmotivmarkierer dient zur Markierung eines in der Partitur erkannten Leitmotivs. Eine vollständige Leitmotivmarkierung dient als Grundlage für die anschließende Anfertigung einer Leitmotivnotiz.

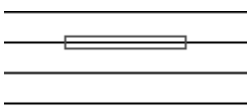
Detailierte Beschreibung

Zu markierende Leitmotivkomponente auswählen

Durch Drücken auf den dementsprechenden Knopf des Leitmotivmarkierers wird ausgewählt, welche der drei möglichen Leitmotivkomponenten als nächstes markiert werden soll. Ohne eine anfängliche Auswahl kann keine Leitmotivkomponente markiert werden. Der Knopf der ausgewählten Komponente wird bis zum Abschluß des Markierungsvorganges hell unterlegt.

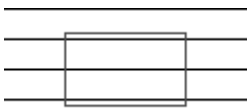
Ausgewählte Leitmotivkomponente in der Partitur markieren

Nach Auswahl einer Leitmotivkomponente kann diese direkt in der im Partituranalysewerkzeug angezeigten Partitur markiert werden. Dabei sind folgende Fälle zu unterscheiden:

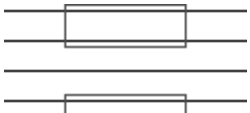


- Die ausgewählte Komponente umfaßt nur eine Stimme:
  Ein Mausklick links der ersten Note und Ziehen bis rechts der letzten Note.

- Die ausgewählte Komponente wechselt die Stimme:
  Ein Mausklick links der ersten Note des ersten Teils und Ziehen bis rechts der letzten Note des ersten Teils. Dann ein Mausklick links der ersten Note des zweiten Teils und Ziehen bis rechts der letzten Note des zweiten Teils.



- Die ausgewählte Komponente umfaßt mehrere benachbarte Stimmen:
  Ein Mausklick links der ersten Note der obersten Stimme und Ziehen bis rechts der letzten Note der untersten Stimme.



- Die ausgewählte Komponente umfaßt mehrere Gruppen benachbarter Stimmen, die ihrerseits nicht benachbart sind:
  Ein Mausklick links der ersten Note der obersten Stimme der ersten Gruppe und Ziehen bis rechts der letzten Note der untersten Stimme der ersten Gruppe. Danach ein Mausklick links der ersten Note der obersten Stimme der zweiten Gruppe und Ziehen bis rechts der letzten Note der untersten Stimme der zweiten Gruppe und so weiter, bis alle Gruppen markiert sind.

Die markierte Leitmotivkomponente wird während und nach dem Markierungsvorgang entsprechend der jeweiligen Leitmotivkomponente farbig eingerahmt:

- Melodische Komponente: rot
- Harmonsiche Komponente: grün
- Rhythmische Komponente: blau

Markierungsvorgang beenden und Notiervorgang beginnen

Druck auf den Knopf "OK". Der Leitmotivmarkierer verschwindet und ein Leitmotivnotizeditor erscheint.

Markierungsvorgang wiederholen

Druck auf den Knopf "Neu". Alle bisher durchgeführten Markierungen der ausgewählten Leitmotivkomponente werden gelöscht.

Markierung verwerfen und mit Leitmotivsuche fortfahren
Druck auf den Knopf "Abbrechen". Alle bisher durchgeführten Markierungen werden gelöscht und der Leitmotivmarkierer verschwindet. Mit der *Leitmotivsuche* kann fortgefahren werden.

Verwendete Verfahrensvorschriften / Algorithmen

Benachbarte Dokumente

- Systemvisionen
  - *Leitmotive suchen*
  - *Leitmotiv notieren*
  - *Leitmotiv markieren*
  - Partituranalysewerkzeug
  - Leitmotivnotizeditor
- Szenarios
  - *Leitmotiv markieren*
- Glossar

