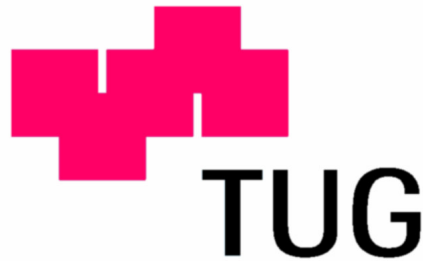


Stefan DAXENBICHLER

**Online-Monitoring für die strategischen IT-
Systeme einer Universitätsklinik am Beispiel
der Tiroler Landeskrankenanstalten**

Diplomarbeit



Institut für Genomik und Bioinformatik

Technische Universität Graz

Krenngasse 37, A-8010 Graz

Vorstand: Univ.-Prof. Dipl.-Ing. Dr. techn. Zlatko Trajanoski

Betreuer und Begutachter: Univ.-Prof. Dipl.-Ing. Dr. techn. Bernhard Tilg

Graz, Februar 2004

Danksagung

An erster Stelle möchte ich mich bei Prof. Dr. Bernhard Tilg für die Betreuung dieser Diplomarbeit und für die Unterstützung bei der Überwindung organisatorischer Hürden bedanken.

Weiterhin bedanke ich mich bei meinen Vorgesetzten Mag. Kaloczy und Dr. Hirsch, dass sie meine Entscheidung zum Abschluss des Studiums mit den damit verbundenen Abwesenheiten vom Dienst vorbehaltlos unterstützt haben.

Mein großer Dank gilt meinen Eltern, die mir das Studium in Graz ermöglicht haben und während dieser Zeit immer für mich da waren.

Ganz besonders möchte ich mich bei meiner Lebensgefährtin Ursula bedanken, die mich immer unterstützt hat ohne deren Vorbild und laufende Motivation ich mein Studium wahrscheinlich nicht wieder aufgenommen und beendet hätte.

Titel: Online-Monitoring für die strategischen IT-Systeme einer Universitätsklinik am Beispiel der Tiroler Landeskrankenanstalten

Kurzfassung: Krankenhäuser besitzen eine komplexe, gewachsene IT-Infrastruktur mit vielen verschiedenen, untereinander kommunizierenden Systemen. Die Verfügbarkeit der zentralen IT-Systeme ist inzwischen unabdingbar für die optimale Behandlung. Daher müssen Maßnahmen zur Erhöhung der Systemverfügbarkeit sowie zu deren kontinuierliche Überwachung ergriffen werden.

An der Universitätsklinik Innsbruck wurde das KIS (Krankenhausinformationssystem) als Ausgangspunkt für ein Überwachungsprojekt genommen. Unter Abwägung aller Faktoren fiel die Entscheidung, als Basis für das Monitoring ein Freeware-Produkt zu verwenden.

Nach Analyse und Definition der zu überwachenden Komponenten wurde das Überwachungssystem angepasst und implementiert.

Die Erfahrungen mit dem System sind sehr positiv. In vielen Fällen konnte die Zeit bis zur Entdeckung eines Problems drastisch verkürzt werden. Durch laufende Anpassungen wird das System weiter verbessert und an Änderungen in der Umgebung angepasst.

Schlüsselwörter: Krankenhausinformationssysteme, Überwachung, 3LGM, Verfügbarkeit

Title: Online Monitoring of strategic IT systems in a university hospital environment

Abstract: Hospital environments have a very complex, grown IT infrastructure with a lot of different systems communicating with each other. Availability of the main IT systems is indispensable in order to guarantee optimal patient treatment. Therefore, measures have to be taken to increase and monitor overall availability.

We took the Clinical Information System (KIS) as starting point for our monitoring project. After balancing all facts, we decided to use a freeware software to implement the monitoring system. After analyzing and defining the items to be monitored, we implemented the monitoring system.

Up to now we made very positive experiences with the system. It helped us to detect and repair system failures more timely in a lot of cases. The system is constantly being refined and adapted to changes in the environment.

Keywords: Clinical Information System, Monitoring, 3LGM, Availability

Inhaltsverzeichnis

1	Einleitung	6
1.1	IT-Landschaft eines Krankenhauses	7
1.1.1	MPI – Master Patient Index	7
1.1.2	RIS – Radiologisches Informationssystem	8
1.1.3	PACS	8
1.1.4	Dokumentenverwaltungssystem	9
1.1.5	KIS – Klinisches Informationssystem	10
1.1.6	LIMS - Laborsystem	10
1.1.7	Kommunikations-System	11
1.1.8	Sonstige Systeme.....	12
1.2	Modellierung mit 3LGM ²	13
1.3	Verfügbarkeit.....	15
1.3.1	Ursachen von Ausfallszeiten.....	16
1.3.2	Methoden zur Erhöhung der Verfügbarkeit.....	18
1.4	Monitoring	21
1.4.1	Aufgaben und Nutzen eines Monitoring-Systems	21
1.4.2	Komponenten eines Monitoring-Systems.....	22
2	Methoden und Ergebnisse	25
2.1	Einleitung	25
2.2	Auswahl eines Monitoring-Tools	26
2.3	Definition der überwachten Komponenten	27
2.3.1	Hardware und Netzwerk	27
2.3.2	Betriebssystem	28
2.3.3	Datenbank	31
2.3.4	Dienste	32
2.3.5	Applikationsebene	34
2.4	Festlegung der Benachrichtigungen.....	38
2.4.1	Alarm.....	38
2.4.2	e-Mail	38
2.4.3	SMS	39
2.5	Automatisierte Aktionen	40
2.6	Aufbau des Monitoring-Systems	41

2.6.1	Monitoring-PC	41
2.6.2	AIX-Agent	41
2.6.3	Windows-Agent	43
3	Diskussion	44
3.1	Zielerreichung.....	44
3.1.1	Verringerung der Reaktionszeit	44
3.1.2	Verkürzung von ungeplanten Ausfallszeiten.....	44
3.1.3	Vermeidung von ungeplanten Ausfallszeiten	44
3.1.4	Kontinuierliche Information des Teams.....	45
3.2	Lehren aus dem Monitoring-Projekt.....	46
3.2.1	Stellenwert des Monitoring.....	46
3.2.2	Freeware-Produkte	46
3.2.3	Kontinuierliche Wartung	46
3.2.4	Aktive Benachrichtigung	47
3.3	Ausblick / Verbesserungsmöglichkeiten.....	48
3.3.1	Bewertung von Auswirkungen.....	48
3.3.2	Redundanz des Monitoring-Systems.....	48
3.3.3	Weitere Verfeinerung und Komplettierung	48
3.3.4	Ausweitung auf weitere Systeme	49
Anhang	50
Anhang A:	3GLM ² -Bilder.....	50
Anhang B:	Screenshots Big Brother.....	52
Anhang C:	Script-Beispiel: OraLock	55
Literatur	58

1 Einleitung

IT-Systeme in Krankenhäusern haben in den letzten Jahren dramatisch an Bedeutung gewonnen. Während vorher vor allem eine Kombination aus zentralem System zur Verrechnung und davon getrennten Insellösungen für einzelne klinische Abteilungen üblich war, geht der Trend anhaltend in Richtung des zentralisierten Betriebs von immer mehr Funktionen. Parallel dazu vertraut man zunehmend auf die papierlose Abwicklung von Vorgängen. Damit begibt man sich aber auch in eine verstärkte Abhängigkeit von IT-Systemen. Ein Ausfall von einzelnen Systemen kann im einfachsten Fall zu Verzögerungen, aber auch zum fast vollständigen Stillstand des normalen Arbeitbetriebs in einzelnen Bereichen führen. Die fehlende Verfügbarkeit von klinischen Patientendaten kann eine nichtoptimale Behandlung und sogar ernsthafte Gefahren zur Folge haben, wenn zum Beispiel durch einen Systemausfall Informationen zu bestehenden Arzneimittelallergien nicht verfügbar sind.

Daher ist die Sicherstellung einer möglichst hohen Systemverfügbarkeit der zentralen Systeme bei gleichzeitiger Niedrighaltung der Betriebskosten besonders wichtig und muss ein zentrales Anliegen der IT-Abteilung sein.

Die Überwachung von Systemkomponenten stellt einen wichtigen Baustein im Bemühen um hohe Verfügbarkeit dar. Diese Arbeit entstand im Zuge eines Projektes zur Überwachung der Verfügbarkeit eines großen Krankenhausinformationssystems.

1.1 IT-Landschaft eines Krankenhauses

IT-Landschaften in Krankenhäusern sind sehr komplex. Das hat mehrere Ursachen:

- Aufgrund der sehr individuellen Anforderungen von klinischen Abteilungen an die Unterstützung durch ein IT-System haben sich zahlreiche Speziallösungen für jede Disziplin entwickelt. Von den Abteilungen wird der Einsatz der jeweils für sie maßgeschneiderten Lösung bevorzugt.
- Diese Systeme, die oft auf völlig verschiedenen Hard- und Softwaregrundlagen basieren, müssen dennoch miteinander kommunizieren, um die Mindestforderung eines abgeglichenen Patientenstamms (Master Patient Index) zu erfüllen.
- In länger bestehenden Systemen steckt bereits so viel über die Jahre akkumulierte Geschäftslogik, dass ein Reengineering oder die Ablöse durch ein anderes System extrem aufwändig ist.
- Durch die lange gesetzliche Aufbewahrungspflicht der Daten (in Österreich derzeit 30 Jahre) müssen patientenbezogene Daten im Allgemeinen mehrere Generationen von Softwaresystemen „überstehen“.
- Speziell in einer Universitätsklinik ist das Durchsetzungsvermögen der Verwaltung beschränkt, da hier innerhalb eines Unternehmens zwei verschiedene Dienstgeber vorhanden sind (einerseits eine Krankenhausbetriebsgesellschaft, meist in Landeseigentum, andererseits der Bund als Dienstgeber der Universitätsangestellten). Das trägt zur Veruneinheitlichung der IT-Landschaft bei.

1.1.1 MPI – Master Patient Index

Der Master Patient Index, also das Referenzsystem für alle Patientenstammdaten innerhalb einer Organisation, ist normalerweise auch das Verwaltungs- und Abrechnungssystem. Meist ist es das erste System, das den Sprung über eine Abteilung hinaus zum flächendeckenden Einsatz getan hat. Aufgrund des Alters sind hier vielfach noch Host-basierte Systeme mit zeichenbasierten Terminals im Einsatz.

Die Applikation, die den MPI zur Verfügung stellt, erfüllt mindestens folgende Aufgaben:

- Aufnahme von Patienten (Erfassung der Stammdaten, Anlegen eines Falls)
- Transferierung (Verlegung in eine andere klinische Abteilung)
- Entlassung

Diese drei Funktionen werden unter dem Kürzel ADT (Admission – Discharge – Transfer) zusammengefasst.

- Zusammenfassen von irrtümlich doppelt angelegten Patienten, Trennen von irrtümlich zusammengefassten Patienten
- Verrechnungsfunktionen

Da gerade die Verrechnungsmodalitäten sehr länderspezifisch und veränderlich sind, muss dieses System laufend an die aktuellen gesetzlichen Grundlagen und Verträge mit den Kostenträgern (vor allem den Krankenkassen) angepasst werden.

Die Anforderungen an die Verfügbarkeit des MPI sind sehr hoch, da die Unmöglichkeit einer Patientenaufnahme auch sämtliche patientenbezogenen Folgeschritte in anderen Systemen verhindert. Daher muss in Subsystemen, die besonders hohe Anforderungen an die durchgehende Verfügbarkeit stellen (z.B. Unfall- und Notfallbereich), eine Möglichkeit zur Off-Line-Aufnahme bestehen.

1.1.2 RIS – Radiologisches Informationssystem

Das RIS (Radiology Information System) ist der Paradefall eines Abteilungssystems. Viele Funktionen, die eigentlich nicht fachspezifisch sind, wurden oft erstmals an radiologischen Stationen implementiert, z. B. Patientenleitsysteme, Terminplanung, Anforderungswesen etc.. Radiologische Abteilungen weisen einige Besonderheiten auf, die sie für den Einsatz eines Informationssystems prädestinieren:

- hoher Patientendurchsatz
- limitierte, teure Ressourcen (z. B. Kernspintomograph), die eine genaue Termin- und Ressourcenplanung notwendig machen
- hohe Verschränkung mit anderen Abteilungen. Die Radiologische Abteilung ist ein interner Dienstleister, der Aufträge von anderen Abteilungen ausführt
- enge Kopplung mit dem PACS

1.1.3 PACS

Das PACS (Picture Archiving and Communication System) ist ein System, das primär der Erfassung von Bilddaten, deren Verteilung und Darstellung sowie Archivierung dient [3].

Der Haupt-Einsatzbereich eines PACS liegt in den radiologischen Abteilungen eines Krankenhauses, wo es zur Verwaltung der Daten folgender Bildquellen dient:

- CR (konventionelles Röntgen)

- CT (Röntgen-Comutertomographie)
- MRI (Magnetische Kernspinresonanz)
- PET und andere bildgebende Verfahren

Auch in der Nuklearmedizin sind PACS verbreitet, z. B. in der Szintigraphie.

In anderen Abteilungen ist der Einsatz noch eher unüblich, aber durchaus sinnvoll. Es bieten sich z. B. Ultraschallaufnahmen (z. B. Gynäkologie), Fotos (Dermatologie, plastische Chirurgie) oder Videosequenzen (insbesondere beim Ultraschall, da hier Einzelbilder weniger Aussagekraft haben) an.

Die Grenze zu einem Dokumentenverwaltungssystem für gescannte Dokumente ist fließend, es gibt auch Bestrebungen, beide Systeme zu integrieren.

1.1.4 Dokumentenverwaltungssystem

Ein eher neuer Zweig im Bereich des patientenbezogenen Krankenhausbetriebes ist die Domäne der Dokumentenverwaltung. Damit ist in diesem Umfeld vor allem das Management jener Dokumente gemeint, die nicht direkt aus einem der Hauptssysteme stammen.

