WHU-Forschungspapier Nr. 15 / November 1992

Wartung und Pflege von Wissensbanken

Privatdozent Dr. habil. Franz Lehner, Mag. Hubert F. Hofmann, Dr. Ralf Setzer
WHU Koblenz, Lehrstuhl für Wirtschaftsinformatik und Universität Zürich, Institut für Informatik

Wissenschaftliche Hochschule für Unternehmensführung Koblenz
Haus d'Ester - Heerstraße 52
5414 Vallendar
Telefon 02 61 / 65 09 - 0
Telefax 02 61 / 65 09 - 1 11

Wartung und Pflege von Wissensbanken

Franz Lehner, Hubert F. Hofmann, Ralf Setzer

WHU Koblenz, Lehrstuhl für Wirtschaftsinformatik
und
Universität Zürich, Institut für Informatik

Inhaltsübersicht

Vorwort

Im Zuge der Einführung neuer Softwaretechnologien am Markt und in den Unternehmen werden oft Fragen oder Folgewirkungen, die am Rande stehen oder die nicht mit der unmittelbaren Nutzung zu tun haben, vernachlässigt. Als Beispiele können die Wartung und die Dokumentation von Softwaresystemen, die mit objektorientierten Entwicklungsumgebungen oder mit Techniken der künstlichen Intelligenz erstellt wurden, genannt werden. Die vernachlässsigten Bereiche sind aber in der Regel von besonderer Bedeutung für den langfristigen Erfolg einer neuen Technologie. Es gilt daher, die Einführung neuer Softwaretechnologien bereits in ein entsprechendes Umfeld, z.B. hinsichtlich des methodischen Vorgehens in der Entwicklung oder der Qualifikation der beteiligten Softwareentwickler, einzubetten.

In diesem Sinn wird zur Zeit vom Verein Deutscher Ingenieure e.V. (VDI) die Richtlinie 5007 "Wissensbanken in der Anwendung" erstellt. Der nachstehende Text entspricht mit leichten Änderungen einem vorläufigen Vorschlag an die VDI für den Abschnitt 5 der genannten Richtlinie, der sich mit der Wartung von Wissensbanken befaßt. Nicht berücksichtigt sind daher die Grundlagen und die Begriffsabgrenzungen für den Themenbereich Wissensbanken. Diese werden, neben weiteren relevanten Themen, in anderen Abschnitten der Richtlinie 5007 behandelt.

Die Autoren des vorliegenden Beitrags planen auf Basis der bisherigen Arbeiten eine weitere Auseinandersetzung mit dem Thema "Wartung von wissensbasierten Systemen".

Vallendar/Zürich, im Dezember 1992

Franz Lehner Hubert F. Hofmann Ralf Setzer

1 Einführung

Die Trennung zwischen der **Wissensbank** und den übrigen Teilen eines **wissensbasierten Systems** bzw. eines **Softwaresystems** mit wissensbasierten Komponenten ist zwar konzeptionell und strukturell möglich, aus der Sicht der Wartung und Pflege (der Einfachheit wegen wird im folgenden nur mehr von Wartung gesprochen) aber wenig zweckmäßig. Dies gilt unabhängig davon, ob man den Begriff Wissensbank eng oder weit faßt. Die Repräsentation und die Speicherung von Wissen und Wissensstrukturen (Meta-Wissen) läßt sich nicht vom verwendeten Repräsentationsformalismus, von der Programmiersprache, von der Entwicklungsumgebung usw. trennen. Die Wartung ist u.a. gerade deswegen eine schwierige Aufgabe, weil diese Verbindungen berücksichtigt werden müssen und weil die Wartungsaktivitäten dadurch unmittelbar beeinflußt werden. Die Wartung von Wissensbanken wird daher in einem umfassenden Sinn verstanden und die Verbindungen zum gesamten Softwaresystem (z.B. Expertensystem) und zur Entwicklungsumgebung miteinbezogen. Damit wird auch der Tatsache Rechnung getragen, daß heute ein Trend in Richtung hybrider System zu beobachten ist, d.h. Softwaresysteme setzen sich aus Komponenten zusammen, die mit herkömmlicher Technologie und mit Hilfe der Expertensystem-Technologie entwickelt wurden. Ein weiteres Argument für die hier gewählte Perspektive ist die Tatsache, daß auch in den Methoden und Techniken der künstlichen Intelligenz viele Elemente der traditionellen Programmierung zu finden sind und umgekehrt. Erfahrungen aus der Wartung herkömmlicher Softwaresysteme können daher häufig übernommen werden.

Wartung ist kein notwendiges Übel und im allgemeinen auch kein Zeichen von Mängeln am Produkt oder bei seiner Entwicklung, sondern meist Folge einer sich ändernden Umwelt (Aufgabenumwelt, Technologie usw.). In der Wartung drückt sich die **Flexibilität** und die **Innovationsfähigkeit** eines Unternehmens aus. Die Übereinstimmung der Software mit der Umwelt muß erhalten bleiben und regelmäßig überprüft werden, wenn der Nutzen der Software gewährleistet bleiben soll. Wenn sich die Umwelt ändert, müssen auch die betroffenen Komponenten des Softwaresystems der neuen Situation angepasst werden. Die Änderungen können aber auch durch neue oder geänderte Anforderungen der Benutzer, durch Fehler in der Software, durch Perfor-

mance-Probleme (z.B. Antwortzeit) und viele andere Ursachen erforderlich werden. Insbesondere Fehler und schlechte Performance können den Einsatz der Software wesentlich behindern oder unmöglich machen und die Benutzer des Systems demotivieren. Die Software bedarf also einer ständigen Pflege und Weiterentwicklung, damit sie nicht an Nutzen verliert.

Als Besonderheit bei der Wartung von Wissensbanken ist die **konzeptuelle Modellierung** hervorzuheben. Die Entwicklung einer Wissensbank kann unter diesem Gesichtspunkt eigentlich nie als abgeschlossen betrachtet werden. Dies bedeutet letztlich, daß die (künstliche) Trennung in Phasen (z.B. Entwurfsphase, Entwicklungsphase und Einsatzphase) für die Wartung wenig nützlich ist. Die Wartung wird derzeit noch allgemein (sowohl bei herkömmlichen Softwaresystemen als auch bei wissensbasierten Systemen) als Teil der Einsatzphase im Lebenszyklus von Softwaresystemen gesehen. Dies ist eine Vereinfachung, die für eine genauere Untersuchung der Problematik keine hinreichende Grundlage bietet. Diese Vereinfachung impliziert, daß auch die Entwicklung wissensbasierter Komponenten unabhängig vom späteren Einsatz und den zu erwartenden Modifikationen der Software betrieben werden kann [vgl. z.B. Buchanan et al. 1983; Harmon & King 89]. Dedizierte Verfahren zum Entwurf wissensbasierter Systeme, z.B. KADS II (ESPRIT II Projekt 5248), haben die strikte Phasenorientierung durch iterative Entwicklungsmethoden, prototyping-orientiertes Vorgehen u.ä. teilweise zurückgenommen. Sie berücksichtigen die Wartung (z.B. Sicherstellung der Erweiterbarkeit im Systementwurf) aber ebenfalls nur ansatzweise.

Für die **Repräsentation des Wissens** in einer Wissensbank stehen unterschiedliche Möglichkeiten zur Verfügung. Am gebräuchlichsten ist die Verwendung von Regeln, Contraints, Objekten bzw. Frames. Das Wissen selbst kann in sehr unterschiedlichen Formen festgehalten werden [vgl. VDI-Richtlinie 5006 1992]: Faktenwissen, kausales Wissen, konzeptuelle Strukturen des Gegenstandsbereiches, Wissen in Regelform, Kontext und Erklärungen zu Regeln, prozedurales Wissen, Rechtfertigungen für die Regelanwendungen, Strategien, Gewichtungen, Evidenzen, Wahrscheinlichkeiten u.a.m. Die Inferenzkomponente dient der Verknüpfung von Wissenselementen, die in der Wissensbank verwaltet werden. Sie muß daher auf die Art und die Besonderheit der Wissensrepräsentation abgestimmt sein. Aus dieser Tatsache ergibt sich ein

weiteres Argument, warum die Wartung nicht auf die Anpassung der Wissensrepräsentation beschränkt werden kann.

Die **Integration der Wartung** in den Prozeß der Entwicklung von Wissensbanken ist u.a. deswegen so schwierig, weil oftmals die Adaptivität menschlicher Spezialisten nicht berücksichtigt wird und weil die Auswirkung wissensbasierter Komponenten auf die Arbeitssituation nur schwer vorhersehbar ist [vgl. z.B. Lamberts & Pfeifer 1992, Becker et al. 1992]. Die Benutzer sind jedoch "lernende Systeme" und unterziehen die Inhalte der Wissensbank einer laufenden Neuinterpretation.

Wartung kann daher als Prozeß verstanden werden, der den gesamten Lebenszyklus einer Wissensbank begleitet. Angestrebt wird eine weitgehende Annäherung der Wartungsaktivitäten an Aktivitäten der Neuentwicklung. Wenn es gelingt, die Wartung mit den gleichen Methoden, Techniken, und Werkzeugen zu betreiben wie die Neuentwicklung von Wissensbanken, dann ergeben sich wichtige Vorteile: Vereinheitlichung und Standardisierung, Vereinfachung der Aus- und Weiterbildung von Mitarbeitern, Synergieeffekte bei der Dokumentation, Einführung und Unterstützung.

Der **Stellenwert der Wartung** und die Anforderungen an ein Wartungsmanagement sind von vielen Kontextfaktoren abhängig. Wenn eine Wissensbank ausdrücklich als experimenteller Prototyp oder als Wegwerf-Prototyp konzipiert wird, kann die Wartung zunächst vernachlässigt werden. In allen anderen Fällen sollte jedoch versucht werden, die Konsequenzen möglichst frühzeitig abzuschätzen und die Wartung sicherzustellen. Die nachfolgenden Ausführungen gehen von der Idealvorstellung einer umfangreichen und intensiv genutzten Wissensbank aus, welche in einem Unternehmen mit einer großen Softwareabteilung entwickelt und eingesetzt wird. Die anschließend beschriebenen Aufgaben, Aktivitäten, Anforderungen, Probleme usw. sind dann entsprechend der unternehmensspezifischen Situation zu gewichten. Das Wartungsmanagement wird also bei einer kleinen Wissensbank oder bei einem explorativen Prototyp anders ausgeprägt sein als bei einem System, das langfristig in einer verteilten Umgebung eingesetzt werden soll und evolutionär weiter entwickelt wird.

1.1 Wartungsbegriff

In der Umgangssprache wird unter Wartung die Reinigung, die Pflege, die permanente Instandhaltung (Reparaturen) und die vorbeugende Instandhaltung (Verschleißvorbeugung) von Geräten und Maschinen verstanden. In der Kostenrechnung kennt man die Begriffe Reinigung und Instandhaltung. Die Auffassung von Wartung im Sinn des Austausches alter, abgenutzter Teile gilt jedoch nicht für Software. Hier besteht die Wartung in der Beseitigung von Fehlern und in der Anpassung an Änderungen der Umwelt oder der Benutzeranforderungen bezüglich Funktionen oder Leistungen.

