Endbericht der Projektgruppe IPSI

Version 1.3 vom 12.02.2001

PG IPSI

Universität Dortmund

WS1999/2000, SS 2000

0 Vorwort

0.1 Internet, WWW und Electronic-Commerce

Das Internet ist ein weltweites, öffentlich zugängliches Rechnernetz, das aus einer Vielzahl verteilter, miteinander gekoppelter Einzelnetze besteht. Sowohl der Umfang der Nutzung der Dienste des Internet, als auch die Teilnehmerzahlen dieses Netzwerks wachsen bis heute stetig [EcOr98]. Viele Internet-Teilnehmer sehen das Internet als Informationsquelle und nutzen es zur Produktberatung, wodurch Kaufentscheide gefördert werden. Durch die Verwendung dieser Möglichkeiten durch Unternehmen, sich selbst und die Produkte und Dienstleistungen darzustellen, begann ca. 1993 die kommerzielle Nutzung des Internets als WorldWideWeb (WWW). Die Dienste des Internet (HTTP, FTP, SMTP, etc.) bieten einen Distributions- und Kommunikationskanal, der dazu benutzt werden kann, neue Kunden zu erreichen und alte Kunden zu binden [La96]. Antworten auf Kundenanfragen können über diesen Kanal schnell und ohne Berücksichtigung von Öffnungszeiten gegeben werden. Das Serviceangebot eines Unternehmens kann erweitert werden, indem beispielsweise mailgestützte Kundenberatung angeboten wird [Al96]. Zusätzlich verliert die tatsächliche geographische Lage eines Unternehmens durch Electronic-Commerce ihre Wichtigkeit und wird für den Kunden völlig transparent.

Seit ca. zwei Jahren gibt es Bestrebungen in Industrie und Forschung, nicht nur die Verkaufsförderung durch das WorldWideWeb zu unterstützen, sondern den gesamten Verkaufsprozess. Unter Electronic-Commerce wird mehr verstanden als ein elektronischer Kaufladen: der Verkaufsprozess von Werbung über Geschäftsanbahnung, Vertragsabschluss, Kaufbestätigung, Bezahlung, Lieferung, Reklamation bis hin zu After-Sales-Services sollte nach Möglichkeit elektronisch abgewickelt werden. Darüber hinaus wird als Electronic-Commerce jede Art von geschäftlichen Transaktionen (Marktvorgängen) bezeichnet, bei denen Geschäftspartner (Anbieter und Kunden) auf elektronischem Weg über ein Datenkommunikationsnetz miteinander verkehren. Sie umfassen die Verkaufsförderung, Verkaufsdurchführung, Distribution und Verkaufsnachbereitung für Produkte und Dienstleistungen. Produkte und Dienstleistungen im Sinne des Electronic-Commerce sind Finanzdienstleistungen (Versicherungen, Aktien, Fonds, etc.), Computerprogramme (Software, Shareware), Kulturgüter (Eintrittskarten, Zeitschriften, Magazine, Bücher, Videofilme, Musik-CDs) und Reisen (Fahr-, Flug- und Fährkarten, Hotels, Ferienhäuser). Electronic-Commerce wird mit dem Ziel des Absatzes von Produkten und Dienstleistungen betrieben, weshalb nachfolgend nicht mehr zwischen diesen beiden Begriffen unterschieden wird. Wenn nicht digitalisierbare Produkte und Dienstleistungen vertrieben werden sollen, muss bei der Lieferung auf den konventionellen Versand zurückgegriffen werden. Digitalisierbare Produkte, wie elektronische Zeitschriften oder Software, lassen sich mit Hilfe der Transportdienste des Internets ausliefern. Das gleiche gilt für After-Sales-Services, wie etwa Wartung oder Installation der Produkte: wenn diese Leistungen digitalisierbar sind, können sie preiswert mit Hilfe des Internets realisiert werden. So können beispielsweise Softwaresysteme (oder Systemkomponenten) von entfernt sitzenden Experten, die über das Internet auf die Zielrechner zugreifen können, installiert und gewartet werden. Die elektronischen Geschäftsbeziehungen zwischen Kunden, Unternehmen und öffentlichen Einrichtungen sind in der Abbildung 1 dargestellt:

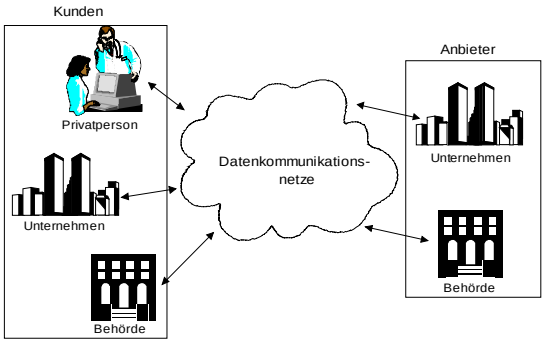


Abbildung 1 - Geschäftsbeziehungen bei Electronic-Commerce

Die elektronischen Geschäftsbeziehungen lassen viele Kombinationsmöglichkeiten zwischen Privatpersonen, Unternehmen und öffentlichen Einrichtungen zu. Jede dieser drei Gruppen kann in einer Geschäftsbeziehung entweder die Rolle des EC-Anbieters oder des Kunden einnehmen. Entsprechend dieser Geschäftsbeziehungen wird zwischen Business-to-Business (B2B), Business-to-Consumer (B2C), Business-to-Administraion (B2A), etc. unterschieden. Der Unterschied liegt darin, dass bei B2C der Kunde einer geschäftlichen Transaktion ein

Endverbraucher ist, während bei B2B der Kunde ein Unternehmen oder eine Behörde sein kann. In beiden Fällen wird davon ausgegangen, dass der Anbieter ein Unternehmen ist. Ein EC-Anbieter bietet Waren oder Dienstleistungen an, die der Kunde kauft und bezahlt.

Die Geschäftsbeziehungen zwischen zwei Unternehmen wurden teilweise auch schon vor Etablierung des Internets elektronisch abgewickelt [EcOr98]. Als Beispiel für eine solche Beziehung werden oft Automobilhersteller und deren Zulieferer genannt, die ihre Geschäfte elektronisch abwickeln. Um dies leichter zu bewerkstelligen, wurde der EDI-Standard eingeführt, der Datenstrukturen zum elektronischen Austausch betrieblicher Daten über Rechnernetze definiert [De95]. Die Geschäftsbeziehungen zwischen Unternehmen und öffentlichen Einrichtungen wurden ebenfalls außerhalb des Internets gepflegt [EcOr98].

Auch Privatkunden können mit Hilfe der über das Internet angebotenen Kommunikationsdienste elektronisch mit Unternehmen und öffentlichen Einrichtungen in Kontakt treten, wodurch auch sie elektronische Geschäftsbeziehungen eingehen können. Durch Electronic-Commerce können andere Wertschöpfer wie etwa Zwischenhändler (Intermediäre) umgangen werden, da die Kommunikation zum Verkauf von Produkten und Dienstleistungen direkt zwischen Kunde und Anbieter stattfindet. Provisionen und Vermittlungsentgelte können dadurch eingespart werden. Die Rationalisierungs- und Erfolgspotentiale für Electronic-Commerce sind in der Abbildung 2 dargestellt.

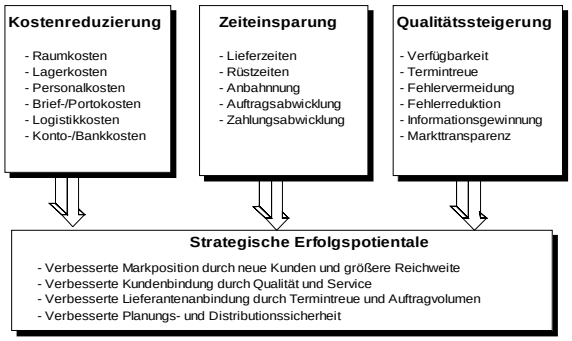


Abbildung 2 - Rationalisierungs- und Erfolgspotentiale von Electronic-Commerce

Electronic-Commerce-Systeme unterstützen die jeweiligen Geschäftspartner bei der elektronischen Abwicklung ihrer geschäftlichen Transaktionen. Sie unterscheiden sich technisch hinsichtlich

- ihrer Client/Serverarchitektur (n-Tier),
- ihres Kommunikationsnetzes (Telefon, GSM, LAN, WAN, VPN),
- ihrer Netztopologie (Stern, Netz, Bus),
- ihrer Netzdistribution (zentral, dezentral),
- ihres Kommunikationsprotokolls (EDI, HTTP, SMTP),
- ihres Kommunikationsdienstes (WWW, FTP, MAIL),
- ihres Sicherheitskonzepts (einfache oder hybride Kryptographie).

Bei Electronic-Commerce-Systemen, die B2C unterstützen, werden Katalogsysteme, Shopsysteme, Auktionssysteme, Ausschreibungssysteme, etc. unterschieden. Einige der genannten Systeme können weiterhin in aktive und passive Systeme klassifiziert werden. In aktiven „Online-Systemen“ wird durch sog. „Agenten“ versucht, eine Aufgabe zu erfüllen, z.B. das preisgünstigste Produkt, welches in verschiedenen Shopsystemen angeboten wird, zu finden. Bei den passiven „Online-Systemen“ übernimmt der Kunde diese Aufgabe. Zu den passiven „Online-Systemen“ zählen die Electronic-Commerce-Portale, durch die eine Menge von einzelnen Electronic-Commerce-Anwendungen und Internet-Anwendungen kundenspezifisch zusammengestellt werden. Wenn als Kunde ein Unternehmen (B2B) agiert, dann werden EDI-Systeme, Warenwirtschaftssysteme (WWS) etc. unterschieden. Electronic-Commerce-Systeme, die bzgl. der Art eines Kunden nicht unterscheiden, sind z.B. Home-Banking-Systeme oder Tracking-Tracing-Systeme.

Nicht alle der genannten Electronic-Commerce-Systeme unterstützen alle Marktvorgänge elektronisch, so wird z.B. vorrangig die Verkaufsförderung und die Verkaufsdurchführung durch die verschiedenen Arten von Katalogsystemen und Shopsystemen unterstützt. Bei EDI-Systemen oder gekoppelten WWS steht nicht die Verkaufsförderung oder die Verkaufsnachbereitung im Vordergrund, sondern im engeren Sinne die Verkaufsdurchführung. Electronic-Commerce-Systeme, die sich ausschließlich mit der Verkaufsförderung und der Verkaufsnachbereitung befassen, ohne selbst Produkte oder Dienstleistungen anbieten, sind Systeme, die ausschließlich In-

formationen, z.B. Nachrichten, Börseninformationen, Wetterdaten, etc., anbieten. Je nachdem wie ein Kunde mit diesen Systemen interagiert, unterscheidet man Push- oder Pull-Systeme. Bei diesen Systemen erfolgt die Verkaufsförderung und Verkaufsnachbereitung durch Werbung mittels Emails, Diskussionsforen, Newsletter oder Banner. Home-Banking-Systeme unterstützen ausschließlich den Aspekt der Zahlungsdurchführung während der Verkaufsdurchführung.

1 Motivation

1.1 Einleitung

Die Projektgruppe IPSI (Internet Portal System for Insurances) wird am Lehrstuhl Softwaretechnologie des Fachbereichs Informatik der Universität Dortmund durchgeführt. Durch das zu konzipierende und zu entwickelnde Internet-Portal sollen eine Vielzahl von Anwendungen (Webanwendungen, EC-Anwendungen, Office-Anwendungen) individuell ausgerichtet für eine spezielle Benutzergruppe sowie aufgabenbezogen integriert werden. Eine Gruppe von Personen, die das Internet in ihren täglichen Arbeitsablauf zur Zeit nur rudimentär integriert hat, sind Versicherungsmakler. Nach Aussage der GdV (Gesamtverband der deutschen Versicherungswirtschaft e.V.) arbeiten Versicherungsmakler in der Regel noch mit konventionellen Arbeitsmitteln (Papier, Telefon, Fax), um Informationen von Versicherungsunternehmen zu sammeln, auszuwerten und dann an Interessenten und Kunden weiterzugeben. Eine weitere Gruppe von Personen, die ähnliche Anforderungen haben, sind die Außendienstmitarbeiter von Versicherungsunternehmen (VAD).

Ein Internet-Portal bietet Informationen oder den Zugang zu Informationen, die ein Versicherungsmakler und ein Versicherungsaußendienstmitarbeiter braucht. Ein Internet-Portal für diese Benutzergruppen umfasst die folgenden Kategorien:

- Web-Links,
- Web-Publishing,
- Web-Procurement (Produkte und Dienstleistungen),
- Web-Communication,
- Web-Applications,
- Versicherungsanwendungen (Legacy-Systeme).

Ein Internet-Portal mit diesen Merkmalen wird in die Klasse der Electronic-Commerce / Electronic-Business-Anwendungen eingeordnet. Durch EC/EB-Anwendungen werden die Techniken des Internets (WorldWideWeb, Electronic-Mail, Electronic-File-Transfer) und der Telekommunikation (SMS, WAP, UTMS) genutzt, um die

- Verkaufsförderung,
- Verkaufsdurchführung,
- Distribution,
- Verkaufsnachbereitung

von Produkten bzw. Dienstleistungen elektronisch zu realisieren. Die Versicherungsaußendienstmitarbeiter und die Versicherungsmakler bilden die Schnittstelle zwischen den Versicherungsunternehmen und ihren Kunden und Interessenten. Durch ein Internet-Portal kann auf Informationen schnell und zielgerichtet am Arbeitsplatz oder vor Ort beim Kunden zugegriffen und damit eine Kompetenzsteigerung erreicht werden.

1.2 Ziele der Projektgruppe IPSI

In vielen Unternehmen finden sich Anwendungen aus den Bereichen E-Business / E-Commerce. Lösungen aus diesem Bereich können unterschiedliche Ausbaustufen (Präsentation, Interaktion, Transaktion, One-To-One Marketing) durchlaufen, die oft nur schwer ineinander überführt werden können, weil die verwendeten Technologien und Lösungsbausteine nicht kompatibel sind.

Dieses Kompatibilitätsproblem wird dadurch verschärft, dass der Markt eine Vielzahl von Technologien und Teillösungen bereitstellt, für die nur mit erheblichem Aufwand ermittelt werden kann, ob sie zueinander passen und welche Rolle sie in einer integrierten E-Business-Lösung eines Unternehmens spielen können.

Einige der Technologien nehmen für sich in Anspruch, die kompletten E-Business-Anforderungen abzudecken. An dieser Stelle sind beispielsweise Contentmanagementsysteme zu nennen, die einen Teil der E-Business-Anforderungen erfüllen können, die jedoch auch mit anderen Lösungen (Workflow, Shop, Zahlungssysteme,

Personalisierung) integriert werden müssen. Tatsächlich ist es so, dass es eine Reihe „überlappender“ Technologien und Bausteine gibt, die integriert werden müssen, um eine Plattform zu bilden, auf der E-Business-Anwendungen effizient entwickelt und betrieben werden können.

Darüber hinaus ist es häufig so, dass E-Business-Anwendungen nicht wie andere Softwareanwendungen behandelt werden. Standards und definierte Verfahren für Entwicklung und Betrieb von Anwendungen werden häufig nicht eingesetzt.

Als Ergebnis entstehen E-Business-Anwendungen,

- die nur unzulänglich miteinander integriert sind,
- die kaum mit den übrigen Anwendungen eines Unternehmens integriert sind,
- deren Architektur nicht als Vorlage für weitere E-Business-Anwendungen dienen kann und
- die hohe Wartungs- und Betriebskosten verursachen.

Auf Grund dieser Situation entsteht zunehmend die Notwendigkeit, E-Business / E-Commerce-Anwendungen zu bündeln und unter einer einheitlichen Oberfläche zusammenzufassen. Solche Lösungen werden als Portallösungen bezeichnet.

Ziel der Projektgruppe IPSI ist es, unter Berücksichtigung der verfügbaren Software und der fixierten Rahmenbedingungen, die Basisdienste und die einzusetzenden Werkzeuge festzulegen, deren Beziehungen und Abhängigkeiten zu definieren und damit eine Portal-Referenzarchitektur festzulegen. Zu dieser Architektur sollen passende Entwicklungs- und Betriebsverfahren entwickelt werden.

Dieses allgemeine Ziel wurde konkretisiert, indem es in einen Anwendungskontext gestellt wurde. Die Projektgruppe IPSI hat sich zum Ziel gesetzt, eine Portallösung für den angestellten Versicherungsaußendienst bereitzustellen. Hierbei kommt es nicht darauf an, eine Lösung bereitzustellen, die zu 100% die Anforderungen eines einzelnen Versicherungsunternehmens erfüllt. Statt dessen wird die Architektur einer solchen Portallösung entwickelt, die einsetzbaren Fremdsysteme werden evaluiert, und es wird ein Prototyp gebaut, der die Funktionalitäten einer Portallösung für den Versicherungsaußendienst demonstriert.

Konkret leistet die Projektgruppe IPSI das folgende:

- Entwurf einer Portallösung für den Versicherungsaußendienst
- Ermittlung eines zu integrierenden Shopsystems zur Unterstützung der Beschaffung zwischen Außendienst und Zentrale
- Auswahl eines Office-Systems zur Unterstützung der Terminverwaltung
- Einsatz eines Contentmanagementsystems als Grundlage der Portallösung
- Integration mindestens eines Legacy-Systems (z.B. zur Partnerverwaltung)
- Entwurf und Realisierung eines Konzeptes zur konsistenten Speicherung harter (beim VU) und weicher Daten (beim Außendienst)
- Dokumentation des Softwareprozesses zur Entwicklung einer Portallösung.

Ein weiteres wichtige Ziel der Projektgruppe ist es, die methodischen Grundlagen der industriellen Softwareentwicklung zu vertiefen und eine Softwareentwicklung auf der Basis eines selbst definierten und dokumentierten Softwareentwicklungsprozesses durchzuführen. Innerhalb der Entwicklung von Anwendungssystemen stellt die Klasse der Electronic-Commerce-Anwendungen eine neue Anwendungsdomäne dar. Es ist daher zu untersuchen, ob die bisher bekannten Softwareentwicklungsprozesse für diese neue Anwendungsdomäne optimal eingesetzt werden können, und wenn nicht, wie ein zielgerichteter Softwareentwicklungsprozess modelliert sein muss.

Die Softwareentwicklungsprozesse für Electronic-Commerce-Systeme unterscheiden sich von den Softwareentwicklungsprozessen für konventionelle Anwendungssoftwaresysteme partiell hinsichtlich

- der Art der Tätigkeiten,
- der Rollen, die Tätigkeiten durchführen,
- der Softwarewerkzeuge, die verwendet werden,

so dass konventionelle Softwareentwicklungsprozesse bzgl. ihrer optimalen Einsetzbarkeit (Zeit, Kosten, Ressourcen) für die Entwicklung dieser neuen Klasse von Softwaresystemen betrachtet werden müssen.

1.3 Die Projektgruppe IPSI

Die Projektgruppe orientiert sich an einer industriellen Vorgehensweise. Das bedeutet, dass die Teilnehmer auf Basis eines Rollenmodells Verantwortlichkeiten und Zuständigkeiten für ein oder mehrere Rollen innerhalb des Softwareentwicklungsprozesses einnehmen. Neben den übergreifenden Rollen „Projekt-/Marketingmanager“ (Carsten Seidel), „Chefarchitekt“ (Stefan Puls), „Systemadministrator“ (Stefan Austen) und „Qualitätssicherer/Softwareprozessdokumentator“ (Martin Mocker) sind die Rollen „Web/GUI-Designer“ (Matthias Book) und „Entwickler“ zu besetzen. Die Rolle „Entwickler“ unterteilt sich aufgrund der fachlichen Anforderungen an das EC-Portal in Entwickler für das Procurement (Wahid Bashirazad, Hasan Ghane), das Publishing (Bernhard Flechtker) und das EC-Portal (Chris Haase, Stefan Göbel, Traugott Dittmann, Christian Leifkes) an sich. Die PG-Teilnehmer haben für gewisse Zeiträume bestimmte Rollen inne. Durch eine kontrollierte Rollenrotation wird sichergestellt, dass jeder Teilnehmer mehrere Rollen kennenlernt und sowohl technische als auch nichttechnische Rollen wahrnimmt.

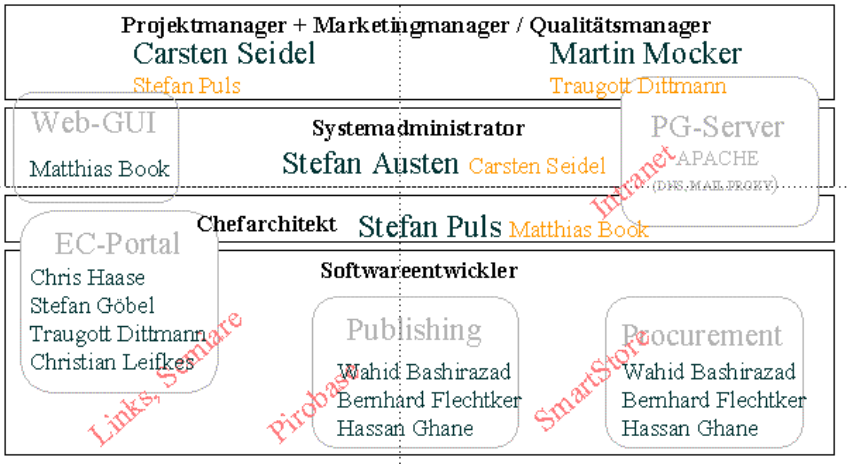


Abbildung 3 - Rollen und Personen

Neben diesen Rollen sind übergreifende Boards vorhanden. Neben dem PG-Board, an dem alle PG-Teilnehmer teilnehmen, wurde das Project-Management-Board (PMB) eingerichtet, das mindestens aus dem Chefarchitekt, dem Projektmanager und dem Qualitätsmanager besteht. Bei Bedarf können weitere Personen dazukommen, wobei die PG-Veranstalter als Berater dazugeladen werden können.

Diese Zuordnung von Personen zu Rollen wurde zu Beginn der Projektgruppe während der Seminarphase getroffen und ist im Laufe der Projektgruppe konkretisiert worden.

Ebenso wie die Entwicklung von konventionellen Anwendungssoftwaresystemen entsprechend eines Softwareentwicklungsprozesses erfolgt, wird ein Softwareentwicklungsprozess benötigt, der die Entwicklung von Electronic-Commerce-Systemen beschreibt. Ein einfacher konventioneller Softwareentwicklungsprozess umfasst mindestens die Phasen Anforderungsanalyse, Systemanalyse, Systementwurf, Implementierung, Test und Betrieb/Wartung.

1.4 Aufgaben / Ziele einer Projektgruppe am FB Informatik

a) Der ausgebildete Informatiker sieht sich in seiner Berufspraxis häufig mit den Fragestellungen konfrontiert, die (aufgrund der raschen Entwicklung des Fachgebietes Informatik und der schnellen Ausweitung und Vermehrung von Einsatzbereichen dieses Fachgebietes) schon relativ bald nach Verlassen der Hochschule für ihn neuartig sind in dem Sinne, dass er nicht ohne weiteres auf Methoden und Techniken zurückgreifen kann, die ihm während seines Studiums vermittelt wurden; der Informatiker ist vielmehr oft gefordert, eine inzwischen erfolgte Weiterentwicklung problemrelevanter Methoden selbständig nachzuvollziehen, diese Methoden seiner Fragestellung anzupassen, sie zu erweitern bzw. neue geeignete Methoden auf der Basis seines Grundwissens zu entwickeln;

b) deren Bearbeitung (aufgrund des zu erwartenden hohen Arbeitsumfanges) der koordinierten Zusammenarbeit vieler Mitarbeiter bedarf, so dass als Arbeitsorganisationsform nur das wohlstrukturierte Mitarbeiterteam sinnvoll erscheint; der Informatiker ist somit oft gefordert, ein konkretes Problem in Teilprobleme zu

zerlegen, die Teilprobleme einer koordinierten Bearbeitung zuzuführen, Lösungen von Teilproblemen zu einer Gesamtlösung zu integrieren.

Lehrveranstaltungsformen herkömmlicher Art (Vorlesungen, Übungen, Seminare, Praktika) können Fähigkeiten zur Bewältigung der geschilderten Schwierigkeiten nur in beschränktem Maße vermitteln. Während sicherlich die Grundlage der bezüglich (a) erforderlichen Flexibilität des Informatikers in Form der Bereitstellung eines breiten, fundierten Basiswissens und in Form der Förderung innovativer Fähigkeiten in Lehrveranstaltungen herkömmlicher Art gelegt werden kann (und muss!), ist doch die konkrete Einübung einer flexiblen Arbeitsweise hier zwangsläufig auf relativ kleine und relativ enge Probleme beschränkt. Darüber hinaus können die bezüglich (b) benötigten Fähigkeiten, Methoden (aus u.U. verschiedenen Spezialgebieten) im Team zu erarbeiten, auszuwählen und anzupassen, diese Methoden in koordinierter Teamarbeit zur Bearbeitung von Teilproblemen einzusetzen und zu einer Gesamtproblemlösung zu integrieren, mit Lehrveranstaltungen herkömmlicher Art kaum gefördert werden. Der Fachbereich Informatik bietet daher seit langem den Studierenden eine spezielle Lehrveranstaltungsform, die *Projektgruppe*, an [oV00]. Die positiven Erfahrungen mit dieser Lehrveranstaltungsform haben inzwischen dazu geführt, den Besuch jeweils einer Projektgruppe für den Studierenden der Informatik zwingend vorzuschreiben.

Projektgruppen zielen darauf ab, Fähigkeiten zur Beherrschung der unter (a) und (b) geschilderten informatikspezifischen Problemstellungen zu vermitteln und einzuüben. Sie bieten einen Rahmen, innerhalb dessen Wissen, Methoden und Techniken aus (u.U. verschiedenen) Teilgebieten der Informatik auf ein konkretes Problem angewandt werden sollen, so dass die Erarbeitung, Anpassung, Erweiterung und Entwicklung problemrelevanter Methoden eingeübt werden kann; ein Problem größeren Umfangs bearbeitet werden soll, so dass das Einarbeiten und Arbeiten in Gruppen (Organisation, Leitung, Koordinierung, Zusammenarbeit, Darstellung und Vermittlung eigener Ideen usw.) eingeübt werden kann; demzufolge die sachlichen und organisatorischen Schwierigkeiten bei der Analyse, Zerlegung und Bearbeitung umfangreicher Probleme deutlich werden und bewältigt werden sollen. Eine Projektgruppe besteht aus einem oder mehreren Projektgruppenveranstaltern und 8-12 Projektgruppenteilnehmern. Der Veranstalter (bzw. zumindest einer der Veranstalter) ist Hochschullehrer oder wissenschaftlicher Mitarbeiter des Fachbereichs Informatik. Bei der Beantragung von Projektgruppen mit externen Betreuern muss ein Hochschullehrer des Fachbereichs Informatik eine Befürwortung der PG und des externen Mitbetreuers abgeben. Die Teilnehmer sind Studierende mit Hauptfach Informatik im zweiten Studienabschnitt (d.h. mit abgeschlossenem Vordiplom). Die Projektgruppe bearbeitet eine konkret und deutlich beschriebene Problemstellung. Im Hinblick auf die Motivierung der Projektgruppenteilnehmer sollte die Problemstellung möglichst realitätsrelevant sein; interdisziplinäre Themen sind ausdrücklich zugelassen; ein externer Produkt- oder Terminzwang ist auszuschließen. Eine Projektgruppe umfasst zwei aufeinanderfolgende Semester. Sie umfasst vom Umfang Arbeiten, die 14-16 Semesterwochenstunden (2 Semester à 6-8 Semesterwochenstunden) herkömmlicher Lehrveranstaltungen entsprechen. Die tatsächliche Arbeitsbelastung des einzelnen Teilnehmers ist so zu bemessen, dass ein paralleler Besuch anderer Lehrveranstaltungen im Gesamtumfang von jeweils 6-10 Semesterwochenstunden möglich ist.