Übersicht | Glossar | Szenario–Übersicht | Systemvision–Übersicht

Systemvision: Leitmotivbetrachter

Stand 18.5.95

Ablaufvisionen, in denen mit dem Softwarewerkzeug gearbeitet wird

1. Leitmotivanalyse
2. Leitmotive suchen

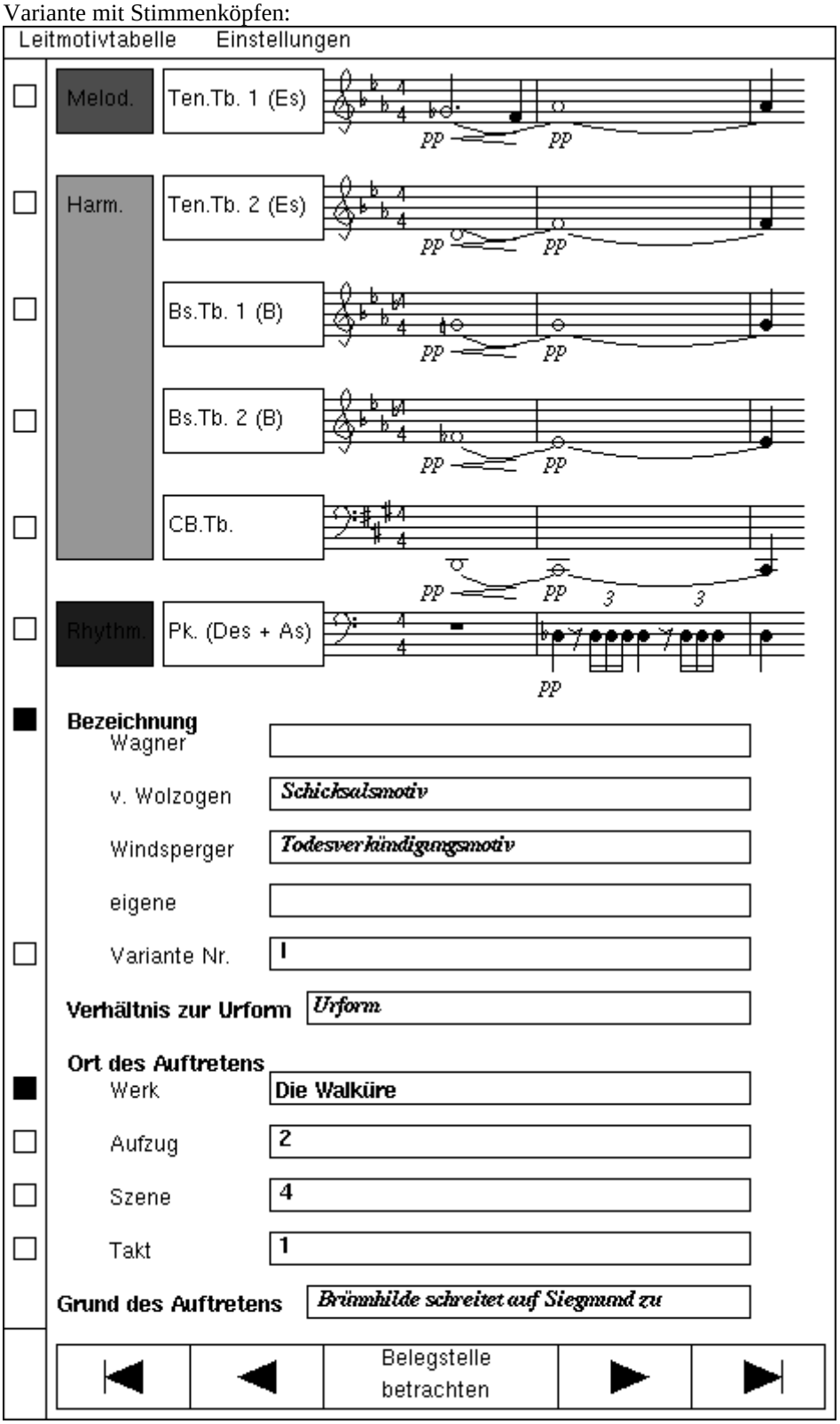
Mögliche Operationen

Nummer der Ablaufvisionen, in der Operation verwendet wird, in Klammern dahinter

- Zu betrachtende Leitmotivtabelle auswählen (1)
- Anzuzeigende Leitmotivnotizbestandteile auswählen (2)
- Für Traversierung relevante Leitmotivnotizbestandteile auswählen (2)
- Durch Leitmotivtabelle traversieren (2)
- Leitmotivbelegstelle in der Partitur betrachten (2)