In diesem Bereich können zwei Ausbaustufen unterschieden werden:

1. Verwaltung von Dokumenten, die in konventioneller Form vorliegen, also im Wesentlichen die Standortverfolgung von Papier-Krankengeschichten, evtl. auch Mikrofilmen.
2. Verfügbarmachen von Dokumenten in elektronischer Form. Das umfasst folgende Schritte:
 - Umwandeln in eine elektronische Form (Scannen), falls das Dokument in Papierform vorliegt
 - Kategorisierung, Beschlagwortung
 - Versetzen mit einem Zeitstempel (hier ist in erster Linie das Datum einer eventuellen Untersuchung relevant, in zweiter Linie das Entstehungsdatum des Dokuments, und erst in letzter Folge der Zeitpunkt der Eingliederung ins Dokumentenverwaltungssystem)
 - Zuordnung zum richtigen Patienten

Die vollständige elektronische Erfassung von bestehenden papiergebundenen Krankengeschichten ist oft aus Kostengründen nicht durchführbar, da sich Kategorisierung und Patientenzuordnung nicht mit der geforderten geringen Fehlerrate automatisieren lassen und daher einen hohen manuellen Aufwand erfordern. Auch scheitert das automatische

Einscannen ganzer Krankengeschichten schon an mechanischen Hürden wie gefalteten, geklammerten oder aus sonstigen Gründen problematischen Dokumenten.

Die vollständige Umstellung auf einen papierfreien Arbeitsablauf kann daher im Allgemeinen nur für neue Patienten und Fälle ab einem bestimmtem Stichtag erfolgen.

1.1.5 KIS – Klinisches Informationssystem

Als KIS bezeichnet man im Allgemeinen das System, mit dem die Elektronische Patientenakte (EPR – Electronic Patient Record) realisiert wird. Es kann folgende Teilbereiche abdecken:

- gemeinsame Darstellung von Daten aus unterschiedlichen Systemen, z. B. LIMS und RIS
- Befund- und Arztbriefschreibung
- strukturierte Dokumentation
- Pflegedokumentation und -planung
- Planung von Patienten-Untersuchungsterminen
- Anforderungswesen (Radiologie, Labor etc.)
- Leistungs- und Diagnosencodierung
- Basissystem für Expertensysteme und Auswertungen

Klinische Informationssysteme gehören zu den umfassendsten Systemen im Krankenhaus, sowohl was ihre geografische Ausbreitung im Krankenhaus als auch die Anzahl der Funktionsbereiche und Anwendergruppen betrifft. Daher stellt die Einführung eines solchen Systems ein Großprojekt dar, das sich über mehrere Jahre erstreckt.

1.1.6 LIMS - Laborsystem

Ein LIMS (Labor-Informations- und Managementsystem) stellt ein Spezialesystem dar, das alle Ebenen der Informationsverarbeitung im Labor abdecken soll. Klinische Labors, besonders die zentralen Großlabors, sind hochautomatisierte Abteilungen mit einem hohen Durchsatz an Probenmaterial bei stringenten Qualitätsstandards. Daher haben sich auch hier schon früh IT-gestützte Lösungen durchgesetzt.

Neben der Unterstützung des Routinebetriebs übernehmen LIMS auch zunehmend Aufgaben der Qualitätssicherung und dienen als Werkzeug für das Labormanagement, um Ansatzpunkte für Kostensenkung und Qualitätssteigerung zu finden.

1.1.7 Kommunikations-System

Alle bisher genannten Systeme sollten im Idealfall miteinander kommunizieren und dabei unter Anderem folgende Daten miteinander austauschen:

- ADT (Patientenstammdaten)
- klinische Ergebnisse (Labor, Befunde, Arztbriefe)
- Anforderungen (Laboranforderung, Radiologieanforderung etc.)
- Verrechnungsdaten (Leistungen und Diagnosen)

Die Kommunikation zwischen Systemen kann dabei grundsätzlich über verschiedene Wege erfolgen:

- direkter Zugriff auf die Datenbank des Fremdsystems: Dieser Weg ist oft am einfachsten durchzuführen, allerdings auch aus architektonischer Sicht am unsaubersten: Jede Änderung an einem System (Änderung am Datenmodell, Versionsänderung der Datenbanksoftware) kann zu notwendigen Änderungen am anderen System führen.
- synchroner Zugriff auf das Fremdsystem über eine definierte Applikations-Schnittstelle
- asynchrone Kommunikation über ein Warteschlangensystem (*Message Queuing*)

Solange nur wenige Kommunikationspartner involviert sind, werden oft Punkt-zu-Punkt-Verbindungen zwischen den Systemen verwendet. Bei einer größeren Anzahl von beteiligten Kommunikationspartnern entwickeln sich diese Verbindungen aber schnell zu einem undurchschaubaren „Spinnennetz“, in dem jede Änderung an einem System nicht mehr zu überblickende Auswirkungen haben kann.

Daher erfolgt die Kommunikation in größeren Installationen heute fast ausschließlich über spezielle Kommunikationssysteme bzw. -server, die Nachrichten von einem System entgegennehmen und asynchron an den Empfänger weiterleiten. Diese Systeme müssen hohen Datendurchsatz und garantierte Nachrichtenzustellung in jeder denkbaren Situation (z. B. auch nach Systemabstürzen) vereinen.

Für die Semantik der Kommunikation zwischen klinischen Systemen hat sich heute auf breiter Basis der Standard HL7 durchgesetzt [2]. Trotzdem müssen auch Mechanismen für die Einbindung von Systemen existieren, die diesen Standard nicht unterstützen, sondern ihre Daten z.B. als Textdateien zur Verfügung stellen.

Eine große Herausforderung in der Kommunikation stellt die durch die einzelnen Partner gewährleistete Datenqualität dar. Überall, wo Patientennamen oder -nummern manuell erfasst

werden, passieren Fehler oder Auslassungen, die die automatische Zuordnung von Daten zum richtigen Patienten beim empfangenden System unmöglich machen. Hier hilft nur die manuelle Abarbeitung von Fehlerprotokollen oder die Umstellung auf Abläufe ohne manuelle Zwischenschritte.

1.1.8 Sonstige Systeme

Neben den genannten größeren Systemen gibt es noch eine Vielzahl von so genannten Subsystemen, die meist sehr spezialisiert sind, z. B. Systeme für die Zahnheilkunde, die Operationsplanung und -dokumentation, Systeme für die Erfassung und Auswertung klinischer Studien und vieles andere. Oft sind diese Programme auch von Mitarbeitern oder kleinen externen Firmen selbst erstellt. Die Einbindung dieser Systeme in eine gesamte Infrastruktur ist meist schwierig und allein schon durch die große Anzahl dieser Systeme mit relativ hohen Kosten verbunden.

1.2 Modellierung mit 3LGM²

Die Voraussetzung für eine eingehende Analyse von IT-Systemen in Bezug auf die Verfügbarkeit ist die Erstellung eines abstrakten Modells der untersuchten Umgebung. Was bei grob strukturierter Betrachtung noch mit einfachen Diagrammen zu veranschaulichen ist, ist bei einer detaillierten Analyse ohne spezielle Hilfsmittel nicht mehr handhabbar.

Aus diesem Bedarf heraus wurde an der Universität Leipzig ein Modell sowie ein entsprechendes Werkzeug zur Unterstützung von systematischen Analysen auf dem Gebiet der Gesundheitsinformatik entwickelt, das Drei-Ebenen-Metamodell (3LGM²) [4], [5], [6], [7], [8], [9].

Dieses Modell beschreibt Krankenhaus-Informationssysteme auf drei Ebenen:

- **Die Fachliche Ebene:** Hier werden die vorhandenen Arbeitsabläufe und die daran beteiligten Personen und Objekte dargestellt. Ein Beispiel ist die Abwicklung einer Patientenaufnahme mit den dazu benötigten Personen und Aktionen.
- **Die Logische Werkzeugebene:** Sie stellt die verwendeten Anwendungen unabhängig von der konkreten Implementierung dar. Es wird dabei unterschieden zwischen rechnerbasierten und nicht rechnerbasierten Bausteinen. Ein Beispiel für einen rechnerbasierten Baustein ist ein Krankenhausinformationssystem. Ein nicht rechnerbasierter Baustein ist z.B. ein Ablagesystem für Papierakten. Auch die Kommunikation zwischen den Systemen mit ihren Schnittstellen und verwendeten Protokollen wird hier beschrieben.
- **Die Physische Werkzeugebene:** Sie beschreibt die konkrete Hardware-Implementierung der verwendeten Anwendungen. Im Falle von rechnerbasierten Anwendungsbausteinen sind das also Netzwerkkomponenten, Server, PCs, Drucker etc.. Auch bei nicht rechnerbasierten Bausteinen können Abhängigkeiten zu Komponenten auf der physischen Werkzeugebene bestehen, z.B. ein Rohrpostsystem oder ein automatisiertes Karteiablagensystem.

Jede dieser drei Ebenen stellt für sich einen bestimmten Aspekt eines KIS konsistent dar. Sie existieren jedoch nicht unabhängig voneinander, sondern sind eng miteinander verknüpft:

Die Fachliche Ebene bedient sich der Werkzeuge auf der Logischen Werkzeugebene, um die Arbeitsabläufe durchzuführen. Die Logische Werkzeugebene wiederum setzt auf die Hardwarekomponenten in der Physischen Werkzeugebene auf.

Aus diesen im konkreten Modell abgebildeten Abhängigkeiten lassen sich ad hoc Antworten auf verschiedene Fragen finden:

- Welche Anwendungsbausteine und physischen Komponenten benötigt ein bestimmter Arbeitsablauf?
- Welche Auswirkungen hat der Ausfall einer physischen Komponente auf den Krankenhausbetrieb?

Das 3LGM ist nützlich für die Modellierung und Darstellung der Abhängigkeiten zwischen den Ebenen und zwischen Systemen.

Als weniger geeignet erweist es sich für die detaillierte Darstellung von redundanten Komponenten, Clustersystemen, Failover-Szenarien und Ähnlichem.

1.3 Verfügbarkeit

Ein System kann als verfügbar bezeichnet werden, wenn es den Anwendern die Durchführung der im Arbeitsprozess notwendigen Schritte erlaubt. Dabei folgt die Verfügbarkeit keinem „alles oder nichts“-Prinzip, es gibt auch Szenarien mit eingeschränkter Verfügbarkeit:

- Ausfall einer Lokation (z.B. Ausfall eines Teils des Krankenhausnetzwerks)
- Ausfall von bestimmten Funktionalitäten (Ausfall einer gewissen Programmfunktion)
- signifikante Verminderung der Systemleistung (höhere Antwortzeiten)

Im Einzelfall kann der Ausfall einer einzelnen, unbedeutend erscheinenden Komponente (z.B. ein Drucker für Patientenetiketten) den Ablauf einer Abteilung massiv beeinträchtigen.

Die Verfügbarkeit einer Komponente kann definiert werden als

$$V = \frac{MTBF}{MTBF + MTTR}$$

MTBF ... Mean time between failure

MTTR ... Mean time to repair

Eine Verfügbarkeit von 1 stellt also eine durchgängige Verfügbarkeit ohne Ausfälle dar. In der Praxis wird die Verfügbarkeit normalerweise in Prozent angegeben.

Tabelle 1: Verfügbarkeitsstufen

<i>Verfügbarkeit</i>	<i>Ausfallszeit pro Jahr</i>
95%	18,25 Tage
99%	3,65 Tage
99,9%	8,76 Stunden
99,99%	52 Minuten
99,999%	5 Minuten

In der Praxis ist für ein System mit der Komplexität und dem dauernden Wandel eines KIS mit vertretbarem Aufwand maximal eine Verfügbarkeit der Applikation beim Endanwender von 99,9% erreichbar. Eine weitere Erhöhung der Verfügbarkeit ist mit überproportionalem Aufwand verbunden bzw. abhängig von den verwendeten Produkten und geplanten Aktionen im Projektverlauf gar nicht möglich.

1.3.1 Ursachen von Ausfallszeiten

Die Angabe einer Prozentzahl für die Verfügbarkeit eines komplexen Systems ist eigentlich eine ungebührliche Vereinfachung, auch wenn sie wegen ihrer plakativen Wirkung oft herangezogen wird.

Für eine praxisnähere Beurteilung müssen zumindest zwei Arten von Ausfällen unterschieden werden:

1.3.1.1 Geplante Ausfallszeiten

Diese Ausfallszeiten haben folgende Eigenschaften:

- Der Ausfall ist den Endanwendern im Vorhinein bekannt, so dass sie notwendige Tätigkeiten umplanen können bzw. Notfall-Arbeitsabläufe vorbereiten können. Zum Beispiel kann eine Abteilung entscheiden, für diese Zeit keine oder weniger Patiententermine zu vergeben.
- Der Ausfall findet im Allgemeinen zu Zeiten geringer Benutzeraktivität statt, also in den Abend- und Nachtstunden bzw. am Wochenende.
- Die Abschaltung dient einem bestimmten Zweck und ist technisch notwendig.
- Alle benötigten Spezialisten sind anwesend bzw. in Bereitschaft.

Folgend die häufigsten Gründe für geplante Ausfälle:

- *Einführung einer neuen serverseitigen Software-Version.* Bei kleinen Software-Änderungen (Fehlerbehebung etc) kann die Installation oft ohne Ausfallszeit erfolgen. Das gilt insbesondere dann, wenn bei einem echten Cluster-Betrieb die Installation rollierend erfolgt und damit immer zumindest ein Knoten des Clusters aktiv ist. Bei vollständig neuen Software-Versionen erfolgt aber häufig ein Umbau in zentralen Teilen der Software-Architektur, der nicht im Betrieb erfolgen kann. Auch die im Zuge eines Upgrades notwendigen Umbauten im Datenmodell (z.B. Änderungen in der Tabellenstruktur) können zu Stillstandszeiten führen. Diese Ausfälle können aber durch den Einsatz aktueller Versionen der gängigen Datenbanksysteme oder durch den Einsatz spezieller Werkzeuge mehr und mehr eliminiert werden.