2 Seminarphase

Um das Wissen aller Teilnehmer einer Projektgruppe auf eine homogene Basis zu stellen, werden in einer Seminarphase verschiedenen Vorträge zu den unterschiedlichsten Aspekten, die sich aus der Zielsetzung der Projektgruppe ergeben, gehalten. Die Seminarphase findet in der Regel vor dem eigentlichen Beginn der Projektgruppe statt. Um darüber hinaus das Kennenlernen der Teilnehmer untereinander zu fördern, findet die Seminarphase in Form einer Exkursion statt.

Durch die einleitenden Vorträge „Was sind Electronic-Commerce Portale?“ von Traugott Dittmann und „Beispiel für EC-Portale“ sollten den PG-Teilnehmern die Thematik EC-Portale allgemein und an Hand von Beispielen näher gebracht werden. Während der erste Vortrag EC-Portale in den Kontext von EC-Anwendungen, WWW- und Internet-Anwendungen und die grundlegende Begriffswelt und Terminologie einordnete, sollte sich der nächste Vortrag mit den fachlichen (Zielgruppe, etc.) und softwaretechnologischen (Architektur, Interoperabilität, Integrierbarkeit, etc.) Merkmalen und Eigenschaften von exisitierenden EC-Portalen auseinandersetzen. Gegenstand der Betrachtung waren hier vier bis fünf verschiedene EC-Portale (McOffice, T-Online, ...). Eine Gegenüberstellung und Bewertung der Merkmale und Eigenschaften wurde am Ende dieses zweiten Vortrags vorgestellt.

Da in der PG ein EC-Portal für eine spezielle Zielgruppe entwickelt werden soll, ist es erforderlich, die Anforderungen an eine EC-Anwendung für diese Zielgruppe zu definieren. Das Ziel des Vortrags „Die Aufgaben des

Maklers und des Außendienstmitarbeiters in der Versicherungswirtschaft“ von Hassan Ghane war es daher, einen umfassenden Überblick über das Arbeitsumfeld und die Tätigkeiten eines Maklers/Außendienstmitarbeiters (manuell und DV-gestützt) zu geben, um hieraus Anforderungen für das zu entwickelnde Softwaresystem gewinnen zu können. Welche WWW-Anwendungen existieren und welche fachliche Unterstützung sie für die spezielle Zielgruppe liefern, war Gegenstand des Vortrags „Überblick über WWW-Anwendungen im Versicherungsumfeld“ von Martin Mocker. Ein gemeinsames weiteres Ziel beider Vorträge war es, die PG-Teilnehmer in die Begriffswelt und Terminologie des fachlichen Kontexts einzuführen.

Bei der Entwicklung des EC-Portals sollen Softwaresysteme für spezielle Anwendungsgebiete eingesetzt werden. Die fachliche Vorstellung dieser Systeme sowie ihre softwaretechnologischen Grundlagen (Architektur, Interoperabilität, Integrierbarkeit, Programmiersprache, Kommunikation, etc.) waren ein Gegenstand der Vorträge „Das Contentmanagementsystem Pirobase“ von Bernhard Flechtker, „Das Shopsystem Intershop“ von Wahid Bashirazad und „One-to-One-Marketsystem Broadvison“ von Carsten Seidel. Ziel der Vorträge war weiterhin die Darstellung der Stärken und Schwächen dieser Systeme für ihr spezielles Anwendungsgebiet (Für was ist das System genau einsetzbar?). Der Hauptschwerpunkt der Vorträge lag aber auf der Darstellung des zugrundeliegenden Prozesses für die Entwicklung von EC-Anwendungen mittels dieser Softwaresysteme.

Neben der Vorstellung von wesentlichen Aspekten aus den verschiedenen nationalen und internationalen standardisierten Richtlinien, Guidelines und Styleguides für Benutzungsoberflächen für Softwaresysteme soll die Übertragbarkeit auf WWW-Anwendungen dargestellt werden. Ein kurzer Abriss über Softwaresysteme, deren Vorteile, Merkmale und Eigenschaften zur Erstellung von GUIs sollte ebenfalls erfolgen. Ein weiterer Schwerpunkt des Vortrags „Richtlinien zur Gestaltung von Benutzungsoberflächen für WWW-Anwendungen“ von Matthias Book lag auf der Vorstellung eines Prozesses für die Entwicklung von Benutzungsoberflächen für WWW-/EC-Anwendungen.

Der Schwerpunkt der Vorträge „Technologien für Intra-/Internet-Anwendungen“ von Stefan Austen, „Implementierungstechniken für WWW-Anwendungen: JAVA, Servlets, Applets“ von Chris Haase und „Implementierungstechniken für WWW-Anwendungen: CGI, Perl, Tcl/TK“ von Stefan Puls lag auf der Darstellung der Vorgehensweise und nicht auf den Sprachbesonderheiten. Die Teilnehmer der PG sollten durch diese Vorträge in die Lage versetzt werden, zu entscheiden, welche Technik aufgrund welcher Vor- und Nachteile für eine Problemstellung eingesetzt werden kann. Wie ist die Vorgehensweise beim Einsatz dieser Technik? Welche Besonderheiten treten auf? (z.B. sehen die Schnittstellen von Klassen für Servlets anders aus; was muss man bei der Einrichtung eines bin-Verzeichnisses auf einem WWW-Server für CGI-Skripts beachten? ...)

In dem Vortrag „Entwicklungsprozesse für die Softwareentwicklung“ von Christian Leifkes sollten verschiedene Entwicklungsprozesse und Vorgehensmodelle für die Softwareentwicklung vorgestellt werden. Zuvor sollten die Merkmale und Eigenschaften von Entwicklungsprozessen skizziert werden, bevor die Unterschiede zwischen den einzelnen verschiedenen Entwicklungsprozesse und Vorgehensmodellen und deren initiale Zielsetzung vorgestellt werden. In dem Vortrag „Rollenbasierte Softwareentwicklung“ von Stefan Göbel sollten die verschiedenen Rollen mit ihren Aufgaben während einer Softwareentwicklung vorgestellt werden. Es sollte vorgestellt werden, über welche Qualifikation die jeweiligen Rolleninhaber verfügen müssen und wie die Kommunikation (fachlich, technisch) zwischen den Personen/Rollen erfolgen soll. Dieser Vortrag wurde mit einem Vorschlag, welche Rollen innerhalb der PG zu besetzten sind, abgeschlossen.

Alle Vorträge sind sowohl in ihrer Vortragsform als auch in der schriftlichen Ausarbeitung auf der Homepage des Lehrstuhls Softwaretechnologie (http://ls10-www.cs.uni-dortmund.de) unter dem Stichwort „Projektgruppen“ zu erhalten. In den folgenden Abschnitten findet sich eine Zusammenfassung jedes Vortrages.

2.1 Technologien für Intra- und Internetanwendungen

Da die Projektgruppe das Ziel verfolgt, ein Internet-Portal für VADs zu konzipieren, war dieser Vortrag darauf ausgerichtet, einen Abriss der verwendeten bzw. verwendbaren Technologien zu geben. Als Einstieg wurde zuerst kurz die Geschichte und Entwicklung des Internets beleuchtet, um ein Verständnis dafür zu bekommen, auf welchen Technologien das Internet an sich basiert. Vorgestellt wurden die verwendeten Basisprotokolle TCP und IP und darauf aufsetzende Internetdienste mit ihren eigenen Kommunikationsprotokollen (z.B. *ftp*, *mail*, *WWW*). Da das Portal webseitenbasiert sein soll, wurde auf Besonderheiten des WWW-Protokolls *http* und der Auszeichnungssprache *html* eingegangen. Anschließend wurden Technologien vorgestellt, die die eingeschränkten Möglichkeiten schlichter Hypertext-Darstellung erweitern können. Dazu gehören z.B. die dynamische Seitengenerierung mittels serverbasierter Programmierung (*ASP*, *Servlets*), Möglichkeiten der Datenverarbeitung

auf Webservern (*CGI*, *Perl*), Anbindung von Datenbanken (*PHP*), Programmiermöglichkeiten für WWW-Browser mittels Programmier- oder Scriptsprachen (*JAVA*, *JavaScript*, *ActiveScripting*).

2.2 Shopsystem INTERSHOP

Als eine mögliche Softwarelösung für das Portal-Subsystem „Procurement“ wurde in diesem Vortrag das Shopsystem INTERSHOP der Intershop Communications GmbH näher betrachtet. INTERSHOP ist ein eigenständiges Produkt, mit dessen Hilfe Shopsysteme für die Benutzung im Internet und Netzen gleicher Technologie (Intranet) erstellt und betrieben werden können. INTERSHOP verfügt über ein breites Leistungsspektrum und wird im Bereich Business-to-Business und Business-to-Consumer eingesetzt. Im Rahmen des Vortrags wurden zunächst die Produktpalette und die sogenannte „Vier-Ebenen Client-Server Architektur“ von INTERSHOP vorgestellt. Anschließend wurde der Funktionsumfang von INTERSHOP bezüglich Skalierbarkeit, Benutzerfreundlichkeit und Sicherheit erläutert. Im nächsten Schritt wurden die von INTERSHOP zur Verfügung gestellten Schablonen und Hilfsassistenten zur Erstellung, Konfiguration und Pflege eines Shops beschrieben. Abschließend wurden die für einen Shop-Kunden verfügbare Funktionalität und die Einsatzmöglichkeiten von INTERSHOP in der Projektgruppe untersucht.

2.3 Richtlinien zur Gestaltung von Benutzungsoberflächen für WWW-Anwendungen

Sowohl bei der Konzeption ganzer Web-Sites wie auch bei der Gestaltung einzelner Seiten helfen dem Web-Designer eine Reihe von Richtlinien, die in diesem Referat vorgestellt wurden. Dazu wurden zunächst die grundlegenden Eigenschaften und Probleme interaktiver Systeme betrachtet und Grundprinzipien aufgeführt, die bei ihrer Gestaltung zu beachten sind. Die gewonnenen Erkenntnisse wurden anschließend auf das WorldWideWeb übertragen: Das Referat untersuchte, welche Elemente aus dem traditionellen Print- und GUI-Design Äquivalente im WWW-Bereich haben und welche neuen, web-spezifischen Aspekte zu beachten sind. Dabei wurden sowohl die Gestaltung der Web-Site als Ganzes als auch der Einsatz konkreter Elemente und Techniken angesprochen. Die entwickelten Gestaltungsregeln wurden im nächsten Schritt in einen umfassenden Entwicklungsprozess für Web-Anwendungen integriert, denn wirklich „brauchbare“ Web-Anwendungen entstehen nicht nur durch sklavische Anwendung der Richtlinien, sondern in einem Prozess, der sich durchgehend an den Eigenschaften und typischen Aufgaben der Nutzer orientiert. Abschließend wurde eine Auswahl von WYSIWYG- und code-orientierten HTML-Editoren im Hinblick auf ihre Nutzbarkeit in der Projektgruppe untersucht.

2.4 Was sind Electronic-Commerce-Portale?

Heute ist „Electronic-Commerce“ kein Fachbegriff mehr für Internetbesessene, sondern auf dem besten Weg, sich in unserem Sprachgebrauch fest zu verankern. Dazu haben auch die Medien mit unzähligen Artikeln und Berichten zu dem Thema ihren Teil beigetragen. Inzwischen weiss jeder, dass das Internet nicht nur zum „Surfen“ geeignet ist, sondern auch ein riesiges Einkaufszentrum darstellt.

Die Begeisterung über die neuen Möglichkeiten und die Menge der angepriesenen Objekte weicht allmählich der Frage, wie man sich in der Flut der Angebote über Wasser halten kann, denn deren unüberschaubare Masse verhindert heute schon in vielen Bereichen eine sinnvolle Nutzung von Electronic-Commerce. Ein Internet-Portal will sich für den Anwender durch die Angebots- und Informationsflut kämpfen und ihm bei der Auswahl helfen. Der Bedarf wird täglich größer.

Die Möglichkeiten und Grenzen des noch jungen Feldes Internet-Portale wurden aufgezeigt und ein Überblick über Electronic-Commerce allgemein gegeben.

2.5 Das Contentmanagementsystem Pirobase

Informationen stellen eine wichtige Entscheidungsgrundlage dar. Sie werden oft mit Hilfe von Webservern bereitgestellt, um jederzeit darauf zugreifen zu können. Bei der klassischen Verbreitung von Informationen über Webserver ergeben sich hohe Kosten für die Pflege des Inhalts. 90 % der Kosten einer Web-Site fallen für die Wartung und Pflege dieser Systeme an. Contentmanagementsysteme ermöglichen hier eine Verbesserung.

Es werden Werkzeuge angeboten, die es jedem ermöglichen, Inhalte wie Text, Bilder, Audio-Videosequenzen für die Website bereitzustellen, ohne dass man sich um die Programmierung kümmern muss.

Technisch basiert Pirobase auf der Trennung von Inhalt und Programmierung einer Website. Eine HTML-Seite wird dynamisch aus einer Datenbank erzeugt und mittels Templates formatiert und ausgegeben. Pirobase benutzt dabei die Three-Tier-Architektur. Auf Tier 1 befinden sich die Clients in Form von gängigen Web-Browsern wie Netscape Navigator oder Internet Explorer. Auf Tier 2 befinden sich der Webserver und Pirobase und auf Tier 3 die einzelnen Dienste und Datenbanken.

2.6 Die Aufgaben des Maklers und Außendienstmitarbeiters in der Versicherungswirtschaft

In der Versicherungswirtschaft spielen Makler und Außendienstmitarbeiter eine wichtige Rolle. Sie sind als Vermittler zwischen Kunden und Versicherungsgesellschaften tätig. Makler stehen den Kunden bei den Entscheidungen zur Wahl des Versicherers zur Verfügung, wobei sie aus einer Vielfalt von Versicherern auswählen.

In diesem Vortrag wurde zuerst versucht, allgemein die Organisation der Versicherungsgesellschaften und Vertriebsorgane eines Versicherungsunternehmens zu verdeutlichen, dann wurde detailliert darauf eingegangen, welche Ziele, Aufgaben und Pflichten Außendienstmitarbeiter bzw. Makler haben.

Zum Schluss folgte die Beziehung zwischen den Vermittlern und dem Internet. Es wurde erläutert, wie die Vermittler das Internet als Hilfsmittel neben Telefon, Fax, SMS, Papier u.a. benutzen, um Verträge abzuschließen, zu verlängern oder neue Kunden zu akquirieren.

2.7 Rollenbasierte Softwareentwicklung

Jeder Teilnehmer der PG IPSI füllt eine oder mehrere Rollen, die für diese Projektgruppe wichtig sind, aus; dabei bestimmt die Zusammenarbeit und Kommunikation der einzelnen Rollen maßgeblich den Erfolg einer Softwareentwicklung. In diesem Vortrag wurden deshalb alle Rollen, die nach Meinung des Autors in der PG IPSI besetzt sein sollten, beschrieben. Dazu wurden sowohl Rollen der klassischen Softwareentwicklung als auch speziell für diese Projektgruppe entwickelte Rollen vorgestellt, indem jeweils die Aufgaben, die benötigte Qualifikation des Rolleninhabers, die zu erstellenden Dokumente und die Kommunikation mit anderen Rollen untersucht wurde. Anschließend folgte die Beschreibung der verschiedenen klassischen bzw. PG-spezifischen Gremien, die regelmäßig einberufen werden sollten.

2.8 JAVA, Servlets, Applets

Die Programmiersprache JAVA hat mit dem Erfolg des WorldWideWeb in den letzten Jahren zunehmend an Bedeutung gewonnen, da sich JAVA als rechner- und betriebssystemunabhängige Sprache besonders gut für den Einsatz im Internet eignet. Mit JAVA können aber nicht nur Anwendungen erstellt werden, die im Browser laufen, sogenannte Applets, sondern auch eigenständig laufende Applikationen. Des weiteren besteht die Möglichkeit, das Request-/Response-Verhalten von Webservern durch Servlets zu unterstützen. Servlets sind Anwendungen, die auf Server-Seite laufen und sind vergleichbar mit CGI-Anwendungen. Mit Hilfe der noch recht jungen Technologie JAVA Server Pages können HTML-Seiten dynamisch generiert werden.

Die Seminararbeit gab zunächst einen kurzen Überblick über die Anwendungen, die mit JAVA erstellt werden können, um dann einige Elemente von JAVA vorzustellen, die für die Erstellung von Applets und Servlets benötigt werden. Beispielhaft sei hier die Gestaltung der Benutzungsoberflächen mit Hilfe des Abstract Window Toolkit bzw. Swing oder die Vorstellung des Interface-Konzeptes genannt. Anschließend wurde die Vorgehensweise zur Erstellung sowohl von JAVA Applets als auch von JAVA Servlets erläutert. Eine Betrachtung des Konzeptes von JAVA Server Pages wurde ebenfalls vorgenommen.

2.9 Entwicklungsprozesse in der Softwaretechnik

Die Durchführung komplexer Softwareprojekte ist mit vielen Teilaufgaben verbunden, wie z.B. der Qualitätssicherung. Zu deren Beherrschung werden seit den 70er Jahren Modelle entwickelt, die versuchen, das Vorgehen bei der Durchführung von komplexen Projekten in Form eines Leitfadens zu beschreiben. Dadurch soll der Prozess übersichtlicher werden, so dass man ihn kontrollieren kann. Zunächst wurden Modelle wie das Wasserfallmodell entwickelt, dass zwar aus Managementsicht erhebliche Vorteile bringt, aber dem tatsächlichen Vorgehen nicht entspricht. Modifikationen dieses Modells versuchen die gravierenden Kritikpunkte auszuräumen. Einige wichtige sind das iterierte und das evolutionäre Wasserfallmodell. Diese Modelle werden heute mehrheitlich

eingesetzt, obwohl auch sie nicht in der Lage sind, den Entwicklungsprozess hinreichend genau nachzubilden. Seit Anfang der 90er Jahre werden objektorientierte Vorgehensmodelle entwickelt. Diese sehr generisch ausgelegten Modelle gehen auf die besonderen Anforderungen objektorientierter Softwareentwicklung ein. In diesem Vortrag wurden, neben den „traditionellen“ Varianten des Wasserfallmodells, drei objektorientierte Vorgehensmodelle beschrieben: der „Rational-Objectory-Process“, das „VModell97“ und der „IBM-OOTC-Process“. Anhand definierter Kriterien wurden diese Modelle miteinander verglichen.

2.10 WWW-Auftritte in der Versicherungsbranche

Aus der Versicherungsbranche sind zur Zeit ca. 150 Unternehmen im WWW präsent.

Dabei kann man drei Ausbaustufen von Electronic-Commerce-Engagements identifizieren: informations-, interaktions- und transaktionsorientierte.

Die meisten Auftritte der Versicherer befinden sich auf der ersten Ausbaustufe, also der Anbietung von größtenteils statischen Informationen. Damit werden die potentiellen Möglichkeiten des Internets als zusätzlichem Vertriebskanal zur Kundengewinnung und -bindung nicht voll ausgeschöpft. Die Probleme, die zu dieser Situation führen, können rechtlicher, gesellschaftlicher, produktspezifischer und technischer, aber vor allem auch organisatorischer Natur sein.

Der Seminarvortrag schlug zunächst eine Klassifizierung von WWW-Auftritten vor. Anschließend wurden sowohl die Möglichkeiten der Ausgestaltung von Web-Auftritten für Versicherer aufgezeigt als auch die dabei auftretenden Hindernisse angesprochen. Abschließend wurden neue Möglichkeiten zur Überwindung dieser Hindernisse skizziert.

2.11 Implementierungstechniken für WWW-Anwendungen: CGI & Perl

Der Seminarvortrag gliedert sich in drei Schwerpunkte: CGI als de facto Standard, Perl als Programmiersprache und der Internet Information Server (IIS) von Microsoft als Beispiel eines möglichen Webservers. Die mit am häufigsten angewandte Technik zum Zusammenstellen dynamischer Web-Dokumente auf einem Webserver ist das Common Gateway Interface (CGI). CGI hat die Stärken, dass viele Sprachen auf unterschiedlichen Systemen verwendet werden können, dass es den Zugriff auf andere Systeme bereitstellt und dass es eine genau definierte Methode zur Ein- und Ausgabe von Daten beinhaltet, mit der seine Features benutzt werden können. Perl ist eine Programmiersprache, die mit CGI verwendet werden kann. Perl ist eine interpretierte Sprache, die für die Verarbeitung von Textdateien, dem Filtern von Informationen aus diesen Dateien und der Ausgabe von auf diesen Informationen basierenden Berichten optimiert ist. Für Perl existiert eine sehr mächtige Klassenbibliothek (CPAN), die für nahezu jeden Bereich Programmierbeispiele und fertige Lösungen liefern kann. Der IIS ist ein Webserver, der unter Windows NT in der Lage ist, sowohl CGI als auch - durch eine zusätzliche Installation - Perl bereitzustellen. Jedoch ist ein gravierender Nachteil dieser Kombination, dass CGI für jede Anfrage an den Webserver einen eigenen Prozess startet. Der IIS stellt aus diesem Grund eine eigene Internet Service API zur Verfügung, die mit einer direkten Anbindung an die Microsoft Entwicklungswerkzeuge zumindestens unter Windows NT eine bessere Lösung für die Projektgruppe IPSI liefert.

2.12 Personalisierungs- und Berechtigungskonzepte im WWW und das Marketingsystem Broadvision One-To-One

Durch die Zunahme von elektronischem Handel im Internet hat sich eine breite Palette von E-Commerce-Systemen entwickelt. Moderne Systeme versuchen, ihr Angebot an jeden einzelnen Benutzer – also Käufer – anzupassen. Jeder Benutzer erhält bei seinem virtuellen Einkaufsbummel eine auf seine Bedürfnisse und Interessen abgestimmte Produktpalette. Diesen Vorgang nennt man Personalisierung.

Ein Problem vernetzter Computer ist, Dienste und Informationen eines Systems nur berechtigten Personen zugänglich zu machen und alle anderen Anfragen abzuweisen.

Der Vortrag führte die Begriffe *Personalisierung* und *Berechtigung* ein und zeigte diese Konzepte anhand realer Beispiele. Im zweiten Teil wurden einige grundlegende Techniken gezeigt, mit denen webbasierte Anwendungen die Ziele von Personalisierung und Berechtigung erreichen. Der dritte Teil stellte das personalisierte Marke-

tingsystem *One-To-One* der Firma Broadvision vor. Dieses System zeichnet sich besonders durch seinen hohen Personalisierungsgrad, seine Präsentationsunabhängigkeit und seine gute Interoperabilität aus.

3 Tutorial

In diesem Kapitel soll die Arbeitsweise der Benutzer des IPSI-Portals (der „fachliche Workflow“) in Form eines Tutorials beschrieben werden.

Hierzu werden zu allen Arbeitsgängen Screenshots des Systems mit kurzen Erläuterungen dargestellt. Nach diesen Arbeitsgängen ist das Kapitel auch gegliedert.

Da es zwei Rollen von Benutzern gibt, ist grundsätzlich zu unterscheiden zwischen den Arbeitsgängen eines Versicherungsaußendienstmitarbeiters (VAD) und denen eines Agenturleiters (AL) bzw. des Versicherungsunternehmens (VU). Die letztgenannten (AL, VU) übernehmen eher administrative Tätigkeiten.

Die Begriffe VAD respektive AL/VU und Benutzer werden in den entsprechenden Abschnitten als synonym betrachtet.

3.1 Arbeitsabläufe eines VAD

Login

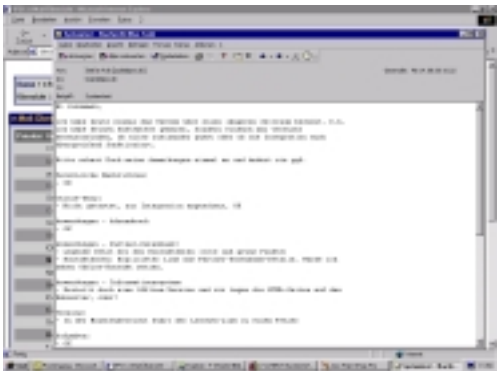
Um die Arbeitsabläufe mit Hilfe des IPSI-Portals durchzuführen, muss sich der VAD zunächst beim IPSI-Portal anmelden. Dies geschieht über den Login-Bildschirm durch Eingabe von Benutzername und Passwort.

Startbildschirm

Nach der Anmeldung befindet sich der Benutzer auf dem Startbildschirm und kann von dort aus alle der im Folgenden beschriebenen Arbeitsabläufe ausführen.
Der Startbildschirm gliedert sich in verschiedene rechteckige Inhaltsbereiche, sog. Portlets. Ein Portlet enthält das Navigationsmenü, die übrigen Portlets enthalten die wichtigsten Inhalte und Funktionen für den Benutzer: die aktuellsten Nachrichten, neueste Shop-Artikel passend zu den Interessenlagen des Benutzers, die nächsten Termine und Aufgaben, sowie Links zu den wichtigsten Anwendungen, die im IPSI-Portal integriert sind, wie das Informationssystem oder die Partnerdatenbank. Schließlich bietet der Startbildschirm die Möglichkeit, über das gesamte Portal mittels der Quicksuche zu suchen.

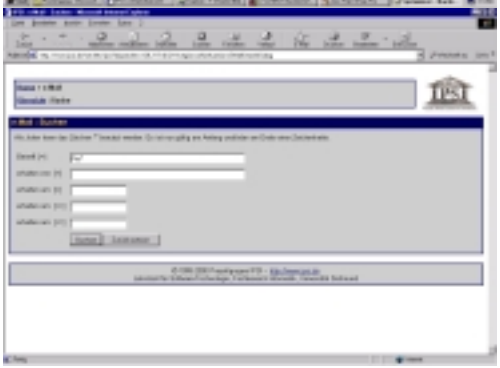
Nachrichten anzeigen

Mit dem IPSI-Portal kann der Benutzer seine sämtlichen Nachrichten (Emails) verwalten, die er auch mit seinem Outlook 2000 System bearbeitet. D.h. IPSI soll Outlook nicht ersetzen, sondern ermöglicht dem Benutzer lediglich den in das Portal integrierten Zugriff auf seine Outlook-Daten ohne das Portal zu verlassen.
Um sich zunächst Nachrichten anzusehen, wählt der Benutzer auf dem Startbildschirm den Menüpunkt „Nachrichten“ (Menüpunkte befinden sich immer im obersten Portlet der IPSI-Masken).
Der Benutzer findet sich dann auf einer Seite mit der Nachrichtenübersicht wieder. Hier wird eine Liste aller Emails angezeigt, die sich im Inbox-Ordner des Outlook 2000 Systems des Benutzers befinden.

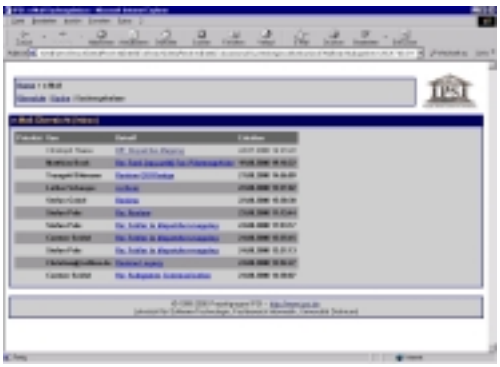


Ein Klick auf eine Nachricht öffnet das Outlook-System des Benutzers und zeigt die gewählte Nachricht in einem eigenen Fenster an. Hier kann der Nutzer nun alle Funktionen durchführen, die ihm von Outlook zur Verfügung gestellt werden, wie z.B. auf die Nachricht antworten, sie drucken oder löschen. Wenn die Nachricht gelesen wurde, schließt der Benutzer das Fenster und befindet sich wieder im IPSI-Portal an der vorherigen Stelle.

Nachrichten suchen

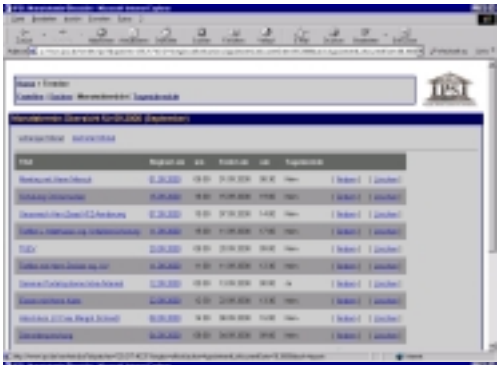


Wenn der VAD eine Menge von Nachrichten besitzt und eine bestimmte Nachricht sucht, die z.B. den Namen des Kunden „Müller“ im Betreff enthält, so kann diese über die Suchmaske gefunden werden, in der das entsprechende Feld mit einem Suchtext gefüllt wird. Ein weiteres Suchkriterium ist das Datum, an dem die Nachricht erhalten wurde. Dieses kann als Intervall oder als festgesetztes Datum angegeben werden.
Die Suchmaske kann über den Menüpunkt „Suche“ erreicht werden.