In den meisten Definitionen und Definitionsversuchen zur Wartung von Softwaresystemen werden sowohl die **Funktionserhaltung** als auch die **Funktionserweiterung** angesprochen. In der Literatur wurde ein Wartungsverständnis, das sowohl Erweiterungen als auch Fehlerreparaturen miteinschließt, mehrfach kritisiert. Für manche Autoren ist die unzureichende Trennung zwischen Funktionserweiterung und Funktionserhaltung ein wesentlicher Grund für viele bisher ungeklärte Fragen im Zusammenhang mit der Wartung. Ohne auf diese Diskussion hier näher einzugehen, werden bei der Funktionserhaltung folgende Wartungsarten unterschieden [vgl. Lehner 1991]:

- **korrigierende Wartung**: hier handelt es sich um die eigentliche Notfallwartung, d.h. um die Korrekturen zunächst unerkannter Fehler, die beim Einsatz des Softwaresystems auftreten.
- **adaptierende Wartung**: Anpassung des Softwaresystems an eine veränderte Umwelt wie z.B. Betriebssystemänderungen, neue Hardware oder geänderte gesetzliche Bestimmungen.
- **perfektionierende Wartung**: Verbesserung der Leistungen im Sinne einer Performanceverbesserung, aber auch Restrukturierung des Systems mit dem Ziel, den zukünftigen Wartungsaufwand zu reduzieren.

Neben den bereits genannten Wartungsarten werden im vorliegenden Fall noch die Funktionserweiterung und die Unterstützung von Benutzern zur Wartung gerechnet.

- Die **Funktionserweiterung** umfaßt den Einbau zusätzlicher Funktionen, die ursprünglich nicht vorgesehen waren sowie die

Integration mit benachbarten, neu hinzugekommenen oder geänderten Systemen.

- Mit **Unterstützung** sind Schulungsmaßnahmen für Benutzer, Hilfestellung bei Bedienungsproblemen, Klärung von Fehlersituationen, Performancemessungen, Abstimmung der Wartungsaktivitäten usw. gemeint.

Als Zeitpunkt der Abgrenzung zwischen Entwicklung und Wartung gilt die Übernahme des gesamten Softwaresystems oder eines Teilsystems in den produktiven Betrieb. Art oder Umfang der Änderung, Weiterentwicklung usw. spielen für die Einordnung in die Kategorie Wartung keine Rolle. Voraussetzung ist allerdings, daß trotz der geplanten Änderung die ursprüngliche Zielsetzung und der unterstützte Aufgabenbereich des Softwaresystems im wesentlichen unverändert bleiben. Auch wenn in Verbindung mit Wissensbanken manchmal diskutiert wird, bestimmte Wartungsaktivitäten zum Benutzer zu verlagern (z.B. Pflege der Wissensinhalte), soll Wartung hier eng gefaßt und auf jene Tätigkeiten beschränkt werden, welche durch Softwarespezialisten ausgeführt werden (z.B. strukturelle Änderungen in der Wissensbank).

1.2 Wartung und Wartbarkeit von Wissensbanken

Die Wartung einer Wissensbank umfaßt die notwendigen Aktivitäten zur Erhaltung oder Wiederherstellung der Leistungsfähigkeit dieser Komponente eines wissensbasierten Systems. Die Leistungsfähigkeit bezeichnet ihren qualitativen und quantitativen Beitrag zur Erfüllung einer bestimmten Tätigkeit im Informations- und Kommunikationsprozeß.

Die Wartung der Wissensbank beinhaltet u.a. eine permanente Aktualisierung des gespeicherten Wissens sowie die Revision der Wissensstruktur. Zur Wartung zählen aber auch die (Nach-)Dokumentation, die (Re-)Strukturierung des Softwaresystems, die Sanierung und die kontrollierte Weiterentwicklung aller Komponenten, die mit der Wissensbank direkt oder indirekt in Verbindung stehen. Die Verbindungen sind im wesentlichen durch die verwendete Programmiersprache bzw. die Entwicklungsumgebung geprägt. Änderungen betreffen zwar möglicherweise nur die Struktur des abgebildeten Wissens, haben jedoch Nebeneffekte in anderen Bereichen (z.B. prozedurale Abläufe, Benutzeroberfläche). Aufgrund dieser engen Verquickung wird in den nachfol-

genden Ausführungen das Wartungsproblem nicht auf die Wissensrepräsentation reduziert, sondern der Kontext der Wartung umfaßt auch die Programmierung und die Systemarchitektur.

Die Begriffe **Wartung** und **Wartbarkeit** stehen in einer engen Verbindung miteinander. Wartung bezeichnet jene Maßnahmen, mit denen ein betriebsbereiter Zustand erhalten oder hergestellt wird. Wartbarkeit bezieht sich auf alle Eigenschaften, die eine effektive und effiziente Wartung unterstützen. Die Wissensbank wird also, bezogen auf Wartung und Wartbarkeit, aus einem unterschiedlichen Blickwinkel betrachtet. Dieser Unterschied in der Perspektive ist ein wichtiger Faktor im Dialog zwischen dem Entwicklungs- und Wartungspersonal einerseits, sowie dem Wartungspersonal und den Benutzern andererseits.

Der Zusammenhang zwischen Wartung und Wartbarkeit kann auch über **Qualitätseigenschaften** hergestellt werden. Unter dem Begriff Wartbarkeit werden jene Eigenschaften zusammengefaßt, welche für die Benutzbarkeit, für den wirtschaftlichen Einsatz und für die Lebensdauer eines Systems von Bedeutung sind. Software-Qualitätssicherung zielt - mehr oder minder direkt - immer auch auf die Sicherstellung der Wartbarkeit ab. Ein allgemein anerkanntes System solcher Qualitätsmerkmale für die Wartbarkeit existiert bisher weder für Software generell noch für spezielle Ausprägungen (z.B. Wissensbanken). Die einschlägige Literatur beinhaltet lediglich eine mehr oder minder genau erläuterte Sammlung qualitativer Merkmale (z.B. Erweiterbarkeit, Anpaßbarkeit, Korrigierbarkeit), deren Relevanz jedoch selten empirisch nachgewiesen wurde. Die Ursache dafür ist, daß viele Software-Qualitätsmerkmale - im Gegensatz zur Hardware - direkten Meßverfahren nicht zugänglich sind. Trotzdem ist eine Präzisierung erforderlich. Erst die Operationalisierung von Qualitätsmerkmalen stellt die Grundlage für die Beschreibung und Beurteilung der Qualität dar.

Besonders wichtig im Zusammenhang mit der Wartbarkeit sind die Auswirkungen von **Methoden und Programmiertechniken.** Die Ergebnisse empirischer Untersuchungen dazu sind widersprüchlich und die Auswirkungen auf die Wartung bzw. Wartbarkeit nicht endgültig geklärt [vgl. Lehner 1991]. Trotzdem besteht heute Einigkeit in der Auffassung, daß bestimmte Strukturen und Anforderungen an ein fertiges Softwareprodukt beachtet werden sollten. Empfohlen werden u.a. die Beachtung der Merkmale Modularität, Programm-

struktur, Datenstruktur, Softwarearchitektur, Dokumentation und Parametrisierung. Die Merkmale Programmstruktur und Softwarearchitektur werden nachfolgend unter Wartungsgesichtspunkten noch etwas näher erläutert.

Programme werden nicht nur für den Computer sondern auch für den Menschen entwickelt. Aufgabe der **Programmstruktur** ist das Zusammenspiel der einzelnen Programmteile. Im Detail geht es um die Art der Kontrollübergabe zwischen Moduln ("Aufruf"), um die Art des Informationsaustausches (z.B. globale Variable oder Parameterübergabe), um die Reihenfolge der Befehlsausführung und um die Art der Fehler- und Ausnahmenbehandlung. Lesbarkeit, Testbarkeit und Komplexität von Programmen werden durch die Wahl der Sprachkonstrukte und die Gestaltung der Kommunikation zwischen Moduln stark beeinflußt. Meist besteht aber nur begrenzter Spielraum, da sich die Auswahlmöglichkeiten auf die in einer Programmiersprache verfügbaren Sprachkonstrukte beschränken. Qualitative Aussagen wie "niedrige Komplexität", "verständliche Struktur", "klarer Programmierstil" u.ä. drücken aus, daß Menschen transparente, lineare und einfache Programmabläufe vorziehen bzw. besser verstehen können. Entscheidend dafür ist ausschließlich die Art der Anordnung der drei elementaren Strukturelemente *Sequenz, Iteration und Selektion*. Sie trägt dazu bei, ob ein Programm wartbar ist oder wie der oft zitierte "Spaghetticode" aussieht. Techniken wie strukturierte, normierte und modulare Programmierung, Verwendung von Namenskonventionen sowie Top-Down-Entwicklung und schrittweise Verfeinerung unterstützen die Erreichung dieses Ziels.

Die genannten Prinzipien der Programmstrukturierung gelten ganz generell für Softwaresysteme. Dabei darf aber nicht übersehen werden, daß sie im Kontext der jeweiligen Programmiersprache u.U. neu interpretiert werden müssen. Es gibt jedoch auch einige Besonderheiten, aus denen sich konkrete Empfehlungen ableiten lassen. Neben "stilistischen Implikationen" für Sprachen der fünften Generation betrifft dies insbesondere die Wahl der Sprache. Als besonders wichtige Eigenschaft wird dabei in Verbindung mit der Wartung die **Portabilität** angesehen. Eine Möglichkeit, Portabilität sicherzustellen, besteht in der Verwendung einer Zwischensprache, welche erst bei der Auslieferung des Systems in die gewünschte Zielsprache übersetzt wird. Das Problem wurde jedoch von den Systemherstellern allgemein erkannt, sodaß heute viele Entwicklungsumgebungen oder Sprachen selbst in C (z.B. ART,

M1/S1), Common LISP (z.B. KEE) oder sonstigen verbreiteten Sprachen implementiert werden, für die wiederum eine breite Hardware- und Systemsoftwarepalette zur Verfügung steht.

Mit **Softwarearchitektur** ist die Gliederung der Software, oder genauer die Zuordnung der Funktionen zu einzelnen Programmen gemeint. Nicht jede Funktion wird genau einem Programm oder Modul zugeordnet; dies würde eine hohe Modulbindung bewirken. Über die Zusammenfassung von Funktionen entscheidet die Zweckmäßigkeit. Beispielsweise kann eine Zusammenfassung logisch ähnlicher Funktionen sinnvoll sein, da ähnliche Funktionen üblicherweise ähnliche Basisfunktionen oder Basisdaten brauchen und daher sehr kompakt an einer Stelle implementiert werden können. Auch die Lokalisierbarkeit von Fehlern oder Änderungen wird erleichtert, wenn man logisch zusammengefaßte Gruppierungen vorfindet.