Für die Verteilung einer neuen Software-Version für den Client (nur im Falle eines *fat client*¹ nötig) ist normalerweise keine Unterbrechung notwendig, wenn der Hersteller auf Rückwärtskompatibilität Wert legt.

- *Einführung einer neuen Datenbank-Version.* Hier gilt das zuvor Gesagte. Kleinere Updates können meist rollierend installiert werden, echte Release-Wechsel erfordern eine Ausfallszeit.
- *Wartungsarbeiten an nur einfach vorhandenen Komponenten.* Das kann jede Art von Hardwarekomponente betreffen, z. B. Server, Netzwerkkomponenten, SAN² etc.. Wo aus Kostengründen oder auch aus Mängeln im technischen Konzept auf eine redundante Auslegung verzichtet wird, muss dieser Verzicht mit höheren Ausfallszeiten bezahlt werden.
- *Großflächige Infrastrukturumbauten.* Dazu zählen z. B. die Totalumstellung der Netzwerkinfrastruktur oder Umbauten an der Stromversorgung. Ein Stillstand kann für solche Szenarien nur durch geographisch entfernte redundante Komponenten verhindert werden.

1.3.1.2 Ungeplante Ausfallszeiten

Wesentlich kritischer sind in der Praxis ungeplante Ausfallszeiten.

- Sie treffen die Anwender unerwartet und mitten im Arbeitsprozess.
- Sie treten entweder in Zeiten hoher Anwenderaktivität auf und verursachen damit großflächige Probleme, oder aber in Randzeiten, wo die Fehlerbehebung durch schwerere oder fehlende Verfügbarkeit von Spezialisten länger dauert.

Die häufigsten Ursachen für ungeplante Ausfallszeiten sind:

- *Softwareprobleme.* Verursacht durch einen Fehler im Serverteil der Anwendung oder abhängiger Komponenten, z.B. Datenbanksoftware, Betriebssystem, Treiber etc.. Diesen Problemen kann kaum mit Redundanz begegnet werden.
- *Hardwarefehler.* Im Falle redundanter Auslegung sollte der Ausfall einer einzelnen Hardwarekomponente zu keinem oder nur zu einem kurzen Ausfall des Gesamtsystems führen. Ein Ausfall entsteht also durch einen der folgenden Gründe

¹ Die lokale Installation der Client-Software auf dem Arbeitsplatz-PC, im Gegensatz zum *thin client*, der über Terminalserver- oder Webtechnologie auf den Server zugreift

² Storage Area Network, eine vom einzelnen Server losgelöste Speicherplatz-Infrastruktur

- die Hardwarekomponente ist nicht redundant ausgelegt bzw. es gibt nicht berücksichtigte Abhängigkeiten von einer bestimmten Instanz der redundanten Komponente
- die Übernahme der ausgefallenen Komponente durch die redundante Komponente funktioniert nicht ordnungsgemäß

So sehr sich geplante von ungeplanten Ausfällen unterscheiden, so ähnlich sind sie sich in den Maßnahmen zu deren Verringerung.

1.3.2 Methoden zur Erhöhung der Verfügbarkeit

1.3.2.1 Verringerung von Abhängigkeiten

Wenn ein Arbeitsschritt von der Verfügbarkeit mehrerer Komponenten abhängt, ergibt sich die Gesamtverfügbarkeit als

$$V_{ges} = \prod_i V_i$$

beziehungsweise der gesamte Ausfallsquotient als

$$A_{ges} = 1 - \prod_i (1 - A_i) \approx \sum_i A_i$$

Diese Formel geht davon aus, dass die Ausfälle der betrachteten Komponenten kausal unabhängig voneinander auftreten. Die angegebene Näherung gilt nur für Verfügbarkeiten nahe 1, was in der Praxis aber gegeben ist.

Aus dieser Formel bestätigt sich die nahe liegende Vermutung, dass eine Erhöhung der Verfügbarkeit nicht zwangsweise über eine höhere Verfügbarkeit der einzelnen Komponenten führen muss, sondern auch und oft am Einfachsten über eine Verringerung der Anzahl der notwendigen Komponenten erreicht werden kann.

1.3.2.2 Redundante Auslegung von Komponenten

Hardware-Redundanz kann auf mehreren Ebenen ansetzen:

- Innerhalb eines Gerätes, z. B. durch redundante Netzteile, Festplatten etc.. Wünschenswert ist, dass diese Komponenten im laufenden Betrieb getauscht werden können (*hot plug*). Bei aktueller Server-Hardware ist diese Forderung im Allgemeinen erfüllt für Festplatten, Netzteile und Schnittstellenkarten wie Netzwerk-Interfaces, SAN-Interfaces etc.. Nur bei wenigen Herstellern sind auch Prozessoren, Hauptspeicher oder Backplanes im Betrieb austauschbar.

- Doppelte Auslegung von ganzen Geräten. Dabei gibt es folgende drei Möglichkeiten des Betriebs:
 - Active/Cold Standby: Nur eine der beiden Komponenten ist im Normalbetrieb aktiv. Im Fehlerfall muss die Standby-Komponente manuell in Betrieb genommen werden. Sinnvoll ist diese Methode nur bei Komponenten mit geringen Auswirkungen bei zeitweiser Nichtverfügbarkeit.
 - Active/Hot Standby: Primäre und sekundäre Komponente sind in Betrieb, aber nur die primäre Komponente ist aktiv. Im Fehlerfall übernimmt die sekundäre Komponente die Aufgaben (normalerweise mit kurzer Unterbrechung).
 - Active/Active Konfiguration: Beide Komponenten sind gleichzeitig aktiv und teilen sich die Last. Im Fehlerfall einer Komponente übernimmt die andere Komponente die ganze Last (ohne bzw. mit minimaler Unterbrechung). Diese Variante ist am schwierigsten zu implementieren, aber am leistungsfähigsten. Die vorhandenen Ressourcen werden optimal genutzt. Transaktionen, die zum Zeitpunkt des Ausfalls auf dem fehlerhaften System in Bearbeitung waren, schlagen fehl und müssen wiederholt werden. Diese Einschränkung trifft bis auf wenige Ausnahmen (fehlertolerante Systeme) zu.
- Doppelte Auslegung ganzer Standorte. Dazu werden Pendants zu sämtlichen benötigten Hardware-Komponenten an einem entfernten Ort redundant gehalten. Damit kann auch beim Ausfall einer ganzen Lokation der Betrieb aufrechterhalten werden. Diese Variante bietet auch einen Schutz vor Problemen wie Naturkatastrophen, längeren Stromabschaltungen oder anderen Szenarien, die einen ganzen Standort betreffen. Die Verbindung zwischen den zwei Standorten muss über eine hohe Bandbreite und Ausfallssicherheit verfügen.

Ein potentielles Problem bei doppelt ausgelegten Servern stellt der Verlust der Kommunikation zwischen den Servern dar. Das kann auch bei Servern am selben Standort auftreten, ist aber vor allem bei entfernten Standorten eine Gefahr. In diesem Fall kann es zu einer so genannten „split-brain“-Situation kommen, bei der beide Partner annehmen, dass das andere System ausgefallen sei und beide unabhängig voneinander Transaktionen auf einer getrennten Kopie der Datenbank ausführen. Diese Situation kann zu nicht mehr reversiblen Dateninkonsistenzen führen und muss daher unbedingt vermieden werden. Dazu gibt es im Wesentlichen zwei Möglichkeiten:

- Verwenden mehrerer, möglichst unabhängiger Verbindungen zur Übertragung des *heartbeat*³, z. B redundante Netzwerkverbindung über (TCP/IP), serielle Verbindungen, Modemverbindung etc..
- Verwenden von drei- oder mehrfach vorhandenen Komponenten, wobei eine Mehrheit der Systeme (das *Quorum*) die Entscheidung über die Verfügbarkeit eines Kommunikationspartners trifft.

1.3.2.3 Bereithalten von Ausfallsszenarien

Trotz aller Bemühungen kann eine Verfügbarkeit von 100% mit vertretbarem Aufwand nicht erreicht werden. Daher müssen für die wichtigsten Prozesse Ausfallsszenarien bzw. alternative Arbeitsabläufe ausgearbeitet werden. Diese Ausfallsszenarien müssen regelmäßig getestet werden, damit sie im Ernstfall funktionieren.

Sie erhöhen zwar nicht direkt die Systemverfügbarkeit, sehr wohl jedoch die „Verfügbarkeit“ der Arbeitsprozesse, die letztlich entscheidend ist.

³ Als *heartbeat* bezeichnet man zwischen Cluster-Systemen ausgetauschte Datenpakete, mit denen sich die Systeme gegenseitig über ihre Verfügbarkeit informieren.

1.4 Monitoring

Als *Monitoring* wird hier die kontinuierliche Überwachung von Systemparametern bezeichnet, die Auswirkungen auf die Verfügbarkeit von Hardware- und Softwarekomponenten haben.

Im Unterschied zu nachträglichen Auswertungen erfolgt das Monitoring kontinuierlich oder zumindest zeitnah. Die Frequenz, mit der einzelne Prüfungen durchgeführt werden, muss dabei individuell festgelegt werden. Sie ist abhängig von

- der Wichtigkeit des überprüften Objekts auf die Gesamtverfügbarkeit
- der Belastung und damit Leistungsverminderung, die durch die Prüfung am Gesamtsystem entsteht
- dem Zeitintervall, in dem ein Problem von der Erkennung erster Anzeichen zu einem kritischen Faktor werden kann

1.4.1 Aufgaben und Nutzen eines Monitoring-Systems

Der Antrieb für den Einsatz eines Monitoring-Systems erfolgt meist aus einem vorhandenen Druck zur Qualitätssteigerung bei geringen Kosten.

Die manuelle Überwachung von Systemen hat einige gravierende Nachteile:

- Da Überprüfungen nur in gewissen Intervallen durchgeführt werden können, vergeht unter Umständen ein großer Zeitraum bis zur Erkennung eines Problems.
- Die Überwachung ist arbeitszeitintensiv und monoton.
- Sie ist abhängig von der aktiven Durchführung durch Mitarbeiter. Wenn eine durchgehende Überwachung gewünscht wird, setzt das einen Schichtbetrieb durch entsprechende Spezialisten voraus.
- Die Dokumentation der Prüfungsergebnisse erzeugt zusätzlichen Aufwand.

Die Einführung eines Monitoring-Systems stellt daher einen klassischen Fall der Automatisierung wiederkehrender Tätigkeiten dar.

Folgendes sind die Hauptzielsetzungen, die ein Projekt zur Einführung eines automatisierten Monitoring verfolgt:

- Sämtliche Probleme, die zu einem Stillstand oder einer Beeinträchtigung des Systembetriebs führen können, sollen zeitnah erkannt werden.
- Nach Erkennung sollen regelgesteuert aktive Benachrichtigungen an entsprechende Personen versandt werden. Insbesondere außerhalb der normalen Bürostunden sollen

- die entsprechenden Personen unter Umgehung des langwierigen konventionellen Benachrichtigungswegs sofort über Probleme informiert werden.
- Die gesamte Zeitspanne vom Auftreten eines Problems bis zu dessen Behebung soll minimiert werden. Im Idealfall wird an der Behebung eines Problems bereits gearbeitet, bevor die ersten Anrufe von Endanwendern die Hotline erreichen.
- Die Gesamtverfügbarkeit des Systems und damit die Benutzerzufriedenheit und Produktivität soll erhöht werden.
- Mit geringem Aufwand sollen Statistiken über die Systemverfügbarkeit generierbar sein.

Eine umfassende Überwachung kann folgende Bereiche einbeziehen:

- Überprüfung der Verfügbarkeit
- Zentralisierte Sammlung, Anzeige und eventuell automatische Klassifizierung der Fehlermeldungen, die von den beteiligten Einzelkomponenten generiert werden
- Protokollierung der quantitativen Systemdaten (Performance-Messung, Speicherplatzbedarf, Auslastung etc.)
- Berechnung und evtl. automatische Bewertung von Performancetrends mit Generierung von entsprechenden Warnungen

1.4.2 Komponenten eines Monitoring-Systems

Trotz der verschiedenen Ansätze und Produktphilosophien der Hersteller ist allen Monitoring-Systemen die grundsätzliche Funktionsweise gemein. Es lassen sich folgende Komponenten unterscheiden:

1.4.2.1 Netzwerkmonitor

Diese Komponente überprüft die Verfügbarkeit von Diensten, indem sie über das Netzwerk Verbindung zu der zu überprüfenden Komponente aufbaut. Damit kann einerseits die generelle Verfügbarkeit einer Hardwarekomponente, andererseits die Funktionsfähigkeit von Diensten inklusive deren Erreichbarkeit über das Netzwerk geprüft werden.

1.4.2.2 Agenten

Als Agenten bezeichnet man Softwarekomponenten, die auf dem zu überprüfenden System selbst installiert werden. Sie führen Prüfungen lokal durch und melden die Prüfergebnisse an den Sammeldienst (s. 1.4.2.3). Das hat den prinzipiellen Nachteil, dass damit ein Eingriff in das System notwendig ist und eine zusätzliche potentielle Problemquelle auf dem Produktivsystem eingeführt wird. Allerdings überwiegen die Vorteile. Durch die Ausführung

auf dem überwachten Server selbst stehen dem Agenten Möglichkeiten offen, die aus technischen oder Sicherheitsgründen über das Netzwerk nicht zur Verfügung stehen.

Agenten müssen in einer spezifischen Version für das Betriebssystem des überwachten Systems verfügbar sein, was bei selteneren Derivaten oft problematisch ist.