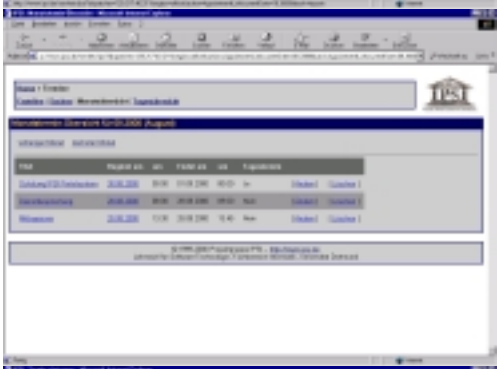


Die den Suchkriterien entsprechenden Ergebnisse werden anschließend in einer Liste angezeigt. Um sich eine der gefundenen Nachrichten anzusehen, klickt der VAD wiederum einfach auf die Nachricht. Diese wird dann, wie schon zuvor beschrieben, in einem eigenen Outlook-Fenster dargestellt.

Termine einsehen



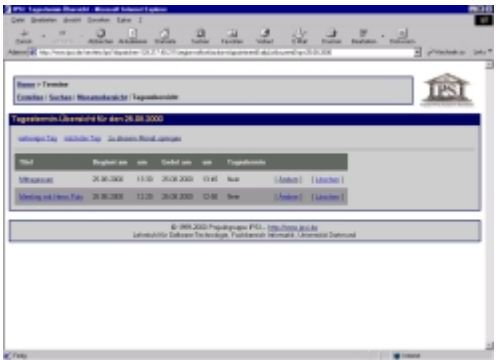
Neben den Nachrichten werden auch die in Outlook 2000 eingetragenen Termine im IPSI-Portal integriert.
Möchte der Benutzer die Termine des aktuellen Monats einsehen, so wählt er im Startbildschirm den Menüpunkt „Termine“. Er erhält sodann eine Liste aller Termine in der Monatsübersicht.
In solch einer Terminliste hat der Benutzer die Möglichkeit, jeden Termin anzusehen, zu ändern oder zu löschen.



Die Terminübersichten der einzelnen Monate können nacheinander eingesehen werden. D.h. der VAD kann nacheinander alle Termine der Monate eines Jahres überblicken.
Ein Klick auf einen Termin zeigt diesen an. Wie ein Termin in der Detailansicht angezeigt wird, zeigt die nächste Abbildung im Bereich „Termin erstellen“.

Termin erstellen / ändern

Um einen Termin zu erstellen, wählt der Benutzer die entsprechend benannte Funktion im Terminbereich und kann dann die in der Eingabemaske dargestellten Eingabefelder ausfüllen, um den Termin zu spezifizieren. Die auszufüllenden Felder sind Titel, Beginn des Termins, Terminende, Erinnerungszeit, Erinnerungsmodus (SMS, Email, Fax), Terminbeschreibung und die Wiederholungsart (für Termine die sich täglich, wöchentlich oder monatlich wiederholen).

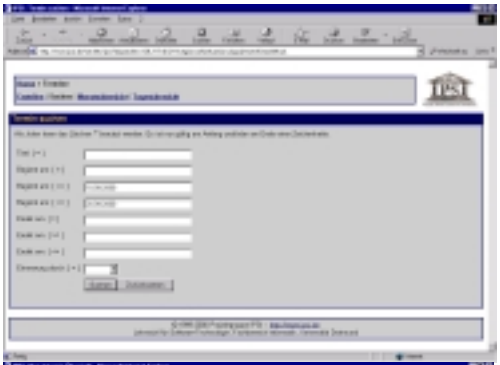


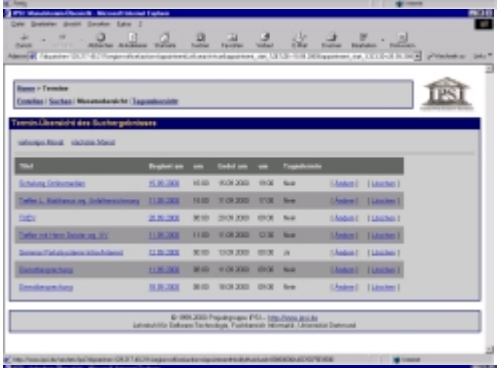
Der Termin wird dann entsprechend in das Outlook 2000 System des Benutzers eingetragen und in der Tagesansicht des IPSI-Portals angezeigt.
Das Ändern von bereits bestehenden Terminen geschieht genauso. Der VAD klickt auf die Funktion „Ändern“ die neben jedem Termin in der Terminliste angezeigt wird. Er sieht dann dieselbe Maske wie beim Erstellen und kann nach den Änderungen einfach den Button „Speichern“ drücken.

Termine löschen

Termine, die nicht mehr relevant sind (z.B. weil sie abgesagt wurden) können aus dem System entfernt werden. Dies geschieht über die Auswahl der Funktion „Termin löschen“ in einer Terminliste. Zur Sicherheit wird der Benutzer gefragt, ob er den gewählten Termin tatsächlich löschen will. Erst wenn der Benutzer diese Sicherheitsabfrage mit „Ja“ beantwortet, wird der Termin endgültig gelöscht.

Termine suchen





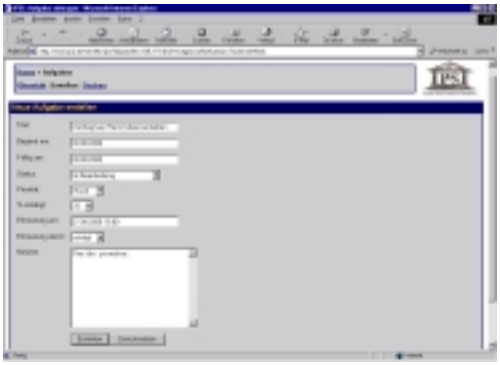
Wenn der VAD nach einem bestimmten Termin sucht, kann er dies durch Angabe von Suchkriterien in der Termin-Suchmaske tun. Diese ist über den Menüpunkt „Suche“ im Terminbereich erreichbar. Der VAD hat die Möglichkeit nach Terminen mit einem bestimmten Titel, einem bestimmten Datum oder einer Uhrzeit zu suchen. Die ausgefüllten Suchfelder werden bei der Suche durch ein logisches UND verknüpft.

Die den Suchkriterien entsprechenden Termine werden nach Beenden des Suchvorganges in einer Terminliste dargestellt und können entsprechend angezeigt, geändert oder gelöscht werden (s.o.). Dies geschieht durch einen Klick auf die hinter dem gewünschten Termin als Link dargestellte Funktion (zum Ändern oder Löschen) bzw. auf den Termintitel (zum Anzeigen).

Aufgaben anzeigen

Aufgaben sind eine weitere Art von Informationen aus dem Outlook 2000 System des Benutzers, die im IPSI-Portal integriert ist.
Der VAD kann sich seine aktuellen Aufgaben in einer listenartigen Übersicht anzeigen lassen. Diese Liste ist über den Menüpunkt „Aufgaben“ zu erreichen. Dargestellt werden der Titel der Aufgabe, deren zeitliche Daten (Beginn und Endtermin), der Status und die Priorität. Der VAD hat die Möglichkeit sich die Aufgabe durch einen Klick auf den Titel anzusehen (s.u.), bzw. durch einen Klick auf die Funktionslinks hinter jeder Aufgabe in der Liste zu löschen oder zu ändern.

Aufgaben erstellen / ändern

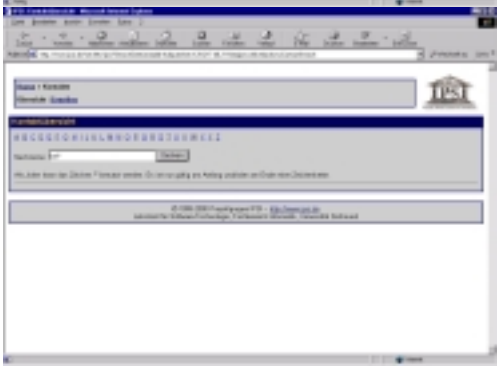


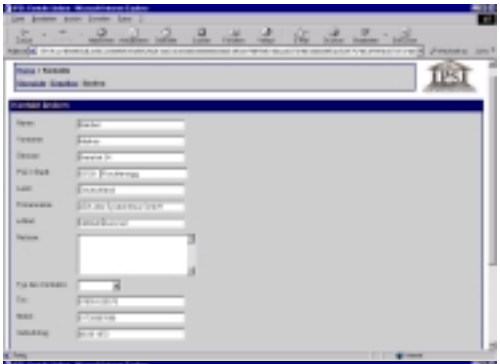
Natürlich kann ein VAD auch neue Aufgaben erstellen oder bestehende Aufgaben ändern.
Dies geschieht analog zum Erstellen/Ändern von Terminen oder Nachrichten in einer entsprechenden Eingabemaske mit Eingabefeldern zur Beschreibung der Aufgabe. Diese Eingabemaske wird im Aufgabenbereich des Portals über den Menüpunkt „Erstellen“ erreicht. Hier sind Titel, Status, Erinnerungsdaten, Beschreibungen und zeitliche Rahmendaten zu spezifizieren.

Aufgaben suchen

Bestimmte Aufgaben können über eine Suchmaske gesucht werden. Diese Suchmaske ist im Aufgabenbereich des Portals über den Menüpunkt „Suchen“ erreichbar. Aufgaben können u.a. über den Titel, einen gemeinsamen Status oder ein gemeinsames zeitliches Datum gesucht werden. Die Eingaben werden bei der Suche mit einem logischen UND verknüpft. Dabei können Intervalle definiert werden. So kann der VAD alle Aufgaben suchen, die zwischen dem 01.01.2000 und dem 01.02.2000 fällig waren und den Status „erledigt“ haben.
Die zur Suchanfrage passenden Aufgaben werden dem VAD in einer Liste angezeigt. In dieser Liste werden die Aufgaben wie gewohnt mit Titel, Anfangs- und Fälligkeitsdatum, Status und Priorität angezeigt. Auch aus dieser Liste kann sich der VAD Aufgaben in der Detailansicht anzeigen lassen, ändern oder löschen.

Kontakte anzeigen

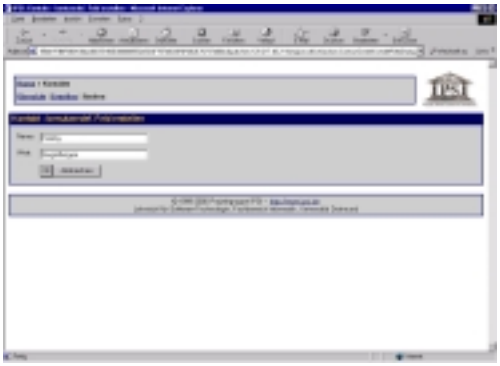




Als letzte Informationsart des Outlook 2000 Systems werden auch Kontakte zu anderen Personen (z.B. zu Kollegen oder Kunden) im IPSI-Portal verwaltet. Um einen bestimmten Kontakt auffinden zu können, sind die Kontakte nach den Nachnamen der Personen alphabetisch geordnet. Entsprechend des Nachnamens kann der VAD den jeweiligen Buchstaben aus einer Liste auswählen und bekommt alle mit diesem Buchstaben beginnenden Kontakte in einer Liste angezeigt.
Ein Klick auf einen Kontakt zeigt dem Benutzer die Detaildaten an. Diese bestehen u.a. aus dem Nach- und Vornamen, den Adressdaten (Straße, PLZ, Ort, Land), Firmenzugehörigkeit, Email, Telefon usw. Wichtig ist auch die Information über den Kontakttyp wie z.B. privat oder geschäftlich.

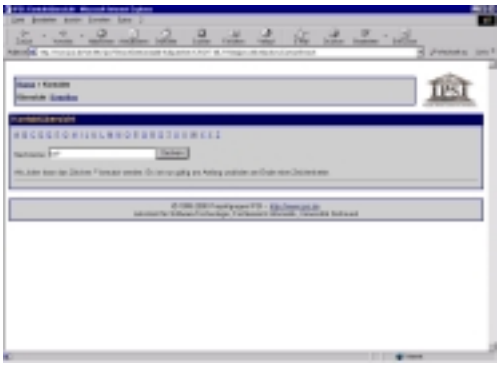
Kontakte erstellen/ändern

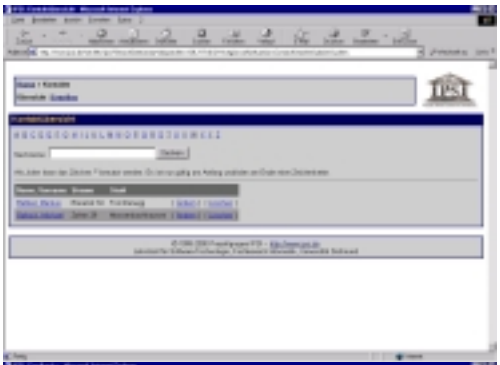
Der VAD kann neue Kontakte erstellen oder ändern. Dies geschieht wiederum über eine Eingabemaske mit den entsprechenden Eingabefeldern zur Beschreibung eines Kontaktes, die schon zuvor vorgestellt wurden. Nach Eingabe aller gewünschten Daten zu dem Kontakt wird dieser über den „Ändern“-Button abgespeichert. Die Eingabemaske wird über die Funktion „Erstellen“ im Kontaktbereich des Portals erreicht.



Insbesondere können auch benutzerdefinierte Felder einem Kontakt hinzugefügt werden, in denen beliebige zusätzliche Informationen abgelegt werden können. Dazu klickt der VAD beim Erstellen oder Ändern eines Kontakts auf den Link „benutzerdefinierte Felder hinzufügen“. In der erscheinenden Maske kann der VAD dann beliebige Schlüssel/Wert-Paare anlegen. Z.B. ist ein Schlüssel „Hobby“ mit dem Wert „Fallschirmspringen“ denkbar. Diese Angabe wird dann mit dem Kontakt abgespeichert.

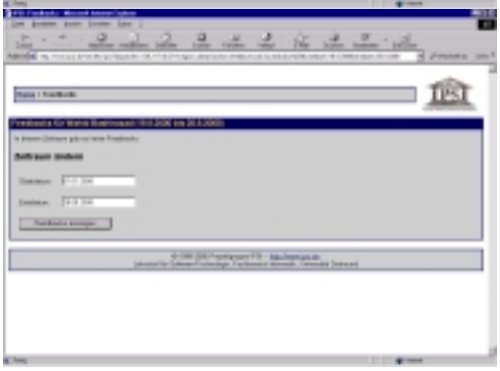
Kontakte suchen



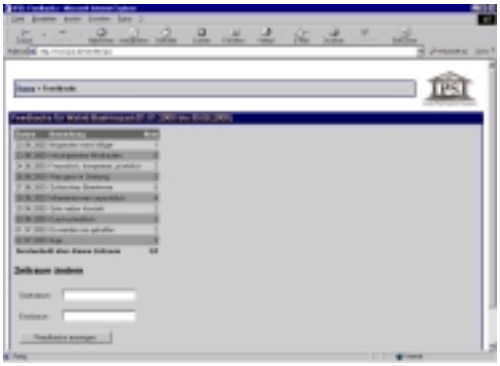


Um einen speziellen Kontakt aufzufinden kann der VAD diesen über eine Suchmaske suchen. Hierzu kann der Nachname als Suchkriterium benutzt werden. Der VAD kann entweder den vollständigen Nachnamen in das entsprechende Feld eingeben oder einen Teil des Nachnamens (wenn er den vollständigen Namen z.B. nicht kennt) und das Jokerzeichen „*“ einsetzen. Mit dem Suchmerkmal „Mo*“ werden so alle Kontakte gefunden, deren Nachname mit „Mo“ beginnt und beliebig endet.
Die Ergebnisse der Suche werden in einer Liste mit Namen und Adressdaten dargestellt. Durch einen Klick auf den Namen des Kontaktes kann sich der VAD die Detailansicht anzeigen lassen. Um einen gefundenen Kontakt zu ändern oder zu löschen klickt der VAD auf die hinter jedem Kontakt dargestellten Funktionslinks.

Kundenfeedbacks anzeigen



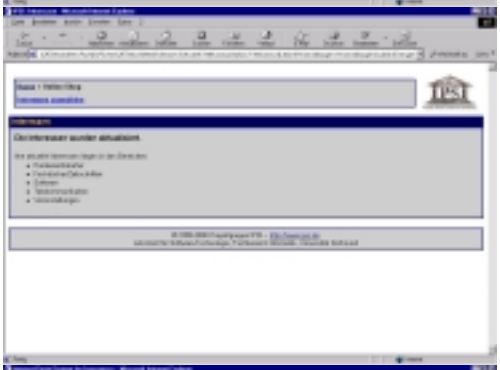
Um die Meinung seiner Kunden über sich selbst im Auge zu behalten, kann sich der VAD alle bei ihm eingegangenen Feedbacks von Kunden ansehen. Der VAD bestimmt den Zeitraum, für den die Feedbacks angezeigt werden sollen.
Der VAD gelangt in den Feedback-Bereich durch Auswahl der Funktion „Feedbacks“ im Startbildschirm.



Die Feedbacks werden samt Datum der Abgabe des Feedbacks Kommentar des Feedback-Gebers und dessen Bewertung in einer Liste angezeigt. Die Bewertungen entsprechen einer Schulnote, sind also aus dem Intervall [1..6]. Am Ende der Liste wird die Durchschnittsnote aller in der Liste angezeigten Feedbacks dargestellt. Auf derselben Seite hat der VAD die Möglichkeit, den Zeitraum der anzuzeigenden Feedbacks zu ändern, also eine erneute Suche nach Feedbacks eines anderes Zeitraums zu starten.

Shopping-Interessen festlegen

Um dem VAD in erster Linie solche Artikel des Shops anzuzeigen, die ihn auch wirklich interessieren, kann der VAD selbst seine Interessensgebiete festlegen. Hierzu ist eine Reihe von Interessen im Portalbereich „Online-Shop“ vorgegeben, der vom Startbildschirm über den Link „Interessen auswählen“ erreichbar ist. Jeder Interessensbereich (z.B. „Schreibwaren“, „Hardware“, etc.) kann über eine Checkbox aktiviert oder deaktiviert werden.



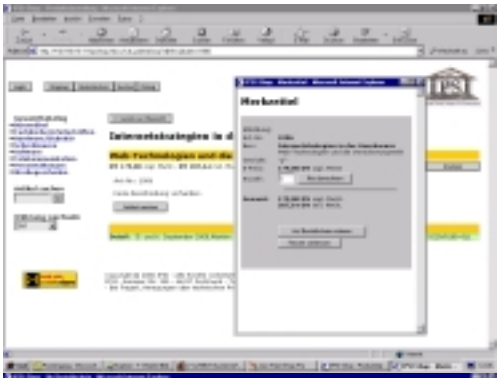
Wenn der VAD seine jeweiligen Interessen angewählt hat, speichert er diese mit einem Klick auf den Button „Eintragen“ ab. Dass die Angaben gespeichert wurden, wird dem VAD in einer Bestätigungsmaske angezeigt. Hier werden die zuvor ausgewählten Interessen nochmals für den VAD wiederholt. Wenn der VAD die Interessen abermals ändern möchte, kann er auf den Menüpunkt „Interessen auswählen“ klicken.

Die Wahl der Interessengebiete schlägt sich direkt in der Anzeige von passenden Artikeln im Shop-Portlet nieder. Hier werden solche Artikel angezeigt, die den vom VAD angewählten Interessen entsprechen. Hat der VAD z.B. „Hardware“ als ein Interessensgebiet angegeben, so werden ihm im Shop-Portlet auf der Startseite vorrangig Hardware-Artikel angezeigt. Natürlich muss der VAD diese Festlegung nur einmal initial vornehmen. Nach der Festlegung werden diese Einstellungen permanent bis zur nächsten Änderung benutzt.

Shopping-Artikel anzeigen

Um sich einen vorgeschlagenen Artikel anzusehen, kann der VAD einfach auf den entsprechenden Vorschlag im Shop-Portlet klicken. Ihm wird dann eine Seite des integrierten Shops mit der Artikelbeschreibung angezeigt. Hier kann der VAD alle Funktionen des Shops benutzen. D.h. er kann den Artikel bestellen, die Artikeldaten drucken oder andere Artikel suchen bzw. durch den Shop „browsen“.

Artikel bestellen



Eine Bestellung eines Artikels funktioniert wie in Shops üblich. Durch einen Klick auf den „Bestellen“-Button wird ein Bestellfenster geöffnet. Nach der Angabe einer Menge kann der Artikel in den Warenkorb gelegt werden. Dort verbleibt der Artikel, bis sich der VAD entschließt den gesamten Warenkorb nun zu bezahlen.

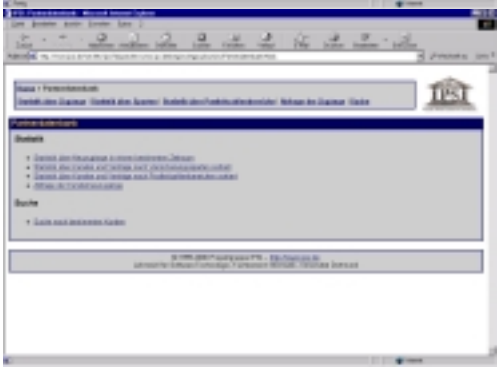
Der Warenkorb kann an der „virtuellen Kasse“ bezahlt werden, die über den entsprechenden Menüpunkt erreicht wird. Hier wird der Bestellschein (oder Warenkorb) erneut angezeigt. Der VAD muss die Richtigkeit der Angaben zunächst bestätigen.

Dabei ist zu beachten, dass der Budgetrahmen des VAD, der für jeden VAD durch den AL festgelegt werden kann, vom System überprüft wird und somit nicht überschritten werden kann. Wenn der Budgetrahmen eingehalten wurde, so kann der VAD die Bestellung endgültig auslösen.

Sollte der individuelle Budgetrahmen durch die Artikelauswahl des VAD überschritten worden sein, so wird der VAD hierauf in einer Maske hingewiesen. Der Benutzer hat hier die Möglichkeit die Bestellung zu korrigieren durch das Löschen einzelner Positionen oder Mengenänderungen. Wenn durch diese Anpassungen der Budgetrahmen wieder eingehalten wird, kann die Bestellung erfolgreich abgeschlossen werden.

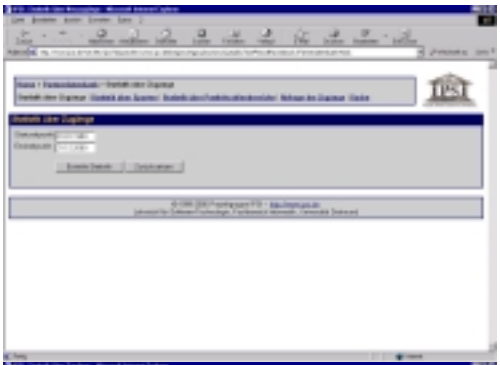
Sobald die Bestellung erfolgreich abgeschlossen wurde, wird dies dem VAD über eine Maske angezeigt. Von hier aus hat der VAD nochmals die Möglichkeit, die Auftragsbestätigung mit der Angabe aller bestellten Artikel einzusehen oder zu drucken.

Partnerdatenbank benutzen

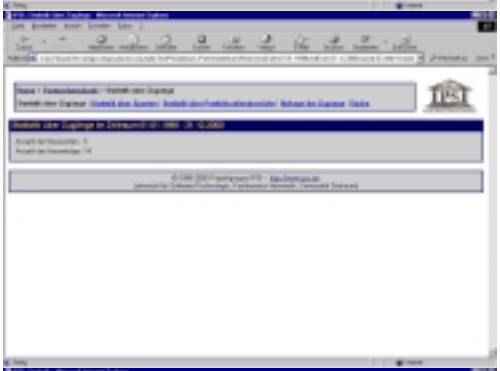


In das IPSI-Portal wurde eine Partnerdatenbank integriert. Zu den in dieser Datenbank enthaltenen Kunden und Vertragsinformationen kann sich der VAD eine Reihe von Statistiken ansehen oder Kunden und Vertrags-Detailinformationen ansehen. Hierzu zählen die Statistiken über Neukunden eines bestimmten Zeitraums, Bestandskunden nach Versicherungssparten oder Postleitzahlengebiete.

Statistiken über neue Kunden



Um sich einen Überblick über Neuzugänge bei den eigenen Kunden zu verschaffen, kann sich der VAD über die integrierte Partnerdatenbank Statistiken über neue Kunden ansehen. Hierzu gibt der VAD einen bestimmten Zeitraum an, für den er sich die Neuzugänge ansehen möchte und berechnet daraufhin die Statistik.

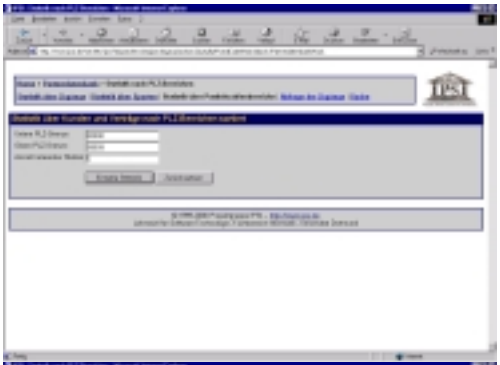


Die Statistik über Neuzugänge eines Zeitraums zeigt dem VAD zum Einen die Anzahl der neuen Kunden innerhalb des gewählten Zeitraums an, zum Anderen auch die Anzahl der Neuverträge (die auch zu in diesem Zeitraum bereits bestehenden Kunden gehören können), die abgeschlossen wurden.

Statistiken über Vertragsarten aller Kunden

Weiterhin kann sich der VAD über die Anzahl von Verträgen in allen Versicherungssparten informieren. Hierzu wird dem VAD eine Tabelle angezeigt. Jede Zeile entspricht einer Versicherungssparte. Zu jeder Sparte werden in den Spalten die Anzahl neuer Kunden und die Anzahl neuer Verträge angezeigt.

Statistiken zur regionalen Verteilung der Kunden

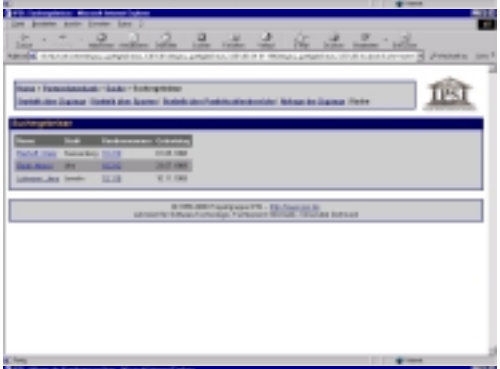


Als letzte Statistik kann sich der VAD die Verteilung von Versicherungen nach Postleitzahlen ansehen. Dazu legt er die Anzahl der relevanten Ziffern der Postleitzahlen fest. Dies geschieht durch die Definition eines Intervalls von Postleitzahlen (z.B. 59379 bis 60000). Der VAD muss die Postleitzahlen nicht mit allen fünf Stellen angeben, sondern kann eine Anzahl signifikanter Stellen spezifizieren.

Das Ergebnis der Abfrage wird dem Benutzer in Form einer Tabelle angzeigt. Jede Zeile entspricht einem Postleitzahlengebiet. Für jedes Postleitzahlengebiet ist die Anzahl der Neukunden und die Anzahl der neuen Verträge angegeben.

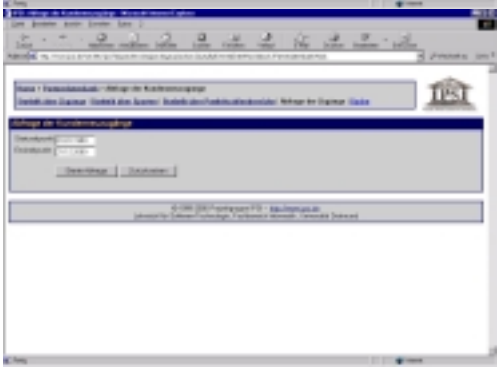
Suche nach Kunden

Der VAD kann nach einem bestimmten Kunden suchen und sich dessen Informationen anzeigen lassen. Hierzu gehören sämtliche Kunden- und Vertrags-Stammdaten, die nach harten und weichen Daten geordnet sind (s.u.). Suchkriterien umfassen u.a. die Kundennummer, Namen, Adressdaten, Bankverbindungen, Geburtstag, Vertragsdaten.