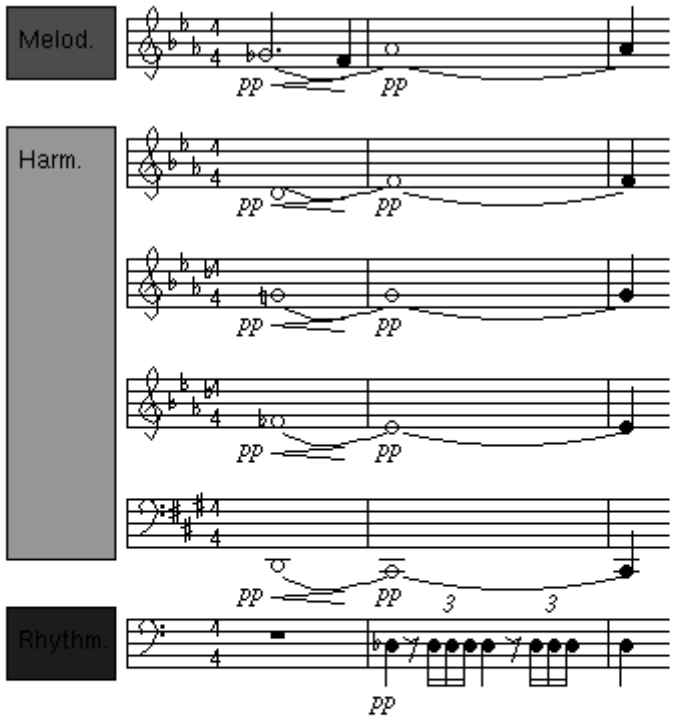
Bildschirmskizze

Variante mit Stimmenköpfen:



Variante ohne Stimmenköpfe:

Leitmotivtabelle Einstellungen



Bezeichnung

Wagner	
v. Wolzogen	*Schicksalsmotiv*
Windsperger	*Todesverkündigungsmotiv*
eigene	
Variante Nr.	I

Verhältnis zur Urform *Urform*

Ort des Auftretens

Werk	**Die Walküre**
Aufzug	**2**
Szene	**4**
Takt	**1**

Stimmen

Melodische Komponente	**Ten.Tb. 1 (Es)**
Harmonische Komponente	**Ten.Tb. 2 (Es), Bs.Tb. 1 (B), Bs.**
Rhythmische Komponente	**Pk. (Des + As)**

Grund des Auftretens *Brünnhilde schreitet auf Siegmund zu*

◀| ◀ Belegstelle betrachten ▶ |▶

Kurzbeschreibung

Der Leitmotivbetrachter dient zur Betrachtung von Leitmotivtabellen. Er kann einzeln als "Leitmotivlexikon" oder in Verbindung mit einem Partituranalysewerkzeug zur *Leitmotivsuche* verwendet werden.

Detailierte Beschreibung

Zu betrachtende Leitmotivtabelle auswählen

Der Analysator wählt zu Beginn der Arbeit mit dem Werkzeug die gewünschte Leitmotivtabelle aus. Dies geschieht über die Auswahl des Menüpunktes "Auswahl" aus dem Menü "Leitmotivtabelle" und kann jederzeit während der Arbeit wiederholt werden.

Anzuzeigende Leitmotivnotizbestandteile auswählen

Leitmotive können maximal so vollständig angezeigt werden, wie in der Leitmotivnotiz hinterlegt. Jeder Leitmotivnotizbestandteil (Notenbild, Name, Positionsangabe) kann über das Menü "Einstellungen" ein- oder ausgeblendet werden. Im Menü erscheinen alle verfügbaren Leitmotivnotizbestandteile - die sichtbaren markiert, der Rest unmarkiert.

Durch Leitmotivtabelle traversieren und für Traversierung relevante Leitmotivnotizbestandteile auswählen

Leitmotive können über die Pfeilknöpfe der Operationenleiste chronologisch vorwärts und rückwärts traversiert werden. Dabei werden nur die Leitmotive angezeigt, die in allen in der am linken Rand befindlichen Auswahlleiste selektierten Bestandteilen mit dem gerade angezeigten Leitmotiv übereinstimmen. Die Auswahlleistenknöpfe wechseln durch einfaches Anklicken ihren Zustand zwischen "für Traversierung relevant" (■) und "für Traversierung irrelevant" (□).

Für die Traversierung (immer bezüglich der aktuellen Selektion) stehen die folgenden Knöpfe zur Verfügung:

-  Zum ersten Eintrag der Leitmotivtabelle
-  Zum letzten Eintrag der Leitmotivtabelle
-  Einen Eintrag zurück
-  Einen Eintrag weiter

Leitmotivbelegstelle in der Partitur betrachten

Durch Druck auf den Knopf "Belegstelle betrachten" wird im Partituranalysewerkzeug die Partiturstelle angezeigt, von der die angezeigte Leitmotivnotiz stammt. Falls gerade kein Partituranalysewerkzeug vorhanden ist, wird eines gestartet.