1.4.2.3 Sammeldienst

Der Sammeldienst nimmt Statusnachrichten von Netzwerkmonitoren und Agenten entgegen und sammelt sie zentral. Dieser Dienst ist auch für die persistente Speicherung der Statusmeldungen zuständig, um die nachträgliche Erstellung Verfügbarkeits-Statistiken zu ermöglichen. Die Speicherung kann dabei in gewöhnlichen Dateien oder in einer Datenbank erfolgen.

1.4.2.4 Anzeigedienst

Der Anzeigedienst ist meist mit dem Sammeldienst integriert und generiert aus den gesammelten Statusdaten eine menschenlesbare Übersicht. Der Anzeigedienst kommuniziert mit dem Monitoring-Client.

1.4.2.5 Monitoring-Client

Der Client stellt die primäre Benutzerschnittstelle für das gesamte Monitoring-System dar. Wenn der Anzeigedienst einen HTTP-Server enthält, kann als Client ein beliebiger Web-Browser verwendet werden. Zusätzlich stehen oft dedizierte Clients zur Verfügung, die mit dem Anzeige- bzw. Sammeldienst über eine proprietäre Schnittstelle kommunizieren und im Funktionsumfang weiter gehen.

1.4.2.6 Benachrichtigungsdienst

Dieser Dienst bewertet empfangene Statusmeldungen und generiert anhand eines Regelsatzes verschiedene Benachrichtigungen wie e-Mail, SMS etc.. Die Flexibilität in der Definition der Benachrichtigungsregeln stellt ein wichtiges Unterscheidungskriterium bei der Auswahl eines Produkts dar.

Bei der Definition von Benachrichtigungsregeln sollten mindestens folgende Parameter verwendet werden können:

- Filterung nach Server, Dienst und Status
- Uhrzeit, Wochentag, hinterlegbarer Feiertagskalender
- Zeitspannen und Eskalationswege bei länger dauernden Problemen

Objekte und Benachrichtigungsziele sollen in Gruppen zusammengefasst werden können, um die Regeldefinition übersichtlicher zu gestalten. Ein Beispiel für einen möglichen abzubildenden Regelsatz wäre:

- Wenn einer der zentralen KIS-Server über das Netzwerk nicht erreichbar ist, schicke eine E-Mail an das Systemteam.
- In der Nacht und am Wochenende schicke zusätzlich eine SMS an den Bereitschaftshabenden.
- Wenn das Problem nach einer Stunde noch immer besteht, verständige die Projektleitung per Mail und SMS.

2 Methoden und Ergebnisse

2.1 Einleitung

Die vorliegende Arbeit entstand im Rahmen eines Projekts zur Einführung eines Krankenhaus-Informationssystems an den Anstalten der Tiroler Landeskrankenanstalten GmbH (Tilak).

Die Tilak ist der primäre Gesundheitsdienstleister im Raum Tirol.(s. Tabelle 2)

Tabelle 2: Tilak Leistungsdaten 2001

Leistungsdaten 2001 *	
Betten (systemisiert)	2.167
Betten (tatsächlich)	2.046
davon Intensivbetten	99
Auslastungsgrad	79,1 %
Stationäre Aufnahmen	91.908
Pflegetage	682.324
Frequenz in Ambulanzen	1,5 Mio.
LDF-Punkte (in Mio.)	3.560

Quelle: www.tilak.at (7.12.2003)

Das Projekt „KIS“ wurde im Jahr 1998 gestartet, die erste Inbetriebnahme erfolgte im Herbst 1999. Inzwischen ist ein Folgeprojekt in Gange, in dem weitere Funktionalitäten der Software eingeführt werden sollen.

Im Zuge des Projektstarts kam es zur Gründung einer externen Beraterfirma, die die Implementierungsdienstleistungen erbringt und unter anderem auch für die Konzepte zum Systembetrieb verantwortlich ist. Der Betrieb der für das System notwendigen Server wurde an einen externen Rechenzentrumsbetreiber vergeben.

Das KIS wurde auch als Ausgangspunkt für das Monitoring-Projekt gewählt. Es vereint die notwendige Komplexität, den großen Anwenderkreis und die hohen Anforderungen an die Verfügbarkeit, die die Einführung eines automatisierten Monitoring sinnvoll machen.

Inzwischen wurde die Lösung auch auf das Abteilungssystem der psychiatrischen Klinik sowie auf die interne IT der für die Einführung des KIS verantwortlichen Beratungsfirma ausgeweitet. Auch diverse Einzelsysteme, die am Rande für das KIS interessant sind (z. B. Gateway zum elektronischen Befundversand, Mailserver) werden mit überwacht.

2.2 Auswahl eines Monitoring-Tools

Es gibt eine Reihe von kommerziellen Monitoring-Lösungen großer Systemhersteller. Die bekanntesten sind hier

- BMC PATROL
- Computer Associates Unicenter
- HP OpenView
- IBM Tivoli

Die Anforderungen an das Monitoring-System in unserem konkreten Einsatzfall waren:

- schnelle Implementierung
- organisationsübergreifende Überwachung von Hardware, da einige zentrale Komponenten von einem externen Rechenzentrum (ARZ) betreut werden
- geringe Kosten
- hohe Flexibilität und Erweiterbarkeit

Unter Abwägung aller Aspekte fiel die Wahl auf die Lösung Big Brother [1]. Diese Software ist in den meisten Fällen lizenzfrei zu benutzen, relativ verbreitet, und es existieren viele Zusatz-Tools und Monitoring-Scripts, die von anderen Benutzern kreiert und zur Verfügung gestellt wurden. Außerdem ermöglichte diese Entscheidung einen schnellen, unbürokratischen Aufbau eines Monitoring-Systems ohne die oft langwierige Entscheidungsfindung bei der unternehmensweiten Einführung eines kommerziellen Produkts.

2.3 Definition der überwachten Komponenten

Um die Aufgabe der Überwachung eines komplexen Systems überschaubar zu machen, empfiehlt es sich, die Vielzahl der zu überwachenden „Einzelteile“ zu strukturieren. Dabei bietet sich einerseits die Einteilung nach der physischen Hardwareeinheit an. Hier wurde allerdings versucht, die Strukturierung vor allem anhand der logischen Ebenen vorzunehmen, auf denen Monitoring stattfinden kann. Dabei haben sich in unserem konkreten Fall folgende Ebenen herauskristallisiert:

- Hardware und Netzwerk
- Betriebssystem
- Datenbank
- Dienste
- Applikation

Diese grobe Einteilung soll im Folgenden detaillierter dargestellt werden.

2.3.1 Hardware und Netzwerk

Auf dieser Ebene wird die grundsätzliche Verfügbarkeit einer Hardwarekomponente überprüft. Im einfachsten Fall wird dazu die Erreichbarkeit über das Netzwerk mittels *ping*⁴ getestet. Wenn sich eine Komponente nicht „anpingen“ lässt, kann davon ausgegangen werden, dass auch sämtliche Funktionen auf höherer Ebene nicht zur Verfügung stehen⁵.

Ein Fehlschlagen dieses *ping*-Tests kann verschiedene Ursachen haben:

- Stromausfall / Netzteildefekt
- Defekt von zentralen Komponenten wie CPU / Hauptspeicher
- Defekt einer Netzwerkkarte bzw. Netzwerkkabel
- Totalabsturz des Betriebssystems

Da sich zwischen der überwachenden und der überwachten Einheit meistens noch Zwischenstationen befinden (Router, Switches), ist es wichtig, auch diese zu überprüfen. Nur so kann man bei einem Fehlschlagen des *ping*-Tests zwischen einem Ausfall des überwachten Gerätes und einem Ausfall der Verbindungsstrecke unterscheiden.

⁴ Das Kommando *ping* setzt eine Verfügbarkeit des TCP/IP-Netzwerkprotokolls auf allen überwachten Geräten voraus. Da sich TCP/IP aber immer mehr als alleiniges Netzwerkprotokoll durchsetzt, stellt dies keine Einschränkung dar.

⁵ Eine Ausnahme stellen Umgebungen mit internen Firewalls dar, die auf Grund ihrer Konfiguration die von *ping* verwendeten Netzwerkpakete selektiv filtern

Natürlich kann eine Nichtverfügbarkeit über TCP/IP auch in Softwareproblemen begründet sein (z. B. extreme Überlastung des Servers, Denial-of-Service-Attacke, Fehlkonfiguration des TCP/IP-Stacks). Das stellt jedoch eher die Ausnahme dar.

Auch wenn eine Komponente über das Netzwerk erreichbar ist, können trotzdem partielle oder latente Hardwareprobleme vorhanden sein (z. B. der Ausfall eines von zwei redundanten Netzteilen). Diese lassen sich über die Auswertung systeminterner Prüfungen ermitteln (siehe Abschnitt 2.3.2.5)

2.3.2 Betriebssystem

Auf dieser Ebene erfolgt die Überwachung des „Gesundheitszustandes“ der primären Grundlagen des Betriebssystems. Die Ermittlung dieses Zustands wird von lokal installierten Monitoring-Agenten übernommen.

Im Einzelnen wurden folgende Tests verwendet:

2.3.2.1 CPU-Belastung

Hier geht es um die Erkennung einer zu hohen Prozessorlast auf einem Server.

Die Lastkurve der CPUs der Applikations- und Datenbankserver folgt einem tageszeitlichen Verlauf, der ziemlich genau die Benutzeraktivität widerspiegelt. Zusätzlich gibt es noch einige Spitzen in der Nacht durch nächtlich laufende Batch-Jobs und die Datenbanksicherung.

Die gemessene CPU-Last kann in zwei Anteile zerlegt werden:

- Kernel- und Benutzerprozesslast: Die eigentlich beanspruchte Rechenleistung
- Wartezyklen, in denen auf die Beendigung von Ein-/Ausgabevorgängen gewartet wird

Eine Überlastung kann mehrere Ursachen haben:

- Das System wird gerade zusätzlich belastet, z. B. durch eine Auswertung
- Ein oder mehrere Prozess belegen aufgrund eines fehlerhaften Zustands (z. B. Endlosschleife) eine der vorhandenen CPUs zu 100%
- Im Falle einer regelmäßigen Überlastung unter normalen Umständen muss eine Aufrüstung der CPU-Leistung erwogen werden.
- Bei einem großen Anteil von Wartezyklen ist meist eine Überlastung des Festplatten-Subsystem die Ursache, z. B. wegen massiver Belastung der Datenbank durch Auswertungen o. ä.

2.3.2.2 Speicherauslastung

Alle aktuellen Betriebssysteme verwenden neben dem physikalisch vorhandenen Hauptspeicher virtuellen Speicher, der über Auslagerung von Speicherseiten auf Festplatten bereitgestellt wird.

Der Zugriff auf ausgelagerte Speicherseiten ist dabei um mehrere Zehnerpotenzen langsamer als der Zugriff auf den Hauptspeicher.

Das Auslagern von selten genutzten Speicherbereichen stellt für sich kein Problem dar. Müssen jedoch häufig zugegriffene Speicherseiten immer wieder aus- und eingelagert werden, kann die Systemleistung bis zum Beinahe-Stillstand sinken.

Dieser Speichermangel kann primär zwei Ursachen haben:

- der Hauptspeicherausbau ist zu gering
- Aufgrund von Fehlern im Programmcode fordert ein Prozess im Laufe des Betriebs immer mehr Speicher an („*memory leak*“). Solange das Problem vom Hersteller nicht behoben ist, bieten sich zwei Behebungsmöglichkeiten an:
 - regelmäßiger automatisierter Neustart des entsprechenden Prozesses
 - Limitierung des dem Prozess zur Verfügung gestellten Hauptspeichers. Eine Überschreitung führt allerdings in Folge meist zum Abbruch des Prozesses durch das Betriebssystem.

2.3.2.3 Verfügbarer Plattenplatz

Den größten Bedarf an Plattenplatz hat in einem Informationssystem naturgemäß die zugrunde liegende Datenbank. Dieser Plattenplatz wird gesondert überwacht (siehe Kapitel 2.3.3.2).

Darüber hinaus benötigen aber auch andere Prozesse laufend Plattenplatz, z. B. zur Speicherung von Log-Files, Auswertungen und sonstigen Daten. Ein Überlauf eines Dateisystems kann daher im schlimmsten Fall bis zum Stillstand des gesamten KIS führen.

Aus diesem Grund wurde der Füllungsgrad der Dateisysteme in die kontinuierliche Überwachung einbezogen. Ab 90% Füllungsgrad wird eine Warnung ausgegeben, ab 95% wird der Zustand als kritisch eingestuft.

Gewisse Dateisysteme wurden von der Überprüfung ausgenommen, wie z. B. ein CD-ROM-Laufwerk (eine finalisierte CD ist per definitionem immer zu 100% voll). Auch die

Überprüfung von NFS⁶-Laufwerken stellte sich als nicht sinnvoll heraus, da bei Nichtverfügbarkeit des entsprechenden NFS-Servers die Überprüfungsroutine „hängt“.

Entsprechende Plattenplatz-Überwachungsscripts wurden implementiert für Windows, UNIX und (in rudimentärer Form) für Novell NetWare.

2.3.2.4 Vorhandensein bestimmter Prozesse bzw. Dienste

Als komplementäre Methode, um die Verfügbarkeit des KIS zu überprüfen, werden die zentralen Betriebssystemprozesse, die die Applikation darstellen, auf ihr Vorhandensein überprüft. Einige Prozesse sollen in genau einer Instanz vorhanden sein, während andere (meist zum Zweck der Lastverteilung) mehrfach vorhanden sind. In diesem Fall wird überprüft, ob sich die Anzahl in einem gewissen Bereich befindet (auch eine Überzahl von Prozessen weist auf ein Problem hin).

Auch eine Anzahl von Prozessen außerhalb der Applikationsserver werden überprüft, wie z.B. Oracle- und WebSphere MQ⁷-Prozesse.