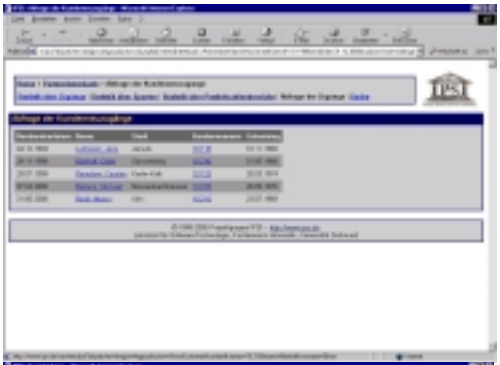


Alle Kunden des VAD, die zu den spezifizierten Suchmerkmalen passen, werden in einer Liste mit Namen, Adressdaten, Kundennummer und Gerburtsdatum angezeigt. Durch einen Klick auf den Namen des Kunden können dessen Kontaktdaten eingesehen werden. Durch einen Klick auf die Kundennummer können dessen Vertrags- und Kundendaten eingesehen werden.

Übersicht über neue Kunden



Alle neuen Kunden in einem bestimmten Zeitraum kann sich der VAD ebenfalls ansehen. Hierzu spezifiziert er einfach den gewünschten Zeitraum in einer Eingabemaske.

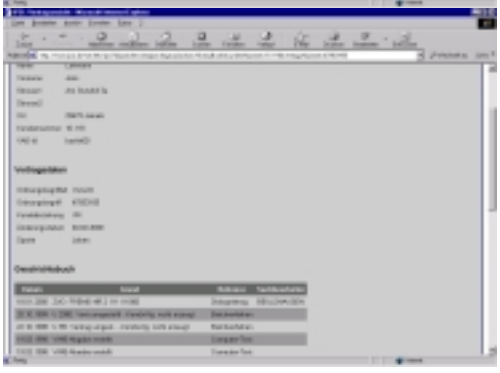


Die passenden Kunden werden in einer Liste angezeigt. Diese enthält zu den oben genannten Daten noch das Datum an dem die jeweilige Person Kunde wurde. Durch einen Klick auf den Namen des Kunden können dessen Kontaktdaten eingesehen werden. Durch einen Klick auf die Kundennummer können dessen Vertrags- und Kundendaten eingesehen werden.

Zu jedem Kunden kann sich der VAD die harten und weichen Daten (Vertrags- und Kundendaten) ansehen. Diese sind unterschiedlich farbig markiert. Weiche Daten sind z.B. Email und Telefon. Harte Daten sind alle vertragsrelevanten Daten wie das Geburtsdatum. Harte Daten werden rot markiert und weiche Daten grün. Durch einen Klick auf die Kundennummer werden die Vertragsdaten des Kunden angezeigt.

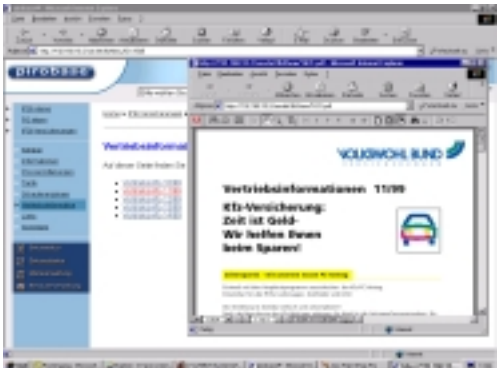
Ebenso kann sich der Benutzer evtl. die für diesen Kunden in Outlook 2000 vorhandenen Kontaktdaten direkt ansehen. Die Daten des Kontaktes werden in der bereits beschriebenen Form mit Namen, Vornamen, Anschrift, Email, Telefon, Notizen und benutzerdefinierten Feldern dargestellt. Der VAD hat direkt hier die Möglichkeit diese Daten zu ändern.

Eine wichtige Information stellen die Verträge dar, die dieser Kunde abgeschlossen hat. Diese werden mit der jeweiligen Sparte, dem Versicherungsunternehmen, dem Datum des Abschlusses und der Versicherungsnummer angezeigt. Durch einen Klick auf die Versicherungsnummer kann sich der VAD die Historie (das Geschichtsbuch) zu jedem Vertrag anzeigen lassen.



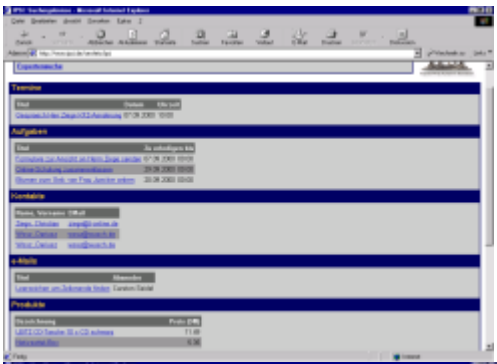
Zu jedem Vertrag wird ein Geschichtsbuch aus der Partnerdatenbank angezeigt. Hierin sind mit Datumsvermerk alle Aktionen, die den jeweiligen Vertrag betreffen in Form einer Liste dargestellt. Vermerkt sind neben dem Datum der Grund der Aktion (z.B. Änderung des Tarifs) und der durchführende Sachbearbeiter.

Einsicht von Informationen der Agentur oder des VU



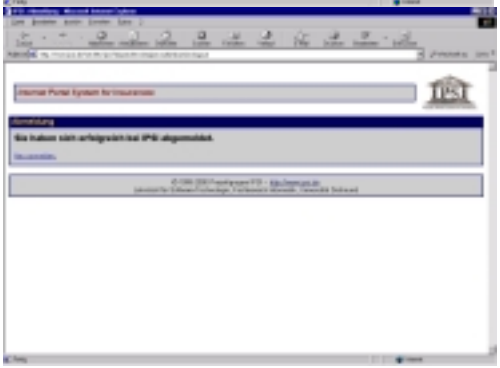
Um die Informationsdistribution zwischen dem VU und den VADs zu vereinfachen, kann der VAD Dokumente im Informationssystem abrufen, die dort vom VU nach bestimmten Kategorien eingestellt wurden. Das Informationssystem wird über den gleichnamigen Menüpunkt von der Startseite des Portals aus erreicht. Im Bereich des Informationssystems kann der VAD anhand von Kategorien zu entsprechenden Informationsseiten navigieren, in denen Dokumente hinterlegt sind.
Durch einen Klick auf ein solches Dokument wird das entsprechende Programm (bei PDF-Dokumenten z.B. der Acrobat Reader) gestartet. Der VAD kann somit alle Dokumente, die vom VU zentral hinterlegt wurden einfach dezentral abfragen. Die Struktur des Informationssystems ist nicht vorgegeben und kann von jedem VU individuell festgelegt werden.

Integrierte Suche



Um abgelegte Informationen jeglicher Art wieder schnell aufzufinden, stellt das IPSI-Portal eine Suche zur Verfügung, die gemäß der angegebenen Suchkriterien über alle gewählten Bereiche hinweg sucht.
Zum Einen gibt es die Quick-Suche, die eine Eingabe eines Stichwortes sowie die Auswahl der zu durchsuchenden Systeme erlaubt.
Zum Anderen gibt es die Expertensuche, die eine Suche nach mehr Kriterien erlaubt (z.B. Suche aller Nachrichten, die ab einem bestimmten Datum eingetroffen sind).
Die Suchergebnisse werden in einem Bildschirm geordnet nach den entsprechenden Bereichen, in denen die Ergebnisse gefunden wurden.
Ein Klick auf das entsprechende Ergebnis führt in den jeweiligen Bereich und zeigt das Informationsobjekt.

Abmeldung



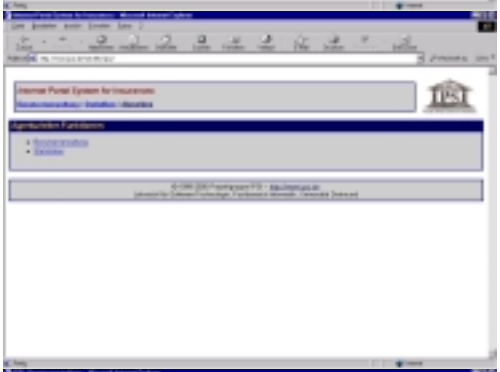
Wenn der VAD alle Arbeitsgänge mit dem System beendet hat, sollte er sich über den Menüpunkt „Abmelden“ vom System abmelden. Es erscheint dann der in der linken Abbildung dargestellte Bildschirm, der den VAD darauf hinweist, dass er sich abgemeldet hat. Hier hat der VAD die Möglichkeit sich erneut anzumelden. Dies ist z.B. sinnvoll, wenn sich der VAD aus Versehen abgemeldet hat oder sich unter einem anderen Login erneut anmelden will (z.B. als Agenturleiter).

3.2 Arbeitssabläufe eines AL/VU

Login

Um die Arbeitsabläufe mit Hilfe des IPSI-Portals durchzuführen, muss sich der AL zunächst beim IPSI-Portal anmelden. Dies geschieht über den Login-Bildschirm durch Eingabe von Benutzername und Passwort. Anhand des Logins erkennt das IPSI-Portal, dass der Benutzer in der Rolle eines AL bzw. VU agiert und präsentiert einen entsprechend anderen Startbildschirm als bei der Anmeldung eines VAD.

Startbildschirm



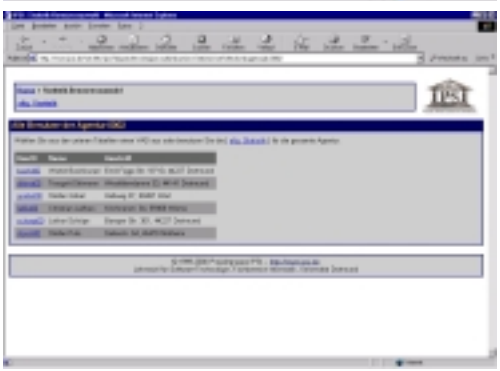
Nach der Anmeldung befindet sich der Benutzer auf dem Startbildschirm und kann von dort aus alle der im Folgenden beschriebenen Arbeitsabläufe ausführen. Diese Abläufe können im Bereich der Benutzerverwaltung oder der Statistikabfrage liegen. Hierfür gibt es zwei Links, die den o.g. Abläufen entsprechend benannt sind. Außerdem kann sich der Benutzer über den Link „Abmelden“ vom System abmelden.

Benutzerverwaltung

In der Benutzerverwaltung kann ein AL die Daten für die IPSI-Portal-Benutzer festlegen.
Alle Benutzer der Agentur werden in einer Liste mit ihrem Login („UserID“), ihrem Namen und ihrer Anschrift dargestellt. Durch einen Klick auf die UserID können die Nutzerdaten eingesehen werden. Durch einen Klick auf die Funktionslinks hinter jedem Nutzereintrag können dessen Daten geändert oder gelöscht werden.

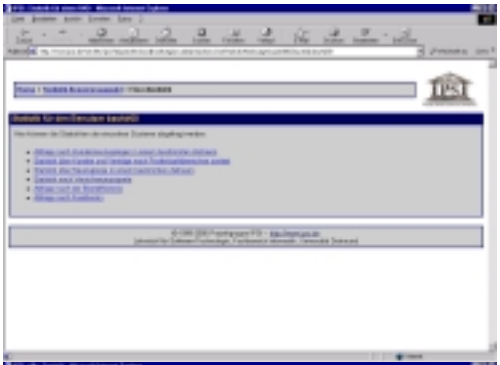
Durch Klick auf einen Benutzereintrag werden dessen Detaildaten dargestellt, die dann auch geändert werden können. Hierzu zählen sowohl die Stammdaten wie Name, Anschrift, Email, Telefon usw. als auch die systembezogenen Daten wie Logins, Passwörter und das Budgetlimit für Einkäufe des VAD im Shop. Für jedes in das Portal integrierte System (Shop, Office, CMS) ist dabei eine Login/Passwort-Kombination nötig. Hierüber wird das Single-sign-on für den VAD ermöglicht.

Statistiken



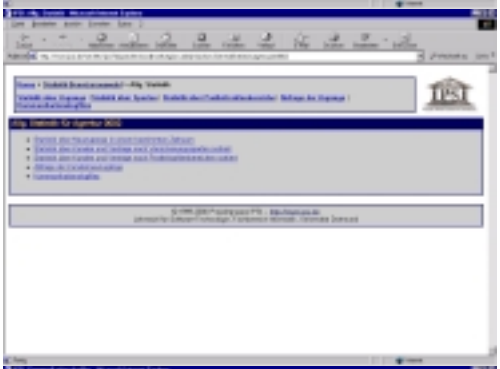
Der AL kann Statistiken sowohl über einzelne VADs ansehen als auch bezogen auf die gesamte Agentur. Für die VAD-bezogenen Statistiken werden alle Benutzer (VADs) der Agentur (bzw. des VU) in einer Liste mit Login, Name und Anschrift angezeigt. Durch einen Klick auf das Login eines VAD werden die VAD-bezogenen Statistiken für diesen VAD angezeigt.

VAD-bezogene Statistiken



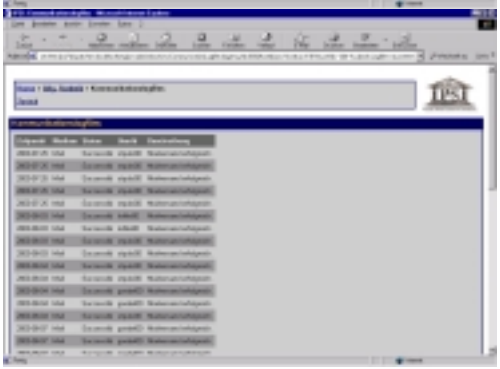
Die VAD-bezogenen Statistiken sind bereits in den Beschreibungen zur Partnerdatenbank im VAD-Bereich beschrieben und werden daher an dieser Stelle nicht erneut dargestellt.

Agenturbezogene Statistiken



Die Agenturbezogenen Statistiken sind den VAD-bezogenen Statistiken ähnlich. Der einzige Unterschied ist, dass hierbei alle Statistiken über alle VADs berechnet werden, die zu der jeweiligen Agentur des ALs gehören. Da dieser Unterschied keine wesentlichen Auswirkungen auf die Darstellungsweise der Statistiken hat, werden die Statistiken an dieser Stelle nicht erneut dargestellt.

Logfiles auswerten



Der AL kann sich Einträge über den Erfolg der Versendung von Nachrichten in einem Logfile ansehen. Hierzu wird eine Liste von Einträgen dargestellt, die einen Zeitpunkt, einen Versandkanal (Mail, SMS oder Fax), den Versender (anhand seines Logins) und den Status der Versendung (erfolgreich oder eine entsprechende Fehlermeldung) anzeigt. Diese Informationen sind technischer Art und können zu Systemadministrationszwecken genutzt werden.

4 Prozessleitfaden

Die Softwaretechnik ist eine Forschungsdisziplin (in der Industrie und in den Hochschulen), die sich mit dem Einsatz, aber auch der Entwicklung von Techniken (Methoden, Verfahren/Prozesse und Hilfsmitteln) für die Softwareentwicklung befasst. Die Techniken werden bei der Entwicklung von Softwaresystemen in einem Softwareentwicklungsprojekt eingesetzt. Ein Softwareprojekt umfasst den organisatorischen und zeitlichen Rahmen (d.h. das Zusammenwirken von Personal, Werkzeugen, Maschinen, etc.), der einer Entwicklung eines Softwaresystems auf Basis eines Softwareentwicklungsprozesses zugrunde liegt. *„[Ein Softwareprojekt ist ein] Vorhaben, das im wesentlichen durch Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist, wie z.B. Zielvorgabe; zeitliche, finanzielle, personelle und andere Bedingungen, projektspezifische Organisation ...“* [Sc91]. Der Softwareentwicklungsprozess legt die Art und die Reihenfolge der durchzuführenden Aktivitäten sowie die Hilfsmittel für die Durchführung der Aktivitäten, aber nicht die beteiligten Personen bei der Durchführung der Aktivitäten fest. Hilfsmittel bei der Durchführung der Aktivitäten können z.B. Softwarewerkzeuge, aber auch Formulare, Interviews, Moderationstechniken, etc. sein. *„Eine Prozessarchitektur legt den allgemeinen Rahmen für die Spezifikation von Softwareentwicklungsprozessen fest. Sie besteht aus einer Standardmenge von fundamentalen Prozessschritten. Regeln bestimmen, wie Prozesse beschrieben und in Beziehung gesetzt werden. Ein Prozess beschreibt Aktivitäten, Methoden und Verfahren, die zur Entwicklung und Überprüfung von Software benötigt werden.“* [Ba98] Das Ergebnis eines Softwareentwicklungsprozesses ist ein Softwaresystem, welches eine Softwaredokumentation (jede Art der Beschreibung, die im Laufe des Softwareentwicklungsprozesses für ein Softwaresystem erstellt wird) und das eigentliche Softwareprogramm (der Objektcode als ausführbare Bitfolge) umfasst.

Durch den Einsatz von Methoden, (*„Methoden sind planmäßig, begründete Vorgehensweisen zur Durchführung von Tätigkeiten mit Hilfe von Beschreibungsmitteln zur Erreichung von [einem festgelegtem Ziel]“* [Ba82])

Verfahren/Prozessen, Prinzipien („*Prinzipien sind Grundsätze, die man seinem Handeln zugrunde legt*“ [Ba82]) und Hilfsmitteln soll die Komplexität bei der Entwicklung von Softwaresystemen reduziert und der Entwicklungsprozess effizienter und produktiver gestaltet werden. Hierdurch soll die Entwicklungsdauer reduziert und die Entwicklungskosten gesenkt werden, bei gleichzeitiger Verbesserung und Steigerung der Qualität.

Die Softwareentwicklungsprozesse haben innerhalb der Softwareentwicklung eine große Bedeutung, denn durch sie können die drei Faktoren Zeit, Kosten, Qualität erheblich beeinflusst werden. Voraussetzung für eine positive Beeinflussung dieser drei Faktoren ist jedoch einerseits eine Beschreibung des Softwareentwicklungsprozesses und andererseits eine automatische Unterstützung dessen. Die Beschreibung von Softwareentwicklungsprozessen wird als Prozessmodellierung bezeichnet, während die automatische Unterstützung von Prozessen allgemein (hierzu zählen auch die Softwareentwicklungsprozesse) durch Softwaresysteme erfolgt. Zu diesen Softwaresystemen zählen CSCW-Systeme (Computer Supported Cooperative Work), insbesondere Workflowmanagementsysteme und Groupwaresysteme, aber auch alle Softwaresysteme, die dieses Ziel verfolgen. Neben der Erhöhung der Qualität des zu entwickelnden Softwaresystems in kürzerer Zeit und bei geringeren Kosten sollen durch die Softwareprozessmodellierung folgende Ziele erreicht werden [CuKe92], [RoMe95]:

- Kommunikations- und Diskussionsgrundlage über Softwareentwicklungsprozesse,
- Wiederverwendbarkeit von Softwareprozessmodellen,
- Weiterentwicklung, Verbesserung und Anpassung von Softwareprozessmodellen,
- Unterstützung des Prozessmanagements,
- Unterstützung des Projektmanagements.

Eine formale Beschreibung eines Softwareentwicklungsprozesses erleichtert zwar eine automatische Unterstützung, ist aber nicht zwangsläufig erforderlich, um einen positiven Effekt für die Softwareentwicklung zu erzielen. Ein gemeinsames Verständnis über den Softwareentwicklungsprozess lässt sich bei allen beteiligten Personen schon durch eine strukturierte und weitestgehend umfassende Beschreibung erreichen. Das Wissen über bewährte Vorgehensweisen wurde und wird oftmals schon in unternehmensinternen Dokumenten und Entwicklungsrichtlinien beschrieben. So schreibt z.B. die ISO 9000 (Teil 1-3) nur die Inhalte der Beschreibung von Vorgehensweisen und Entwicklungsrichtlinien vor, jedoch nicht die Notation [Th97], [Th98]. Allerdings birgt eine Beschreibung in einer natürlichen Sprache die Gefahr in sich, dass es zu Fehl- und Missinterpretationen kommt. Dies liegt in der Tatsache begründet, dass diese Art der Beschreibung in der Regel einen enormen Umfang hat und dass durch die Verwendung der natürlichen Sprache bestimmte Sachverhalte, Abhängigkeiten und Voraussetzungen nicht immer präzise ausgedrückt werden oder werden können.

Der Softwareentwicklungsprozess der Projektgruppe wird definiert durch die Erstellung eines Prozessleitfadens. Ein Prozessleitfaden stellt alle Aktivitäten in einer bestimmten Reihenfolge, die benötigten Werkzeuge und die erstellten (Zwischen-) Produkte dar, die zur Erreichung des Zweckes des Prozesses benötigt werden.

Ein Prozessleitfaden ist somit konkreter als ein Prozessmodell, indem er auf ein spezielles Projekt zugeschnitten (sog. „Tailoring“) ist. Ein Prozessleitfaden kann jedoch auf einem Prozessmodell basieren.

Ein Prozess dagegen ist eine Ausführung (in objektorientierter Denkweise eine Instanziierung) eines Prozessleitfadens. Es handelt sich also um die tatsächliche Durchführung der Tätigkeiten, die im Prozessleitfaden beschrieben sind.

Der Prozessleitfaden der Projektgruppe kann durch folgende Eigenschaften charakterisiert werden:

- anwendungsfallgetrieben
  Die Nutzerforderungen nehmen beim Vorgehen eine zentrale Bedeutung ein. Sie bestimmen den Zweck des zu erstellenden Systems. Diese Bedeutung wird im Prozessleitfaden dadurch deutlich, dass die Anforderungen – in Form von Anwendungsfällen - Eingang in viele Aktivitäten finden.
- architekturzentriert
  Die Architektur des zu entwickelnden Portals wurde schon vor Beginn der Projektgruppe im Groben vorgegeben. Diese Architektur wurde im Laufe der Zeit immer weiter verfeinert, wobei unterschiedliche Techniken, wie z.B. Prototyping, eingesetzt wurden.
- prototypbasiert
  Zur Verifizierung der Architektur und der Entscheidungen über den Einsatz bestimmter Techniken ist es empfehlenswert, in verschiedenen Phasen (Analyse, Realisierung) zielgerichtete Prototypen (Durchstichprototypen, Middlewareprototypen) zu erstellen. Damit verfolgte die PG das Ziel, Risiken der Integration früh dadurch zu eliminieren, dass Vermutungen über die Fähigkeiten von Techniken schnell durch Wissen ersetzt wurden.
- inkrementell
  Ein inkrementeller Prozessleitfaden sieht die Aufteilung des zu entwickelnden Systems in Teile vor, die möglichst unabhängig und eventuell sogar parallel entwickelt werden können. Für eine Projektgruppe ist dieser Aspekt unerlässlich, da von Anfang an eine konstante Anzahl von 12 Personen zur Verfügung stand, die genutzt werden sollte. Hierzu ist ein unabhängiges, in weiten Teilen paralleles Vorgehen so weit wie möglich durchzuführen. Eine inkrementelle Vorgehensweise ermöglicht genau dies. Diese Eigenschaft hängt direkt mit der Architekturzentriertheit zusammen, da Inkremente in der Regel Subsysteme des Gesamtsystems sind.
- iterativ
  Ein iterativer Prozessleitfaden sieht vor, dass das Softwaresystem evolutionär erstellt wird. Dies bedeutet, dass die Phasen von der Anforderungsanalyse über die Implementierung bis zum Test mehrfach durchlaufen werden und das System (oder das Inkrement) bei jedem Durchlauf erweitert wird. Hierzu eignet sich ganz besonders der Einsatz von Anwendungsfällen; diese werden priorisiert, und die Anwendungsfälle mit der höchsten Priorität werden in der ersten Iteration bearbeitet. Danach folgen die Anwendungsfälle mit der nächst niedrigeren Priorität usw.

Es werden bei der Projektgruppe die folgenden Kernprozesse unterschieden:

- PG-Entwicklung,
- Systementwicklung,
- Qualitätsmanagement,
- Projektplanung.

Während das Qualitätsmanagement und die Projektplanung originär eng mit der Systementwicklung verbunden sind, spielt die PG-Entwicklung eine übergeordnete Rolle. Das Ziel der Projektgruppe ist die Entwicklung eines Softwaresystems. Die Tätigkeiten zur Verfolgung dieses Ziels werden in der Systementwicklung beschrieben. Die PG-Entwicklung beschreibt die Rahmenbedingungen, die einerseits zur Initiierung einer Projektgruppe führen, aber auch andererseits den organisatorischen Ablauf zur Durchführung einer Projektgruppe, damit diese das Ziel strukturiert und sinnvoll verfolgen kann.

Nachfolgend werden die vier Kernprozesse detaillierter erläutert. Die Darstellung der Prozesse folgt einer Notation, die in

Anhang A – Leseanleitung zu den Prozessmodellen genauer beschrieben ist.

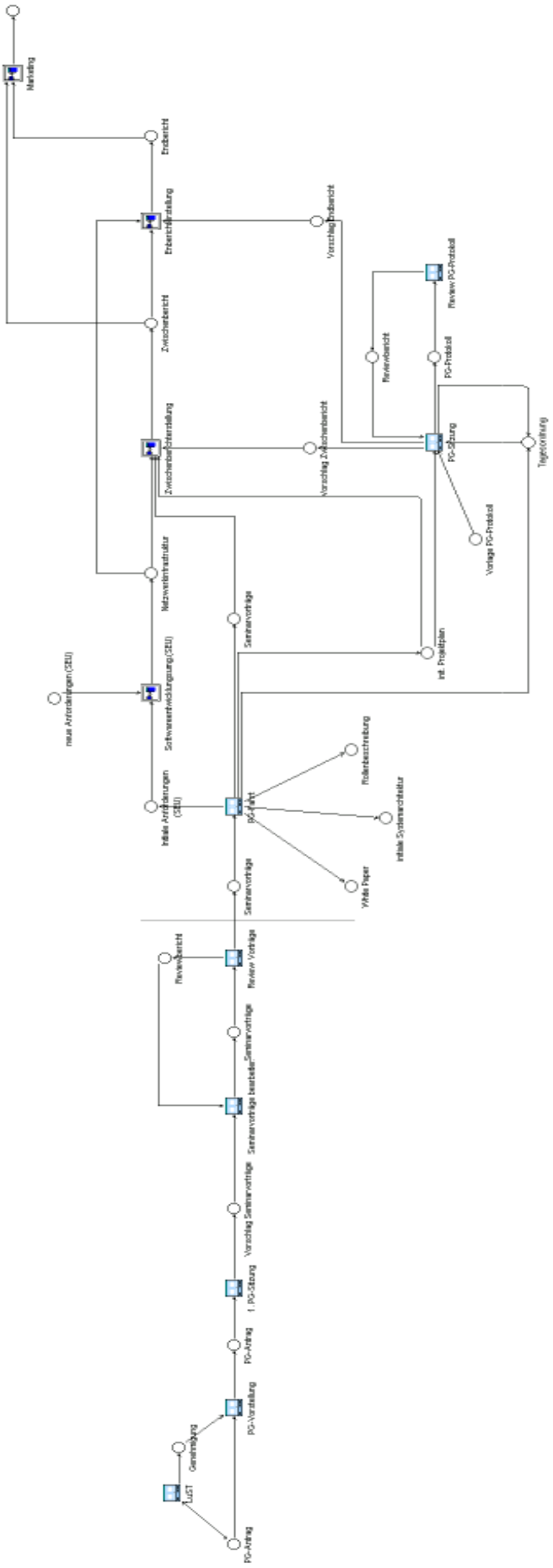
4.1 PG-Entwicklung

Ausgehend von einem Antrag auf die Durchführung einer Projektgruppe im Hauptstudium Informatik, der von der Kommission „Lehre und Studium“ (LuSt) genehmigt werden muss, findet eine erste Präsentation der Ziele und Themen der Projektgruppe vor einer Menge von interessierten Studierenden statt. Im Anschluss daran, wenn sich mindestens acht Personen bereiterklärt haben an der PG teilzunehmen, findet die erste Projektgruppensitzung statt. Ziel dieser ersten Projektgruppensitzung ist die Konsolidierung eines Projektgruppenteams und die Festlegung von Seminarvorträgen. Als Ergebnis einer Diskussion im Anschluss an die Seminarvorträge (die in der Regel während einer Projektgruppenfahrt gehalten werden) wird die initiale Systemarchitektur des zu entwickelnden Softwaresystems (bei dieser Projektgruppe die Architektur des Portals) entworfen. In einer initialen Anforderungsliste (Whitepaper) werden neben ersten Anforderungen auch ein initialer Projektplan aufgestellt und die technischen Rahmenbedingungen beschrieben. Die ersten technischen Rahmenbedingungen führen zum

Aufbau einer Softwareentwicklungsumgebung bestehend aus Hardware- und Softwarekomponenten. Der initale Projektplan ist der Ausgangspunkt für die regelmäßige Projektgruppensitzung. In den Projektgruppensitzungen werden neben Entscheidungen, die die Systementwicklung betreffen, auch Vorschläge für Struktur und Inhalt des Zwischen- und Endberichts der Projektgruppe erarbeitet. Sowohl der Zwischenbericht als auch der Endbericht wird in dieser PG zu Marketingzwecken verwendet, um Präsentationsveranstaltungen zur Vorstellung der Ergebnisse der Projektgruppen vorzubereiten.