Verwendete Verfahrensvorschriften / Algorithmen

Eine Datenbank bietet sich für die Verwaltung der Leitmotive an.

Benachbarte Dokumente

- Systemvisionen

    - *Leitmotivanalyse*
    - *Leitmotive suchen*
    - Partituranalysewerkzeug
- Szenarios
    - *Leitmotive suchen*
- Glossar

Handhabungsprototyp

Stand 23.6.95

Zugrundeliegende Systemvisionen

- alle

Beschreibung

Der Prototyp dient lediglich der Erprobung der Handhabung und implementiert keine der in den Systemvisionen beschriebenen Funktionalitäten.

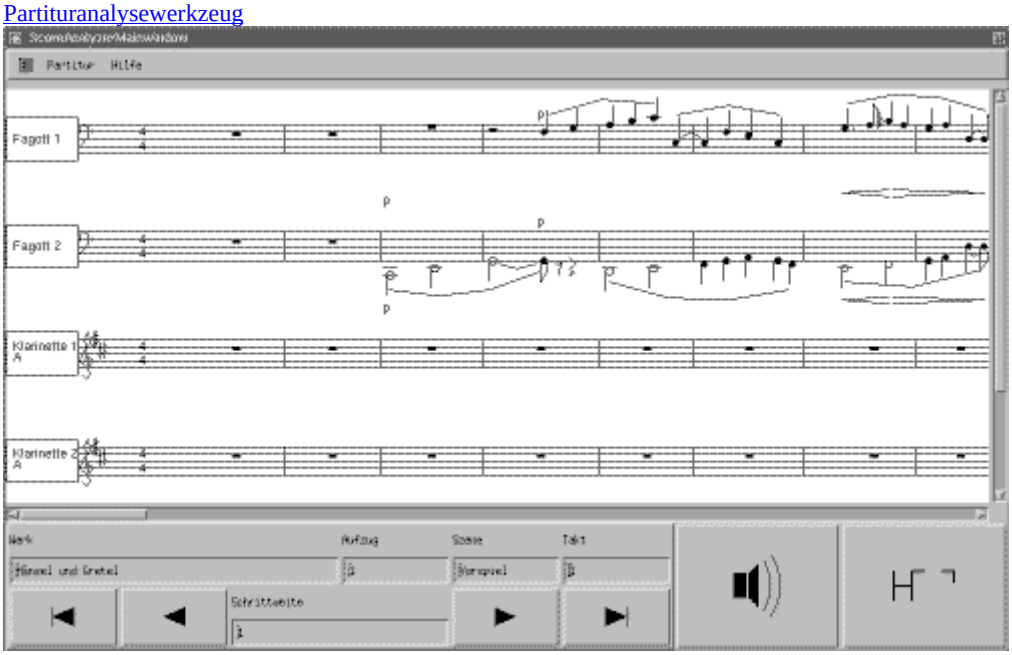
Zusätzlich zu den in den Systemvisionen beschrieben Softwarewerkzeugen wurde ein weiteres Werkzeug zur Koordinierung der vorhandenen Werkzeuge implementiert. Es stellt den leeren Schreibtisch mit geschlossen Leitmotivtabellen und Partituren dar und dient ausschließlich dazu, diese zu öffnen. Dies geschieht über die Menüpunkte `Partituranalysewerkzeug` und `Leitmotivbetrachter` aus dem Menü `Neues Werkzeug`.

Alle Werkzeuge besitzen ein `Hilfe`-Menü, nach Auswahl dessen einzigen Menüpunkts *Mosaic* mit der zum Werkzeug gehörigen Systemvision gestartet wird.

Beendet werden kann die Arbeit durch Auswahl des Menüpunkts `Arbeit beenden` aus dem Menü ganz links mit dem Symbol.