Die eigentliche Schwierigkeit besteht darin, die Strukturierung so vorzunehmen, daß zukünftige Adaptionen und Erweiterungen eindeutig bestimmbaren und bestimmten Teilen zugeordnet werden können. Die Erreichung dieser Ziele, die auf die Modularität und auf die Flexibilität einen wesentlichen Einfluß haben, kann durch Konzepte wie Datenabstraktion, strukturierte Analyse, Schichtenarchitektur, Information-Hiding u.ä. unterstützt werden. Für eine erfolgreiche Umsetzung sind zwei Voraussetzungen zu beachten: Funktionen mit einer hohen Änderungswahrscheinlichkeit müssen bekannt oder identifizierbar sein, und es muß möglich sein, diese Funktionen softwaretechnisch in einem Modul zusammenzufassen. In den meisten modernen Programmiersprachen wird die technische Umsetzung durch ein Modul-Konzept unterstützt. Schwieriger ist dagegen die Erfüllung der ersten Forderung. Bei konsequenter Beachtung kann damit jedoch eine wesentlich verbesserte Wartbarkeit erzielt werden. Eine Hilfe ist dabei die Konzentration auf die Verbindung zur Umwelt der Software sowie auf die modellmäßige Abbildung dieser Umwelt (z.B. Datenstrukturen, Schnittstellen zu externen Geräten oder anderen Anwendungen, Auswertungen von Datenbeständen und Zugriffsberechtigungen einzelner Benutzer). Da diese Komponenten von einer Änderung dieser Umwelt unmittelbar betroffen sind, sind sie auch häufig Gegenstand von Wartungsarbeiten. Voraussetzung für vorbeugende Maßnahmen ist die Identifikation des hier aufgezeigten Änderungspotentials. Eine Verminderung des Wartungsumfangs kann dann durch Techniken wie Tabellensteuerung oder Para-

metrisierung erreicht werden. Trotz der Vielfalt der Möglichkeiten zur Reduzierung der Wartung ist es meist technisch nicht machbar und wirtschaftlich nicht sinnvoll, jegliche Wartung vermeiden zu wollen. Die aufgezählten Maßnahmen haben vor allem eine Verminderung der Wartungsnotwendigkeit und des Arbeitsaufwands bei der Durchführung der Wartung zum Ziel.

Eine wichtige Möglichkeit, den Wartungsaufwand zu reduzieren, ist die **vorbeugende Wartung** (präventive Wartung). Dies bedeutet, daß aufgrund der Wahrscheinlichkeit, daß eine Wartungsaktivität erforderlich wird, vorbeugende Maßnahmen getroffen werden (z.B. Verhinderung des Wartungsfalls). Grundsätzlich können folgende Arten von Maßnahmen, die zur Prävention der Wartung beitragen können, unterschieden werden:

- *Softwaretechnische Maßnahmen*: Methodeneinsatz, Programmiertechniken, Programmiersprachen usw.;
- *Technische Maßnahmen*: Einsatz von Hilfsmitteln und Werkzeugen;
- *Organisatorische Maßnahmen*: Einführung von Standards, Wartungsorganisation, aber auch Reengineering;
- *Kontrollierende Maßnahmen*: Durchführung von analytischen Maßnahmen zur Software-Qualitätssicherung wie Audits oder Reviews.

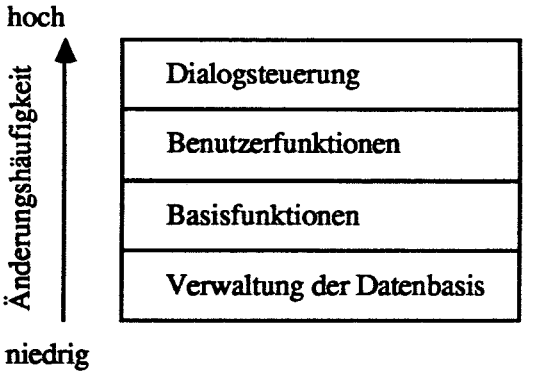

Abb. 1.2-1: Schichtenarchitektur

Ziel und Wirkung präventiver Wartungsmaßnahmen soll am Beispiel einer softwaretechnischen Maßnahme, nämlich der Schichtenarchitektur, etwas näher dargestellt werden. Die **Schichtenarchitektur** muß bereits in der Entwurfsphase in Form von Abstraktionsschichten vorbereitet werden, damit bei späteren Änderungen Funktionen möglichst ohne Seiteneffekte hinzugefügt oder weggenommen werden können. Abbildung 1.2-1 zeigt ein Beispiel für eine derartige Schichtenarchitektur. Die unterste Schicht stellt die Grundoperatio-

nen zur Datenverwaltung zur Verfügung und kann durch ein Datenbankverwaltungssystem realisiert werden. Die nächsthöhere Schicht umfaßt die anwendungsspezifischen Basisfunktionen (z.B. Datenaufbereitung in einer bestimmten Form und Reihenfolge). Die Benutzerfunktion enthält die eigentliche anwendungsspezifische Verarbeitung der Daten. Die Präsentation der Ergebnisse, die Verbindungen zwischen Benutzerfunktionen, die Steuerung des Dialogablaufs usw. erfolgen schließlich durch die Dialogsteuerung.

Weitere Maßnahmen mit präventivem Charakter sind der Einsatz von Verfahren zur Datensicherung, Plausibilitätskontrollen bei der Dateneingabe, Wiederanlauf-Verfahren und Checkpoint-Restart-Prozeduren. Eine Verminderung der Wartung kann aber auch durch mehr Redundanz (z.B. Ausfallssicherheit durch mehrfache Verfügbarkeit von Funktionen, Konzept der Datenspiegelung) oder durch Erhöhung des Testabdeckungsgrads (Reduktion von Fehlern) erreicht werden.

Die Sicherung gegen Verfahrensfehler aufgrund mangelhafter Systemplanung oder Programmierung ist nahezu unmöglich. Die Erfahrung mit vorbeugender Wartung auf anderen Gebieten unterstreicht diese Ansicht. So wurde z.B. im Bereich der Luftfahrt einige Zeit versucht, durch eine total vorbeugende Wartung jede Art von ungeplanter Wartung zu vermeiden. Anhand der empirisch ausgewerteten Ergebnisse wurde jedoch festgestellt, daß die vorbeugende Wartung in 68% aller Fälle die Zuverlässigkeit der Komponenten eher reduziert als verbessert hat, bei 14% war sie ineffektiv und bei 11% unwirtschaftlich [vgl. Lehner 1991]. Vorbeugende Wartung bei Software sollte daher hauptsächlich als Bereitstellung von Hilfsmitteln für den Wartungsfall verstanden werden. In der Fachliteratur hat sich für diese Vorgehensweise der Begriff ***defensive Programmierung*** durchgesetzt. Techniken der defensiven Programmierung sind u.a. Assertionen, Reservierung von Ressourcen (z.B. Speicherplatz), die Verwendung von Statusfeldern über den Systemzustand und Trace-Aufzeichnungen. Eine andere Möglichkeit ist die Eigenschaft der ***Selbst-Prüfung*** von Programmen. Dazu müssen alle Bedingungen, die zu einem abnormalen Ende der Programmausführung führen können (z.B. Tabellenüberlauf, Division durch Null, Stromausfall, zu wenig externer Speicherplatz) überlegt und als Teil der Programmlogik berücksichtigt werden.

Im Umfeld der vorbeugenden Wartung sind auch die Eigenschaften der **Adaptivität** (adaptivity) und der **Adaptierbarkeit** (adaptability) zu sehen [vgl.

Oppermann 1992]. Aufgrund der gewählten Begriffe könnte allerdings leicht ein falscher Bezug zur Wartung hergestellt werden. Gemeint sind **Eigenschaften der Benutzeroberfläche**, welche eine Anpassung des Systems an das Verhalten des Benutzers unterstützen. Um diese Anpassungsleistung grundsätzlich erbringen zu können, benötigen adaptive Systeme Wissen über die unterstützte Aufgabe, über den Benutzer und über das System selbst. Ein Teil dieses Wissens kann allerdings nur im interaktiven Betrieb, d.h. erst während der Systemnutzung erworben werden. Je nachdem, ob der Mensch oder das System die Initiative ergreift, bezeichnet man das System als adaptierbar bzw. als. adaptiv. Die Effektivität solcher Systeme, Probleme der Irritation durch Vorschläge seitens des Systems sowie andere Fragen sind allerdings bisher noch kaum untersucht worden.

Die Forschungsarbeiten auf dem Gebiet adaptiver und adaptierbarer Systeme werden im deutschsprachigen Raum vor allem bei der GMD in Bonn vorangetrieben (Projekt "Assistenzcomputer"). Die bisherigen Ergebnisse führten dazu, daß einige der angestrebten **Assistenzeigenschaften** bereits exemplarisch demonstriert werden können (z.B. FLEXEL als Adaption von EXCEL). Die grundlegenden Forschungen konzentrieren sich auf die Suche nach neuen Formen der Arbeitsteilung zwischen Mensch und Maschine. Die Ausstattung mit Unterstützungssystemen (sogenannte Assistenzcomputer) soll durch Wissen über Aufgabenzusammenhänge, Benutzer und Systeme dem Menschen eine bessere Unterstützung bei der Aufgabenausführung bieten.

1.3 Wartungsmanagement

Das Wartungsmanagement (üblich sind auch die Bezeichnungen Change-Management, Request-Management u.a.) umfaßt alle Aufgaben, welche die systematische Planung, Überwachung und Steuerung aller Wartungsaktivitäten zum Ziel haben. Dazu zählen u.a. [vgl. Lehner 1991]:

- Wahl der Strukturorganisationsform und Einsetzen eines Wartungsausschusses,
- Bildung von Änderungsklassen,
- Datenaufzeichnung und Berichtswesen,
- Festlegen der Wartungsstrategie,
- Wartungsplanung (z.B. Kosten, Termine, Personal),

- Wirtschaftlichkeitsprüfung (z.B. Soll-/Ist-Vergleiche, Aufwandsschätzung für Wartungsprojekte, Ablöseentscheidungen),
- vertragliche Absicherung,
- Konfigurations- und Versionsmanagement,
- Qualitätssicherung (z.B. Reviews, Werkzeugeinsatz),
- Personal und Führung (z.B. Wartungsproduktivität, Personalentwicklung, Motivation)
- formelle Abnahme bei einem Release-Wechsel und beim Austausch einzelner Systemkomponenten.

Voraussetzung für ein funktionierendes Wartungsmanagement ist die **Formalisierung des Wartungsablaufs**, welche die einheitliche und effiziente Behandlung der Änderungsanträge und Wartungsvorhaben gewährleisten. Dieses Verfahren wird meistens durch ein Formular unterstützt. Wegen seines integrativen Charakters für den gesamten Wartungsprozeß kommt diesem Formular ein besonderer Stellenwert zu. Es darf keine Änderung geben, die nicht nach dem festgelegten Verfahren abgewickelt wird, egal wie unbedeutend die Änderung erscheinen mag. Damit soll einerseits sichergestellt werden, daß bei der Durchführung von Änderungen mit entsprechender Sorgfalt vorgegangen wird und andererseits, daß die Änderungen dokumentiert und mit anderen Anwendungen oder Unternehmensbereichen abgestimmt werden. Es besteht allerdings die Gefahr, daß der Änderungsprozeß äußerst formalistisch gehandhabt wird und in Verbindung mit einer zeitaufwendigen Entscheidungsbildung (Kompetenzstreitigkeiten, Interessenswahrung usw.) zu Verzögerungen führt. Ziel ist es also, den Ablauf so zu gestalten, daß der Entscheidungsprozeß im Einzelfall möglichst schnell abgeschlossen werden kann. Abbildung 1.3-1 zeigt einen solchen Wartungsablauf in Form eines integrierten Verfahrensmodells.

Die Erkennung von Fehlern, Problemen oder Änderungspotentialen kann durch verschiedene Personengruppen erfolgen. Das Bedienungspersonal (z.B. Operator, Benutzer) kann Fehler in der Software selbst (z.B. bei Fehlermeldungen oder bei abnormalem Systemverhalten) und durch die Analyse des dynamischen Verhaltens erkennen. Letzteres stellt eine agierende Aufgabe dar, bei der aktiv nach potentiellen Verbesserungsmöglichkeiten gesucht wird (z.B. durch Beobachten des Antwortzeitverhaltens, Streßtests, Auswertung von Log-Dateien). Seitens der Systemprogrammierung werden Anforderungen aufgrund von Änderungen der Hardware und der Systemsoftware formuliert. Das Management erstellt Vorgaben bezüglich der Funktionalität der Software zur

Sicherung wettbewerbsrelevanter Systemeigenschaften. Zur Problemerkennung gehört aber auch das Ermitteln der Daten, mit denen die Existenz eines Problems nachgewiesen werden kann. Damit soll ausgedrückt werden, daß es oft notwendig ist, unklare Problem- und Fehlerbeschreibungen zu präzisieren oder zu validieren. Die Behandlung des Fehlerberichts, der Problemmeldung oder des Änderungswunsches erfolgt entweder durch den Wartungsausschuß oder einen Spezialisten, der damit beauftragt wird.