Der oben beschriebene Sachverhalt ist in der neben stehenden Abbildung grafisch dargestellt.

In den weiteren Unterabschnitten werden die Verfeinerungstätigkeiten genauer beschrieben. Auf eine Abbildung der graphischen Modelle wurde verzichtet, da die Prozessteile wenig Komplexität aufweisen.



4.1.1 Softwareentwicklungsumgebung

Der Aufbau und die Pflege der Softwareentwicklungsumgebung ist Teil einer Systementwicklung. Aus der Systementwicklung heraus ergeben sich im Laufe der Zeit neue Anforderungen, die eine Modifikation der Softwareentwicklungsumgebung erforderlich machen.

4.1.2 Zwischen-/Endberichterstellung

Im Zwischenbericht und im Endbericht werden die bisher erarbeiteten Ergebnisse zusammengefasst und in einer strukturierten Form aufgeschrieben.

4.1.3 Marketing

Innerhalb des Marketings werden die notwendigen Voraussetzungen geschaffen, um die Ergebnisse der Projektgruppe entweder in Form eines Prototyps oder in Form einer Systemvorführung einem interessierten Publikum vorzustellen.

4.2 Systementwicklung

Die Tätigkeit Anforderungsanalyse wird initiiert durch eine initiale Systemarchitektur und einer initialen Anforderungsliste (Whitepaper). Diese beiden Dokumente sind das Ergebnis des Prozesses PG-Entwicklung. Das Ziel der Anforderungsanalyse ist die Konkretisierung der initialen Anforderungen. Das Ergebnis der Anforderungsanalyse sind konsolidierte Anforderungen für das Portal (es wird beschrieben, „was“ entwickelt werden soll). Aus den Anforderungen an das Portal werden Anwendungsfälle abgeleitet. Durch Anwendungsfälle wird das Verhalten des Portals beschrieben. Die Architektur des Portals wird durch die initiale Systemarchitekur grob vorgegeben. Bestandteile dieser Architektur sind Subsysteme. Die in der Anforderungsanalyse konsolidierten Anforderungen sind nach den Subsystemen strukturiert.

Die Entwicklung eines Benutzungsoberflächenprototyps dient einerseits zu Marketingzwecken und anderseits zur Verifikation der Anforderungen.

Auf Basis der Anforderungen für Subsysteme muss entschieden werden, ob existierende Softwaresysteme den Anforderungen genügen oder ob eine Eigenentwicklung erforderlich ist. Nach der Entscheidung, existierende Softwaresysteme für Subsysteme einzusetzen, wurden nach einer Marktanalyse, bei der nichtfunktionale Kriterien wie z.B. Preis, Verfügbarkeit, Support oder Plattform, eine Rolle spielten, existierende Softwaresysteme, z.B. MS Outlook 2000, SmartStore und Pirobase ausgewählt. Für diese Softwaresysteme musste nun eruiert werden, ob sie über eine Programmierschnittstelle (API) verfügen oder ob eine Schnittstelle entwickelt werden kann. Dies erfolgt durch die Entwicklung von Prototypen auf der Basis der Anwendungsfälle und der ermittelten Anforderungen.

Die initiale Systemarchitektur wird im Grobentwurf konkretisiert, in dem festgelegt wird, welche Subsysteme durch eine Eigenentwicklung realisiert werden und für welche Subsysteme externe Softwaresysteme eingesetzt werden. Eine Entscheidungshilfe hierfür liefert die Prototypentwicklung.

Parallel zum Grobentwurf erfolgt der Entwurf der graphischen Benutzungsoberfläche (GUI) für das Portal. Wiederum werden die Anwendungsfälle verwendet. Dieser GUI-Entwurf wird beim Feinentwurf und der Realisierung verwendet.

Der Grobentwurf wird durch einen Feinentwurf konkretisiert, um eine Basis für die Realisierung zu liefern. Der GUI-Entwurf, aber auch die Anwendungsfälle und die grobe Systemarchitektur (unterteilt nach eigenentwickelten Subsystemen und existierenden Softwaresystemen) sind die Voraussetzungen für den Entwurf einer detaillierten Systemarchitektur.

In der Realisierung wird der Entwurf, genauer die konkrete Systemarchitektur mittels einer Programmiersprache umgesetzt. Bei dieser Umsetzung werden elementare Bestandteile der Systemarchitektur (Controller, Adaptor und Formatter) inkrementell umgesetzt. Diese werden als Komponenten bezeichnet und einem Test unterzogen. Die getesteten Komponenten werden nach der Realisierung in der Integrationsphase zum Portal integriert und einem Systemtest unterzogen, bevor das fertige Softwaresystem ausgeliefert wird.

Bei der Auslieferung wird das erstellte Softwaresystem mit dem Benutzertutorial ausgeliefert.

Die bis hierhier textuell beschriebene Vorgehensweise ist in der nachfolgenden Abbildung grafisch dargestellt.

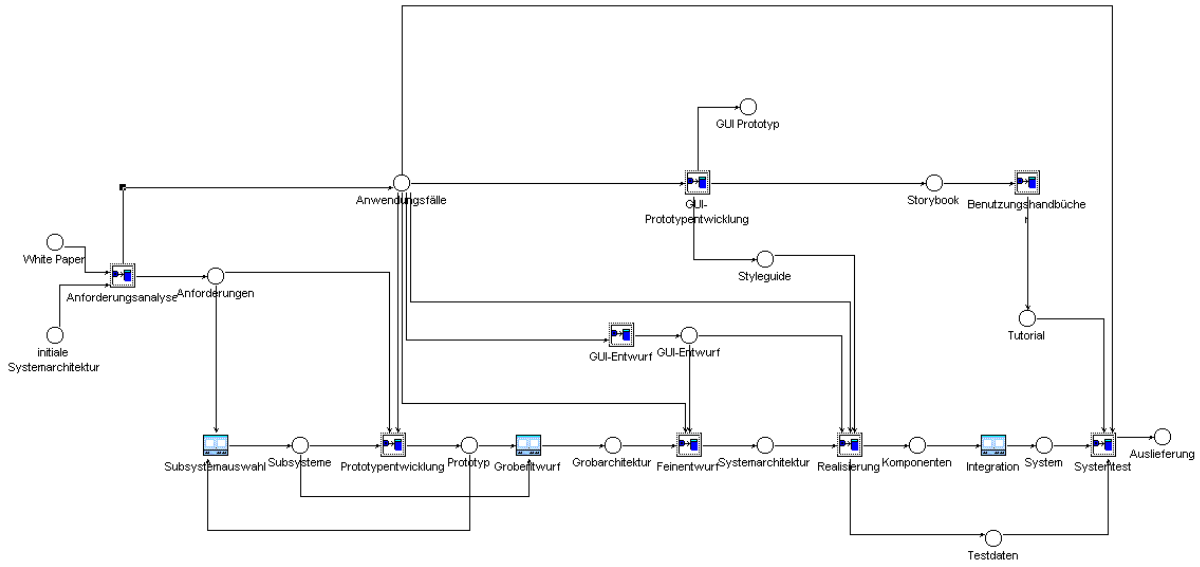


Abbildung 4 - Prozessmodell Systementwicklung

4.2.1 Anforderungsanalyse

Die initiale Systemarchitektur und die initiale Anforderungsliste (Whitepaper) sind der Ausgangspunkt zur Erarbeitung eines Anforderungskatalogs für das gesamte zu entwickelnde Softwaresystem. Dieser Anforderungskatalog wird auf verschiedene Art und Weise auf Widerspruchsfreiheit, Vollständigkeit und Redundanz geprüft. Hierzu werden in Form von Interviews Anwender und Betreiber befragt. Anwender sind die Personen und Personengruppen, die das Portal letztendlich benutzen sollen, während Betreiber die Personen und Personengruppen sind, die das Portal den Anwendern zur Verfügung stellt. Im Kontext der Projektgruppe sind hiermit Versicherungsaußendienstmitarbeiter (VAD) und Versicherungsunternehmen (VU) gemeint. Sowohl Anwender als auch Betreiber haben unterschiedliche, unter Umständen auch konkurrierende Anforderungen. Einige Versicherungsunternehmen setzen schon Unterstützungssysteme für den Versicherungsaußendienst ein. Diese Systeme wurden genauer betrachtet, um weitere Anforderungen zu ermitteln. Nach der Konsolidierung der einzeln ermittelten Anforderungen wird bei Bedarf der Anforderungskatalog korrigiert, evtl. erweitert und eine Überprüfung der Anforderungen erneut vorgenommen. Die Architektur des Portals wird durch die initiale Systemarchitekur grob vorgegeben. Für die dort identifizierten einzelnen Subsysteme werden die Anforderungen, bestehend aus funktionalen und nichtfunktionalen Anforderungen, beschrieben. Zusätzlich werden die Anforderungen an die Schnittstellen beschrieben, d.h. welche Informationen werden zwischen den Subsystemen voraussichtlich ausgetauscht. Die letztendlich abschließend konsolidierten Anforderungen werden dann zur weiteren Verwendung zur Verfügung gestellt.

Dieses Vorgehen stellt die nachfolgende Abbildung dar.

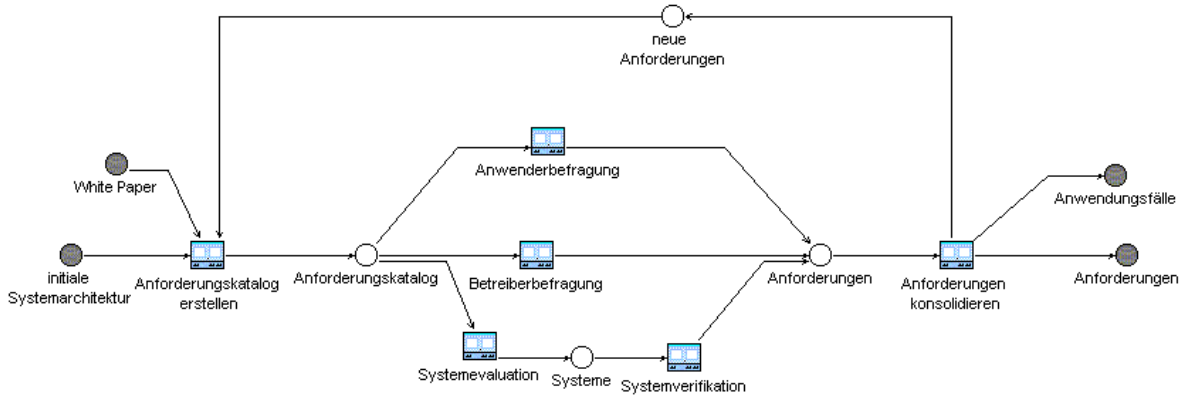


Abbildung 5 - Prozessmodell Anforderungsanalyse

4.2.2 Prototypentwicklung

Das Ziel der Prototypentwicklung ist die Ermittlung der Möglichkeiten zur Integration der Softwaresysteme untereinander. Diese Ermittlung erfolgt durch die Realisierung von Prototypen für jedes ausgewählte Softwaresystem. Für jedes Softwaresystem resp. jeden Prototyp werden Anforderungen, sog. Keyfeatures, festgelegt, die durch den Prototyp realisiert werden sollen. Diese Anforderungen an die Prototypen werden aus den Anwendungsfällen und den Anforderungen aus der Anforderungsanalyse erstellt. Auf der Basis der erarbeiteten Subsysteme werden zunächst existierende Softwaresysteme evaluiert, die eingesetzt werden können. Die Prototyprealisierung erfolgt durch Realisierung von Funktionen für die Softwaresysteme, die die aufgestellten Anforderungen (Keyfeatures) erfüllen. Durch die Prototypen soll nachgewiesen werden, ob über eine Programmierschnittstelle auf Funktionen des zugrundeliegenden Softwaresystems zugegriffen werden kann.

Das nachfolgend abgebildete Modell skizziert diese Vorgehensweise.

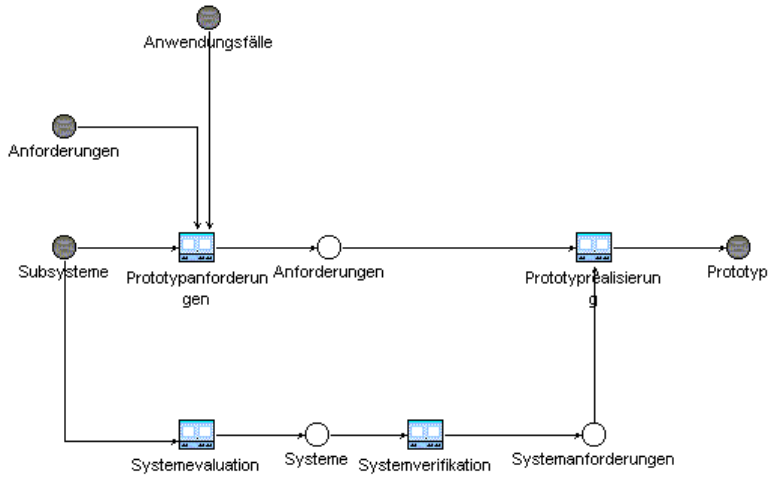


Abbildung 6 - Prozessmodell Prototypenentwicklung

4.2.3 GUI-Prototypentwicklung

Ausgehend von den Anwendungsfällen wird in der GUI-Prototypentwicklung zunächst ein Storybook erstellt. Dieses Storybook wird in der GUI-Prototypentwicklung zur Erstellung eines Styleguides und zusammen mit diesem zur Realisierung des Prototyps verwendet.

In der folgenden Abbildung ist die GUI-Prototypenentwicklung in einem einfachen Prozessmodell beschrieben.

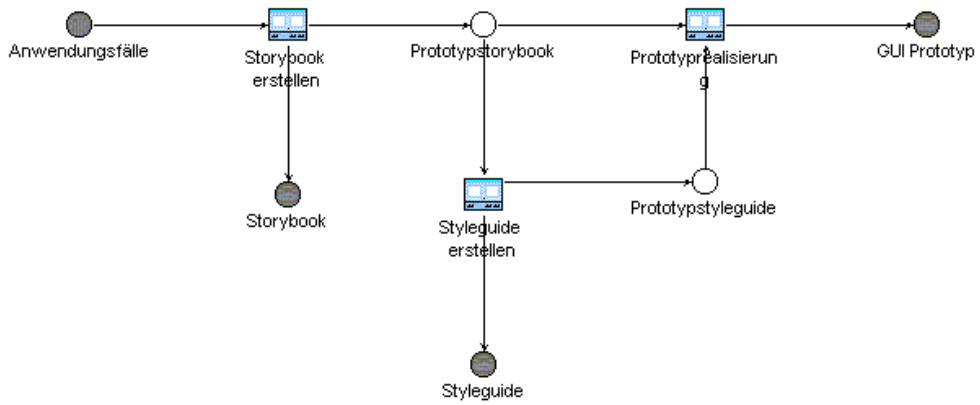


Abbildung 7 - Prozessmodell GUI-Prototypenentwicklung

4.2.4 GUI-Entwurf

Ziel des GUI-Entwurfs ist die Beschreibung der Veränderungen der Benutzungsoberfläche nach dem Eintreten eines Ereignisses. Je nach Art des Ereignisses bzw. der Interaktion, die von einem Benutzer oder vom Portal selbst ausgelöst werden kann, findet an der Benutzungsoberfläche eine Zustandsänderung statt. Sowohl die Interaktionen als auch die Zustandsänderungen werden beschrieben und ergeben den GUI-Entwurf.

Das folgende Modell zeigt die Vorgehensweise beim Entwurf der Benutzeroberfläche.

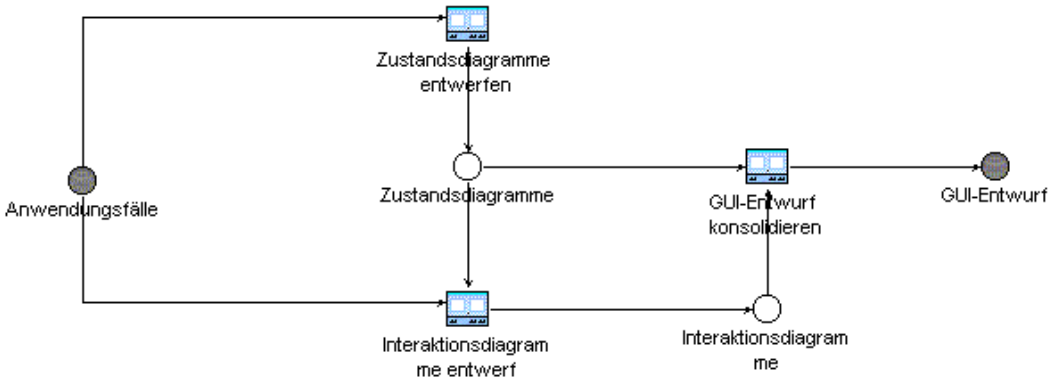


Abbildung 8 - Prozessmodell GUI-Entwurf

4.2.5 Feinentwurf

Durch die grobe Systemarchitektur wird beschrieben, für welche Subsysteme existierenden Software verwendet wird und welche Subsysteme durch eine Eigenentwicklung realisiert werden. Der Feinentwurf umfasst daher den Entwurf von Adaptoren und Controllern. Adaptoren sind Schnittstellen für existierende Softwaresysteme. Sie ermöglichen den Zugriff auf Funktionen dieser Systeme. Daneben wird durch Controller der Workflow, resultierend aus den Anwendungsfällen, beschrieben. Für jeden Controller werden ein oder mehrere Formatter benötigt, die die Eingabe und Darstellung von Informationen an der Benutzungsoberfläche ermöglichen. Der Entwurf erfolgt mittels UML [Oe97] durch die Beschreibung von Klassen. Durch die Identifizierung der Fassadenklassen und die Komposition von Klassen zu Paketen wird die letztendliche Systemarchitektur entworfen.

Der Feinentwurf ist in der nachfolgenden Abbildung dargestellt.

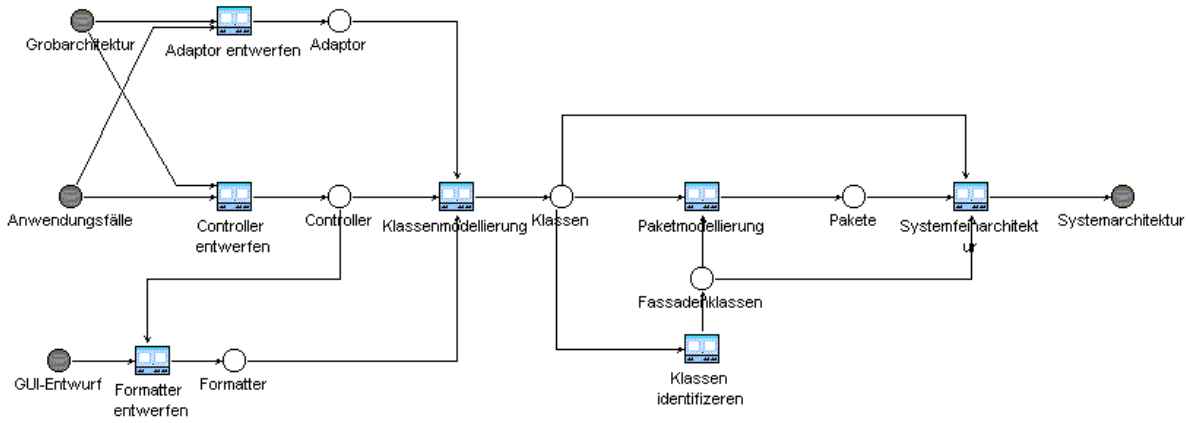


Abbildung 9 - Prozessmodell Feinentwurf

4.2.6 Realisierung

Die Implementierungsphase der entworfenen Systemarchitektur, i.d.S. der Controller, Adaptoren und Formatter erfolgt in der Realisierung. Bei der Realisierung der Formatter muss der vorgegebene Styleguide beachtet werden. Eine weitere notwendige Voraussetzung für die Realisierung der Formatter ist die Realisierung einer GUI-Library. Durch die GUI-Library werden die erforderlichen elementaren Funktionen für die Ein- und Ausgabe von Informationen bereitgestellt, die bei der Realisierung der Formatter verwendet werden können. Durch einen Middlewareprototyp wird vor Beginn der Realisierung der Controller und Adaptoren eruriert, welche Middlewaretechnik (CORBA, RMI) zur Kommunikation eingesetzt werden kann. Alle realisierten Komponenten durchlaufen abschließend einen Komponententest. Auf der Grundlage der Anwendungsfälle werden Testdaten erstellt, die zum Testen der Komponenten dienen.

Das nachfolgende Bild stellt die Vorgehensweise in der Implementierungsphase vor.

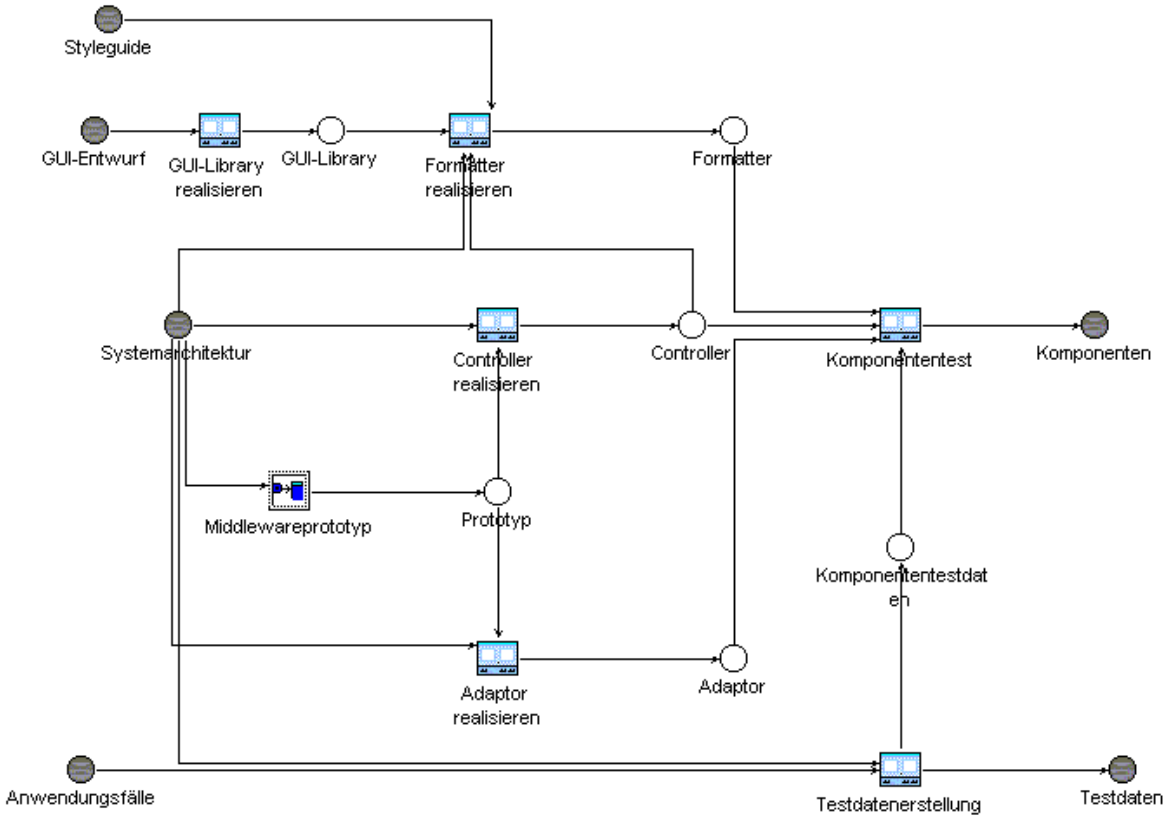


Abbildung 10 - Prozessmodell Realisierung

4.2.6.1 Middlewareprototyp

Um entscheiden zu können, welche Middlewaretechnik eingesetzt werden kann, werden zunächst die Anforderungen an die Middlewaretechnik aufgestellt. Hierbei werden nur Anforderungen erfasst, die für den speziellen Anwendungskontext auch relevant sind. Auf dieser Basis werden Systeme ausgewählt und verifiziert. Für diese Systeme werden Prototypen zum Nachweis der Machbarkeit der aufgestellten Anforderungen realisiert. Ein realisierter Prototyp wird zur weiteren Verwendung und Verbesserung zur Verfügung gestellt.

Das hier textuell beschriebene Vorgehen verdeutlicht die nachfolgende Abbildung in grafischer Form.

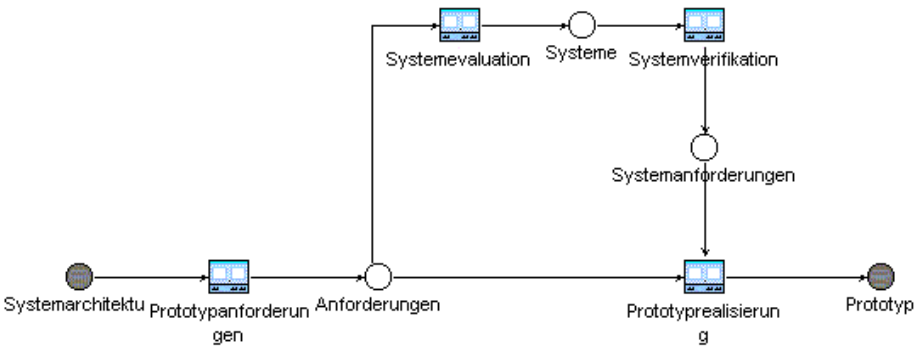


Abbildung 11 - Prozessmodell Middlewareprototyp

4.2.7 Benutzungshandbücher

Bei der Erstellung von Benutzungshandbüchern wird zwischen Benutzungstutorials und Referenzhandbüchern unterschieden. Zur Gestaltung von Benutzungshandbüchern wird ein Styleguide Benutzerdokumentation verwendet. Hierin wird beschrieben, wie das Erscheinungsbild der fertigen „Benutzerdokumentation“ aussehen soll. Für die Erstellung eines Tutorials wird das Storybook verwendet.

4.2.8 Systemtest

Das vollständig integrierte System durchläuft einen System- und Integrationstest. Hierzu werden Testdaten aufgebaut respektive die schon für den Komponententest verwendeten Testdaten ergänzt. Mit diesen Testdaten wird der Systemtest vollzogen. Nach erfolgreichem Test wird in der Systemauslieferung das System zusammen mit einem Benutzungstutorial an den Kunden ausgeliefert.

Das Vorgehen beim Systemtest ist durch das nachfolgende Modell dargestellt.