Das Fehlen bestimmter Prozesse kann zum Stillstand des KIS führen.

2.3.2.5 Auswertung von Fehlerprotokollen

Viele Komponenten eines Systems schreiben aufgetretene Fehler in eine Log-Datei. Einige dieser Logs wurden in die Überprüfung mit aufgenommen.

- Errpt-Log des AIX-Betriebssystems: hier werden Fehler vom Hardwarelevel bis zu Betriebssystemproblemen aufgezeichnet
- System Log der Windows Server. Hier werden ebenfalls Probleme auf verschiedenen Ebenen von Hardware und Betriebssystem vermerkt
- Oracle-Alertlog (siehe Kapitel 2.3.3.1)
- Applikationsspezifische Logfiles

Ein prinzipielles Problem bei der Überwachung von Log-Files besteht darin, dass in einem Log-File *Ereignisse* aufgezeichnet werden, während in unserem Monitoring-Tool *Zustände* dargestellt werden. Daher muss in irgendeiner Form eine Wandlung Ereignis → Zustand stattfinden.

Für diesen Zweck wurde die Visual Basic-Applikation *LogCheck* entwickelt, die als benutzerdefinierter Test in den Big Brother Client für Windows integriert werden kann.

⁶ NFS: Network File System, ein unter UNIX üblicher Standard zum Zugriff auf Dateien über ein Netzwerk

⁷ WebSphere MQ ist ein Produkt zur asynchronen Prozesskommunikation

Diese Applikation liest in einem Konfigurationsfile definierbare Log-Files und überprüft sie auf das Vorhandensein von bestimmten Zeichenketten, wobei zwischen Zeichenketten mit Warnstufe und solchen mit Fehlerstufe unterschieden werden kann.

Das Auftreten einer dieser Zeichenketten führt dazu, dass die Prüfroutine für eine je nach Zeichenkette einstellbare Zeitspanne den Zustand „Warnung“ bzw. „Fehler“ an Big Brother meldet.

2.3.3 Datenbank

Im Zentrum der meisten größeren Informationssysteme steht eine Datenbank. In unserem Fall findet Oracle[®] 8i Verwendung, eines der am weitesten verbreiteten RDBMS⁸. Da ein Stillstand der Datenbank auch den Totalausfall des KIS bedeutet, wurden hier einige spezielle Prüfmaßnahmen eingebaut. Da diese Routinen nicht Bestandteil von Big Brother sind, mussten sie selbst entwickelt werden.

2.3.3.1 Allgemeine Datenbankfehler

Oracle zeichnet kritische Ereignisse im so genannten *alert log* auf. Da dieses Log ein einfaches Textfile ist, das außerhalb der Datenbank existiert, kann es auch ausgewertet werden, wenn ein Zugriff auf die Datenbank nicht möglich ist.

Hier wurde zur Überprüfung ein einfacher Ansatz verfolgt: Wenn sich in den letzten 50 Zeilen des Logs ein Fehler befindet (erkennbar an der Zeichenfolge **ORA-**), wird an Big Brother der Status „Fehler“ übergeben. Diese Festlegung auf eine gewisse Zeilenanzahl hat sich in der Praxis im Falle des *alert logs* nicht als Nachteil herausgestellt, da dieses Log relativ kontinuierlich fortgeschrieben wird.

Wie kritisch ein Fehlereintrag tatsächlich ist, lässt sich automatisiert kaum ermitteln, da dies von vielen Faktoren abhängt. Insbesondere im Fall von internen Datenbankfehlern kann die Relevanz oft nur vom Datenbankhersteller selbst beurteilt werden.

In jedem Fall stellt ein Eintrag im *alert log* ein Ereignis dar, das der genauen Abklärung bedarf.

2.3.3.2 Füllungsgrad der Datenbank

Von seiner Konzeption her speichert Oracle seine Daten in *tablespaces*, die ihrerseits aus Dateien bestehen. Diese Dateien werden in vielen größeren Oracle-Installationen als *raw*

⁸ RDBMS: Relationales Datenbank-Managementsystem

devices angelegt, die die Dateisystem-Routinen des Betriebssystems umgehen. Diese Dateien werden mit einer fixen Größe angelegt und können nicht automatisch vergrößert werden.

Wenn im Laufe des Betriebs die Tabellen und Indices der Datenbank anwachsen, füllen sich diese Dateien mehr und mehr.

Ist eine Datei voll, kann für neu hinzukommende Daten kein Platz mehr alloziert werden. Das führt dazu, dass entsprechende Transaktionen abgebrochen werden.

Eine Aufgabe des Datenbankmanagements ist es, den zur Verfügung stehenden Platz zu überwachen und Erweiterungen proaktiv vorzunehmen.

Für die Überwachung wurde ein Skript entwickelt, das den Füllungsgrad aller Tablespaces überwacht und eine Warnung an Big Brother meldet, sobald ein Tablespace zu mehr als 90% befüllt ist.

2.3.3.3 Datenbanksperren

Wenn ein Benutzer bzw. Prozess in der Datenbank Änderungen vornimmt, sind die entsprechenden Datensätze vor gleichzeitiger Änderung durch andere Prozesse geschützt, bis der erste Prozess seine Transaktion abgeschlossen hat.

Der zweite Prozess wird bis zu diesem Zeitpunkt unterbrochen und erst fortgesetzt, wenn die Sperre aufgegeben wird, indem der erste Prozess die Transaktion bestätigt (*commit*) oder die Änderungen rückgängig macht (*rollback*).

Wenn ein Prozess aufgrund eines Programmfehlers seine Transaktion nie beendet, kann dies daher zum Stillstand der ganzen Applikation führen. Dieser Zustand kann besonders leicht entstehen, wenn im System vom Implementierungsteam auch interaktive Datenbanksitzungen verwendet werden und ein Benutzer den Abschluss seiner Transaktion vergisst.

Aus diesem Grund wurde für diese Art von Fehlern ein eigenes Überwachungs-Skript entwickelt, das entsprechende Fehlerbedingungen entdeckt und an Big Brother meldet (siehe Anhang).

2.3.4 Dienste

Als Dienste im engeren Sinn bezeichnen wir in diesem Kontext Prozesse, die auf einem Server laufen und auf Anfrage von Clients oder auch anderen Server-Prozessen gewisse Leistungen anbieten. Im häufigsten Fall erfolgt die Verbindung zu einem Dienst über TCP/IP zu einem bestimmten IP-Port des Servers.

In unserem Fall wurde auf den Datenbank- und Applikationsservern die Überwachung folgender Dienste implementiert:

2.3.4.1 Telnet (Terminaldienst)

Der Telnet-Dienst erlaubt den Aufbau von Benutzersitzungen zu Wartungszwecken. Ein Ausfall hat auf den laufenden Betrieb zwar keinen Einfluss, zeigt aber normalerweise tiefer liegende Probleme an.

Das Rechenzentrum, das die Server für das KIS beherbergt, hat den Telnet-Zugang vor einiger Zeit bis auf wenige Ausnahmen (Fernzugriff durch die Herstellerfirma) aus Sicherheitsgründen vollständig gesperrt, da dieses Protokoll potentiellen Angreifern viele Angriffspunkte bietet (unverschlüsselte Übertragung von Passwörtern etc.). Daher wurde auch die Überprüfung dieses Dienstes deaktiviert, da sie konstant einen Fehlerzustand meldete. Statt dessen wird nun SSH verwendet, eine Möglichkeit zum gesicherten Aufbau von Benutzersitzungen mit verschlüsselter Übertragung (siehe Abschnitt 2.3.4.3).

2.3.4.2 FTP (file transfer-Dienst)

Der FTP-Dienst wird in unserem Fall für die Übertragung von Daten zwischen verschiedenen Systemen verwendet. Dabei geht es im laufenden Betrieb primär um Daten, die sich nicht über den HL7-Standard abbilden lassen und daher auch nicht über den zentralen HL7-Kommunikationsserver geroutet werden.

Außerdem wird er für die Dateiübertragung zu Wartungszwecken durch das Implementierungsteam verwendet.

Ein Ausfall dieses Dienstes führt nicht zu sofortigen Problemen, sondern zu nicht erfolgreichem Datenaustausch mit gewissen Fremdsystemen.

Auch dieser Dienst wird schrittweise von der sicheren Variante SFTP abgelöst.

2.3.4.3 SSH / SFTP

Diese Protokolle sind die sicheren Pendanten zu Telnet bzw. FTP. Ein Ausfall führt zu den jeweils dort angeführten Problemen. Die Verfügbarkeit des SSH-Dienstes wurde in das Monitoring aufgenommen, da ein Ausfall meist gravierende Systemprobleme anzeigt.

2.3.4.4 HTTP und HTTPS (Webserver-Dienst)

Diese Dienste spielen auf den Applikationsservern derzeit keine Rolle. Bei einer mittelfristig geplanten Web-Benutzerschnittstelle zu den KIS-Applikationen werden diese Dienste allerdings von zentraler Wichtigkeit sein.

2.3.4.5 Oracle Listener

Dieser Dienst stellt die Schnittstelle zum Zugriff auf die Oracle[®]-Datenbank des KIS über das Netzwerk dar. Eine Nichtverfügbarkeit bedeutet, dass sich Applikationsserver-Prozesse nicht zur Datenbank verbinden können und führt daher zum Stillstand des Systems.

Andererseits zeigt die Verfügbarkeit des Listener-Diensts allein nicht an, dass die Datenbank funktionsfähig ist. Für die genauere Analyse des Datenbankstatus werden die im Kapitel 2.3.3 beschriebenen Tests verwendet.

2.3.4.6 LPR (Druckdienste)

Der LPR-Dienst ist unter UNIX[®] die gebräuchlichste Form des Druckens über das Netzwerk. Auch die Client-PCs und Druckerserverboxen stellen diesen Dienst für Ausdrücke zur Verfügung.

Die Überwachung des LPR-Dienstes wurde bei einem dedizierten Druckserver implementiert. Ein Stillstand dieses Servers hat zur Folge, dass keine vom Applikationsserver generierte Druckaufträge abgearbeitet werden können, was im Arbeitsablauf zu massiven Problemen führen kann.

2.3.4.7 DNS (Namensauflösung)

Eine zentrale Komponente jedes IP-basierenden Netzwerks ist die Namensauflösung über DNS. Dieser Dienst ist für die Lokalisation der Server durch die Clients einerseits und die Kommunikation der Server-Knoten im Cluster andererseits unverzichtbar.

Eine Nichtverfügbarkeit hat den Stillstand des KIS zur Folge.

Da diese Komponente so zentral ist, wird eine mehrfach redundante DNS-Infrastruktur verwendet. Sämtliche verwendete DNS-Server wurden in die Überwachung mit einbezogen.

2.3.4.8 Applikationsspezifische Dienste

Die Kommunikation von KIS-Clients zu den Applikationsserver-Prozessen läuft in unserem Fall über ein proprietäres Protokoll. Trotz Unkenntnis der Interna dieses Protokolls kann dennoch überprüft werden, ob die vom Protokoll verwendeten IP-Ports verfügbar sind.

Eine Nichtverfügbarkeit hat einen vollständigen Stillstand des KIS zur Folge.

2.3.5 Applikationsebene

2.3.5.1 Serverseitige Überwachung der Applikation

Das betrachtete KIS ist serverseitig als eine Anzahl von Applikationsserver-Prozessen implementiert, die auf eine Datenbank zugreifen.

Der Hersteller der KIS-Software liefert einige Kommandozeilen-Werkzeuge mit, mit denen gewisse Statusinformationen und Statistiken über den Zustand der Serverprozesse ermittelt werden können. Diese Werkzeuge sind allerdings für den interaktiven Einsatz gedacht und für die Einbindung in eine Monitoring-Lösung in dieser Form nicht brauchbar. Daher mussten für die Verwendung dieser Werkzeuge *wrapper*-Scripts erstellt werden, die sie mit Parametern und Benutzereingaben versorgen und aus der generierten Bildschirmausgabe einen Status extrahieren.

Für diesen Zweck wurden Scripts in der Sprache *Expect* [10] verwendet. Expect ist eine Erweiterung der Sprache *Tcl* [11], die sich besonders für die Simulation von Benutzersitzungen in interaktiven Kommandozeilenprogrammen eignet. Es erlaubt, abhängig von der Bildschirmausgabe eines Programms Benutzereingaben zu simulieren. Damit können auch Kommandozeilen-Programme, die nicht für einen Aufruf im Batch-Modus vorgesehen sind, automatisiert verwendet werden.

Mit Hilfe dieser Werkzeuge wurden unter anderem Scripts für die Entdeckung folgender Fehlerzustände entwickelt:

- Ein Applikationsserver-Prozess läuft nicht
- Ein Applikationsservice bearbeitet Anfragen nicht so schnell, wie sie eintreffen (die zugehörige Warteschlange kann nicht mehr abgebaut werden)
- Das Subsystem zur asynchronen Nachrichtenverarbeitung kann Nachrichten nicht in genügender Geschwindigkeit abbauen
- Eine Schnittstelle zu einem Fremdsystem kann keine Verbindung aufbauen
- Eine Schnittstelle kann Nachrichten nicht schnell genug verarbeiten
- über eine Schnittstelle treffen über längere Zeit keine Nachrichten ein, obwohl das zu erwarten wäre

2.3.5.2 Clientseitige Überwachung der Applikation

Die höchste Ebene, auf der Monitoring sinnvoll ist, ist das clientseitige Testen auf Applikationsebene. Diese Tests sind sehr spezifisch für eine bestimmte Anwendung und oft sogar für eine bestimmte Installation.

Diese Art von Tests hat den Vorteil, dass sie eine echte „End-to-End“-Überwachung darstellt. Aus Verfügbarkeit der zu Grunde liegenden Komponenten auf die Verfügbarkeit der Applikation für den Endanwender zu schließen, ist nämlich in der Praxis nicht zulässig, da eine vollständige Erfassung und Überwachung aller notwendigen Bedingungen nicht möglich ist.