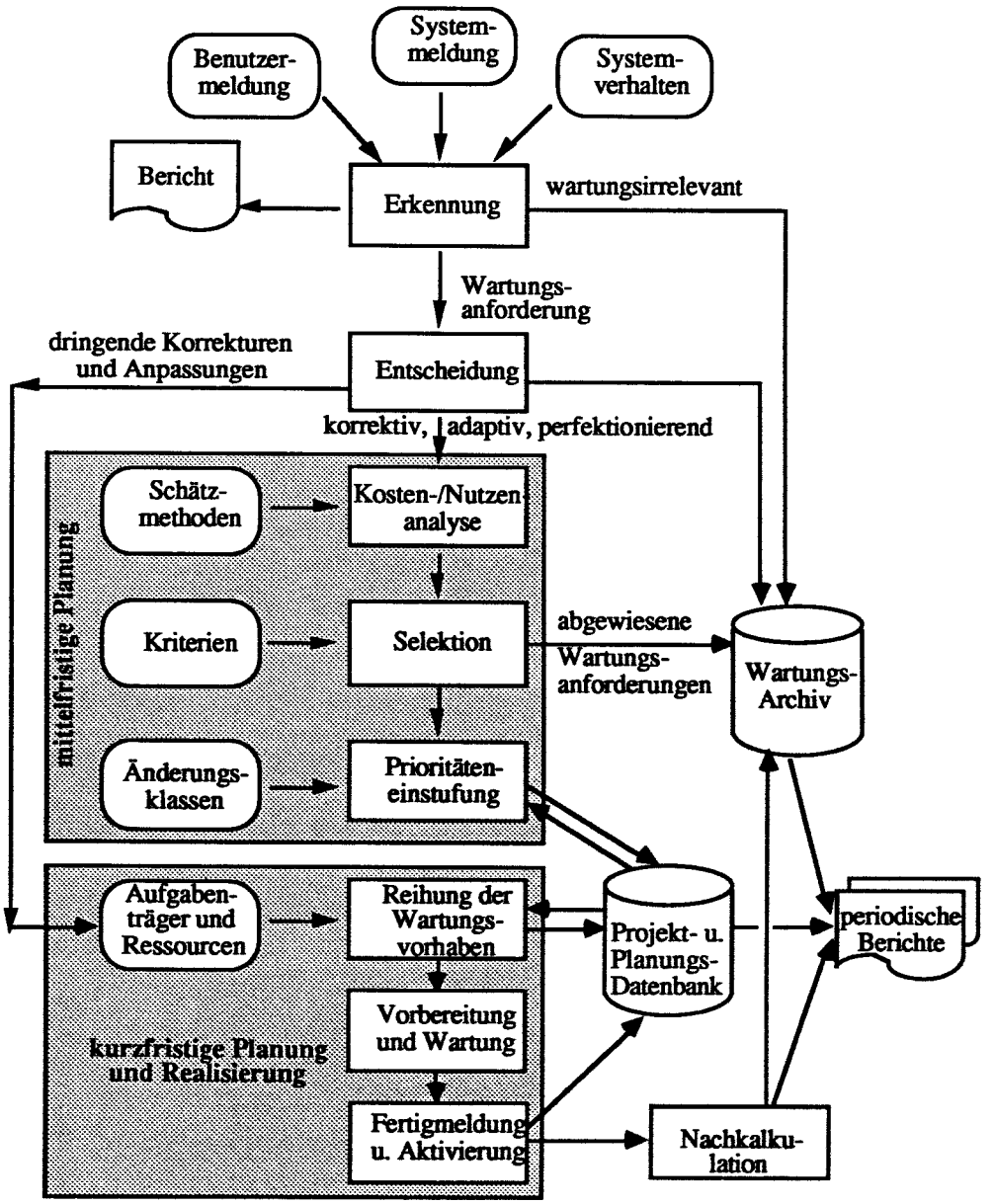

Abb. 1.3-1: Integriertes Verfahrensmodell für den Wartungsablauf

Aufgabe des **Konfigurations- und Versionenmanagements** ist es, Ordnung in die Vielfalt der Zustände von Teilsystemen zu bringen und Bezugspunkte für Änderungsprozesse (z.B. Releasewechsel) zu definieren. Eine wichtige Frage dabei ist, welche Teile miteinander kombiniert werden können, sodaß die volle Funktionsfähigkeit der Software gewährleistet ist. Um dies zu ermöglichen ist im Rahmen des Konfigurationsmanagements ein Register mit allen beantragten und genehmigten Änderungen zu führen sowie der Stand der Änderungen einschließlich ihrer Einführung in den laufenden Betrieb aufzuzeichnen.

Eine besondere Aufgabe im Rahmen des Wartungsmanagements ist die Bildung von **Wartungsklassen**. Es gibt mehrere Möglichkeiten der Klassenbildung, die innerhalb des Wartungsmanagements wiederum unterschiedlichen Zwecken dienen. Die Gemeinsamkeit besteht im Ziel der bewußten Steuerung der Wartungsaktivitäten [vgl. Lehner 1991]. Folgende Klassifikationsschemata werden nachfolgend näher erläutert:

- Differenzierung nach Wartungsarten,
- Bildung von Prioritätsklassen,
- Bildung von Benutzerklassen.

Bei der **Differenzierung nach Wartungsarten** geht man im allgemeinen von einem der bekannten Klassifikationsschemata aus. Die Klassenbildung ist ein wichtiger Prozeß, wobei die Folgen oft erst später sichtbar werden. In welcher Klasse und Kategorie eine Änderung eingeordnet wird, kann wesentlich davon abhängen, ob man beim Entwurf bereits an ihre Möglichkeit gedacht hat. Dieses wird dann unter Bezugnahme auf die konkrete Situation und Zielsetzung angepaßt. Häufig geht es dabei um Datenaufzeichnungen für Managementzwecke (z.B. Aufzeichnungen für Fehlerberichte). Geht man von den drei klassischen Wartungsarten aus, so ist zwischen korrigierender, adaptierender und perfektionierender Wartung zu unterscheiden. Während die erstgenannten Wartungsanforderungen unverzüglich durchzuführen sind, erlauben die beiden anderen ein geplantes Vorgehen. Beispiele für weitere Klassifikationsmerkmale, die von der konkreten Umgebung und anderen Faktoren abhängen, sind die Tätigkeitsart (z.B. Analyse, Programmierung, Test, Dokumentation) sowie die Zuordnung zu einer Kostenstelle oder einem Aufgabengebiet.

Das **Bildung von Prioritätsklassen** ist notwendig, um bei beschränkten Ressourcen eine möglichst effiziente und geordnete Abwicklung der anstehenden Wartungsaufträge zu gewährleisten. Sie ist ein wichtiges Instrument für Entscheidungen des Wartungsausschusses. Die unterschiedliche Behandlung von Änderungsanforderungen soll vor allem die Durchlaufzeiten für Wartungsaufträge beschleunigen. Prioritätsklassen können aber auch anderen Zielen dienen, z.B. der statistischen Auswertung oder der Überwachung der Änderungsaktivitäten. Die Klasseneinteilung darf jedoch nicht dazu führen, daß z.B. langfristige Änderungen wegen Überlastung u.U. nicht zum Zug kommen. Merkmale für die Bildung von Prioritätsklassen sind u.a. die Komplexität der Änderung, die Möglichkeit einer einfachen Rücknahme, die potentiellen Auswirkungen und die Dringlichkeit für den laufenden Betrieb [vgl. Lehner 1991].

Eine andere Einteilung ergibt sich, wenn man die **Bildung von Benutzerklassen** als Grundlage nimmt. Bei dieser Einteilung, die bei herkömmlichen Softwaresystemen nicht üblich ist, die also eine Besonderheit von wissensbasierten Systemen darstellt, lassen sich folgende Klassen unterscheiden [vgl. Thuy & Schnupp 1989, 247]:

- Wissen, das ohne Änderungen der Struktur oder des logischen Konzepts integriert werden kann,
- Wissen, das Änderungen in der Systemstruktur oder der Implementierung erfordert (z.B. neue Inferenzstrategie, externe statt hauptspeicherresidente Datenhaltung),
- Wissensinhalte, die so stark vom gewählten Rahmen abweichen, daß ein völlig neues Systemkonzept erstellt werden muß.

Aufbauend auf diese Klassifikation können Benutzerklassen mit entsprechenden Rechten definiert werden. Eventuell müssen die Rechte zur Änderung von Konzepten innerhalb der Benutzerklasse noch weiter differenziert werden (z.B. nach Anwendungsgebieten oder nach Änderungsfunktionen, also Hinzufügen neuer Konzepte, Ändern bestehender Konzepte u.a.). Außerdem muß wegen der vielfältigen Abhängigkeiten zwischen den Konzepten (Klassen u.a.m) für Abstimmungsmechanismen gesorgt werden.

Änderungen auf der Faktenebene bestehen aus dem Hinzufügen, Löschen oder Modifizieren von Wissen (z.B. Fakten, Instanzen von Schemata). Solche Ände-

rungen können bei einfachen Wissensbanken durch den Benutzer nach einer entprechenden Unterweisung im Gebrauch von Update-Funktionen vorgenommen werden. Wenn eine gefundene Inkonsistenz nicht auf Fehlern bei der Instantiierung von Konzepten beruht, erfordert ihre Behebung genaue Kenntnisse sowohl der Wissensbank selbst als auch der Repräsentationsformalismen und Schlußfolgerungsmechanismen. Die Änderung kann daher im allgemeinen nicht mehr vom Benutzer vorgenommen werden und erfordert die Hilfe von Personen mit entsprechendem Wissen einschließlich der formalen Berechtigung zur Änderung der Wissensbank.

Die Änderung von Konzepten (z.B. Klassen mit ihren Eigenschaften, Regeln und Constraints) erfordert neben dem notwendigen Anwendungswissen auch Erfahrung im Modellieren von Wissen. Daher sollten diese Änderungen einem eingeschränkten Kreis von Personen vorbehalten sein. Diese Personen müssen profunde Kenntnisse des Sachgebiets und der darin verwandten Terminologie sowie Kenntnisse über die Wissensrepräsentation und die Schlußfolgerungsmechanismen besitzen.

Änderungen an der Funktionalität der Wissensbank (z.B. Regelinterpreter, interne Repräsentation von Schemata) sollten dem Systemadministrator (Knowledge Engineer) vorbehalten sein. Zu dieser Klasse gehören auch Strategiekomponenten zur Regelauswahl und spezielle Schlußfolgerungskomponenten oder Diagnosefunktionen. Der Systemadministrator muß neben den erforderlichen Systemkenntnissen über gute Kenntnisse auf dem Gebiet der Wissensrepräsentation und der Schlußfolgerungsmechanismen verfügen.