Vorhandene Operationen von Softwarewerkzeugen

- Partituranalysewerkzeug



  - Zu analysierende Partitur auswählen

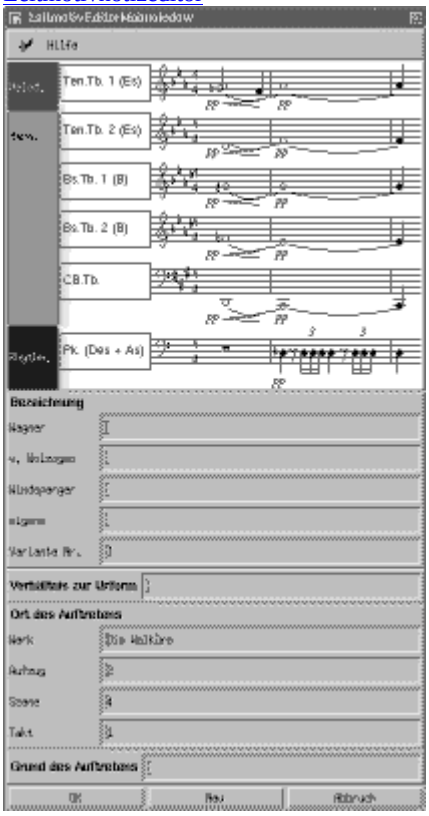
- Über das Menü `Partitur/Auswählen` kann eine Partitur ausgewählt werden. Es erscheint ein Dateiauswahlfenster, in dem nur `HG.sco` (für *Hänsel und Gretel*) erscheint. Durch Doppelklick darauf werden die ersten 60 Takte der Stimmen *Fagott 1*, *Fagott 2*, *Klarinette 1* und *Klarinette 2* geladen (willkürliche Auswahl im Rahmen des Prototyps) und stehen nach einigen Sekunden zur Verfügung.

- Stimmen ein- und ausblenden
  - nicht implementiert, da mit echten Noten zu aufwendig
- Stimmen vertikal neu anordnen
  - nicht implementiert, da mit echten Noten zu aufwendig
- Partitur positionieren
  - gemäß Systemvision, allerdings nur innerhalb der ersten 60 Takte
- Partitur akustisch wiedergeben
  - nicht implementiert, da CD-Ankoppelung noch nicht vollzogen
- Leitmotivmarkierung einleiten
  - gemäß Systemvision

- Leitmotivmarkierer

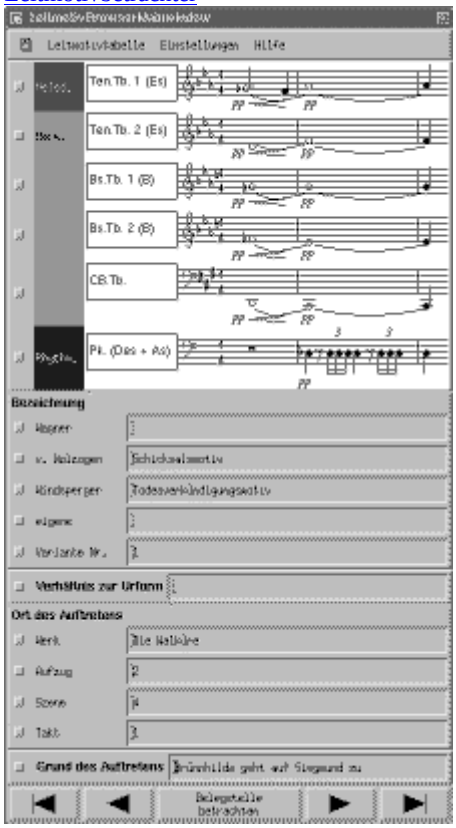
  - Zu markierende Leitmotivkomponente auswählen
  - Ausgewählte Leitmotivkomponente in der Partitur markieren
    - gemäß Systemvision: Das Ziehen nach rechts und links geschieht durch Klicken auf hervorgehobene Griffe auf dem Markierungsrahmen und anschliessendem Ziehen an die gewünschte Position
  - Markierungsvorgang beenden und Notiervorgang beginnen
    - gemäß Systemvision
  - Markierungsvorgang wiederholen
    - gemäß Systemvision
  - Markierung verwerfen und mit Leitmotivsuche fortfahren
    - gemäß Systemvision