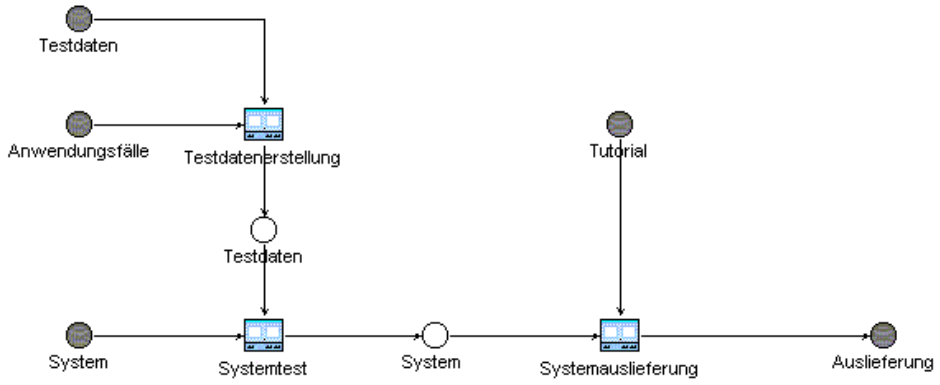


Abbildung 12 - Prozessmodell Systemtest

4.3 Qualitätsmanagement

Das Qualitätsmanagement ist nicht explizit definiert worden.

4.4 Projektplanung

Die Projektplanung ist nicht explizit definiert worden.

4.5 Integration

Die beschriebenen Prozesse (PG-Entwicklung, Systementwicklung, Qualitätsmanagement, Projektplanung) laufen nicht unabhängig voneinander ab. Sie haben Schnittstellen zueinander, die bedingen, dass Tätigkeiten innerhalb von Prozessen auf Ergebnisse von Tätigkeiten anderer Prozesse warten oder diese Ergebnisse bearbeiten, wenn diese vorliegen. Auf eine explizite Modellierung wurde aus Gründen der Verständlichkeit und Übersichtlichkeit verzichtet. Daher werden diese Schnittstellen nur tabellarisch dargestellt.

Name der Schnittstelle	erzeugender Prozess	bearbeitender Prozess
Anforderungen	Systementwicklung	PG-Entwicklung
Anwendungsfälle	PG-Entwicklung	PG-Entwicklung
Endberichtvorlage	Qualitätsmanagement	PG-Entwicklung
Grobarchitektur	Systementwicklung	PG-Entwicklung
GUI-Prototyp	Systementwicklung	PG-Entwicklung
Initiale Systemarchitektur	PG-Entwicklung	Systementwicklung
Neue Anforderungen (SEU)	Systementwicklung	PG-Entwicklung

Projektplan	Projektplanung	PG-Entwicklung
Prototypbeschreibung	Systementwicklung	PG-Entwicklung
Prozessleitfaden	Qualitätsmanagement	PG-Entwicklung
Rollenbeschreibung	PG-Entwicklung	Projektplanung
Storybook	Systementwicklung	PG-Entwicklung
Styleguide	Systementwicklung	PG-Entwicklung
Systemarchitektur	Systementwicklung	PG-Entwicklung
Testdatenbeschreibung	Systementwicklung	PG-Entwicklung
Tutorial	Systementwicklung	PG-Entwicklung
Vorlage PG-Protokoll	Qualitätsmanagement	PG-Entwicklung
Whitepaper	PG-Entwicklung	Systementwicklung
Zwischenberichtvorlage	Qualitätsmanagement	PG-Entwicklung

Darüber hinaus sei angemerkt, dass alle in der Systementwicklung erzeugten Dokumente, bevor sie in der Systementwicklung weiter verwendet wurden, durch das Qualitätsmanagement einem Review unterzogen wurden. Das Qualitätsmanagement erstellt ein Reviewprotokoll, in dem die Anmerkungen zu dem Dokument (z.B. Anforderungen, Styleguide, Klassendiagramm, JAVA Quellcode...) klassifiziert und priorisiert werden. Dieses Vorgehen wurde nicht eigens modelliert.

5 Projektplan

Dieses Kapitel stellt den Projektplan des IPSI-Projekts chronologisch dar. Die folgende Tabelle enthält u.a. Gruppen von Aktivitäten, die jeweils aus mehreren Einzelaktivitäten bestehen können (z.B. Implementierung und Test einer Komponente). Für eine Übersicht über den Zeitplan von IPSI ist diese Zusammenfassung sinnvoll, weil die Aktivitätsgruppen gut den Phasen der traditionellen Softwareentwicklung entsprechen und besser verglichen werden können.

In der Tabelle sind keine Aktivitäten aufgeführt, die nicht mit der eigentlichen Entwicklung zu tun haben, wie z.B. Auswahl der Entwicklungswerkzeuge, Erstellen der Rechner- und Entwicklungsumgebung, usw. Diese Aktivitäten laufen parallel zur eigentlichen Entwicklung ab und beeinflussen den eigentlichen Zeitplan daher nur indirekt.

Der Projektplan entstand weitgehend parallel zur Definition des Softwareprozesses. Daher gab es in den ersten Wochen nach Projektstart nur einen rudimentären Plan, der die groben Schritte der Softwareentwicklung (Anforderungsanalyse, Systemanalyse, Systemdesign, Implementierung und Test) beinhaltete und versuchte, die Phasen in den Gesamt-Projektzeitraum durch Schätzen unter Einbringung von Erfahrungswerten einzuordnen. Dieser Plan war wenig hilfreich und wurde eher im Nachhinein an die tatsächlich stattfindenden Aktivitäten angeglichen. Erst mit Fortschreiten des Softwareprozesses konnten die darin definierten Aktivitäten in den Projektplan aufgenommen und mit Terminierungen versehen werden. Die Termine richteten sich weiterhin nach Schätzungen der Projektteilnehmer und wurden während jeder laufenden Aktivität und nach deren Abschluss korrigiert. Es stellte sich dabei heraus, dass mit Fortschreiten des Projekts die Terminierungen immer exakter wurden und auch der real benötigten Zeit gut entsprachen.

Tätigkeit	Zeitraum	Beschreibung
Softwareentwicklungs-prozess	Durchgängig	Der Softwareentwicklungsprozess (SEP) definiert die einzelnen Schritte der Softwareentwicklung. Er wird während der gesamten Projektlaufzeit (weiter-)entwickelt.
Projektplanung	Durchgängig	Die Projektplanung greift die Schritte des SEP auf und terminiert sie. Der Projektplan dient den Projektteilnehmern als verbindlicher Terminplan bei der Durchführung der einzelnen Aktivitäten des SEP. Der PP entwickelt sich sukzessive mit dem SEP.
Anforderungsanalyse	11.10.-18.11.1999	Die Anforderungen und Ziele des IPSI-Portals werden festgelegt. Dazu werden Interviews mit Partner aus der Versicherungsbranche durchgeführt und daraus die Muss- und Kann-Anforderungen des Systems bestimmt.
1. GUI-Prototyp	19.11.-15.12.1999	Der erste GUI-Prototyp besteht aus eine Serie von statischen HTML-Seiten, die ein mögliches Design von IPSI darstellen.
Make-Or-Buy-Entscheidung	25.11.-14.12.1999	Nachdem die Komponenten des Systems identifiziert sind, wird in dieser Phase entschieden, welche Subsysteme entwickelt und

		welche zugekauft und integriert werden.
Machbarkeitsprüfung	27.12.1999-18.01.2000	Ziel der Machbarkeitsprüfung ist es, die Make-Or-Buy-Entscheidungen zu verifizieren. Es werden für jede Komponente eine Menge von Keyfeatures definiert, deren Machbarkeit mit der gewählten Methode „Make“ oder „Buy“ durch einen Cut-Through-Prototyp (CTP) zu beweisen ist.
Zwischenbericht	18.01.-28.03.2000	Der Zwischenbericht spiegelt die bisherigen Ergebnisse des Projekts wider und dient als Grundlage für den Endbericht.
Systemanalyse und Systemdesign der Subsysteme	02.02.-28.03.2000	Grob- und Feinentwurf jeder der Subsysteme Office, CMS, Procurement, Kommunikation, Administration, Suche, Core, Legacy.
Implementierung und Test der Subsysteme	22.03.-19.05.2000	Die Entwürfe der Subsysteme werden realisiert und getestet.
Integration Legacy	17.05.-30.06.2000	Das Legacy-Subsystem wird in das System integriert.
Dateneingabe	12.06.-30.06.2000	Die Subsysteme CMS, Procurement und Legacy werden mit sinnvollen, präsentationsfähigen Daten gefüllt.
GUI-Styleguide	12.06.-23.06.2000	Es wird ein Styleguide für das Design und die Bedienung von IPSI definiert.
Masken-Entwurf	12.06.-04.07.2000	Die HTML-Bildschirmmasken werden entworfen.
Entwicklung GUI-Klassen	23.06.-18.07.2000	Low- und High-Level-Klassen zur Realisierung des GUIs werden entworfen, realisiert und getestet.
Entwicklung Controller/Formatter	04.07.-08.08.2000	Die Controller und Formatter werden entworfen und mit Hilfe der GUI-Klassen implementiert.
Systemtest	08.08.-22.08.2000	Das Gesamtsystem wird gegen die Anforderungen getestet.

6 Anforderungsanalyse

In der Anforderungsanalyse wurden die fachlichen und technischen Anforderungen an das Portal definiert. Dies geschah teilweise durch Interviews mit Versicherungsunternehmen, teilweise auch durch eigene Vorstellungen von der Funktionalität eines solchen Portals.

Ausgangspunkt der Anforderungsanalyse war ein Vorschlag der Betreuer für zu verwendende Portalbestandteile, die sich typischerweise in Internet-Portalen finden. Diese sind im ersten Abschnitt dieses Kapitels dargestellt. Die Anforderungen die sich hieraus, aus den eigenen Vorstellungen und den Interviews mit Versicherungsunternehmen ergaben, sind in tabellarischer Form in den weiteren Abschnitten erläutert. Die Spalten der Tabelle geben folgende Informationen wider:

- Anforderungsname: zur eindeutigen Identifizierung jeder Anforderung. Der Anforderungsname wurde nach folgender Form gewählt: /X/Y<Nr1>/<Nr2>, wobei X ein Kürzel für das Subsystem darstellt (z.B. O für Office), Y entweder F (funktionale Anforderun) oder NF (nicht-funktionale Anforderung) ist. <Nr.1> und <Nr.2> vergeben fortlaufende Nummern / Unternummern.

- Aufgabenstellung: Inhalt der Anforderung

- Typ: identifiziert verschiedene Arten von Anforderungen: Muss-Anforderungen, die auf jeden Fall erfüllt werden müssen, Kann-Anforderungen, die nur bei ausreichender Zeit erfüllt werden sollen und inverse Anforderungen, die darlegen, was das System nicht enthalten darf

- Begründung: stellt die Rationale hinter der Anforderung dar, die meist fachlich motiviert ist

- Angabe über die Realisierung: gibt rückblickend an, ob die Anforderung letztlich umgesetzt wurde oder nicht; um die Tabellen nicht ein weiteres Mal abzudrucken, wurde diese Information bereits an dieser Stelle aufgenommen

- Begründung der Abweichung: begründet, warum eine Anforderung nicht umgesetzt wurde

6.1 Bestandteile des Internet-Portals

In der ersten Phase der Informationsanalyse wurde erkannt, dass das Internet-Portal als Integrationsplattform für verschiedene heterogene Anwendungen dient (Legacy-Anwendungen, Web-Anwendungen, Internet-Anwendungen, etc.). Diese sind in der folgenden Abbildung mit beispielhaften Funktionalitäten dargestellt.

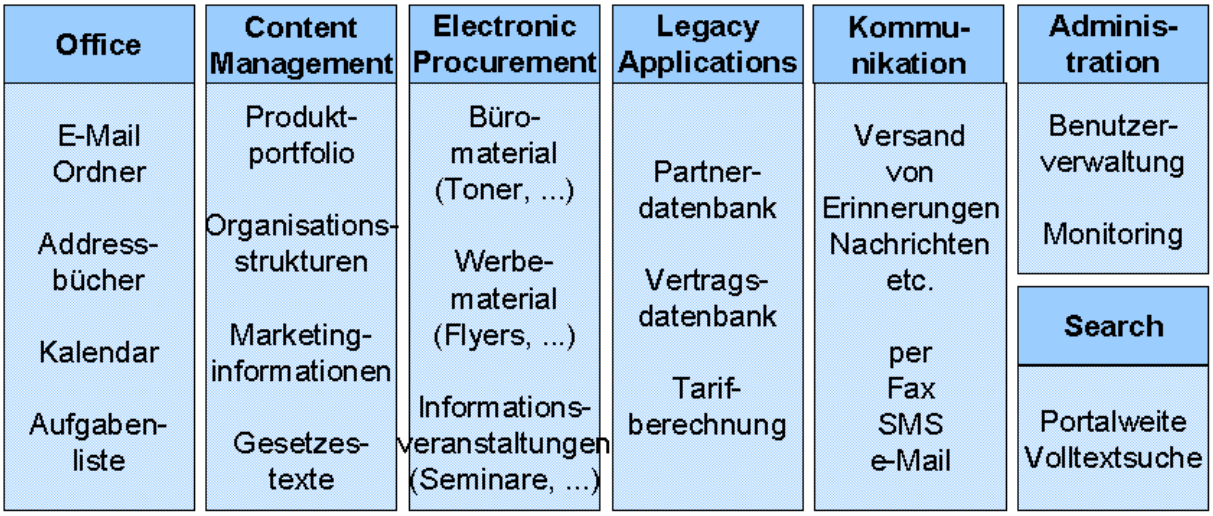


Abbildung 13 - Bestandteile des Portals

Ausgehend von einer 3-Ebenen-Architektur, bei der die Benutzungsoberfläche und die Datenverwaltung von der funktionalen Anwendungslogik getrennt wird, wurden in der Ebene der funktionalen Anwendungslogik zunächst die folgenden Bestandteile eines Portals für diese Zielgruppe klassifiziert:

- Portal-Administrations-System (PDS)
  Durch das PDS werden Funktionen zur Verfügung gestellt, die das Auswerten von Protokolldateien und deren graphische Präsentation ermöglichen. Darüber wird durch das PDS sichergestellt, dass sich der Benutzer des Portals nur einmal dem Portal gegenüber identifizieren muss.
- Kommunikationssystem (KS)
  Das Kommunikationssystem stellt die Schnittstelle zu den Techniken der Telekommunikation dar. Durch Funktionen können Texte und Benachrichtigungen in Form von Emails oder SMS-Nachrichten, aber auch als Fax versandt werden. Durch das Terminplanersystem gesteuert, werden diese Texte und Benachrichtungen jeweils zu vorher bestimmten Zeitpunkten versandt. Das Kommunikationssystem stellt ebenfalls den Zugriff auf das Internet-Portal mittels des Wireless-Application-Protocol (WAP) zu Verfügung, so dass auch direkt von Mobiltelefonen auf Informationen des Internet-Portals (soweit sinnvoll) zugegriffen werden kann.
- Suchsystem (SS)
  Um im gesamten Internet-Portal nach Inhalten suchen zu können, bietet das Suchsystem Funktionen zur kontextsensitiven Suche an. Dies umfasst neben der Möglichkeit einer Volltextsuche ebenso die Suche nach bestimmten Merkmalen von Informationen.
- Terminplanersystem/Adressverwaltungssystem (TAS)
  Die Verwaltung von Adressen und Terminen wird durch diese beiden Systeme gewährleistet. Im Gegensatz zu den sogenannten „harten Daten“, die durch das Partnermanagementsystem eines Versicherungsunternehmens für das Internet-Portal zur Verfügung gestellt werden, werden durch diese beiden Systeme nur die „weichen Daten“ verwaltet. Eine Synchronisation der Daten ist aber durchaus möglich.
- Contentmanagementsystem (CMS)
  Die Bereitstellung von Informationen durch das Versicherungsunternehmen für die VADs, aber auch durch die VADs für das Versicherungsunternehmen und andere VADs wird durch das CMS verwaltet. Auf der Basis eines Berechtigungskonzepts sind Innendienst- und Außendienstmitarbeiter in der Lage, sämtliche größeren Informationen (Prospekte, Handbücher, Gesetzestexte, Urteile, Marketingstrategien, etc.) zur Verfügung zu stellen, so dass der VAD auf diese Informationen zur Unterstützung seiner Arbeit zugreifen kann.
- Procurement-System (PS)
  Die Beschaffung von Verbrauchsmaterial einerseits und die Inanspruchnahme von Dienstleistungen (Seminare, Kurse, Veranstaltungen, Schulungen, Immobilienvermittlung) anderseits ist für Versicherungsunternehmen, wenn sie zentral erfolgt, ein wichtiger Kostenfaktor. Sämtliche VADs aber auch Innendienstmitarbeiter können aus einem zentralen Katalog Produkte und Dienstleistungen auswählen und bestellen.

In den folgenden Tabellen sollen nun die einzelnen Anforderungen aufgelistet werden, die in der Anforderungsanalyse an das Gesamtsystem identifiziert wurden. Gleichzeitig wird mit der Einnahme einer Retrospektive ein Abgleich mit den gestellten Anforderungen vorgenommen. D.h. es ist in den Tabellen dokumentiert, welche Anforderungen tatsächlich realisiert wurden und bei welchen es Abweichungen gab. Diese Abweichungen werden zusätzlich begründet.

Die Gliederung dieses Abschnitts erfolgt nun nach den identifizierten Subsystemen. Jedem Subsytem ist nachfolgend ein eigener Abschnitt gewidmet.

6.2 Anforderungen Office

6.2.1 Funktionale Anforderungen

Anforderungsname	Aufgabenstellung	Typ	Begründung	realisiert?	Begründung bei Abweichung / Erläuterung
/O/F1	Der VAD muss ein Kunden- & Interessenten-Adressbuch haben, um Daten zu speichern.	MUSS	Dient der Sicherung und besseren Verwaltung der harten und weichen Daten für den VAD, hier in Bezug auf die Adressen der Kunden bzw. der Interessenten.	ja	
/O/F1/1	Im Kunden- & Interessenten-Adressbuch muss die Möglichkeit bestehen, den Namen (Vorname, Zuname, Firma, ...), die Anschrift (Postfach, Straße, PLZ, Stadt, ...) die Kontaktmöglichkeiten (Telefon, Fax, Email, ...) und Notizen zu speichern	MUSS	Keine notwendige Information über einen Kunden oder Interessenten sollte fehlen.	ja	
/O/F1/2	Das Kunden- & InteressentenAdressbuch muss weiterhin die Möglichkeit haben zusätzliche Informationen zu speichern. Der VAD muss Felder frei definieren und nach seinen Wünschen füllen können.	KANN	Das System muss so flexibel wie möglich für die Bestände des VADs ausgelegt sein.	ja	Benutzerdefinierte Felder
/O/F1/3	Der VAD muss in dem Kunden- & InteressentenAdressbuch nach Daten (z.B. nach einem Namen) suchen können.	MUSS	Bei einem großen Adressbestand ist das Adressbuch sonst kaum handhabbar.	ja	Es gibt eine vordefinierte Suche nach dem Anfangsbuchstaben des Namens
/O/F1/4	Zusätzlich zum Speichern von Daten muss die Möglichkeit vorhanden sein, Daten zu ändern (Adresse einer Person ändert sich) und ggf. zu löschen (Interessent schließt keinen Vertrag ab).	MUSS	Daten müssen geändert und gelöscht werden können.	ja	
/O/F1/5	Das Kunden- & InteressentenAdressbuch muss die Möglichkeit bieten, Gruppen anzulegen.	MUSS	Die Einteilung der Adressen nach Gruppen wie z.B. Interessenten für Lebensversicherungen hilft, die Übersicht zu bewahren.	nein	Diese Funktionalität ist in Outlook natürlich möglich, wird aber in unserem Frontend nicht berücksichtigt. Allerdings haben wir drei feste Gruppen definiert: „Interessent“, „Kunde“ und „Privat“
/O/F1/6	Der VAD muss Gruppen anlegen, ändern und löschen können.	MUSS	VAD sollte Entscheidung über die Einteilung der Gruppen nach seiner persönlichen Organisationsstruktur vornehmen	nein	s.o.

			können.		
/O/F1/7	Der VAD muss über das Kunden- & InteressentenAdressbuch eine Möglichkeit der Ranking-Verteilung vornehmen können.	MUSS	Der VAD gibt wichtigen Kunden eine höhere Priorität als Kunden mit einem geringen Vertragsvolumen.	nein	Es ist möglich, beliebige benutzerdefinierte Felder zu definieren. Hier ist auch ein Ranking denkbar.
/O/F1/8	Der VAD muss eine Ranking-Verteilung anlegen, ändern und löschen können.	MUSS		nein	s.o.: Ranking ist nicht explizit realisiert
/O/F1/9	Jeder Adresse müssen Gruppen und ein Ranking zugeordnet werden können.	MUSS		nein	s.o.Grundsätzlich aber über benutzerdef. Felder
/O/F1/10	Der VAD muss die Daten in dem Kunden- & InteressentenAdressbuch jederzeit abrufen können, d.h. sowohl online als auch offline müssen die Daten zur Verfügung stehen.	MUSS	Die Daten im Adressbuch zählen zu den wichtigsten Arbeitsmitteln des VADs, zu denen er auch Zugang braucht, wenn er nicht online sein kann.	ja	Da die Daten in Outlook auf dem Rechner des VAD gespeichert werden, stehen diese auch Offline zur Verfügung – allerdings nicht über das Portal, sondern nur über Outlook selber.
/O/F1/11	Die Präsentation des Kunden- & InteressentenAdressbuchs kann auf unterschiedlichen Detailstufen erfolgen.	KANN	Die Sicht des VADs auf die Adressen sollten übersichtlich (z.B. sortiert nach Gruppe oder Ranking) innerhalb des Portals erfolgen.	nein	
/O/F2	Der VAD kann auf ein Adressbuch mit Daten von allen Kollegen lesend zugreifen.	KANN	Die Kontaktaufnahme mit seinen Kollegen soll unterstützt werden.	nein	Ein Zugriff auf das Adressbuch eines Kollegen schien in mehreren nachfolgenden Gesprächen nicht sinnvoll, da die weichen Daten eines VAD dort gespeichert sind, wo niemand außer ihm Zugriff haben sollte.
/O/F2/1	Der VAD kann folgende Informationen über seine Kollegen abrufen: Adresse, Telefon/Fax, Email, SMS, Schwerpunkte (z.B. Lebens-, KFZ-Versicherung).	KANN	Die Adresse ist wichtig, wenn der VAD per Post Kontakt mit anderen VADs aufnehmen möchte. Die Telefon-/Fax-Nr., Email bzw. SMS dienen der elektronischen Kontaktaufnahme.	nein	Kollegen können als normale Kontakte in Outlook angelegt werden und auch über das Portal abgerufen werden.

			Über die Schwerpunkte eines VADs können diese besser eingeordnet werden, und ein gezielterer Erfahrungsaustausch wird ermöglicht.		
/O/F2/2	Der VAD kann über Suchkriterien Kollegen in dem Adressbuch suchen und das auch mit Einsatz von Wildcards.	KANN	Eine schnelle Suche von Kollegen ist insbesondere bei einem umfangreichen Adressbuch erforderlich.	nein	Nicht explizit über Kollegen. Nur über alle Kontakte.
/O/F2/3	Der VAD kann über eine Anbindung des Adressbuches an das Kommunikationssystem einfach Kontakt mit anderen VADs aufnehmen. So kann nach Auswahl eines/einiger VADs direkt eine Email generiert werden, wobei der Adressat automatisch eingetragen wird. Es kann außerdem eine SMS verschickt werden.	KANN	Die elektronische Kontaktaufnahme wird erleichtert.	nein	s.o. Nur allgemein für Kontakte.
/O/F2/4	Der VAD kann über einen Exportfilter ausgewählte Adressen in sein Textverarbeitungsprogramm übernehmen (Serienbrieffunktionalität).	KANN	Die schriftliche Kontaktaufnahme wird erleichtert.	nein	Dies kann aber über die Standard-Exportfunktion von Outlook erfolgen.
/O/F2/5	Der VAD kann Terminanfragen an ausgewählte Adressen versenden (Anbindung an Terminverwaltung).	KANN	Die Terminabsprache wird erleichtert.	nein	Es besteht keine Anbindung an andere Terminverwaltungen als die eigene. Die Anfrage müsste also manuell erfolgen (z.B. per Email).
O/F2/6	Der VAD muss die Daten in dem Mitarbeiteradressbuch jederzeit abrufen können, d.h. sowohl online als auch offline müssen die Daten zur Verfügung stehen.	MUSS	Die Daten im Adressbuch zählen zu den wichtigsten Arbeitsmitteln des VADs, zu denen er auch Zugang braucht, wenn er nicht online sein kann.	ja	Grundsätzlich wie bei Kontakten.
/O/F2/7	Die Präsentation des Mitarbeiteradressbuchs kann auf unterschiedlichen Detailstufen erfolgen.	KANN	Die Sicht des VADs auf die Adressen sollte übersichtlich erfolgen.	nein	s.o.
/O/F3	Der VAD muss über einen Terminkalender verfügen können.	MUSS		ja	
/O/F3/1	Termine müssen hinzugefügt und wieder gelöscht werden können.	MUSS		ja	
/O/F3/2	Ein Termin muss verschiedene Parameter aufweisen (z.B. Zeit, Ort, Teilnehmer, Alarm, Wiederholung, Thema, Notizen).	MUSS	Ein Termin besteht nicht nur aus einem Zeitpunkt, sondern wird auch durch andere interessante Informationen charakterisiert.	ja	
/O/F3/3	Parameter eines Termins müssen verändert werden können.	MUSS	Da Termine sich ändern können, müssen ihre Parameter ebenfalls geändert werden können.	ja	
/O/F3/4	Es müssen „normale Termine“ definiert werden können.	MUSS	Normale Termine haben eine definierte Start- und Endzeit.	ja	