Den Abschluß zum Wartungsmanagement bilden zwei Hinweise, deren Beachtung den Aufwand und den Erfolg erheblich beeinflussen können:

- Trotz der Vielzahl an möglichen Problemen, Aufgaben und Lösungen bei der Wartung stellt das **Personal** den eigentlichen **Engpaß** dar. Es ist außerordentlich schwierig, Programmierer zu finden, welche bei Sprachen der fünften Generation Erfahrung und Kompetenz besitzen, und diese dann noch für Wartungsaufgaben zu motivieren.

- Die Wartung darf nicht zum Selbstzweck werden. Manche Wartungsanforderungen (z.B. adaptierende und perfektionierende Wartung) können hinsichtlich des Umfangs und des Zeitpunktes ihrer Durchführung beein-

flußt werden. Die **Relation** zwischen **Kosten und Nutzen** sollte als Steuerungsgröße verwendet werden.

1.4 Fehlerquellen, Änderungsanforderungen und Probleme

Eine systematische Gliederung wichtiger oder typischer Fehlerquellen, Änderungsanforderungen und Probleme ist aus heutiger Sicht für Wissensbanken noch nicht möglich. Die nachfolgende Darstellung hat daher exemplarischen Charakter und soll einen Eindruck von der Komplexität der Wartung einer Wissensbank vermitteln. Potentielle Fehlerquellen in Wissensbanken sind in der Regel auch kaum zu überschauen und im voraus nicht konkret oder vollständig zu beschreiben. Dies zeigt sich unter anderem daran, daß mit dem Testen, der gebräuchlichsten Form der Fehlerfindung, meist nicht mehr als 50 Prozent der vorhandenen Fehler in einem System gefunden werden [vgl. Jones 1983]. Die nachstehenden Darstellungen, welche teils aus den Erfahrungen in eigenen Entwicklungsprojekten und teils aus der Literatur abgeleitet wurden, können daher nur einen Überblick in Form von exemplarischer Fehlerquellen, Änderungsanforderungen und Problemen geben. Manche der genannten Punkte können bei der Wartung von Softwaresystemen generell beobachtet werden, manche sind spezifisch für Wissensbanken. Sie werden nachfolgend gemeinsam beschrieben, weil eine klare Trennung meist nicht möglich und auch nicht sinnvoll ist.

Exemplarische Fehlerquellen und Änderungsanforderungen beschreiben wir zunächst anhand von drei Beispielen aus der Entwicklung von IAS (Investment Advisory System). IAS ist ein kooperatives Anlageberatungs-System, das von einem der Autoren entwickelt wurde und das die anlage-orientierten Aufgaben des Portfolio Managements unterstützt [Hofmann 1992]:

- IAS erhebt und bewertet persönliche und finanzielle Daten und identifiziert die Anlageziele und Randbedingungen eines Investors. IAS transformiert diese Anlageziele und Randbedingungen zu Zielwerten, die Attributen von bestimmten Anlageinstrumenten entsprechen. Die Zielwerte bilden das Anlegerprofil. In diesem Prozeß müssen die Informationen in der Wissensbank vollständig ausgenützt werden; d.h. automatische Konsistenzprüfung der Antworten des Anlegers, Konstruktion von Erklärungen entsprechend dem kognitiven Modell des Anlegers, usw. Die Identifizierung von Transformationsregeln,

deren korrekte Anwendung im Beratungsprozeß, sowie die notwendige Konsistenzprüfung stellen einerseits vordringliche Fehlerquellen dar und bedingen andererseits durch z.B. neue Finanzinstrumente ständige Änderungsanforderungen.

- Viele Anleger besitzen gewisse Vorstellungen über den Anwendungsbereich - diese Vorstellungen können realistisch sein oder auch nicht. Eine erfolgreiche Beratungssitzung setzt die Übereinstimmung in zumindest grundsätzlichen Annahmen über den Anwendungsbereich voraus. Um diese Übereinstimmung zu erreichen, muß IAS ein konsistentes konzeptuelles Modell relevanter Phänomene im Anwendungsbereich präsentieren. Da derartige Modelle per definitionem unvollständig sind und relevante Phänomene von der Individualität des Portfoliomanagers abhängen, müssen permanent Änderungsanforderungen berücksichtigt werden.
- Wesentliche Anforderungen an IAS sind Adaptivität und Flexibilität und die Kontrolle des Portfoliomanagers über die durchgeführten Operationen. Zum Beispiel, muß IAS den Stand der Beratungssitzung sowie dessen Kontext speichern, wenn der Kunde oder der Portfoliomanager des Thema wechselt. Darüber hinaus muß der neue Gesprächskontext indentifiziert werden und bei Bedarf zum vorherigen Themengebiet zurückgekehrt werden können. Geeignete situative Strategien zu entwickeln und ihre gegenseitige Abhängigkeit zu erkennen waren in diesem Zusammenhang die wesentlichen Fehlerquellen.

Exemplarische Fehlerquellen:

- Eine typische Fehlerquelle, deren Ursachen in der mangelhaften Spezifikation zu suchen sind, ist die unzureichende Dimensionierung von Variablen. So werden z.B. Teile von Datensätzen zur weiteren Verarbeitung in Matrizen geladen, welche auf eine ursprünglich geringere Anzahl von Datensätzen ausgelegt waren.
- Eine weitere Fehlerquelle sind die Eingabeüberprüfungsroutinen hinsichtlich der Benutzereingaben. Sie sind ein wesentlicher Bestandteil der Integritätssicherung von Wissensbanken. Eine Durchlässigkeit hinsichtlich falscher Wissenselemente kann zu falschen Ergebnissen in der späteren Wissensverarbeitung führen.

- Insbesondere bei der Entwicklung von Expertensystemen kann durch Syntaxfehler bei der Eingabe von Wissenselementen (z.B. beim Kopieren von Fakten oder Regeln, mit dem Ziel, den Eingabeaufwand niedrig zu halten) ein späteres "Pattern Matching", d.h. die erfolgreiche Überprüfung der syntaktischen Übereinstimmung mit dem Ziel, die entsprechenden Wissenselemente konstruktiv zu verketten, unmöglich gemacht werden. Dies ist insbesondere dann schwerwiegend, wenn der Benutzer keine detaillierten Kenntnisse über die Inhalte bzw. Struktur der Wissensbank besitzt und damit das Ergebnis nicht oder nur eingeschränkt kontrollieren kann.
- In den Entwicklungssystemen (z.B. Shells, Programmierumgebungen) von Expertensystemen können Programmierfehler enthalten sein, welche auf unzureichende Qualitätsstandards bei deren Entwicklung zurückzuführen sind. Weiterhin können Fehler auftreten, welche durch Inkompatibilitäten zwischen den Entwicklungssystemen und dem jeweiligen Betriebssystem verursacht werden. Diese Fehler lassen sich, sofern nicht neuere, aufeinander abgestimmte Versionen der Entwicklungsumgebung und des Betriebssystems zur Verfügung stehen, nur durch die Änderung der Implementierung des Softwaresystems umgehen.

Exemplarische Änderungsanforderungen:

- Eine häufige Änderungsanforderung bei formatierter Datenspeicherung ist die der Änderung des Formates, in welchem die Daten gespeichert werden. Beispiele hierfür sind die Einführung der fünften Stelle in den Postleitzahlen der Bundesrepublik Deutschland oder der nächste Jahrtausendwechsel, welche Änderungen in bestehenden Softwaresystemen verursachen.
- Eine ebenfalls häufige Änderungsanforderung ist das Einführen neuer Objekte, welche in eine bestehende Objektklasse eingeordnet werden. Wurde zum Beispiel bisher ein Objekt KUNDE verwaltet, so kann es durch geänderte Anforderungen notwendig werden, dieses Objekt in zwei Objekte mit unterschiedlichen Definitionen eines Kunden aufzuspalten.
- Einen gewissen Anteil an den Aufwendungen für die Wartung haben Umstrukturierungen in den Wissensbanken, welche auf Veränderungen in der Umwelt des Softwaresystems zurückzuführen sind. Ein Beispiel

hierfür ist die Zusammenführung der Inhalte verschiedener Datenmodelle in einem gemeinsamen Datenmodell, welches schließlich in einer Datenbank (z.B. relational, deduktiv, objektorientiert) abgebildet wird.

- Besonders bei Wissensbanken, welche bei der Entwicklung von Expertensystemen erstellt werden, ist aufgrund der Verwendung eines explorativen Erstellungsparadigmas mit häufigen Änderungsanforderungen zu rechnen. Eine höhere Komplexitätsstufe der Aktivitäten in der Wartung wird außerdem erreicht, wenn in derartige Wissensbanken nicht nur Wissen in bestehende Strukturen eingefügt, sondern neue Strukturen und damit häufig auch eine Revision der vorhandenen Struktur zur Aufnahme des Wissens benötigt werden.

Exemplarische Probleme:

Abbildung 1.4.-1 zeigt die Einordnung und Systematisierung von Problemen, die beim Einsatz von Wissensbanken auftreten können.

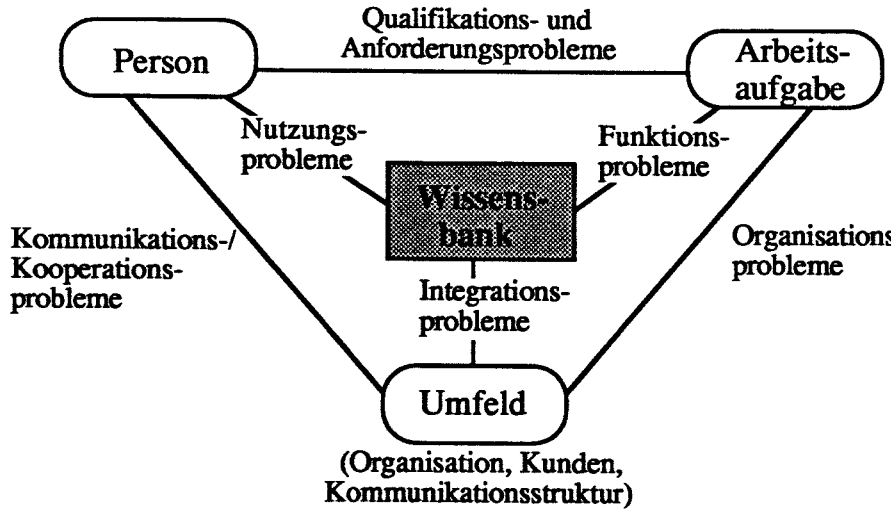

Abb. 1.4.-1: Problemfelder bei der Nutzung von Wissensbanken [nach Becker et al. 1992]

Für die einzelnen Problemgruppen werden nachfolgend jeweils einige Beispiele genannt [vgl. Becker et al. 1992].

- Nutzungsprobleme: im Dialog weit zurückreichende Eingaben oder Änderungen, Kopplung von Beratungs- und Handbuchkomponente,

Überprüfung der Leistungsfähigkeit und der Aktualität der Wissensbank;

- Funktionsprobleme: Anzeige von Schwellwerten, Kennzeichnen vorläufiger Werte, Ablage von Notizen, Modifikation ausschließlich durch Entwickler, Beantworten von Warum-nicht-Fragen;
- Integrationsprobleme: Kopplung von Beratungs- und Handbuchkomponente, Kennzeichnen vorläufiger Werte;
- mangelnde Transparenz: Transparenz der Inhalte (Anzeige von Inhalten, Modifikation und Vollständigkeit der Inhalte), Transparenz der Funktionalität (Wirkung von Systemfunktionen, Anwendbarkeit von Systemfunktionen).