Es gibt prinzipiell zwei Möglichkeiten für die Überwachung auf Applikationsebene:

- Eine spezielle Test-Applikation, die ohne Benutzerschnittstelle auskommt. Sie imitiert dabei in der Kommunikation mit der serverseitigen Applikation eine typische Benutzerinteraktion, indem sie die selben Funktionsaufrufe verwendet, die auch von den „echten“ Client-Applikationen benutzt werden.
- Die Steuerung einer echten Applikation mithilfe von kommerziellen Test-Werkzeugen. Diese Werkzeuge simulieren im Wesentlichen Tastatur- und Mausektionen abhängig von gewissen Ereignissen wie z. B. Erscheinen eines Fensters und extrahieren aus der Bildschirmausgabe der Applikation Werte, die in einen Überwachungsstatus übersetzt werden.

An beide Arten von Testprogrammen werden folgende Anforderungen gestellt:

- Der serverseitigen Software soll durch die Testapplikation eine normale Benutzeraktion vorgespiegelt werden.
- Es sollen keine unerwünschten Seiteneffekte auf die produktiven Daten entstehen. Insbesondere muss bei schreibenden Zugriffen auch die Anhäufung von „Datenmüll“ vermieden werden. Wenn zum Beispiel alle 5 Minuten ein Dokument zu einem Testpatienten erzeugt wird, entstehen dadurch im Jahr über 100.000 Testdokumente. Abgesehen vom Platzbedarf kann das auch zu Artefakten führen (z. B. übermäßig lange Antwortzeiten bei diesem Testpatienten oder Überschreitung von bestimmten Limits).
- Dem für die Tests verwendeten Benutzerkonto sollten nur jene Rechte zugewiesen werden, die für die Durchführung der Tests notwendig sind. Zugangsdaten (Benutzername/Kennwort) für die Anmeldung an der Echtumgebung müssen möglichst geschützt hinterlegt sein, um missbräuchliche Verwendung durch physische Personen zu vermeiden.
- Der Tests soll einfach, aber repräsentativ sein. Beispiel:
 - Start der KIS-Applikation
 - Suche eines Patienten
 - Beenden der Applikation
- Die Test-Applikation soll robust gegenüber zu erwartenden Fehlern sein und in jedem Fall in definierter Zeit ein Ergebnis liefern. Eine Situation, in der die „echte“ Applikation nicht mehr reagiert, soll bei der Testapplikation zu einem definierten Abbruch und zur Rückgabe des entsprechenden Fehlercodes an die Überwachungssoftware führen.

Um diese Forderungen erfüllen zu können, wurde eine Testapplikation in Visual Basic entwickelt, die eine kurze Benutzersitzung imitiert.

Eine Schwierigkeit stellte dabei die nicht vorhandene Dokumentation des Herstellers über Programmschnittstellen für die proprietäre Client-Server-Kommunikation dar. Deshalb mussten nach dem trial-and-error-Prinzip die korrekten Aufrufe der entsprechenden Funktionen ermittelt werden.

Die Testapplikation simuliert folgende Schritte:

- Start einer KIS-Applikation
- Benutzerauthentifizierung
- Durchführung einer Datenbankabfrage
- Abmeldung und Beenden

Bei der Entwicklung wurde besonders darauf Wert gelegt, im Falle von Fehlerbedingungen immer in einer definierten Zeit ein Testergebnis zu liefern. Auch im Falle von unmäßig hohen Antwortzeiten, Timeouts, Fehlerabbrüchen o. ä. wird ein entsprechender Status zurückgeliefert.

2.4 Festlegung der Benachrichtigungen

Die automatisierte Überwachung von Systemen ist nutzlos, wenn eine Fehlerbedingung zwar vom System erkannt wird, aber niemand darauf reagiert. Daher sind Lösungen nicht akzeptabel, die nur darauf aufbauen, dass ein Operator gelegentlich einen Blick auf eine Konsole wirft, auf der Fehlermeldungen aufscheinen. Eine aufgetretene Fehlerbedingung soll möglichst *sofort* zum Einleiten einer entsprechenden Aktion führen. Das Ziel ist, schon an der Behebung des Problems zu arbeiten bzw. es im Idealfall schon behoben zu haben, bevor der erste Anruf von Endanwendern an der Hotline eingeht.

Eine spezielle Notwendigkeit stellen automatische Benachrichtigungen dar, wenn aus Kostengründen kein 24-Stunden-Betrieb des IT-Teams möglich ist und stattdessen für Nacht- und Wochenendbetrieb ein Rufbereitschaftsdienst eingerichtet wird. In diesem Fall ist eine automatische Benachrichtigung sogar unerlässlich, da es sonst zu erheblichen Stillstandszeiten kommen kann.

Folgende aktive Benachrichtigungsmethoden wurden in unserem Fall verwendet:

2.4.1 Alarm

Auch wenn sich ein zuständiger Systemadministrator an seinem PC-Arbeitsplatz befindet, muss eine aktive Benachrichtigung erfolgen. Die Monitoring-Applikation ist unter Umständen von anderen Fenstern verdeckt und wird auch nicht laufend beobachtet.

Als praktische Lösung für diesen Einsatzzweck hat sich ein kleines Programm mit dem Namen BBWatch bewährt, das laufend den Status der Big Brother-Webseite abfragt und über ein Symbol in der Windows-Taskleiste darstellt. Beim Eintreten einer Fehlerbedingung wird außerdem ein wählbares akustisches Signal ausgegeben. Da dieses Programm auf den Arbeitsplätzen mehrerer Mitglieder des KIS-Teams ständig läuft, werden Fehler schnell bemerkt.

2.4.2 e-Mail

Je nach Fehlerstatus wird an bestimmte Empfänger in einstellbaren Intervallen eine e-Mail mit einer Beschreibung des Fehlers versendet. Diese Art der Benachrichtigung setzt eine funktionierende Netzwerkverbindung und einen funktionierenden Mailserver voraus.

Die Benachrichtigung per e-Mail ist wesentlich weniger unmittelbar als die Verwendung eines Alarms. Gerade bei Mitarbeitern in diesem Umfeld, die täglich unzählige e-Mails erhalten, ist die Aufmerksamkeit für eintreffende Warnungen gering.

Trotzdem wurde diese Art von Benachrichtigung implementiert, um auch für jene Benutzer eine Benachrichtigung zu ermöglichen, die die Alarm-Applikation nicht installiert haben oder unterwegs sind und nur auf ihre e-Mails Zugriff haben.

2.4.3 SMS

Das Versenden von SMS ist das Mittel der Wahl für die Alarmierung, wenn sich der Systemadministrator nicht an seinem Arbeitsplatz befindet, sei es bei einer Besprechung im Nebenraum oder unterwegs. SMS ist eine bewährte, kostengünstige Technologie mit hohem Verbreitungsgrad und guter Netzabdeckung.

Zum Versenden von SMS gibt es prinzipiell zwei Möglichkeiten:

- Senden einer e-Mail an ein e-Mail-SMS-Gateway. Diese Methode ist je nach GSM-Provider unterschiedlich bzw. gar nicht verfügbar, aber einfach und kostengünstig zu implementieren. Die Funktion dieser Methode ist von der Verfügbarkeit von Netzwerk und Mailserver abhängig.
- Direktanschluss eines Mobiltelefons an den Überwachungsrechner. Der Anschluss erfolgt über die serielle Schnittstelle und ist mit nahezu allen Typen von Mobiltelefonen möglich. Für einen kontinuierlichen Betrieb muss das Datenkabel gemeinsam mit dem Ladekabel an das Mobiltelefon ansteckbar sein. Die Funktion ist nicht von externen Komponenten abhängig. Allerdings ist jede einzelne Benachrichtigung kostenpflichtig.

In unserem Fall wurde aus Gründen der Einfachheit vorläufig die Implementierung über ein e-Mail-SMS-Gateway gewählt.

Ein prinzipieller Nachteil von SMS-Nachrichten ist, dass sie derzeit bei keinem österreichischen Betreiber auf ein anderes Mobiltelefon umgeleitet werden können.

2.5 Automatisierte Aktionen

Für einzelne Fehlerbedingungen sind automatisierte Aktionen zur Behebung denkbar. Dazu müssen allerdings einige Voraussetzungen gegeben sein:

- Der Fehlerzustand tritt öfter und immer in der exakt gleichen Art auf.
- Das Entstehen des Fehlers lässt sich nicht vermeiden (z. B. durch regelmäßig laufende Skripts).
- Die Behebung des Fehlers ist ein einfacher, exakt beschreibbarer Vorgang, der automatisiert werden kann.

Ein unüberlegter Einsatz von automatisierten Fehlerbehebungen kann leicht in ein Desaster führen. Als Beispiel soll eine Regel dienen, die bei Überlauf eines Filesystems dieses automatisch vergrößert. Wenn ein fehlerhaftes Programm immer mehr Plattenplatz belegt, würde dieser Automatismus dazu führen, dass das entsprechende Filesystem solange vergrößert wird, bis sämtlicher zur Verfügung stehende Plattenplatz verbraucht ist. Damit würden die effektiven Auswirkungen also nur verschlimmert.

Da es im vorliegenden Fall keinen wirklich sinnvoll erscheinenden Einsatzzweck für automatisierte Aktionen gab, wurde bewusst darauf verzichtet.

Tätigkeiten, die die Systemkonstanz verbessern, wie das Entfernen veralteter Einträge aus Datenbank und Dateisystem, werden als zeitgesteuerte Jobs regelmäßig ausgeführt, anstatt sie durch das Erreichen gewisser Schwellwerte auszulösen.

2.6 Aufbau des Monitoring-Systems

2.6.1 Monitoring-PC

Als Hardware für die gewählte Monitoring-Lösung konnte aufgrund der geringen Hardwareanforderungen ein ausrangierter PC verwendet werden. Der PC wurde allerdings zur Sicherstellung eines kontinuierlichen Betriebs in einem Serverraum platziert.

Die Software „Big Brother“ existiert in Versionen für *Microsoft Windows*[®] und *Linux*. Aufgrund des größeren Reifegrades von Big Brother unter Linux fiel die Wahl auf diese Plattform. Verwendet wurde Red Hat[®] Linux Release 7.1.

Dieser PC erfüllt alle drei Aufgaben, für die ein PC bei Big Brother verwendet werden kann:

- **Monitor:** Das eigentliche Überwachen von Netzwerk-Adressen und Diensten auf anderen Servern. Auf diese Weise ist nur eine Überprüfung von Diensten möglich, die direkt über das Netzwerk angesprochen werden können.
- **Display:** Die übersichtliche Darstellung des aktuellen Status über ein Webinterface. An diesen Dienst senden sowohl der Monitor-Dienst als auch aktive Big Brother-Clients auf anderen Servern ihre Statusmeldungen
- **Alert:** Dieser Dienst übernimmt den Versand von e-Mails und SMS-Nachrichten.

Zusätzlich läuft auf diesem PC auch der Big Brother-Client, das heißt der PC überwacht sich auch selbst. Das ist bei einem Ausfall des PCs natürlich nicht sinnvoll, kann aber z. B. vor dem Überlauf eines Dateisystems warnen.

2.6.2 AIX-Agent

Der Big Brother-Client existiert auch in vorkompilierten Fassungen für das IBM-Betriebssystem AIX[®], unter dem sowohl die Oracle[®]-Datenbank als auch die Applikationsserver-Prozesse des KIS laufen. Dieses System stellte den weitaus komplexesten Teil des KIS-Gesamtsystems dar und war daher auch bei der Implementierung des Monitorings am zeitaufwändigsten.

Die Installation des Agent war problemlos möglich. Sinnvollerweise wird der Agent über einen Eintrag in der Datei `/etc/inittab` beim Systemstart automatisch geladen. Zusätzlich zum Agent musste auf den Servern noch die Sprachen *Tcl* und *Expect* installiert werden, um die unter 2.3.5.1 beschriebene Überprüfung auf Applikationsebene durchführen zu können. Da die Installation aller drei Softwarepakete keinen tiefen Eingriff in das System darstellt, wurde das Risiko in Kauf genommen.

In Tabelle 3 finden sich die konfigurierten Überprüfungsroutrinen für diesen Agent mit den Auswirkungen auf die Verfügbarkeit des KIS, auf die ein Fehlschlagen des jeweiligen Tests hinweisen kann.

Tabelle 3: Funktionen des AIX-Agents

Überwachungs-Item	Mögliche Fehlerursachen	Mögliche Auswirkungen auf die Verfügbarkeit des KIS
CPU-Belastung	„echte“ Überlastung Fehlerhafte Prozesse	längere Antwortzeiten
Dateisystem-Überlauf	nicht rechtzeitige Anpassung der Dateisystemgröße Fehlerhafte Prozesse	harmlos bis Totalausfall je nach Dateisystem
Fehlen von wichtigen Prozessen	Softwareproblem	bis zu Totalausfall
AIX-Systemlogs	Hardwareproblem gravierendes Softwareproblem	harmlos bis Totalausfall
Oracle-Alert-Log	Oracle-Problem	harmlos bis Totalausfall
Oracle-Tablespaces	versäumte rechtzeitige Erweiterung	bei 100% Füllgrad Totalausfall
Oracle-Locks (s. Anhang C)	Softwarefehler Bedienungsfehler	bis zu Totalstillstand
KIS-Schnittstellen	Problem bei Fremdsystem Problem im Kommunikationsserver Softwareproblem bei den KIS-Schnittstellen	fehlende Kommunikation zu anderen Systemen
Serverprozess-Gesundheit	Softwareproblem	harmlos bis Totalausfall

Wie aus der Tabelle ersichtlich ist, kann aufgrund eines Fehlerstatus bei der Überwachung nicht sofort auf die tatsächliche Schwere der Auswirkungen geschlossen werden. In jedem

Fall soll die Fehlerursache auch in harmlosen Fällen behoben werden, damit schwere Fehler nicht von harmlosen Fehlern „verdeckt“ werden.