- Leitmotivnotizeditor



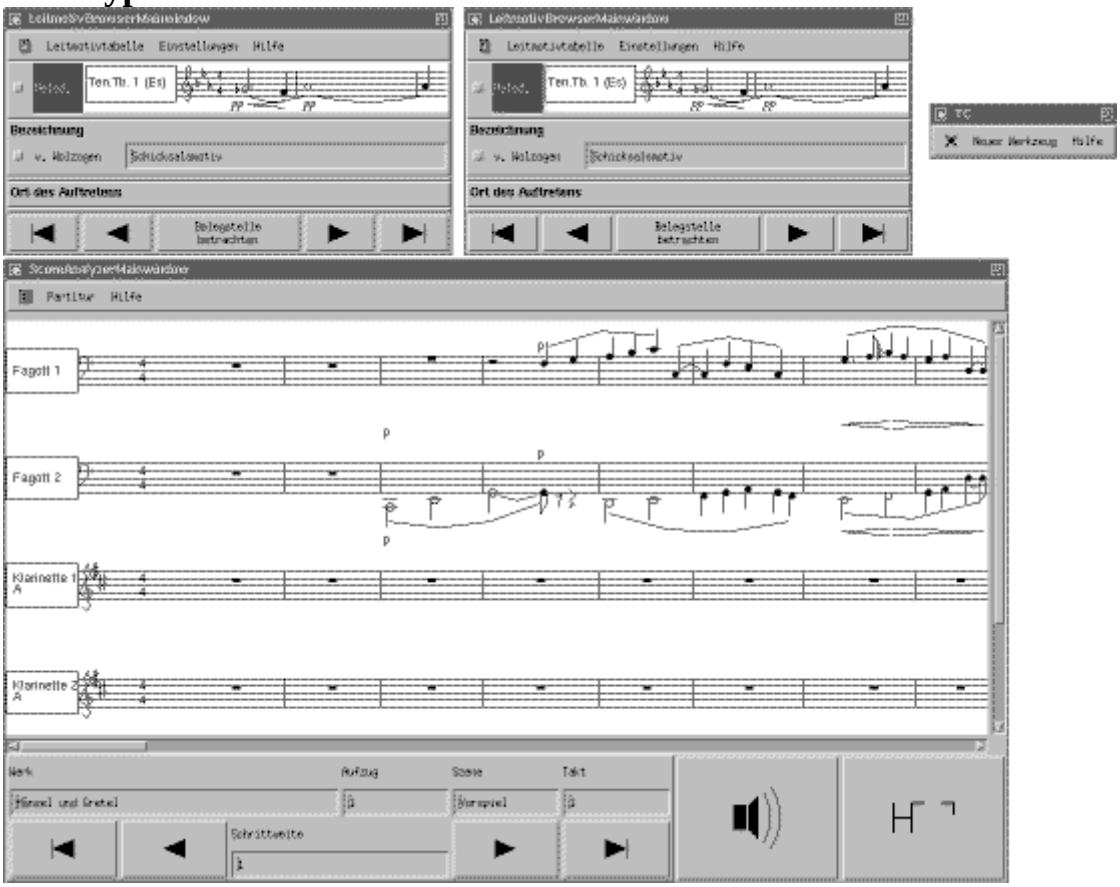
  - Leitmotivnotizbestandteile eintragen
    - gemäß Systemvision. Das bearbeitete Leitmotiv entspricht dabei nicht dem in der Partitur markierten Leitmotiv, sondern ist beim Prototyp stets das *Schicksalsmotiv* aus *Die Walküre*.
  - Notiervorgang beenden und Leitmotivnotiz in eigene Leitmotivtabelle übernehmen
    - gemäß Systemvision
  - Leitmotivnotiz verwerfen und auf Basis der Markierung neu erstellen
    - gemäß Systemvision
  - Leitmotivnotiz verwerfen und mit Leitmotivsuche fortfahren
    - gemäß Systemvision

- Leitmotivbetrachter



  - Zu betrachtende Leitmotivtabelle auswählen
    - nicht implementiert. Das angezeigte Leitmotiv ist stets das *Schicksalsmotiv* aus *Die Walküre*.
  - Anzuzeigende Leitmotivnotizbestandteile auswählen
    - gemäß Systemvision
  - Für Traversierung relevante Leitmotivnotizbestandteile auswählen
    - nicht implementiert
  - Durch Leitmotivtabelle traversieren
    - nicht implementiert
  - Leitmotivbelegstelle in der Partitur betrachten
    - nicht implementiert

Prototyp starten



Technische Voraussetzungen

1. Betriebssystem *SunOs*
2. in der persönlichen `.mailcap`-Datei muß sich die Zeile
   `application/octet-stream; sh %s`
   befinden, damit der WWW-Browser (*Mosaic*, *Netscape*,...) weiß, daß es sich bei dem Prototypen um eine ausführbare Datei handelt.

Benachbarte Dokumente

- Systemvisionen
  - *Leitmotivanalyse*
- Szenarios
  - *Leitmotivanalyse*
- Glossar

Übersicht | Glossar | Szenario–Übersicht
Systemvision–Übersicht