Daß die Wartungssituation bei Wissensbanken, die sich im Praxiseinsatz befinden, ein durchaus ernstzunehmendes Problem ist, belegen die wenigen Berichte, die es zu diesem Thema gibt. Beispielsweise waren für das System XCON/-R1, das die Konfiguration von DEC-Computern unterstützt, nach vier Jahren Einsatz noch immer vier Mitarbeiter vollzeit mit der Erweiterung und Änderung des Wissens beschäftigt. Der Wartungsaufwand hatte in diesem Zeitraum nicht abgenommen sondern sogar zugenommen. Allerdings wurde der Nutzen, der sich aus dem Systemeinsatz ergab, noch immer höher bewertet als die Wartungskosten. SIRATAC, ein System zur Entscheidungsunterstützung beim Schädlingsbefall von Baumwollpflanzen wurde nach mehrjährigem Einsatz durch ein neu konzipiertes System abgelöst, weil das gespeicherte Wissen so zugenommen hatte, daß es zu Schwierigkeiten bei der Wartung kam [vgl. Jansen & Compton 1989].

Ein anderes Problemfeld, das ebenfalls häufig genannt wird, ist die Situation bei der Dokumentation. Oft existiert für die Wissensbank überhaupt keine Dokumentation. Dazu kommt noch, daß in bestimmten Programmiersprachen verschiedene Wissensarten in einer Regel vermischt werden können. Die Sprache OPS5 erlaubt z.B. Wissen über den Aufgabenbereich, Inferenzwissen, Heuristiken u.a.m. in einer Regel darzustellen. Neben der Vermischung von Wissensarten hat sich als problematisch erwiesen, daß für bestimmte Wissensarten keine explizite Repräsentationsform existiert. Die jeweilige Lösung ist von der verwendeten Sprache abhängig. Dies führt dann unter Umständen dazu, daß z.B. die Heuristik der Problemlösung ihren Niederschlag in der Reihenfolge der Regelanwendung findet. Die Änderung einer einzigen Regel

kann unvorhersehbare Ergebnisse zur Folge haben [vgl. Angele et al. 1990]. In Verbindung mit einer mangelhaften oder fehlenden Dokumentation ist dies eine bsonders heikle Aufgabe für die Wartung.

2 Wartungsaktivitäten bei Wissensbanken

2.1 Gliederung der Wartungsaktivitäten

Die durch das Wartungsmanagement zu initiierenden und zu überwachenden Wartungsaktivitäten werden nachfolgend in Form von Aktivitäten strukturiert und dargestellt. Folgende Aktivitäten werden identifiziert (siehe Abbildung 2.1-1):

- Modifikation und Erweiterung,
- Test und Verifikation,
- Validierung,
- Wiedereinführung und Schulung.

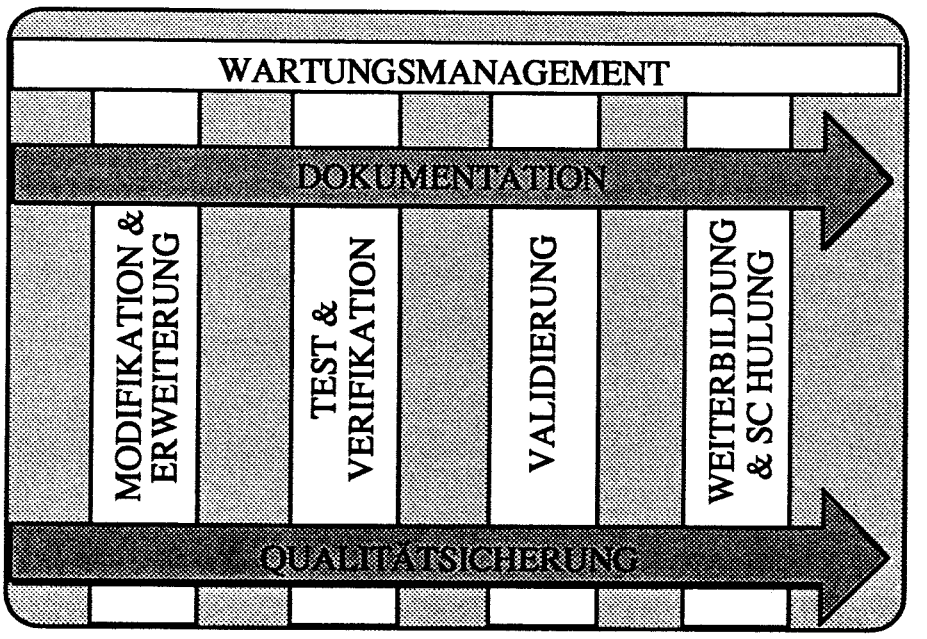

Abb. 2.1-1: Wartungsmanagement

Diese Wartungsaktivitäten werden durch zwei Querschnittsfunktionen ergänzt, nämlich die Dokumentation und die Qualitätssicherung. Beide sind in jeder Wartungsaktivität zu berücksichtigen. Sie werden aber nicht dem alle War-

tungsaktivitäten überspannenden Wartungsmanagement zugeordnet, da es sich bei ihnen um die operative Durchführung in der einzelnen Aktivität handelt, nicht um ihre Berücksichtigung innerhalb der Managementprozesse. Nachfolgend werden zuerst die Dokumentation und die Qualitätssicherung beschrieben, um ihren Querschnittscharakter zu unterstreichen, anschließend werden die einzelnen Wartungsaktivitäten dargestellt.

2.2 Dokumentation

Wesentliche Voraussetzung zur qualitativ hochwertigen und wirtschaftlichen Durchführung der Wartung ist die Dokumentation der vorhandenen Wissensbank. Im Idealzustand ist diese Dokumentation parallel zu der Entwicklung der Wissensbank und eventuell vorangegangener Wartungsprozesse entstanden, bzw. weitergeführt worden. Auch innerhalb eines konkreten Wartungsprozesses ist das Aktualisieren der Dokumentation innerhalb der einzelnen Aktivitäten notwendig. Ein nachträgliches Dokumentieren von Änderungen, d.h. Dokumentation nach Abschluß der Wartung, führt zu einer unvollständigen oder nicht aktuellen Dokumentation. Dies kann zu Nachteilen hinsichtlich der Qualität und der Wirtschaftlichkeit führen. Es ist z.B. zu erwarten, daß ein Test wesentlich zielgerichteter durchgeführt werden kann, wenn aufgrund der Dokumentation geänderte Teile der Wissensbank eindeutig identifizierbar sind. Darüber hinaus kann die änderungsbegleitende Dokumentation aufgrund ihrer anderen Darstellungsform im Vergleich zum formalen Quellcode frühzeitig ein Hinterfragen der Spezifikation und ihrer softwaretechnischen Umsetzung bewirken.

Die Dokumentation innerhalb eines Wartungsprozesses geschieht in der Regel durch Anpassung bzw. Fortführung der Dokumente oder Dateien, welche bei der erstmaligen Dokumentation erstellt wurden. Hilfsmittel sind z.B. Textverarbeitungssysteme für Fließtextdokumentation, Hypertextsysteme, Datenlexika mit Datenbeschreibungssprachen für die formale Beschreibung von Datenstrukturen und Editoren für das Einfügen von Programmkommentaren.

Neben der Dokumentation von z.B. Datenstrukturen, Entscheidungslogik, Schnittstellen und externer Referenzen ist für das Verständnis von Wissensbanken ein konzeptionelles Modell des Anwendungsgebietes von entscheidender Bedeutung. Ein konzeptuelles Modell wird bei der Analyse des Anwendungs-

bereiches entwickelt, d.h. die Interaktion aller Beteiligten, z.B. Software Engineer und Benutzer, ist wesentlich. Der Benutzer bringt vor allem sein Fachwissen über den Anwendungsbereich ein, während der Software Engineer, z.B. durch sein Fachwissen über Strukturierung, zur Modellierung beiträgt. Ein konzeptuelles Modell definiert aufgabennahes Wissen, strukturiert den Anwendungsbereich und repräsentiert ein partielles, meist unvollständiges Modell des Anwendungsbereiches [vgl. Newell 1981; Norman 1983]. Konzeptuelle Modelle reflektieren somit Expertenwissen über den Aufgabenbereich, sie repräsentieren jedoch nicht notwendigerweise die kognitiven Prozesse eines Aufgabenspezialisten [Winograd & Flores 1986].

2.3 Qualitätssicherung

Mit der Einführung von Prinzipien, Methoden und Vorgehensmodellen des Software Engineering wird das Ziel verfolgt, einen gewissen Qualitätsstandard bei der entwickelten Software sicherzustellen. Somit ist die Software-Qualitätssicherung eine Teildisziplin des Software Engineering und demnach auch geplant und systematisch in der Wartung von Softwareprodukten zu berücksichtigen. [vgl. Wallmüller 1990, 6].

Unter (Software-)Qualität werden diejenigen Eigenschaften eines (Software-) Produktes verstanden, welche geeignet sind, die Erreichung der an das Produkt gestellten Anforderungen sicherzustellen [vgl. Heinrich & Roithmayr 1992; DIN 55350, Teil 11; ANSI/ASQC A3-1978]. Neben den Produkteigenschaften werden häufig auch die Kosteneinhaltung und die Termineinhaltung in der Softwareproduktion als Qualitätseigenschaften angesehen. Dies bedingt eine erweiterte Sicht, in der der Qualitätsbegriff auf den Prozeß der Softwareentwicklung ausgedehnt wird. Es werden eigene Qualitätsziele für den Entwicklungsprozeß definiert, welche sich aus den Qualitätszielen des Produktes ableiten lassen [vgl. Wallmüller 1990, 6].

Zur Planung der Softwarequalität, deren Berücksichtigung in den Tätigkeiten der Phasen des Lebenszyklus und der Kontrolle der Berücksichtigung bedarf es einer Operationalisierung des Qualitätsbegriffes. Hierfür wurden verschiedene Systeme bzw. Begriffshierarchien von Qualitätsmerkmalen entwickelt [eine Auflistung verschiedener Systeme findet sich z.B. bei Sneed 1987, 92ff.].

Ein Beispiel hierfür ist die nachstehende Bildung von Klassen von allgemeinen Qualitätsmerkmalen:

- Portabilität,
- Zuverlässigkeit,
- Effizienz,
- Benutzerfreundlichkeit,
- Testbarkeit,
- Verständlichkeit,
- Änderbarkeit.

Diese Qualitätsmerkmale werden dann im Zuge einer weiteren Operationalisierung in spezifischere Merkmale unterteilt (z.B. Geräteunabhängigkeit, Vollständigkeit, Genauigkeit, Konsistenz, Geräteeffizienz, Verfügbarkeit). Die Vielzahl von Systematiken zur Beschreibung und Operationalisierung der Softwarequalität machen deutlich, daß es noch an einer allgemein akzeptierten Klärung des Begriffs Softwarequalität und der entsprechend durchzuführenden Tätigkeiten mangelt [vgl. Lehner 1991, 155].