2.6.3 Windows-Agent

Der Windows-Agent von Big Brother wurde auf einem Windows-Server installiert, der primär für die Abarbeitung von Druckaufträgen zuständig ist, die von den KIS-Servern initiiert werden. Dabei gibt es einerseits Druckjobs, die über den LPR-Dienst abgewickelt werden, andererseits Jobs, die von einem speziellen auf diesem Server installierten Dienst abgearbeitet werden, der *Microsoft Word* zum Formatieren der Ausdrucke verwendet. Zusätzlich wurde dieser Server auch gleich mit verwendet, um den clientseitigen Test der KIS-Applikation („End-to-End“-Test, s. Abschnitt 2.3.5.2) durchzuführen.

Im Einzelnen wurden auf diesem Server die in Tabelle 4 angeführten Überprüfungsrouitinen konfiguriert:

Tabelle 4: Funktionen des Windows-Agents

Überwachungs-Item	Mögliche Fehlerursachen	Mögliche Auswirkungen auf die Verfügbarkeit des KIS
CPU-Belastung	„echte“ Überlastung fehlerhafte Prozesse	langsamere Abarbeitung von Druckjobs
Dateisystem-Überlauf	nicht rechtzeitige Erweiterung oder Bereinigung des Plattenplatzes fehlerhafte Prozesse	Stillstand der Druckfunktionen
Fehlen von wichtigen Prozessen / Diensten	Softwareproblem	Stillstand der Druckfunktionen
LogCheck	Fehler im KIS-Dienst zur Formatierung von Ausdrucken	Stillstand von bestimmten Druckfunktionen
„End-to-End“-Test	fehlende Erreichbarkeit der KIS-Server gravierendes Serverproblem	Totalstillstand des KIS

3 Diskussion

3.1 Zielerreichung

Folgende Ziele waren für das Monitoring-Projekt gesteckt:

1. Verringerung der Reaktionszeit vom Auftreten eines Problems bis zur Behebung
2. Verkürzung der ungeplanten Ausfallszeiten
3. Vermeidung von Ausfallszeiten durch frühzeitiges Erkennen von potentiellen Problemen
4. kontinuierliche Information über den Systemzustand

3.1.1 Verringerung der Reaktionszeit

Ein Ziel war, die Zeit, die vom Auftreten eines Problems bis zum Beginn der Fehleranalyse und -behebung verstreicht, zu verringern. Dieses Ziel konnte voll und ganz erreicht werden.

Sehr häufig war die Behandlung eines aufgetretenen Problems durch die ausgelöste Alarmierung des KIS-Teams bereits im Gange, bevor der erste Anruf eines Endanwenders die Hotline erreichte. Der Hauptnutzen dieser Alarmierung ist natürlich die effektive Stillstandszeit zu verringern. Ein positiver Nebeneffekt ergibt sich aber auch durch die gewonnene Kompetenz, die der Hotline und dem Team zugeschrieben wird, wenn diese sofort über bestehende Probleme Bescheid weiß und nicht erst durch die Anwender darauf aufmerksam gemacht werden muss. Das wiederum erhöht die Akzeptanz des Systems und des ganzen KIS-Projekts.

3.1.2 Verkürzung von ungeplanten Ausfallszeiten

Dieses Ziel konnte prinzipiell erreicht werden, da durch die Verkürzung der Reaktionszeit auch die Dauer der Ausfallszeit verringert wird. Allerdings muss eingeräumt werden, dass diese Verkürzung prozentuell gesehen eher gering ist, da im Falle eines massiven Problems die Behebung vielfach länger dauert als die Zeitspanne, die durch die automatisierte Benachrichtigung gewonnen wird.

3.1.3 Vermeidung von ungeplanten Ausfallszeiten

Dieses Ziel wurde erreicht. In häufigen Fällen konnte ein potentielles Problem durch die automatisierte Benachrichtigung entdeckt und behoben werden, bevor es sich auf die Anwender auswirkte. Das betrifft vor allem Warnungen wie der drohende Überlauf eines Dateisystems, aber auch Stillstände von Hintergrundaktivitäten wie den Schnittstellen zu anderen Systemen, die sonst unter Umständen längere Zeit unbemerkt geblieben wären, aber

aufgrund der nicht übermittelten Informationen trotzdem problematisch sein können. So kann z. B. die Tatsache, dass neue Laborbefunde im System nicht verfügbar sind, durchaus für die Behandlung eines Patienten relevant sein.

3.1.4 Kontinuierliche Information des Teams

Dieses Ziel bestand darin, dass sich eine Anzahl von Personen jederzeit und einfach einen Überblick über die „Systemgesundheit“ verschaffen können. Auch dieses Ziel wurde erreicht. Durch das verwendete Webinterface kann sich jedes Mitglied des KIS-Teams jederzeit über den Zustand des Systems informieren. Die graphische Darstellung ist dabei so intuitiv, dass sofort erkannt werden kann, in welchem Bereich ein Problem vorliegt. Das erwies sich als großer psychologischer Nutzen des Projekts und verringerte auch die Anzahl von telefonischen Rückfragen.

Allerdings zeigte sich hier auch, dass die einfache Benutzeroberfläche nicht ein tieferes Wissen über die überwachten Objekte ersetzen kann. So ist die Einschätzung, wie kritisch ein angezeigtes Problem wirklich ist, nur über eine nachfolgende genauere Analyse mit den dazu notwendigen Kenntnissen über den Systemaufbau möglich.

3.2 Lehren aus dem Monitoring-Projekt

3.2.1 Stellenwert des Monitoring

Als das Monitoring-Projekt gestartet wurde, hatte es den Status eines kleinen Zusatzes zum KIS, der dem Support-Team eine Arbeitserleichterung bringt.

Inzwischen hat sich die Benutzeranzahl des KIS und auch die Intensität der Nutzung drastisch erhöht, und jeder Systemausfall führt zu sofortigen Reaktionen. In diesem Zuge hat sich das Monitoring zu einem für den Betrieb des KIS unverzichtbaren Systembestandteil entwickelt.

3.2.2 Freeware-Produkte

Am Beginn des Projektes herrschte Skepsis darüber, ob der Einsatz einer frei verfügbaren Software zur Überwachung eines kritischen Systems rund um die Uhr sinnvoll ist. Auch wenn der Einsatz von freier Software, vor allem dank der Verbreitung von Linux, nicht mehr kategorisch abgelehnt wird, bestanden doch gewisse Bedenken bezüglich der Professionalität der jetzigen Lösung.

Es hat sich aber im Projektverlauf gezeigt, dass die gewählte Lösung stabil und zufrieden stellend funktioniert. Es scheint auch, dass das verwendete Grundprodukt gar nicht entscheidend ist, solange ein gewisser Funktionsumfang vorhanden ist. Das Wichtigste ist vielmehr der Aufwand, der in die Konfiguration und die kontinuierliche Pflege investiert wird.

Ein Nachteil des gewählten Produkts ist, dass die Konfiguration über weite Strecken durch das Bearbeiten von Textdateien erfolgen muss. Hier wäre auch für einen erfahrenen Administrator ein grafisches Tool nützlich.

Auch die Synchronisation der Konfiguration über alle beteiligten Knoten muss manuell erfolgen, hier wäre eine Automatisierung wünschenswert.

3.2.3 Kontinuierliche Wartung

Ein Monitoring-System macht nur dann Sinn, wenn es regelmäßig gewartet wird. Überprüfung und eventuelle Änderungen am Monitoring-System müssen zumindest in folgenden Fällen erfolgen:

- eine Hardwarekomponente (Server o. ä.) wird hinzugefügt oder entfernt
- größere Änderungen an Betriebssystem, Applikationssoftware oder der darunter liegenden Datenbank
- neue Funktionen oder Systemkomponenten werden in Betrieb genommen

- es kommt wiederholt zu Fehlalarmen wegen Fehlern in Überprüfungs-Scripts oder zu einer falschen Bewertung von Zuständen

Die Bedeutung dieser kontinuierlichen Wartung ist vielleicht der wichtigste Punkt für den Erfolg eines solchen Projekts. Wichtig ist auch, dass die Wartung unbürokratisch funktioniert. Wenn für eine Änderung mehrere Personen, womöglich aus verschiedenen Abteilungen, notwendig sind, kann eine Änderung lange dauern oder womöglich gar nicht passieren. Das ist vor allem im Fall von Fehlalarmen problematisch. Sobald sich aber niemand mehr aktiv um die Vermeidung von Fehlalarmen kümmert, wird die Aussagekraft der Benachrichtigungen bezweifelt und diese über kurz oder lang ignoriert.

3.2.4 Aktive Benachrichtigung

Nur durch aktive Benachrichtigung kann einer der großen Vorteile des Monitorings erreicht werden, nämlich verkürzte Reaktionszeiten. Auch bei gewissenhafter Durchführung ist das manuelle Überprüfen des Systemzustands in keinem Fall ausreichend.

Als Vorteil hat sich erwiesen, wenn die aktive Benachrichtigung mehrere Personen erreicht. So kann auch im Falle von kurzen oder längeren Abwesenheiten einer Person zeitnah auf einen Fehler reagiert werden.

3.3 Ausblick / Verbesserungsmöglichkeiten

Es hat sich gezeigt, dass die Entscheidung für die bestehende Lösung richtig war. Trotz minimaler Hardware- und überschaubarer Personalkosten wurde bereits mehrmals eine Fehlerbedingung frühzeitig bemerkt und behoben, bevor ein Anruf die Hotline erreichte.

Trotzdem gibt es noch genügend Raum für Verbesserungen.

3.3.1 Bewertung von Auswirkungen

Die einzelnen verwendeten Überprüfungen, die zu Statusanzeigen im Monitoring führen, setzen auf verschiedenen logischen Ebenen von Hardware bis Applikationsverfügbarkeit auf. Es ist für einen neuen Benutzer nicht ersichtlich, welche Auswirkungen auf den Betrieb und auf die Endanwender das Vorhandensein eines Fehlerzustandes hat. Hier wäre es wünschenswert, wenn sich über eine Koppelung an ein 3LGM-ähnliches Modell die exakten Effekte eines Problems ermittelt ließen. So ist eine Meldung wie „Patientenaufnahme nicht möglich“ anstatt „Fehler xy bei Server yz“ denkbar.

Der Aufwand für Modellierung, Koppelung und Wartung eines solchen Modells wäre allerdings hoch, was die Sinnhaftigkeit relativiert.

3.3.2 Redundanz des Monitoring-Systems

Der Monitoring-PC selbst ist derzeit nur einfach vorhanden. Bei Ausfall dieses PCs bzw. seiner Netzwerkanbindung ist das Monitoring unterbrochen. Da die verwendete Monitoring-Software Big Brother alle Voraussetzungen für eine redundante Auslegung dieses PCs mitbringt, erscheint dies als sinnvolle Erweiterungsmöglichkeit, die mit relativ wenig Aufwand machbar wäre.

Eine Verbesserung der Verfügbarkeit des Benachrichtigungs-Systems könnte durch die Verwendung eines Mobiltelefons für den Versand von SMS-Nachrichten an Stelle des zur Zeit benutzten e-Mail-Gateways erreicht werden.

3.3.3 Weitere Verfeinerung und Komplettierung

Obwohl derzeit bereits über 70 einzelne Tests regelmäßig durchgeführt werden, sind weitere Überprüfungen sinnvoll. Obwohl die meisten gravierenden Probleme von der Monitoring-Software erfasst und gemeldet wurden, gab es eine kleine Anzahl von Ausfällen, die nicht bemerkt wurden. Auf der anderen Seite gibt es noch Verbesserungsmöglichkeiten bei der Einschätzung der Kritizität gewisser Fehlerzustände. Hier können noch die Erfahrungen im

Betrieb und eine eventuell erweiterte Programmlogik eingesetzt werden, um Fehlalarme zu verringern.

3.3.4 Ausweitung auf weitere Systeme

Das Monitoring umfasst derzeit die Systeme, die direkt oder mittelbar mit der Verfügbarkeit des KIS zu tun haben. Zusätzlich wird noch ein Subsystem und die Infrastruktur des KIS-Teams überwacht.

Aufgrund der guten Erfahrungen und des im Projektverlauf gewonnenen Know-hows würde es durchaus Sinn machen, das Monitoring auf andere Systeme auszuweiten.