/O/F3/5	Es müssen Tagestermine (Ereignisse, Events) definiert werden können.	MUSS	Tagestermine unterscheiden sich von normalen Terminen darin, dass sie keine zeitlich angegebene Start- und Endzeit besitzen, sondern ganz allgemein an diesem Tag stattfinden.	ja	
/O/F3/6	Sämtliche Termine müssen wiederkehrend definiert werden können.	MUSS	Ein Geburtstag ist z.B. ein wiederkehrender Tagestermin.	nein	Dies ist zwar möglich und auch realisiert, jedoch lassen sich die einzelnen Instanzen einer Termin-Serie mit der Suchfunktion nicht finden. Grund ist die Outlook-API. So machen wiederkehrende Termine nicht Sinn.
/O/F3/7	Das Wiederholungsintervall muss frei wählbar sein.	MUSS	Sinnvoll wären z.B. täglich, wöchentlich, monatlich, jährlich.	nein	Es ist nur täglich, werktäglich, wöchentlich, monatlich (nach Datum), monatlich (nach Woche + Wochentag) und jährlich realisiert.
/O/F3/8	Das Wiederholungsintervall muss eine Terminierungsmöglichkeit aufweisen.	MUSS	Es müssen sowohl zeitlich unbegrenzt wiederkehrende Termine möglich sein (Geburts-/Feiertage) als auch z.B. PG-Sitzungen über 52 Wochen, dann nicht mehr.	ja	
/O/F3/9	Der VAD muss über das „Kommunikationssystem“ an Termine erinnert werden können.	MUSS	Dadurch wird verhindert, dass Termine versäumt werden.	ja	
/O/F3/10	Es muss möglich sein, die Alarmzeit unabhängig von der Terminzeit festlegen zu können.	MUSS	Man möchte gerne auch vor einem Termin an diesen erinnert werden	ja	Die Erinnerungszeit kann in Minuten vor dem Startzeitpunkt angegeben werden
/O/F3/11	Es muss verschiedene Sichten auf Termine geben.	MUSS	Alle Termine eines Tages, einer Woche oder eines Monats müssen überschaubar sein.	ja	Es gibt eine Tages- und eine Monatsübersicht
/O/F3/12	Überschneidungen von Terminen müssen erkennbar sein.	MUSS	Eine Überschneidung ist nicht zwangsläufig unerwünscht, ihre Hervorhebung hilft aber, mögliche Kollisionen zu vermeiden.	nein	In der Terminübersicht kann der Benutzer das nur selber herausfinden.
/O/F3/13	Überschneidungen von Terminen können bei der Eingabe entdeckt	KANN	s.o.	nein	

	werden.				
/O/F3/14	Es muss eine Suchfunktion im Terminkalender geben.	MUSS	Man muss Termine ohne großen Aufwand wiederfinden können.	ja	
/O/F3/15	Es können Termindaten aus dem Suchergebnis hinaus zur Bearbeitung genommen werden.	KANN	Es wäre damit möglich, sämtliche Termine mit Meier zu löschen und somit effizienter zu arbeiten.	ja	
/O/F3/16	Termine müssen „privat“ oder „geschäftlich“ sein können.	MUSS	Dadurch kann eine unterschiedliche Behandlung privater und geschäftlicher Termine gewährleistet werden.	nein	
/O/F4	Der VAD muss über eine ToDo-Liste verfügen können.	MUSS		ja	Eine ToDo-Liste wird in IPSI als Aufgaben-Liste bezeichnet
/O/F4/1	Der VAD muss Actions (Aufgaben) zu der ToDo – Liste hinzufügen oder ändern können.	MUSS		ja	
/O/F4/2	Eine Aufgabe besteht aus: Name, Priorität, Klassifikation, zugehörigen Terminen, Notizen, Historie, Status.	MUSS	Für zusätzliche Funktionen werden neben dem Namen noch weitere Attribute benötigt.	ja	
/O/F4/3	Der VAD muss anhand der Priorität die Dringlichkeit und Notwendigkeit einer Action festlegen können.	MUSS	Dadurch kann das System auf besonders wichtige Actions hinweisen und Actions nach Prioritäten sortiert anzeigen.	ja	
/O/F4/4	Der VAD muss Actions nach bestimmten Kriterien klassifizieren können.	MUSS	Durch eine Klassifizierung können verschiedenartige Actions in einer Liste gehalten werden und getrennt voneinander angezeigt werden, z.B. privat, geschäftlich.	nein	
/O/F4/5	Der VAD kann Kriterien zur Klassifizierung individuell definieren.	KANN	Dadurch wird eine Unterteilung möglich, die über die Klassifizierung in private und geschäftliche Actions hinausgeht.	nein	
/O/F4/6	Der VAD muss Actions mit bestimmten Terminen oder Zeiträumen verknüpfen können.	MUSS	Beim Erstellen einer Action kann ein Termin direkt in den Terminkalender eingefügt werden, z.B. Vorbereitung auf die Präsentation am 16.12.	nein	Dies wird von der Outlook-API nicht unterstützt
/O/F4/7	Der VAD muss zusätzliche Bemerkungen oder Verknüpfungen mit einer Action verbinden können.	MUSS	Diese Beschreibung kann ausführlichere Informationen zu einer Action enthalten. Hier können auch Verknüpfungen mit bestimmten Kunden oder Datenbank - Objekten hergestellt werden.	nein	Zu einer Aufgabe können Notizen verfasst werden, allerdings nicht mehr
/O/F4/8	Der VAD muss sich die aktuellen Actions nach Kriterien sortiert oder	MUSS	Dies steigert die Übersicht der Action-	ja	Eine Suche nach allen

	gefiltert anzeigen lassen können. Kriterien könnten sein: Priorität, Klasse, Status, Erstellungszeitpunkt oder alle Actions, die bis zu einem bestimmten Termin erledigt sein müssen. Auch die Kombination ist möglich.		Liste.		Attributen ist möglich – ohne Sortierung.
/O/F4/9	Der VAD muss den Status einzelner Actions ändern können. Beispiele für Stati: open, in progress, frozen, closed, deleted	MUSS		ja	
/O/F4/10	Der VAD kann auf eine ToDo – Historie zugreifen.	KANN	Dadurch wird es ihm ermöglicht, seine erledigten Actions zu betrachten.	ja	
/O/F4/11	Der VAD muss Filterregeln definieren können, anhand derer er über bestimmte Actions informiert wird (Reminder).	MUSS	So wird der VAD an wichtige oder dringende Aufgaben erinnert.	ja	
/O/F4/12	Der VAD kann Filterregeln definieren, anhand derer neue Actions z.B. in der Partnerdatenbank gefunden und automatisch in die Liste eingefügt werden.	KANN	Ständig wiederkehrende Aufgaben können so automatisch als Actions eingetragen werden. Z.B. Jahresbericht, Geburtstage von VIP-Kunden etc.	nein	
/O/F4/13	Es kann eine Schnittstelle zur Terminverwaltung zum Anlegen neuer Actions bereitgestellt werden. (oder andersherum: zum Anlegen neuer Termine)	KANN	Termine und Actions können sich gegenseitig beeinflussen.	nein	
/O/F4/14	Der Vorgesetzte (z.B. Agenturleiter) kann evtl. Actions vorschlagen, aber nicht die anderen einsehen.	KANN	Der Vorgesetzte kann so z.B. Berichte anfordern.	nein	

6.2.2 Schnittstellen-Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	realisiert ?	Begründung bei Abweichung / Erläuterung
/O/S1/1	Es muss eine Schnittstelle zwischen Adressbuch (Mitarbeiter-, Kunden- und Interessenten-Adressbuch) und dem Kommunikationssystem geben.	MUSS	Sinnvoll, wenn z.B. über das Kommunikationssystem aus dem Adressbuch heraus an einen Kunden eine Email verschickt werden soll.	ja	
/O/S1/2	Es muss eine Schnittstelle zwischen der Terminverwaltung und dem Kommunikationssystem geben.	MUSS	Sinnvoll, wenn z.B. über das Kommunikationssystem aus der Terminverwaltung eine SMS oder ein FAX verschickt werden soll.	ja	

/O/S1/3	Es muss die Möglichkeit bestehen, aus dem Adressbuch und aus der Terminverwaltung notwendige Übergabeparameter an das Kommunikationssystem weiterzuleiten, wie z.B. Adressierung (Email Adresse, Handy Nr., Fax Nr., ...) Art des Mediums (Email, SMS, Fax, ...) Mitteilung (Subjekt des Termins)	MUSS	Das Kommunikationssystem stellt dem Adressbuch und der Terminverwaltung eine Plattform zur Verfügung, Mitteilungen über ein Medium an eine bestimmte Adresse zu versenden. Dazu müssen Parameter übergeben werden.	ja	
/O/S2/1	Es muss eine Schnittstelle zwischen dem Adressbuch und der Terminverwaltung geben.	MUSS	Die Terminverwaltung interagiert direkt mit dem Adressbuch und umgekehrt.	nein	Dies wird durch die Outlook-API nicht unterstützt
/O/S2/2	Aus der Terminverwaltung muss es möglich sein, auf die Bestände des Adressbuches zuzugreifen, um eventuell Personen aus dem Adressbuch in einen Termin einzufügen.	MUSS		nein	s.o.
/O/S2/3	Aus dem Adressbuch muss ersichtlich sein, ob Termine mit einer Person existieren.	MUSS	Die Sicht aus dem Adressbuch heraus auf die Termine mit einer Person steigert die Benutzerfreundlichkeit des Portals, da diese Informationen nicht extra nachgeschlagen werden müssen.	nein	s.o.
/O/S3/1	Es muss eine Schnittstelle zwischen dem Suchsystem, dem Adressbuch und der Terminverwaltung geben.	MUSS	Siehe /O/S3/2	ja	
/O/S3/2	Das Suchsystem muss die Möglichkeit haben, eine Suche nach definierten Attributen an das Adressbuch und an die Terminverwaltung zu stellen.	MUSS	Das Suchsystem gibt die Suchanfrage (Attribute müssen definiert sein und mit Suchkriterien verbunden werden können) an das Adressbuch und an die Terminverwaltung weiter.	ja	
/O/S3/3	Die Ergebnismenge der Suche muss an das Suchsystem zurückgegeben werden. Nur bestimmte (Kern-)Attribute sollen dabei zurückgegeben werden.	MUSS	Das Adressbuch und die Terminverwaltung bearbeiten die Suchanfrage und geben das Ergebnis an das Suchsystem zurück. Um eine praktikable Präsentation des Ergebnisses zu sichern, werden nur bestimmte Attribute zurückgeliefert, aus denen man auf eine Detailansicht „verlinken“ kann.	ja	Durch das Search-Subsystem realisiert

/O/S4/1	Das Adressbuch und die Terminverwaltung müssen eine Schnittstelle an die Benutzungsoberfläche haben.	MUSS	Daten müssen angezeigt werden.	ja	
/O/S4/2	Die Benutzungsoberfläche muss Formulare integrieren, um Daten des Adressbuchs und der Terminverwaltung bearbeiten (insert, modify, delete) zu können.	MUSS	Formulare dienen der Eingabe von Daten und der Anweisung von Aktionen.	ja	
/O/S4/3	Die Benutzungsoberfläche muss Daten (in Form von Tabellen, Listen, ...) des Adressbuchs und der Terminverwaltung präsentieren können.	MUSS	Die Benutzungsoberfläche dient auch der Präsentation von Daten.	ja	
/O/S5/1	Das Adressbuch (Mitarbeiter-, Kunden- und Interessenten-Adressbuch) kann eine Schnittstelle an die Partnerdatenbank haben.	KANN	Die Verbindung der Daten kann dem VAD Arbeitserleichterung bringen, da er bestimmte Daten nicht extra nachschlagen muss.	ja	Realisiert im Legacy-System
/O/S5/2	Das Adressbuch kann mit der Partnerdatenbank des Versicherungsunternehmens synchronisiert werden.	KANN	Dies kann die Aktualität des Adressbuches steigern, z.B. wird ein Interessent nach Abschluß des Vertrages zum Kunden.	nein	Es werden sowohl harte als auch weiche Daten integriert angezeigt, allerdings ist eine Synchronisierung nicht mehr erwünscht gewesen
/O/S5/3	Der Zugriff auf die Partnerdatenbank darf nur lesend, aber nicht schreibend erfolgen.	INV	Sicherheit der Partnerdaten steht im Vordergrund.	ja	

6.3 Anforderungen Procurement

6.3.1 Funktionale Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	realisiert?	Begründung bei Abweichung / Erläuterung

				?	
/EP/F1/	Der VAD muss die Möglichkeit haben, für seine Arbeit erforderliches *Material* (z.B. Verbrauchsmaterial, Werbemedien, Büroausstattung etc.) über das Portal beim VU zu bestellen.	MUSS	Die Bestellung über das Portal verringert die „Papierarbeit“ des VADs in allen Phasen des Bestellprozesses von der Warenauswahl bis zur Abrechnung und bietet z.B. durch Personalisierung Zusatznutzen.	ja	Es wird das Shopsystem SmartStore integriert.
/EP/F1/1	Die dem VAD übergeordnete Instanz (VU oder AL) kann die Möglichkeit haben, Default-Regeln für bestimmte Nutzergruppen zu definieren (z.B. ist für einen in einer Agentur arbeitenden VAD das Angebot an Teppichen in den Unternehmensfarben nicht interessant, wohl aber für den AL).	KANN	Bei einem breiten Artikelspektrum ist die Definition von Ein-/Ausschlussregeln relativ aufwendig. Diese Arbeit kann dem VAD durch Default-Regeln abgenommen werden.	nein	Das SmartStore-System bietet diese Möglichkeit nicht
/EP/F1/2	Der VAD kann die Möglichkeit haben, die Default-Regeln an seine Bedürfnisse anzupassen (z.B. kein Interesse mehr an Tintenpatronen, seit der neue Laserjet auf dem Tisch steht).	KANN	Da Default-Regeln voraussichtlich relativ grob sind, ist die individuelle Anpassung wichtig.	ja	Auf der Portalseite werden bei der Auflistung von Shopangeboten die Interessen des VADs berücksichtigt.
/EP/F1/3	Alle Artikel müssen dem VAD in einem zentralen Bereich des Portals („Shop“) angeboten werden.	MUSS	Durch die Sammlung aller Artikel in einem zentralen Shop muss der VAD nicht im gesamten Portal nach dem gewünschten Artikel suchen, sondern hat eine Anlaufstelle für alle Bestellungen.	ja	
/EP/F1/3/1	Beim „Betreten“ des Shops oder auf Abruf muss der VAD auf neue Artikel und Sonderangebote aufmerksam gemacht werden.	MUSS	Da das Sortiment des Shops auf die Bedürfnisse der VADs abgestimmt ist, wird es sich voraussichtlich nicht ständig ändern. Gerade deshalb ist es wichtig, auf die Änderungen hinzuweisen.	ja	Dies geschieht über die Eingangsseite des Shop.
/EP/F1/3/2	Alle Artikel müssen im Shop über eine geeignete Hierarchiestruktur erreichbar sein (z.B. Bürobedarf: Drucker-Verbrauchsmaterial: Lasertoner: Kartusche für HP Laserjet 1100)	MUSS	Die Hierarchie bietet dem VAD eine gute Übersicht über das Sortiment und erlaubt das schnelle Navigieren zwischen Artikeln.	ja	
/EP/F1/3/3	Alle Artikel müssen im Shop zusätzlich über eine Volltext-Suchfunktion erreichbar sein (z.B. Suche nach „Lasertoner“).	MUSS	Die Suche hilft dem VAD, falls er den gewünschten Artikel in der Hierarchie nicht finden kann.	ja	
/EP/F1/3/4	Zu jedem Artikel müssen weitere artikelspezifische Daten neben Artikelbezeichnung und Preis abrufbar sein.	MUSS	Der VAD benötigt die artikelspezifischen Daten i.d.R., um eine Kaufentscheidung treffen zu können.	ja	
/EP/F1/3/4/1	Zu jedem Artikel darf nur der für den bestellenden VAD gültige Preis angezeigt werden.	MUSS	Für den VAD ist nur der für ihn gültige Preis relevant, zusätzliche ungültige Prei-	nein	Es wird nicht zwischen verschiedenen, benutzer-

			se verringern die Übersicht und können Missverständnisse auslösen.		bezogenen Preisen unterschieden
/EP/F1/3/5	Der VAD muss gewünschte Artikel im Shop zunächst unter Angabe der Bestellmenge in einem „Warenkorb“ sammeln (Durch das Ablegen im Warenkorb wird noch keine Bestellung ausgelöst).	MUSS	Der Warenkorb hilft dem VAD, einen Überblick über den aktuellen Einkauf zu behalten (s.u.).	ja	
/EP/F1/3/5/1	Der VAD muss die Möglichkeit haben, den Warenkorb jederzeit einzusehen, die Bestellmenge von Artikeln darin zu ändern oder Artikel daraus zu entfernen. Im Warenkorb müssen zu jedem Artikel Einzel- und Gesamtpreise sowie die Bestellsumme angezeigt werden.	MUSS	Der Warenkorb ermöglicht dem VAD, Kaufentscheidungen zu überdenken und rückgängig zu machen, bevor die Bestellung tatsächlich abgeschickt wird.	ja	
/EP/F1/3/6	Der VAD muss das Abschicken der Bestellung über die im Warenkorb liegenden Artikel explizit veranlassen.	MUSS	Ohne explizit ausgedrücktes Einverständnis des VADs dürfen keine Bestellungen abgeschickt werden, um Missverständnissen vorzubeugen.	ja	
/EP/F1/3/6/1	Der VAD kann die Möglichkeit haben, seiner Bestellung Kommentare, Fragen oder Hinweise beizufügen.	KANN	In Sonderfällen kann es für den VAD hilfreich sein, mit dem Bearbeiter der Bestellung in Kontakt treten zu können.	ja	
/EP/F1/4	Artikel können dem VAD auch ausserhalb des zentralen Shops an geeigneter Stelle „dezentral“ angeboten werden (z.B. unmittelbare Bestellmöglichkeit der gedruckten Version eines langen Dokuments, das im Content-Bereich angeboten wird).	KANN	Wenn der VAD irgendwo im Portal einen benötigten Artikel findet, muss er nicht extra in den Shop wechseln, sondern kann ihn gleich bestellen, ohne seinen aktuellen Kontext verlassen zu müssen.	ja	Die Artikel werden außer im Shop auch in der Eingangsseite des Portals und als Ergebnisse in der Portalsuche angeboten.
/EP/F1/5	Der VAD kann die Möglichkeit haben, eine gerade getätigte Bestellung auszudrucken.	KANN	Ausdrucke können als Archiv, Backup und Beleg bei Systemstörungen dienen.	ja	
/EP/F1/6	Der VAD muss die Möglichkeit haben, eine Historie seiner Bestellungen (incl. Bestellsummen) über einen gegebenen Zeitraum abzurufen.	MUSS	Diese Daten helfen dem VAD bei der Bedarfsplanung und der Kostenkontrolle.	nein	Die Bestellhistorie eines VADs ist nur für VU bzw. AL zugänglich.
/EP/F1/7	Der AL kann die Möglichkeit haben, Budgetgrenzen für Bestellungen festzulegen, die ein VAD nicht überschreiten darf.	KANN	Auf diese Weise hat der AL die Möglichkeit, das Kaufverhalten seiner VADs zu steuern (z.B. um es an ihre Leistung anzupassen).	ja	
/EP/F1/7/1	Der VAD muss vor dem Abschicken einer Bestellung darauf hingewiesen werden, wenn sie seine Budgetgrenze überschreitet (das Abschicken muss aber dennoch möglich sein, siehe Folgepunkt).	MUSS	Wenn eine Budgetgrenze besteht, sollte der VAD vor Überschreitungen gewarnt werden, um seine Bestellung ggf. noch verändern zu können.	nein	Bei Überschreitung der Bestellgrenze ist eine Bestellung nicht möglich.
/EP/F1/8/	Das VU muss die Möglichkeit haben, in einem Administrationsbereich das Shop-Angebot zu editieren.	MUSS	Da das Angebot variabel ist, ist eine Änderungsfunktion unerlässlich.	ja	
/EP/F1/8/1/	Das VU muss die Möglichkeit haben, im Administrationsbereich	MUSS	Da diese Daten variabel sind, ist eine	ja	

	Artikeldaten zu editieren (d.h. Artikeldaten hinzufügen, löschen, ändern).		Änderungsfunktion unerlässlich.		
/EP/F1/8/2/	Das VU muss die Möglichkeit haben, im Administrationsbereich die Kategoriehierarchie zu editieren (d.h. Kategorien anlegen, löschen, umbenennen).	MUSS	Da diese Struktur variabel ist, ist eine Änderungsfunktion unerlässlich.	ja	
/EP/F2/	Der VAD muss die Möglichkeit haben, für seine Arbeit erforderliche *termingebundene Veranstaltungen* (z.B. Schulungen), die vom VU vermittelt werden, über das Portal zu buchen.	MUSS	Die Buchung über das Portal verringert die „Papierarbeit“ des VADs in allen Phasen des Buchungsprozesses von der Veranstaltungsauswahl bis zur Abrechnung und bietet z.B. durch Personalisierung Zusatznutzen.	nein	Der Bereich Veranstaltungen ist aus Zeitgründen nicht realisiert worden. Allerdings könnten Veranstaltungen behilfsweise als Artikel abgelegt werden.
/EP/F2/1	Die dem VAD übergeordnete Instanz (VU oder AL) kann die Möglichkeit haben, Default-Regeln für bestimmte Nutzergruppen zu definieren (z.B. ist für einen in einer Agentur arbeitenden VAD das Angebot an IPSI-Schulungen für Agenturleiter nicht interessant, wohl aber für den AL selbst).	KANN	Bei einem breiten Veranstaltungsspektrum ist die Definition von Ein-/Ausschlussregeln relativ aufwendig. Diese Arbeit kann dem VAD durch Default-Regeln abgenommen werden.	nein	Das SmartStore-System bietet diese Möglichkeit nicht
/EP/F2/2/	Alle Veranstaltungen müssen dem VAD in einem zentralen Bereich des Portals („Ticket Center“) angeboten werden.	MUSS	Durch die Sammlung aller Veranstaltungen in einem zentralen Ticket Center muss der VAD nicht im gesamten Portal nach der gewünschten Veranstaltung suchen, sondern hat eine Anlaufstelle für alle Buchungen.	nein	Alle Veranstaltungen werden innerhalb einer Warengruppe im Shop angeboten.
/EP/F2/2/1/	Beim „Betreten“ des Ticket Centers oder auf Abruf muss der VAD auf neue Veranstaltungen aufmerksam gemacht werden.	MUSS	Da das Angebot des Ticket Centers sich ständig ändert, ist es wichtig, die Neuerungen gesammelt darzustellen.	ja	Auf der Eingangsseite des Portals werden die aktuellen Veranstaltungen angezeigt, wenn die Warengruppe „Veranstaltungen“ zu den Interessen des VADs gehören.
/EP/F2/2/2/	Alle Veranstaltungen müssen im Ticket Center über eine geeignete Hierarchiestruktur erreichbar sein (z.B. Schulungen: EDV: Textverarbeitung: MS Word)	MUSS	Die Hierarchie bietet dem VAD eine gute Übersicht über das Angebot und erlaubt das schnelle Navigieren zwischen Veranstaltungen.	nein	Es gibt keine Unterteilung innerhalb der Warengruppe „Veranstaltungen“
/EP/F2/2/3/	Alle Veranstaltungen können im Ticket Center zusätzlich über eine Volltext-Suchfunktion erreichbar sein (z.B. Suche nach „Rhetorikseminar“).	KANN	Die Suche hilft dem VAD, falls er die gewünschten Veranstaltung in der Hierarchie nicht finden kann.	ja	Die Veranstaltungen sind über die Portalsuche und Shopsuche erreichbar.
/EP/F2/2/4/	Zu jeder Veranstaltung müssen weitere leistungsspezifische Daten	MUSS	Der VAD benötigt die leistungsspezifi-	ja	

	neben Titel und Honorar abrufbar sein.		schen Daten i.d.R., um eine Buchungsentscheidung treffen zu können.		
/EP/F2/2/4/1/	Zu jeder Veranstaltung darf nur der für den bestellenden VAD gültige Preis angezeigt werden (wird z.B. für einen Veranstaltung eine vom Umsatz des VADs abhängige Subvention gewährt, so darf nur der für den bestellenden VAD gültige Preis angezeigt werden).	MUSS	Für den VAD ist nur der für ihn gültige Preis relevant, zusätzliche ungültige Preise verringern die Übersicht und können Missverständnisse auslösen.	nein	Es wird nicht zwischen verschiedenen, benutzerbezogenen Preisen unterschieden
/EP/F2/2/5/	Der VAD muss gewünschte Veranstaltungen im Ticket Center zunächst in einem „Warenkorb“ sammeln	MUSS	Der Warenkorb hilft dem VAD, einen Überblick über die aktuellen Buchungen zu behalten (s.u.).	ja	
/EP/F2/2/5/1	Der VAD muss die Möglichkeit haben, den Warenkorb jederzeit einzusehen, Buchungen darin zu ändern oder daraus zu entfernen. Im Warenkorb müssen zu jeder Veranstaltung Honorar und Stornogebühren sowie die Buchungssumme angezeigt werden.	MUSS	Der Warenkorb ermöglicht dem VAD, Kaufentscheidungen zu überdenken und rückgängig zu machen, bevor die Buchung tatsächlich abgeschickt wird.	ja	
/EP/F2/2/6/	Der VAD muss das Abschicken der Buchungen über die im Warenkorb liegenden Veranstaltungen explizit veranlassen.	MUSS	Ohne explizit ausgedrücktes Einverständnis des VADs dürfen keine Buchungen abgeschickt werden, um Missverständnissen vorzubeugen.	ja	
/EP/F2/2/6/1/	Der VAD kann die Möglichkeit haben, seiner Buchung Kommentare, Fragen oder Hinweise beizufügen.	KANN	In Sonderfällen kann es für den VAD hilfreich sein, mit dem Bearbeiter der Buchung in Kontakt treten zu können.	ja	
/EP/F2/3/	Veranstaltungen können dem VAD auch ausserhalb des zentralen Ticket Centers an geeigneter Stelle „dezentral“ angeboten werden (z.B. unmittelbare Buchungsmöglichkeit eines Seminars zu einem Thema, über das der VAD gerade im Content-Bereich liest).	KANN	Wenn der VAD irgendwo im Portal einen interessante Veranstaltung findet, muss er nicht extra in das Ticket Center wechseln, sondern kann sie gleich buchen, ohne seinen aktuellen Kontext verlassen zu müssen.	ja	Die Veranstaltungen werden außer im Shop auch in der Eingangsseite des Portals und als Ergebnisse in der Portalsuche angeboten.
/EP/F2/4/	Gebuchte termingebundene Veranstaltungen können automatisch in den Terminkalender eingetragen werden.	KANN	Die automatische Eintragung erspart dem VAD die manuelle Eingabe und beugt späteren Terminkonflikten vor.	nein	Es gibt keine Schnittstelle zum Office-Subsystem
/EP/F2/5/	Der VAD kann die Möglichkeit haben, eine gerade getätigte Buchung auszudrucken.	KANN	Ausdrucke können als Archiv, Backup und Beleg bei Systemstörungen dienen.	ja	
/EP/F2/6/	Der VAD kann die Möglichkeit haben, in einer Übersicht alle gebuchten, aber noch nicht in Anspruch genommenen Veranstaltungen jederzeit einzusehen und einzelne Veranstaltungen (soweit möglich) zu stornieren. Zu jeder Veranstaltung sind Honorar und Stornogebühren anzuzeigen.	KANN	Durch die Möglichkeit der Stornierung im Ticket Center kann der VAD Buchungsentscheidungen überdenken und bei Bedarf rückgängig machen.	nein	Das SmartStore-System bietet diese Möglichkeit nicht
/EP/F2/7/	Der VAD muss die Möglichkeit haben, eine Historie seiner Bu-	MUSS	Diese Daten helfen dem VAD bei der	nein	Die Bestellhistorie eines

	chungen (incl. Honorar bzw. Gebühren) über einen gegebenen Zeitraum abzurufen.		Bedarfsplanung und der Kostenkontrolle.		VADs ist nur für VU bzw. AL zugänglich.
/EP/F2/8/	Der AL kann die Möglichkeit haben, Budgetgrenzen für Buchungen festzulegen, die ein VAD in einem bestimmten Zeitraum nicht überschreiten darf.	KANN	Auf diese Weise hat der AL die Möglichkeit, das Buchungsverhalten seiner VADs zu steuern (z.B. um es an ihre Leistung anzupassen).	ja	Die Budgetgrenze gilt insgesamt für Artikel und Veranstaltungen.
/EP/F2/8/1/	Der VAD muss vor dem Abschicken einer Buchung darauf hingewiesen werden, wenn sie seine Budgetgrenze überschreitet.	MUSS	Wenn eine Budgetgrenze besteht, sollte der VAD vor Überschreitungen gewarnt werden, um seine Buchung ggf. noch verändern zu können.	ja	Bei Überschreitung der Bestellgrenze ist eine Bestellung nicht möglich.
/EP/F2/9/	Das VU muss die Möglichkeit haben, in einem Administrationsbereich das Ticket-Center-Angebot zu editieren.	MUSS	Da das Angebot variabel ist, ist eine Änderungsfunktion unerlässlich.	ja	Im Administrationsbereich des Shops
/EP/F2/9/1/	Das VU muss die Möglichkeit haben, im Administrationsbereich Veranstaltungsdaten zu editieren.	MUSS	Da diese Daten variabel sind, ist eine Änderungsfunktion unerlässlich.	ja	s.o.
/EP/F2/9/2/	Das VU muss die Möglichkeit haben, im Administrationsbereich die Kategoriehierarchie zu editieren.	MUSS	Da diese Struktur variabel ist, ist eine Änderungsfunktion unerlässlich.	ja	s.o.
/EP/F3/	Der AL muss die Möglichkeit haben, eine Historie aller Bestellungen und Buchungen seiner VADs, aufgeschlüsselt nach VAD und Artikel- bzw. Veranstaltungskategorien, abzurufen.	MUSS	Diese Daten helfen dem AL bei der Bedarfsplanung, Kostenkontrolle und dem Monitoring seiner VADs.	ja	s.o.

6.3.2 Nichtfunktionale Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	realisiert?	Begründung bei Abweichung / Erläuterung
/EP/N1/	Artikel und Veranstaltungen können im gleichen Softwaresystem verwaltet werden.	KANN	Die Datenstrukturen und Funktionen sind sehr ähnlich.	ja	Tatsächlich können Veranstaltungen wie Artikel in SmartStore behandelt werden. Es gibt allerdings keinen eigenen Veranstaltungsbereich
/EP/N2/	Der Shop muss im Portal organisatorisch vom Ticket Center getrennt sein.	MUSS	Das gemeinsame Anbieten von Artikeln und Veranstaltungen würde dort zu Problemen führen, wo die Bereiche Shop und Ticket Center sich unterscheiden (z.B. Terminkalender-Anbindung)	nein	Es gibt kein Ticket Center.