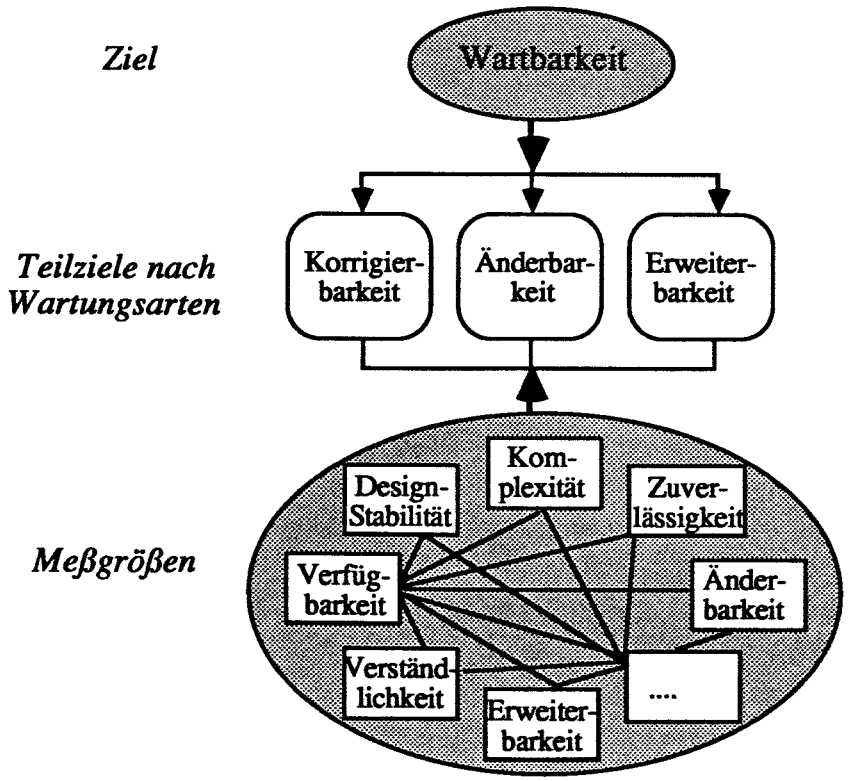

Abb. 2.3-1: Allgemeines Qualitätsmodell für die Softwarewartung [Lehner 1991, 156]

Unter Beachtung dieser Einschränkung ist festzustellen, daß es Qualitätsmerkmale gibt, welche für die Wartung von besonderer Bedeutung sind. Als allgemeine Merkmale seien hier für ein Qualitätsmodell der Softwarewartung die Korrigierbarkeit, die Änderbarkeit und die Erweiterbarkeit aufgeführt. Diese lassen sich im Zuge einer weiteren Operationalisierung in spezifischere Merkmale zerlegen. Welche spezifischen Merkmale zu berücksichtigen sind und wie diese Eingang in den Prozeß der Softwareproduktion finden, hängt von den konkreten Projekten, bzw. Produkten, ab. Es ist daher Aufgabe des Wartungsmanagements, ein entsprechendes Qualitätsmodell für die Wartung zu entwickeln (vgl. Abbildung 2.3-1), welches nicht nur in der Wartung selbst verwendet wird, sondern hinsichtlich seiner Konsequenzen auch in der (Erst-)Erstellung eines Softwareproduktes, also insbesondere in den Phasen Design und Implementierung, berücksichtigt wird. Umgekehrt sind auch in den Wartungsaktivitäten nicht nur die Qualitätsmerkmale zu berücksichtigen, welche die Wartbarkeit in späteren Wartungsprozessen sicherstellen, sondern auch jene, welche eine unmittelbare Bedeutung für die Nutzung des Softwareproduktes haben (z.B. Laufzeiteffizienz, Speichereffizienz, Robustheit, Benutzerfreundlichkeit).

Die Forderung nach intersubjektiver Überprüfbarkeit und maschineller Nachprüfbarkeit der Softwarequalität, bzw. der festgelegten Qualitätsmerkmale, führte zur Entwicklung einer Disziplin, die als Software-Metrik, Softwaremetrie oder manchmal auch als Software-Physik bezeichnet wird. Das Ziel dieser Disziplin ist das Definieren und Anwenden von Qualitätsmaßen, welche die Qualitätsmerkmale objektiv und möglichst maschinell meßbar werden lassen. Aus den schon fast unüberschaubar gewordenen Qualitätsmaßen für Software sind folgende unter Wartungsgesichtspunkten von besonderem Interesse [vgl. Lehner 1991, 154 ff.]:

- Design-Stabilitätsmaße (z.B. PDS von Yau/Collofello),
- Komplexitätsmaße (z.B. MacCabe, Halstead, Rechenberg),
- Änderbarkeits- und Erweiterbarkeitsmaße (z.B. LSP von Yau/Collofello, Modifiability Metric von Yau/Chang),
- Zuverlässigkeitsmaße (Fehlervorhersagemaß von Ottenstein, Zuverlässigkeitsmaß von Nelson),
- Verfügbarkeitsmaße (Error-Day, MTBF, Maß von Littlewood),

- weitere Maße (z.B. Portabilitätsmaße, Modularitätsmetrik, Maße für die Benutzungsfreundlichkeit, Maße für die Code-Verständlichkeit, Maße für die Testbarkeit und Maße für die Dokumentations-Qualität).

Ein spezielles Forschungsfeld befaßt sich noch mit der Entwicklung von Metriken zur Evaluierung von wissensbasierten Komponenten [vgl. O'Keefe et al. 1987; Ayel & Laurent 1991]. Zwei Arten von Metriken werden dabei unterschieden: aufgabengebiet-bezogene Metriken und Metriken für wissensbasierte Komponenten [vgl. Preece 1990]. Aufgabengebiet-bezogene Metriken messen das Verhalten von wissensbasierten Komponenten und sind analog zu den Kriterien, die von Spezialisten verwendet werden, um andere Spezialisten zu beurteilen. Metriken von wissensbasierten Komponenten sind unabhängig von einem spezifischen Softwaresystem, und befassen sich mit dem Inhalt und der Struktur der wissensbasierten Komponente (siehe z.B. GATEWAY Projekt, IED Projekt 1751).

Die Erfahrung im Bereich der Software-Metrik zeigt allerdings, daß es derzeit nicht möglich ist, eine umfassende Liste aller Qualitätsmerkmale (und damit auch der Qualitätsmaße) zu erstellen, und daß jede Organisation bzw. jede Entwicklungsumgebung ihre eigenen, spezifischen Charakteristika besitzt. Dies führt dazu, daß für eine Metrik oft lediglich Richtlinien vorgegeben werden können. Oft wird auch übersehen, daß die eigentliche Schwierigkeit in der Interpretation der Meßergebnisse besteht; hier existieren im Augenblick noch keine sehr fundierten Erfahrungen und Ergebnisse. Dennoch kann heute davon ausgegangen werden, daß Metriken für diese Zwecke verstärkt eingesetzt werden. Sie sind z.T. bereits fester Bestandteil objektorientierter Entwurfsumgebungen, die gerade im Bereich wissensbasierter Systeme eine zentrale Stellung einnehmen.

Innerhalb der einzelnen Wartungsaktivitäten sind die Vorgaben des durch das Wartungsmanagement vorgegebenen Qualitätsmodells, bzw. der Qualitätsmerkmale, umzusetzen. Zu Unterstützung dieser Arbeiten stehen verschiedene Automatisierungswerkzeuge zur Verfügung. Genannt werden u.a. [vgl. Deutsche Gesellschaft für Qualität 1986; Wallmüller 1990, 111 ff.): Data-Dictionaries, Syntaxgesteuerter Editor, Datenbankgenerator, Assertionsgenerator, Prototyping-Werkzeuge, Dokumentationssysteme, Pretty Printer, Cross-Reference- und Listengenerator, Disassembler, Werkzeuge zur statischen und dynamischen Programmanalyse, Flußdiagramm-Generator, Kontrollfluß-Analysa-

tor, Datenfluß-Analysator, Schnittstellen-Analysator, Prüfwerkzeuge für die Testabdeckung, Pfadausdrucksgenerator, Testdatengenerator, Assertionsauswerter, Werkzeuge für die symbolische Programmausführung, Debugger, Performance Analysator, Softwaremonitor.

2.4 Modifikation und Erweiterung

Die Erweiterung des Wissens im Anwendungsbereich (perfektionierende Wartung), Modifikation bestehender Funktionen und Daten (adaptive Wartung) und die Behebung von aufgedeckten Abweichungen (korrigierende Wartung) führen zu einem ständigen Anpassungsprozeß, in dem Fachexperten, Benutzer und Systemdesigner die Wissensbanken gemeinsam betreuen, aktualisieren und verbessern.

Diese Modifikationen und Erweiterungen einer Wissensbank müssen in der Wissensbank selbst dokumentiert und jederzeit nachvollziehbar sein. Zur Modifikation und Erweiterung der Wissensbank gehören auch Änderungen der Benutzerdokumentation, der technischen Dokumentation, sowie Änderungen an Übersichtsmaterialien (z.B. Marketingunterlagen, Schulungsunterlagen, usw.).

Nach traditionellen Wartungskonzepten sind diese Aktivitäten so auszuführen, daß der Entwurf und die Implementation der Wissensbank soweit als möglich erhalten bleibt. Die Wartungsaktivitäten sind zumeist vom Erkenntnisstand zur Zeit der Neuentwicklung dominiert, d.h. mit veralteten Methoden werden veraltete Wissensbanken bearbeitet.

Das vorgestellte Wartungskonzept hat hingegen einen anderen Ausgangspunkt. Vor der Modifikation bzw. Erweiterung wird die Wissensbank auf dem neuesten gesicherten Stand der Technik gebracht. Daher kann das Wartungspersonal mit der modernisierten Wissensbank wie mit einer neuentwickelten Wissensbank verfahren und die neuesten Hilfsmittel und Methoden einsetzen. Die Modifikationen werden spezifiziert, entworfen, und implementiert, genauso wie es von Neuentwicklungen von Wissensbanken bekannt ist.

Eine idealtypische Vorstellung besteht darin, daß nicht das Wartungspersonal sondern der Anwender nach Übernahme des Systems, die Modifikation mit geeigneten Werkzeugen selbst durchführt. Diese Art der Wartung (Tuning) muß

allerdings bereits bei der Neuentwicklung berücksichtigt werden [Greenbaum & King 1991].

2.5 Test und Verifikation

Nach der Durchführung der Modifikation und Erweiterung sind die veränderten Wissensbanken zu testen und zu verifizieren [vgl. Pfeifer et al. 1992; Ayel & Laurent 1991].

Testen ist eine Aktivität zum Nachweis der Richtigkeit von Ergebnissen im Softwareentwurf, z.B. experimentelle Ausführung zur Feststellung vordefinierter Eigenschaften. Interessant ist in diesem Zusammenhang die Erkenntnis, daß das Testen ein weit weniger wirksames Mittel zum Finden von Fehlern ist, als gemeinhin angenommen wird [vgl. Lehner 1989]. Formale Entwurfs- und Codeinspektionen, die von kleinen Gruppen erfahrener Spezialisten durchgeführt werden, sind bei der Lokalisierung von Fehlern vergleichsweise wirkungsvoller.

Die Testaktivitäten in der Wartung unterscheiden sich vom Test in der Neuentwicklung in erster Linie dadurch, daß eine Testumgebung bereits existiert und daß repräsentative Testdaten zur Verfügung stehen. In der Wartung müssen Testumgebung und Testdaten jedoch angepaßt bzw. erweitert werden.

Verifikation bezeichnet Tätigkeiten, die rein auf der "internen" Repräsentation einer Wissensbank durchgeführt werden können, also das Feststellen der Konsistenz, Redundanzfreiheit, und "Korrektheit" im Rahmen des gewählten Repräsentationsformalismus. Verifikation behandelt die Wissensbank als 'white box', d.h. es gilt zu zeigen, daß die interne Struktur der Wissensbank ihrer formalen Spezifikation entspricht.