Anhang

Anhang A: 3GLM²-Bilder

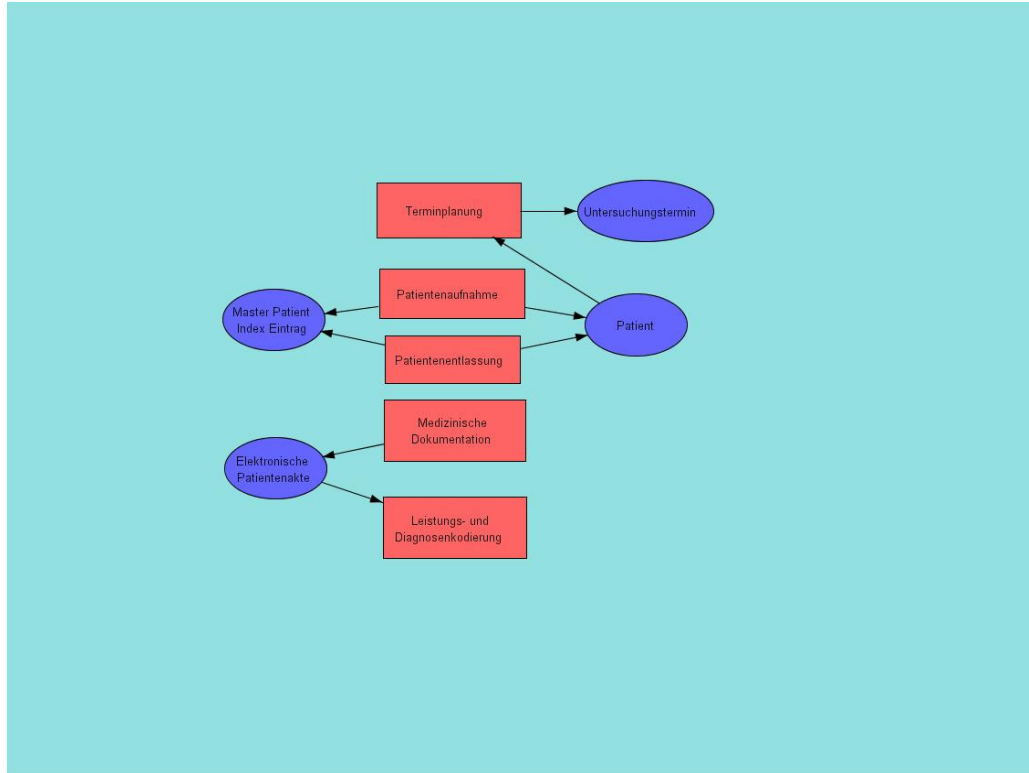


Abbildung 1: 3LGM - Fachliche Ebene (Ausschnitt)

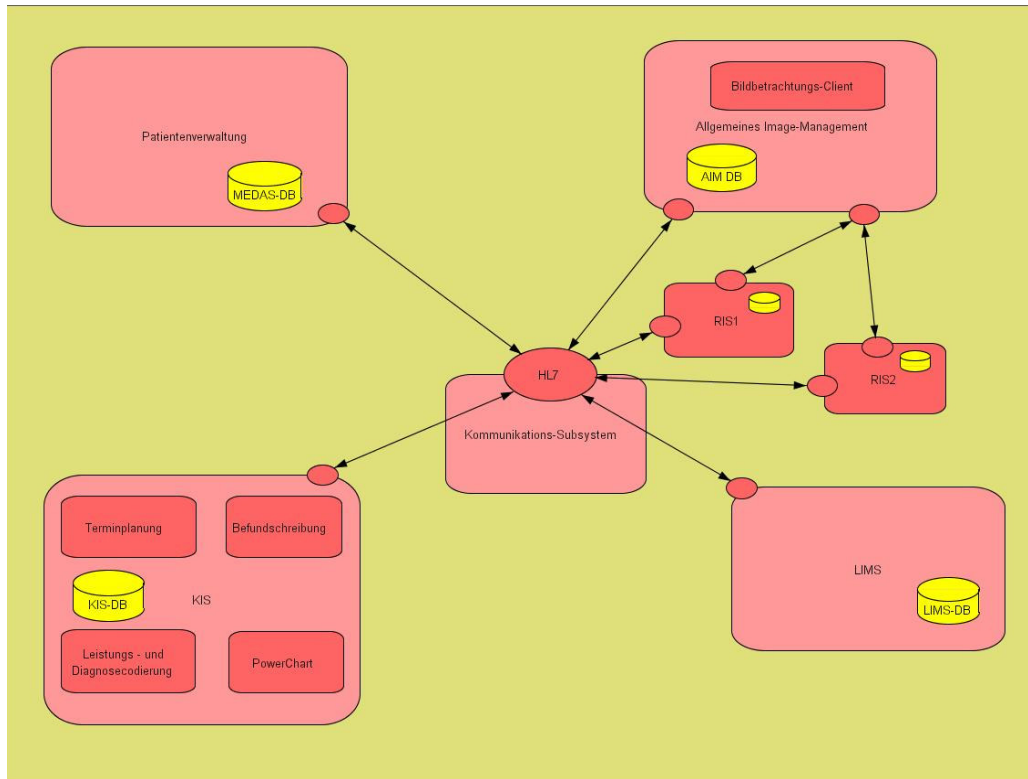


Abbildung 2: 3LGM - Logische Werkzeugebene

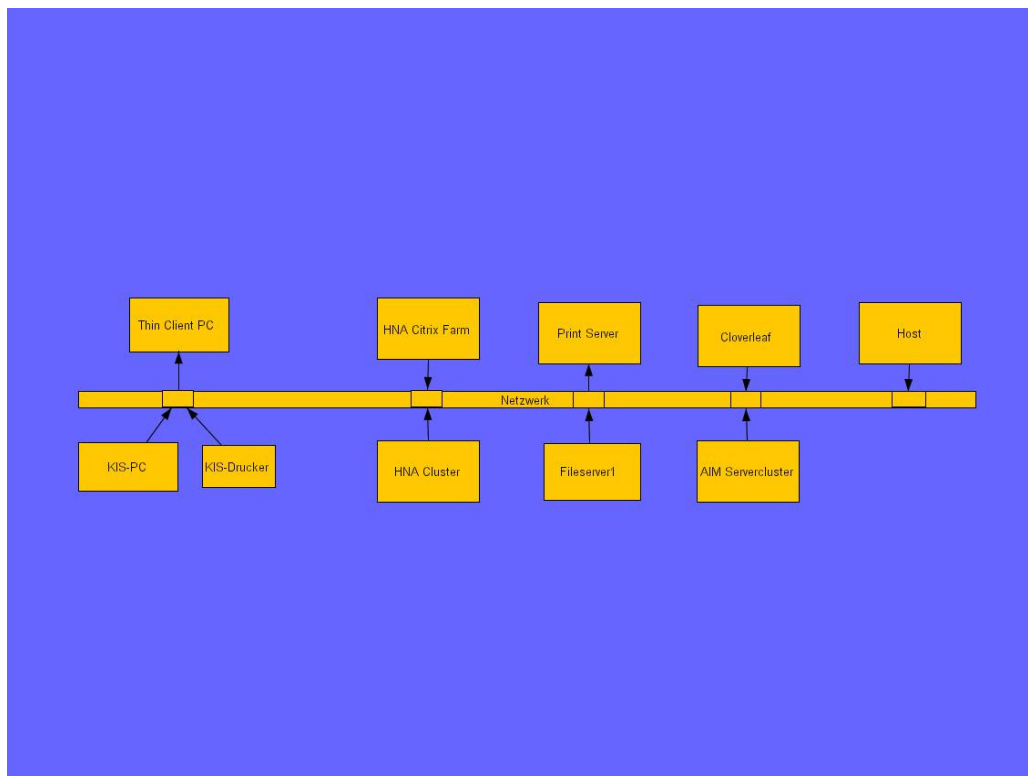


Abbildung 3: 3LGM - Physische Werkzeugebene



Abbildung 5: Big Brother - Kurzansicht



Abbildung 6: Big Brother - Verfügbarkeitsstatistik

Anhang C: Script-Beispiel: OraLock

Im Folgenden findet sich ein Beispiel für einen benutzerdefinierten Test, der in Big Brother eingebunden wurde. Er überprüft, ob sich auf der Oracle-Datenbank Satzsperrungen befinden, die den Betrieb beeinträchtigen können.

Der Test besteht aus zwei Komponenten, nämlich einem AIX-Shellscript, das von Big Brother aufgerufen wird (Listing 1), und einem Programm in der Sprache CCL, das vom ersten Script aufgerufen wird und die eigentliche Überprüfung durchführt (Listing 2).

Listing 1: oralock - Unix Shell Script

```
#!/bin/sh
#
# Script: oralock
# Autor: Stefan Daxenbichler
# Zweck: Überprüft, ob auf der Oracle-Instanz locks existieren
#       Bei längeren Locks oder bei Locks, die einen Prozess
#       zum Warten zwingen, wird der Status "red" zurückgegeben
# Ver   Datum       Autor   Kommentar
# 1.0   2003-10-02   SDA     Erstversion

#-----
# BigBrother-Definitionen

BBPROG=oralock; export BBPROG
TEST="oralock"

if test "$BBHOME" = ""
then
    echo "BBHOME is not set... exiting"
    exit 1
fi

if test ! "$BBTMP"                # GET DEFINITIONS IF NEEDED
then
    . $BBHOME/etc/bbdef.sh        # INCLUDE STANDARD DEFINITIONS
fi

#-----
# Status über CCL-Script ermitteln und in temp. File schreiben

echo "sd_lockcheck \"/tmp/$$tmp\" go" | $cer_exe/cclora

LINE="status $MACHINE.$TEST `cat /tmp/$$tmp`"

#-----
# Ergebnis an BB-Server schicken
$BB $BBDISP "$LINE"              # SEND IT TO BBDISPLAY

# temp. File löschen
$RM -f /tmp/$$tmp

#-----EOF
```

Listing 2: oralock – CCL-Programm sd_lockcheck.prg

```

; Program: sd_lockcheck
; Autor:   Stefan Daxenbichler (ITH)
; Zweck:  testet auf Oracle-Locks für BigBrother
;
; Ver  Datum      Autor  Beschreibung
; 1.0  2003-12-10  SDa   Erstversion
; 1.1  2003-12-11  SDa   Timeouts erhöht
; 1.2  2003-12-15  SDa   PLAN_TABLE ausgenommen

drop program sd_lockcheck go
create program sd_lockcheck

PROMPT "Output to: " = "MINE"

; Variablendeklaration
DECLARE sStatusColor=vc
DECLARE sHeader=vc
DECLARE sLine=vc
DECLARE sOutput=vc
DECLARE Status=i4
DECLARE SumStatus=i4

; Oracle-Select für bestehende LOCKs
SELECT into "nl:" *
FROM V$LOCK L,
     ALL_OBJECTS O,
     V$SESSION S
WHERE (L.ID1 = O.OBJECT_ID)
      AND (O.OWNER!= "SYS" )
      AND (O.OBJECT_NAME!= "V$*"  AND O.OBJECT_NAME != "PLAN_TABLE")
      AND (L.SID=S.SID)

DETAIL
; wenn lock > 5 min oder blockierend --> Status "red"
if (L.CTIME>300) OR (L.CTIME>5 AND S.LOCKWAIT != NULL)
  Status=2
elseif (L.CTIME > 30)
  Status=1
else
  Status=0
endif

; Output-Zeile formatieren
if (Status>=1)
  if (Status=1)
    sLine="&yellow "
  else
    sLine="&red "
  endif
  sLine=CONCAT(sLine, " ", format(S.OSUSER, "##### |"))
  sLine=CONCAT(sLine, " ", format(S.MACHINE, " ##### |"))
  sLine=CONCAT(sLine, " ", format(S.PROCESS, " ##### |"))
  sLine=CONCAT(sLine, " ", format(O.OBJECT_NAME, " ##### |"))
  sLine=CONCAT(sLine, " ", format(S.PROGRAM,
    "##### |"))
  sLine=CONCAT(sLine, " ", format(BUILD(S.LOCKWAIT," "),
    " ##### |;P-"))
  sLine=CONCAT(sLine, " ", format(L.CTIME, " #####"))
  sOutput=CONCAT(sOutPut,sLine, CHAR(10))
endif

if (Status > SumStatus) ; kumulativen Status
  SumStatus=Status
endif

WITH NOCOUNTER, NOFORMAT

```

```
; -----  
, Output für Big Brother erzeugen  
  
if (SumStatus=0)  
    SET sStatusColor="green"  
    SET sOutput="no bad locks"  
elseif (SumStatus=1)  
    SET sStatusColor="yellow"  
else  
    SET sStatusColor="red"  
endif  
  
SET sHeader=CONCAT(sStatusColor, " ", format(cnvtdatetime(curdate, curtime3),  
    "dd-mmm-yyyy hh:mm:ss;;D"), CHAR(10))  
  
SELECT into VALUE($1)  
DETAIL  
    CALL PRINT(sHeader)  
    CALL PRINT(sOutput)  
WITH maxcol=32000, FORMAT=VARIABLE  
  
end go
```

Literatur

- [1] Big Brother Website <http://www.bb4.com/>
- [2] Website des HL7-Boards: <http://www.hl7.org>
- [3] http://www.kodak.com/global/plugins/acrobat/en/health/pdf/prod/svcs/eamer/khi136_1100.pdf
- [4] Buchauer A, Ammenwerth E, Winter A, Haux R (1997): 3LGM: Method and Tool to support the Management of Heterogeneous Hospital Information systems. In: Kacki (edt): Computers in Medicine 4, 1997. Volume 1. Lodz, 1997
- [5] Buchauer A (1995): Methoden und Werkzeuge zur Anwendung des graphenbasierten Drei-Ebenen-Modells für die Beschreibung, Bewertung und Planung von Krankenhausinformationssystemen. Bericht 2/95 der Abteilung Medizinische Informatik, Universität Heidelberg.
- [6] Dujat C (1996): Zur digital-optischen Archivierung von medizinischen Dokumenten im Krankenhaus. Abteilung Medizinische Informatik, Universität Heidelberg, Bericht 2/1996
- [7] Lagemann A (1996): Integration des Verfahrens Pflegedokumentation in ein Klinisches Arbeitsplatzsystem. Abteilung Medizinische Informatik, Universität Heidelberg, Bericht 3/1996.
- [8] Winter A (1994): Beschreibung, Bewertung und Planung heterogener Krankenhausinformationssysteme. Abteilung Medizinische Informatik, Universität Heidelberg, Bericht 7/1994
- [9] Winter A, Haux R (1995): A Three Level graph-based Model for the Management of Computer-Supported Hospital Information Systems. Methods of Information in Medicine, 34(4), 378-396
- [10] The Expect Home Page: <http://expect.nist.gov/>
- [11] Tcl Developer Xchange: <http://www.tcl.tk/>
- [12] Dejraj V (2000): Oracle 24x7 Tips & Techniques. Berkeley: Osborne/McGraw Hill, 2000
- [13] Dreyer C (2002): Citrix MetaFrame und Windows Terminal Services. Bonn: mitp-Verlag, 2002
- [14] Washburn K (1997): TCP/IP Aufbau und Betrieb eines TCP/IP-Netzes. Bonn: Addison-Wesley-Longman, 1997