Beim Bekanntwerden neuer Fakten oder Regeln sollte die Wissensbasis mit einem vertretbaren Aufwand berichtigt werden können. Zur Unterstützung dieser Aktivitäten sind im Bereich logischer Wissensrepräsentationen sogenannte Truth-Maintenance-Systeme entwickelt worden. Diese Systeme prüfen bei jeder Erweiterung der Wissensbank (alle) ihre Inhalte auf Konsistenz und berichtigen diese gegebenenfalls. Die Notwendigkeit derartiger Systeme ergibt sich, wenn einmal als wahr abgeleitete und in der Wissensbank gespeicherte Schlußfolgerungen sich durch Hinzufügung neuen Wissens als falsch heraus-

stellen können. Von monotoner Logik spricht man, wenn einmal als wahr abgeleitete und in der Wissensbasis abgespeicherte Schlußfolgerungen sich durch Hinzufügung neuen Wissens nicht mehr als falsch herausstellen können; allenfalls können sich neue wahre Fakten ergeben. Wenn diese Voraussetzung (Monotonizitätsannahme) nicht gilt, dann müssen bei jeder Erweiterung der Wissensbasis alle ihre Inhalte auf Konsistenz geprüft und gegebenenfalls berichtigt werden. Diesen Vorgang bezeichnet man als truth maintenance und Systeme die dies unterstützen als Truth-Maintenance-Systeme. Ohne truth maintenance führt eine nichtmonotone Logik zu der untragbaren Situation, daß abgeleitete und in der Wissensbasis abgelegte Fakten oder Regeln unwahr werden können [Thuy & Schnupp 1989, 77].

2.6 Validierung

Die Validierung wird oft mit der Verifikation verwechselt oder vermengt. Die Verifikation dient dem Nachweis, daß das Verhalten der Wissensbank deren Spezifikation entspricht und daß das relevante Wissen repräsentiert wird. Die Validierung befaßt sich hingegen mit der Bewährung des Systems in der Anwendungsumgebung [vgl. z.B. O'Keefe et al. 1987]. Es geht dabei um die Frage, ob die Probleme auch im Kontext der betrieblichen Aufgabenstellung (die nicht mit der Spezifikation identisch sein muß) "korrekt" gelöst werden. Validierung behandelt die Wissensbank als 'black box', d.h. die interne Struktur der Wissensbank wird in der Untersuchung nicht berücksichtigt.

Der Aufwand, der Im Rahmen der Wartung für die Validierung erforderlich ist, wird durch das Aufgabengebiet bestimmt. Ein klar abgegrenztes Aufgabengebiet, welches z.B. in einem wohldefinierten Repräsentationsformalismus abgebildet werden kann, verlangt vorrangig Verifikation; wird das Aufgabengebiet jedoch erst durch die Wissensbank strukturiert, dann werden die Validierungsaktivitäten überwiegen.

2.7 Wiedereinführung und Schulung

Die Eingliederung der modifizierten Wissensbank in den betrieblichen Ablauf konzentriert sich auf die Aktivitäten: Wiedereinführung und Schulung.

Bei der Wiedereinführung wird die Übereinstimmung von Systemverhalten und geforderter Leistungsfähigkeit in der Arbeitssituation anhand qualitativer und quantitativer Merkmale untersucht. Ein wichtiges Beurteilungskriterium ist, ob die Wiedereinführung der Wissensbank eine Verbesserung in der Arbeitssituation der Benutzer oder im unterstützen Geschäftsbereich herbeiführt. Diese Verbesserung kann anhand der Merkmale Qualität, Nutzen und Benutzbarkeit beschrieben werden.

Die Qualitätsbewertung sollte durch eine Expertengruppe, die an der Wartung der Wissensbank nicht beteiligt war, durchgeführt oder anhand quantitativer Leistungskriterien und empirischer Tests vorgenommen werden. Der Nutzen der Wissensbank wird durch Benutzer und Management des Geschäftsbereichs festgelegt. Die Benutzbarkeit wird z.B. durch Kriterien wie Benutzerakzeptanz, Verständlichkeit der Systemantworten und Adaptivität der Wissensbank bestimmt.

Die Wiedereinführung gewarteter Wissensbanken wird durch Schulungsaktivitäten unterstützt. Aufgrund der (technischen) Dokumentation, die aus der Wissensbankanalyse resultiert, sollte die Schulung einfacher sein als bei unstrukturierten Vorgehensweisen und die dabei verbesserte Qualität der Wissensbank sowie die damit verbundene Zuverlässigkeit sollte weniger Unterstützung der Benutzer notwendig machen.

3 Schlußfolgerungen

Systematische Untersuchungen oder Daten zur Wartungsituation bei Wissensbanken existieren bisher nicht. Dies trifft auch auf den Einsatz von Methoden und Werkzeugen zu. In der Literatur über Expertensysteme finden sich bisher nur wenige Berichte und Vorschläge, die sich mit der Wartungsproblematik beschäftigen. Dies erklärt sich primär durch die Tatsache, daß sich nur sehr wenige Wissensbanken wirklich im betrieblichen Einsatz befinden. In den wenigen verfügbaren Berichten kommt klar zum Ausdruck, daß die Wartung von Wissensbanken einen entscheidenden Erfolgsfaktor in bezug auf Verwendbarkeit und Wirtschaftlichkeit des Systems darstellt.

Soweit die Ursache von Wartungsproblemen bei Wissensbanken eine mangelhafte oder fehlende Dokumentation ist, ist die Situation mit der herkömmlichen Softwarewartung vergleichbar und erfordert z.T. auch ähnliche Maßnahmen. Die übrigen Probleme stellen sich als Kombination mangelnder Erfahrung und technischen Restriktionen dar. Eine Verbesserungen der Situation werden von neuen Entwicklungen im Bereich ausführbarer konzeptioneller Modelle sowie von Knowledge Dictionaries erwartet. Konzeptuelle Modelle, die dem ablauffähigen System möglichst ähnlich sehen, haben in bezug auf die Wartbarkeit u.a. folgende Vorteile: Erklärungsfähigkeit, Selbstdokumentation und Verständlichkeit. Der Beitrag von Knowledge Dictionaries kann mit der Unterstützung der Wartung durch Data Dictionaries in herkömmlichen Entwicklungsumgebungen verglichen werden. Besondere Anstrengungen sind noch auf dem Gebiet der Qualitätssicherung (z.B. Programmierkonventionen, Programmstrukturen) und der Qualitätsmessung (z.B. Software-Metriken für objekt-orientierte Sprachen) erforderlich.

Literatur

Angele, J., Fensel, D., Studer, R.: What could the knowledge engineer learn from the software engineer? In: Ehrenberg, D. et al. (Hrsg.): Wissensbasierte Systeme in der Betriebswirtschaft. Berlin 1990, 285-303

Ayel, M., Laurent, J.-P. (Hrsg.): Validation, Verification and Test of Knowledge-based Systems. John Wiley & Sons, Chichester/New York 1991

Becker, B., Herrmann, Th., Steven, E. (Hrsg.): Zur Diskrepanz zwischen Expertensystementwicklung und -einsatz. GMD Arbeitspapier Nr. 641, GMD Bonn/St. Augustin, April 1992

Benbasat, I.; Dhaliwal, J.S.: "A framework for the validation of knowledge acquisition". Knowledge Acquisition, vol. 1, no. 2, 1989, 215-233.

Boehm, B.W.: Software Engineering. In: IEEE Transactions on Computing, vol. 25, no. 12, 1976, 1226-1241

Buchanan, B.G.; Barstow, D.; Bechtal, R.; Bennett, J.; Clancey, W.; Kulikowsky, C.; Mitchell, T.; Waterman, D.: "Constructing an Expert System". In: Hayes-Roth, F.; Waterman, D. (eds.): Building Expert Systems, Addison Wesley, London, 1983, 127-167.

Buth, A.: Softwaremetriken für objekt-orientierte Programmiersprachen. GMD Arbeitspapier Nr. 545, Bonn/st.Augustin Juni 1991

Deutsche Gesellschaft für Qualität (Hrsg.): Software-Qualitätssicherung. Berlin, 1986.

Greenbaum, J.; Kyng, M.: "Design at work". Hillsdale, N.J.: Erlbaum, 1991

Harmon, P.; King, D.: "Expertensysteme in der Praxis: Perspektiven". Oldenbourg, München, 1989

Harmelen, F. van: $(ML)^2$: A formal language for KADS models of expertise. In: Knowledge Akquisition, 4/1992, 127-161

Heinrich, L. J., Roithmayr, F.: Wirtschaftsinformatik-Lexikon. 4. Aufl., München/Wien 1992

Hofmann, H. F.: IAS - Investment Advisory System Prototype (Investor Counseling). Diplomarbeit, Universität Linz, Institut für Wirtschaftsinformatik, Linz 1992

Jansen, B., Compton, P.: Data dictionary approach to the maintenance of expert systems: The Knowledge Dictionary. In: Knowledge-Based Systems, Vol. 2, 1/1989, 14-26

Jones, T. C.: Verhütung und Beseitigung von Programmierfehlern. In: Elektrisches Nachrichtenwesen, 4/1983, 295-300.

Karbach, W., Linster, M.: Wissensakquisition für Expertensysteme. Techniken, Modelle und Softwarewerkzeuge. München 1990

Lamberts, K.; Pfeifer, R.: Computational models of expertise. Accounting for routine and adaptivity in skilled performance. In: Gilhooly, K.; Keane, M. (Hrsg.): Advances in psychology of thinking. New York: Simon & Schuster, in Druck.

Lehner, F.: Softwarewartung. München 1991

Lehner, F.: Nutzung und Wartung von Software. München, 1989.

Matthews, M. H.: Maintenance & Language Choice. In: AI Expert, 9/1987, 42-49

Newell, A.: "The Knowledge Level". AI-Magazine, vol. 2, no. 2, 1981, 1-20

Norman, D.: "Some Observations on Mental Models". In: Gentner, D. (ed.): Mental Models, Hillsdale, New Yersey, 1983, 72-113.

O'Keefe, R. M.; Balci, O.; Smith, E. P.: Validating Expert System Performance. In: IEEE Expert, Winter 1987, 81-89.

Oppermann, R.: Adaptively supported Adaptability. In: GMD-Nachrichten, GMD Bonn/St. Augustin, AC Special, Nr. 11, Juli 1992, 1-16

Pfeifer, R.: Hofmann, H.F.; Rademakers, R.; Vinkhuyzen, E. Validation of knowledge based systems. KI-Magazin 4/92 (in Druck).

Preece, A.: Towards a Methodology for Evaluating Expert Sytem. In: Expert Systems, vol. 7, no. 4, 215-223.

Sneed, H. M.: Software-Management, Köln 1987

Thuy, N. H. Ch., Schnupp, P.: Wissensverarbeitung und Expertensysteme. München/Wien 1989

Van Meegen, M., Schless, P.: Programmierkonventionen für C++. In: Softwaretechnik-Trends, hrsg. von den Fachgruppen "Software Engineering" und "Requirements Engineering" der Gesellschaft für Informatik (GI), Band 12, Heft 3, 1992, 29-47

Verein Deutscher Ingenieure (Hrsg.): VDI-Richtlinie 5006: Bürokomunikation - Expertensysteme in betriebswirtschaftlichen Anwendungen., VDI, Düsseldorf, September 1992

Wallmüller, E.: Software-Qualitätssicherung in der Praxis. München 1990

Wielinga, B. J., Schreiber, A. Th., Breuker, J. A.: KADS: a modelling approach to knowledge engineering. In: Knowledge Akquisition, 4/1992, 5-53

Winograd, T.; Flores, F.: "Understanding Computers and Cognition". Ablex, New York, 1986