/EP/N3/	Historie-Informationen und Filterregeln können zentral gespeichert werden.	KANN	Eine dezentrale Speicherung ist nicht notwendig, da es sich nicht um weiche Daten handelt.	ja	Diese Informationen liegen zentral im Shopsystem, da sie hier benötigt werden.
/EP/N4/	Ist der VAD offline, so muss der Zugriff auf Shop und Ticket Center verhindert werden.	MUSS	Zur Offline-Nutzung müssten die kompletten Artikel- und Veranstaltungsdatenbanken auf den Client übertragen werden, und die gesamte Shop-Software müsste auf dem Client laufen. Da Bestellungen und Buchungen jedoch sehr wahrscheinlich am Arbeitsplatz (=online) und nicht mobil (=offline) getätigt werden, ist dieser Aufwand nicht gerechtfertigt.	ja	es ist kein Offline-Zugriff auf das Portal möglich.
/EP/N5/	Bei der Navigation in der Artikel- bzw. Veranstaltungshierarchie muss die aktuelle Position in der Hierarchie jederzeit offensichtlich sein.	MUSS	Der VAD muss sich jederzeit in der Hierarchie orientieren können, um effizient mit ihr arbeiten zu können.	ja	
/EP/N6/	Im Shop bzw. Ticket Center muss ständig ein Button zur Anzeige des Warenkorbs zur Verfügung stehen.	MUSS	Der Warenkorb ist ein wichtiges Hilfsmittel für den VAD, um Einkäufe und Buchungen zu planen.	ja	
/EP/N7/	Die Suchmaschinen im Shop und im Ticket Center müssen Schnittstellen bereitstellen, um Anfragen der globalen „Metasuchmaschine“ des Portals entgegenzunehmen und ihr Ergebnisse liefern zu können.	MUSS	Die niedrige Priorität ist damit begründet, dass *globale* Suchanfragen sich wahrscheinlich eher auf Inhalte von Contentmanagementsystem, Terminkalender und Adressbuch beziehen als auf die Daten im Shop.	ja	Dies wurde durch eine Anpassung der ASP-Seiten des Shops realisiert. Es bestand keine andere Schnittstelle.

6.3.3 Schnittstellen-Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	realisiert ?	Begründung bei Abweichung / Erläuterung
/EP/S1/	Es muss eine Schnittstelle zwischen Shop bzw. Ticket Center und Benutzungsoberfläche existieren	MUSS	Das Portal muss über eine einheitliche Oberfläche bedient werden können.	ja	aber nicht für das Ticket-Center (s.o.)
/EP/S1/1	Alle Funktionalitäten des Shops bzw. Ticket Centers müssen in einer einheitlichen Darstellung in die Benutzungsoberfläche integriert werden.(z.B. durch Frames)	MUSS	Der Nutzer des Portals braucht eine einheitliche Oberfläche um sich in dem Portal zurechtfinden zu können.	ja	Frames wurden nicht benutzt. Der Shop wurde allerdings graphisch angepasst.
/EP/S2/	Es muss eine Schnittstelle zwischen dem Suchsystem und dem Shop bzw. Ticket Center existieren.	MUSS	Um aus dem Suchsystem auf die spezielle Suche innerhalb des Shops bzw. Ticket	ja	Die integrierende Suche umfasst auch den Shop.

			Centers zugreifen zu können, muss eine Schnittstelle existieren.		
/EP/S2/1	Die Suchanfragen aus dem Suchsystem müssen an die Suche des Shops bzw. Ticket Centers weitergereicht werden.	MUSS	Die Suche innerhalb des Shops bzw. Ticket Centers wird über die Oberfläche des Suchsystems in Form einer Metasuche angesprochen.	ja	
/EP/S2/2/	Die Suchergebnisse müssen über eine Schnittstelle zum Suchsystem (Metasuchmaschine) des Portals zurückgegeben werden.	MUSS	Die Metasuchmaschine stellt eine Suchanfrage, die von der Suche innerhalb des Shops bzw. Ticket Centers bearbeitet wird. Die Ergebnisse werden dann zurückgegeben, um aus der Metasuchmaschine weitere Suchen zu ermöglichen.	ja	Über SearchResultset
/EP/S3/	Der Shop bzw. das Ticket Center kann über eine Schnittstelle zur Kommunikationskomponente verfügen.	KANN	Dadurch wird der VAD über die Änderungen im Shop bzw. Ticket Center informiert.	nein	Subsysteme sind passiv. D.h. sie werden nur angesprochen (von Controllern), sprechen aber selbst keine anderen Subsysteme an
/EP/S3/1	Der Shop bzw. das Ticket Center kann Mitteilungen an die Kommunikationskomponente weiterleiten.	KANN	Informationen über die Neueinträge und Änderungen im Shop und Ticket Center können mittels SMS, FAX, Email oder Popup-Fenster an VADs geschickt werden.	nein	s.o.
/EP/S4/	Der Shop bzw. das Ticket Center muss über eine Schnittstelle zur Portaladministration verfügen.	MUSS	Damit einheitliche Zugriffsrechte gewährleistet sind. Einmal Anmelden muss genügen.	ja	Die Benutzeradministration hat eine Schnittstelle zur Portaladministration. Die Artikeladministration des Shops erfolgt allerdings über den Shop selber.
EP/S6/	Zwischen dem CMS und dem Shop bzw. Ticket Center muss eine Schnittstelle bestehen.	MUSS	Es muss aus dem Shopsystem auf die Daten aus dem Content Bereich zugegriffen werden können, um auch Dokumente (z.B. aus dem Dienstleistungskatalog des VUs) anbieten zu können.	nein	das CMS konnte nicht integriert werden (s.u.)

6.4 Anforderungen Contentmanagement

6.4.1 Funktionale Anforderungen

Nr.	Aufgabenstellung	Prior.	Typ	Begründung
C/F1	VADs müssen auf das CMS zugreifen können.	1	MUSS	Dort sind Informationen enthalten, die er für seine Arbeit braucht.
C/F2	Das VU muss in der Lage sein, Content zu verwalten.	1	MUSS	Dadurch kann der VU bestimmen, worüber die VADs informiert werden sollen.
C/F2/1	VU muss Content ändern, hinzufügen und löschen können.	1	MUSS	Content muss immer aktuell sein
C/F2/2	VU muss die Möglichkeit haben, den einzelnen Benutzern bzw. Benutzergruppen Berechtigungen für bestimmte Bereiche des CMS zu vergeben.	1	MUSS	Der Gefahr der Manipulation wird vorgebeugt und die Daten können geschützt werden.
C/F3	Der VAD muss in Offline-Betrieb auf von ihm gewählte Dokumente zugreifen können (Downloadfähigkeit).	1	MUSS	Der VAD will auf bestimmte Informationen jeder Zeit zurückgreifen, ohne dass er Online sein muss.
C/F3/1	Der VAD muss Dokumente runterladen oder direkt drucken können.	1	MUSS	Dadurch hat der VAD auch die Möglichkeit die Dokumente in Papierform zu verwenden
C/F3/2	Der VAD muss explizit Tariftabellen runterladen können.	1	MUSS	Damit er sie auch Offline zur Verfügung hat.
C/F3/3	Der VAD kann auf Dokumente, die er runtergeladen hat aber nicht mehr aktuell sind, hingewiesen werden.	2	KANN	Dadurch wird vermieden, dass der VAD z.B. mit ungültigen Tarifen arbeitet.
C/F4	Es muss dem VAD eine intelligente Suchmaschine zur Verfügung stehen, mit deren Hilfe er die gewünschten Dokumenten finden kann.	1	MUSS	Das Content-System kann sehr schnell unübersichtlich werden, so dass es ohne Suchfunktionen sehr mühsam wird, bestimmte Dokumente zu finden.
C/F4/1	Die Suchfunktion muss durch Auswahl von boolschen Verknüpfungen, Datumseinschränkungen, Themengebiete, etc. optimierbar sein.	1	MUSS	Das Finden von gesuchten Dokumenten wird dadurch beschleunigt.
C/F5	Durch Berücksichtigung von Personalisierungsregeln können die für die einzelnen VADs relevanten Dokumente hervorgehoben werden.	2	KANN	VADs finden die Dokumente, die sie für ihre Arbeit benötigen, schneller.
C/F5/1	Es muss auf Neuigkeiten in der Content-Datenbank gesondert hingewiesen werden. (unter Berücksichtigung von Personalisierungsregeln)	1	MUSS	Neuer Content enthält meist wichtige und aktuelle Informationen.

6.4.2 Schnittstellen-Anforderungen

C/S1	Es muss eine Schnittstelle zwischen Suchsystem und CMS geben.	1	MUSS	Damit der VAD die Möglichkeit hat, Inhalte des CMS durchzusuchen.
C/S1/1	Die Suchmaschine des CMS muss die Anfragen des Suchsystems bearbeiten und die Suchergebnisse zurückliefern.	1	MUSS	Das Suchsystem erhält die Suchergebnisse aus einzelnen Komponenten und stellt sie dar.
C/S1/2	Es muss in CMS einen Verweis auf das Suchsystem geben.	1	MUSS	Damit der VAD von jeder Stelle innerhalb des CMS auf Suchfunktionen zugreifen kann.

C/S2	Es muss eine Schnittstelle zwischen CMS und Portal-Administrationssystem geben.	1	MUSS	Dadurch ist es gewährleistet, dass die Benutzer- und Rechteverwaltung der beiden Systemen übereinstimmen.
C/S2/1	Die im Portal definierten Benutzer und deren Rechte müssen dem CMS bekannt sein.	1	MUSS	Dadurch wird vermieden, dass sich der Benutzer für die Nutzung des CMS zusätzlich identifizieren muss.

Die Anforderungen an das Contentmanagementsystem sind nur insoweit erfüllt worden, als das ein solches CMS dem VAD zur Verfügung gestellt wird, um darin zu suchen und Daten abzulegen. Die Integration mit dem Portal (z.B. im Hinblick auf eine Einbeziehung in die übergreifende Suche) konnte nicht realisiert werden, da eine integrationsfähige Version des CMS (Pirobase 4.0 ist ansprechbar über CORBA, die vorliegende Version 3 allerdings nur über weitere Servlets) bis zur gesetzten Deadline nicht zur Verfügung stand und eine nachträgliche Integration des CMS die Dauer der Projektgruppe für alle PG-Teilnehmer unerträglich verlängert hätte. Die Integrierbarkeit wurde jedoch durch einen CTP (Cut-Through-Prototype) gezeigt. Die jetzige Einbindung besteht lediglich in einem Link auf das CMS-System.

6.5 Anforderungen Suchen

6.5.1 Funktionale Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	realisiert?	Begründung bei Abweichung / Erläuterung
Suche1	Es muss eine Suchfunktion angeboten werden, die in allen Bereichen von IPSI eine Suche durchführt.	MUSS	So kann der VAD schnell einen umfassenden Überblick zu einem Thema erhalten.	ja	
Suche2	Die Suche muss auf folgende Bereiche eingeschränkt werden können: Contentmanagement: Dokumentenbeschreibung und Dokumenteninhalte Legacy-System Office: Termine, Kontakte, und Nachrichten Procurement: Produktbezeichnungen und Produktbeschreibungen	MUSS	Eine Überflutung mit unerwünschten Informationen wird verhindert, so dass der VAD keine Zeit für das Filtern der Informationen aufwenden muss.	ja	bis auf das CMS, das nicht integriert werden konnte (s.o.)
Suche3	Für die Suche müssen Suchkriterien zur Verfügung stehen, die die individuelle Beschaffenheit der Informationen im jeweiligen System berücksichtigen.	MUSS	Sonst ist der VAD unter Umständen gezwungen, doch noch einmal im Teilsystem zu suchen. Beispiele für individuelle Suchkriterien: Office – Termin: Datum Procurement - Produkt: Preis	ja	Neben einer Volltextsuche steht auch eine Expertensuche für die einzelnen Subsysteme zur Verfügung.

6.5.2 Nichtfunktionale Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	realisiert	Begründung bei Abweichung / Erläuterung

				siert ?	
Suche4	Es muss leicht sein, später weitere Suchbereiche anzubinden.	MUSS	Da IPSI eine Integrationsplattform ist, muss mit der Anbindung weitere Systeme gerechnet werden. (Beispiele: Partnerdatenbank, Provisionsverwaltung)	ja	Die Schnittstelle zum Suchsystem ist eindeutig definiert

6.5.3 Schnittstellen-Anforderungen

Das Subsystem Suche braucht Schnittstellen in allen Subsystemen, in denen nach Inhalten gesucht wird. Nach den Anforderungen sind das die folgenden Subsysteme: Contentmanagement Legacy-System Office Procurement Das Subsystem Suche übergibt einen Teil einer Suchanfrage und erhält ein tabellarisches Suchergebnis.	ja	bis auf CMS (s.o.)
Für die Benutzungsoberfläche werden die Suchfunktionalitäten zur Verfügung gestellt. Das Subsystem Suche erhält eine Suchanfrage und gibt ein tabellarisches Ergebnis zurück.	ja	

6.6 Anforderungen Kommunikation

6.6.1 Funktionale Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	reali-siert?	Begründung bei Abweichung / Erläuterung
Kom1	Kommunikation VU – VAD (Push) Neue Inhalte und Nachrichten müssen dem VAD (wenn er zu einer bestimmten Benutzergruppe gehört) durch die diversen Push - Arten mitgeteilt werden.	MUSS	Der VAD muss auch erreichbar sein, wenn er nicht online ist.	nein	Die Funktionalität wird vom Subsystem Kommunikation bereitgestellt, aber derzeit nicht benutzt.
Kom2	Kommunikation VU – VAD (Pull) Neue Inhalte müssen dem VAD nach dem Einloggen im Portal angezeigt werden.	MUSS	So muss der VAD nicht selber nach neuen Inhalten suchen.	ja	dies geschieht auf der Startseite
Kom3	Diese Übersicht (Kom2) kann auch mit der WAP-Technik abgerufen werden.	KANN	Die WAP-Technologie ermöglicht es, sich mobil über neue Inhalte zu informieren.	nein	die WAP-Integration ist generell möglich, wurde aber aus Zeitgründen nicht weiter verfolgt.
Kom4	Kommunikation VU – VAD (Pull)	MUSS	Das VU ist für das System verantwortlich	ja	Dies ist über das CMS

	Das VU muss dem VAD Zugang auf eine FAQ-Liste und eine Hilfe zum Portal anbieten.		und muss sich um die Nutzer kümmern.		möglich.
Kom5	Kommunikation VAD – VAD (Push) Für die Kommunikation zwischen den VADs müssen ihnen die diversen Push-Arten angeboten werden.	MUSS	So können sich die VADs noch einfacher absprechen.	ja	
Kom6	Kommunikation VU – Kunde (Pull) Das VU kann dem Kunden eine FAQ-Liste zu den Angeboten zur Verfügung stellen. Dorthin gelangt der Kunde über die Homepage des VADs.	KANN	Der VAD kann nicht selber alle Infos zu den Versicherungen anbieten.	nein	Die Anbindung von Kunden wurde nicht berücksichtigt. Fokusgruppe sind die VADs
Home1	Jeder VAD muss über eine automatisch generierte Homepage für seine Kunden verfügen können.	MUSS	So kann er seine Kunden besser betreuen.	nein	Aus Zeitgründen wurde dieser Bereich nicht betrachtet.
Home2	Dem VAD muss eine Maske zum Editieren der Homepage-Inhalte angeboten werden.	MUSS	So präsentieren sich alle Homepages im gleichen Design.	nein	s.o.
Home3	Die Homepage muss folgendes enthalten können: Name, Anschrift, Telefon, Email Freier Text Bild(er)	MUSS	Wichtige Infos sind immer an der gleichen Stelle. Dazu hat der VAD noch genügend Freiheit für individuelle Inhalte.	nein	s.o.
Home4	Die Homepage muss Kunden und Interessenten die Möglichkeit bieten, sich in die Emailingliste des VADs einzutragen.	MUSS	Falls der Kunde es wünscht, kann er sich in die Emailingliste eintragen und über interessante Neuigkeiten informieren lassen. Andererseits kann der VAD so auch an Adressen neuer Interessenten kommen.	nein	s.o.
Home5	Der Eintrag eines Kunden in die Emailingliste muss eine Benachrichtigung an den VAD per Email nach sich ziehen.	MUSS	So kann der VAD schnell reagieren, z.B. jemand besuchen.	nein	s.o.
Home6	Die Homepage muss dem Kunden ein Feld zum Verschicken von Emails (SMS, Fax) bieten.	MUSS	Nicht alle Kunden haben Email. Dazu ist es so einfacher für sie.	nein	s.o.
Home7	Die Homepage muss einen Link zum VU enthalten.	MUSS	Der VAD muss nicht selber über alle Produkte informieren. Außerdem hat der Kunde da die Möglichkeit zu einem Feedback (siehe Monitoring)	nein	s.o.
Home8	Die Homepage kann frei editierbare Links enthalten.	KANN	Hier kann der VAD auch auf eigene individuellere Seiten verweisen.	nein	s.o.

6.6.2 Nichtfunktionale Anforderungen

Anf.Name	Aufgabenstellung	Typ	Begründung	reali-	Begründung bei Ab-

				siert?	weichung / Erläuterung
KoTech1	Alle Subsysteme müssen über eine einheitliche Schnittstelle die verschiedenen Push-Arten zum Informationsversand nutzen können.	MUSS	Das erleichtert die Bedienung (und die Programmierung).	ja	
KoTech2	Die angebotenen Push-Arten sind: Email SMS Fax	 MUSS KANN KANN	So kann der VAD auch ohne ständige Online-Verbindung informiert werden.	ja	SMS und Fax sind vorbereitet. Die entsprechenden Klassen wurden allerdings nicht vollständig implementiert, da der PG an der Universität kein Fax/Modem zur Verfügung stand.

6.6.3 Schnittstellen-Anforderungen

Das Subsystem Kommunikation bietet den anderen Subsystemen die Möglichkeit, Nachrichten zu verschicken. Wie bereits in der nichtfunktionalen Anforderung KoTech1 beschrieben, ist daher eine einheitliche Schnittstelle zu allen anderen Subsystemen erforderlich, die den Versand von Mitteilungen unterstützt. Diese Schnittstelle ist über die Fassaden-Klasse des Subsystems realisiert worden.

6.7 Anforderungen Administration/Monitoring

6.7.1 Funktionale Anforderungen

Anf. Name	Aufgabenstellung	Typ	Begründung	realisiert?	Begründung bei Abweichung / Erläuterung
Admin1	Ein Administrator im VU muss Nutzern die Rechte eines „Agenturleiters“ zuteilen bzw. wegnehmen können. Dabei müssen sie entscheiden können, ob die untergeordneten VADs auch ihre Rechte verlieren.	MUSS	Nur sie haben einen Überblick über die Agenturen. Außerdem könnten sich Agenturen auflösen bzw. fusionieren.	ja	
Admin2	Die Agenturleiter und das VU müssen ihren Mitarbeitern die Rechte eines VADs zuteilen und wegnehmen können.	MUSS	Falls das VU die VADs einer Agentur nicht kennt, muss der Agenturleiter die VADs löschen bzw. einfügen können.	ja	
Admin3	Der Benutzer muss sich beim Start des Portals durch einen Benutzernamen und ein Passwort authentifizieren.	MUSS	Alle Bereiche des Portals müssen aus Sicherheitsgründen passwortgeschützt sein. Außerdem	ja	Der LoginController realisiert dies.

			erleichtert die Authentifizierung die Personalisierung des Portals.		
Admin4	Der Benutzer muss die Möglichkeit haben, sich beim Verlassen des Portals explizit abzumelden.	MUSS	Alle laufenden Sitzungen werden so auf der Stelle beendet; somit ist Missbrauch schwieriger.	ja	
Admin5	Der Benutzer kann sein Portal-Passwort ändern.	MUSS	Falls das Passwort irgendwie bekannt geworden ist, kann der VAD so Missbrauch vermeiden.	nein	Dies kann nur der AL.
Monitor1	Das VU kann die Entwicklung des Kundenstammes einer Agentur tabellarisch und graphisch sehen. Zur Entwicklung gehört: Zahl der neuen Vertragsabschlüsse (gegliedert nach Art) Zahl der Vertragsabläufe ohne Verlängerung Zahl der insgesamt betreuten Kunden	MUSS	Nur durch graphische Anzeige werden die Daten plastisch.	ja	Dies ist über die Statistiken realisiert.
Monitor2	Dem Versicherungskunden kann ein Feedbackfeld zur Verfügung gestellt werden, indem die Betreuung durch den VAD / die Agentur bewertet werden kann. Die Bewertung sollte ein frei editierbares Feld sein und Möglichkeiten zur Benotung enthalten.	KANN	So lässt sich die qualitative Arbeit bewerten und so bekommt das VU Zahlen in die Hand, die ihre Entscheidungen unterstützen können.	ja	
Monitor3	Das VU kann die Bewertung und ermittelte Durchschnitte einsehen.	MUSS	Das dient der Bewertung der Kundenbetreuung eines VADs / einer Agentur.	ja	
Monitor4	Ein Administrator muss die Zahl der Zugriffe auf das Portal und die Zahl der Zugriffe auf die einzelnen Teilfunktionen des Portals einsehen können. Diese Sicht muss auch bezüglich einzelner VADs und Agenturen abrufbar sein. Teilfunktionen sind: Benutzung des Kalenders Benutzung der Kundendatenbank Benutzung des Shops Benutzung der zur Verfügung gestellten Informationen und Marketingunterstützung.	MUSS	Nur so lässt sich bewerten, ob das Portal genutzt wird, also die VADs unterstützt, und welche Funktionen überflüssig sind.	nein	Dies hätte ein Logging für jeden Schritt eines Benutzers vorausgesetzt.
Sonstige1	Es muss die technische Möglichkeit bestehen, dass die sensiblen Daten der VADs eines Tages vom VU zentral verwaltet werden können.	MUSS	So bleibt die Entscheidung über den Umgang mit sensiblen Daten den Betroffenen überlassen.	nein	Die weichen Daten werden in Outlook gespeichert.

6.7.2 Nichtfunktionale Anforderungen

Anf. Name	Aufgabenstellung	Typ	Begründung	realisiert	Begründung bei Abweichung / Erläuterung

NFAdmin1	Die Administrationschnittstelle muss in das Portal integriert sein.	MUSS	So bekommen die Administratoren ein Gefühl für die Situation der VADs. Außerdem wirkt das Portal homogener.	ja	
NFAdmin2	Der VAD muss sich nur beim Start des Portals authentifizieren, danach nicht mehr.	MUSS	So wird das Portal benutzerfreundlicher.	ja	Ein Single-Sign-On wurde realisiert.

6.7.3 Schnittstellen-Anforderungen

Anf. Name	Aufgabenstellung	Typ	Begründung	realisiert	Begründung bei Abweichung / Erläuterung
SSMonitor1	Es muss eine Schnittstelle zur GUI geben.	MUSS	Die Visualisierung übernimmt die GUI.	ja	
SSMonitor2	Es muss eine Schnittstelle zur Kundendatenbank geben.	MUSS	Wird für die Generierung der Entwicklung des Kundenstamms eines VADs / einer Agentur benötigt.	ja	Realisiert durch die Fassadenklasse des Legacy-Subsystems.
SSMonitor3	Es muss eine Schnittstelle zur Homepage des VADs / der Agentur geben.	MUSS	Das Feedbackfeld für den Kunden wird auf der Homepage des VADs / der Agentur zur Verfügung gestellt.	nein	Die Hompage wurde nicht realisiert (s.o.).
SSMonitor4	Es muss eine Schnittstelle zum Kalender (Subsystem Office) geben.	MUSS	Wird für die Erstellung der Portalbenutzungsstatistiken benötigt.	nein	Diese Statistik erschien nicht sinnvoll.
SSMonitor5	Es muss eine Schnittstelle zum Subsystem Shop geben.	MUSS	Wird für die Erstellung der Portalbenutzungsstatistiken benötigt.	ja	
SSMonitor6	Es muss eine Schnittstelle zum Subsystem CMS geben.	MUSS	Wird für die Erstellung der Portalbenutzungsstatistiken benötigt.	nein	Das CMS konnte nicht integriert werden.

6.8 Anforderungen GUI

Die Präsentationsanforderungen definieren eine einheitliche und benutzerfreundliche Oberfläche für alle Komponenten des Portals.

Anf.Name	Aufgabenstellung	Typ	Begründung	realisiert?	Begründung bei Abweichung / Erläuterung

/GUI/1/	Die Oberfläche muss die *Grundsätze der Dialoggestaltung* [Di88], [Is93] erfüllen, soweit anwendbar.	MUSS	Diese Grundsätze bilden die genormte Basis für benutzerfreundliche Oberflächen, unabhängig von ihrer konkreten Ausprägung.	ja	
/GUI/1/1/	Dialoge müssen *aufgabenangemessen* sein, d.h. die Erledigung der eigentlichen Arbeitsaufgabe des Benutzers unterstützen, ohne ihn durch Eigenschaften des Systems zusätzlich zu belasten.	MUSS	Aufgabenangemessenheit ist die Grundvoraussetzung für effiziente Arbeit mit dem System.	n/a	dies ist eine nicht messbare Anforderung, die hätte quantifiziert werden müssen.
/GUI/1/2/	Dialoge müssen *selbsterklärungsfähig* sein, d.h. unmittelbar verständlich sein oder dem Benutzer auf Verlangen Einsatzzweck und Einsatzweise des Dialogs erläutern können.	MUSS	Sowohl neue als auch erfahrene Nutzer sollten das System bedienen können, ohne ständig auf die Dokumentation zurückgreifen zu müssen.	n/a	s.o.
/GUI/1/3/	Dialoge müssen *steuerbar* sein, d.h. dem Benutzer erlauben, die Geschwindigkeit des Ablaufs, die Auswahl und Reihenfolge von Arbeitsmitteln sowie Art und Umfang der Ausgabe zu beeinflussen.	MUSS	Da verschiedene Benutzer auf unterschiedliche Weise arbeiten, sollte das System sie nicht auf eine bestimmte Arbeitsweise festlegen.	n/a	s.o.
/GUI/1/4/	Dialoge müssen *erwartungskonform* sein, d.h. den Erwartungen des Benutzers entsprechen, die er aus Erfahrungen mit Arbeitsabläufen mitbringt und die er sich während der Benutzung des Systems bildet.	MUSS	Werden die Erwartungen des Nutzers nicht erfüllt, so ist die Bedienung des Systems kaum erlernbar.	n/a	s.o.
/GUI/1/5/	Dialoge müssen *fehlerrobust* sein, d.h. trotz erkennbar fehlerhafter Eingaben muss das beabsichtigte Arbeitsergebnis mit minimalem oder ohne Korrekturaufwand erreichbar sein. Dazu müssen dem Benutzer die Fehler zum Zweck der Behebung verständlich gemacht werden.	MUSS	Werden fehlerhafte Eingaben nicht korrigiert, so ist die Datenintegrität und Systemstabilität nicht mehr gewährleistet. Die Unterstützung des Benutzers bei der Korrektur ist wichtig, damit er durch Fehler nicht in eine Sackgasse gerät.	ja	Dies geschieht durch passende Fehlermeldungen an den entsprechenden Stellen.
/GUI/1/6/	Dialoge können *individualisierbar* sein, d.h. den individuellen Bedürfnissen und Fähigkeiten des Benutzers anpassbar sein.	KANN	Individualisierbare Dialoge erlauben besonders erfahrenen Nutzern, effizienter zu arbeiten.	nein	Der Aufwand hierfür schien zu hoch.
/GUI/1/7/	Dialoge müssen *erlernbar* sein, d.h. den Benutzer in den Lernphasen unterstützen und führen.	MUSS	Um produktiv zu arbeiten, sollte der Nutzer sein eigenes Fachwissen problemlos auf das neue System übertragen können, ohne	n/a	s.o.

			besonderes Vorwissen über das System zu benötigen.		
/GUI/2/	Bei der Beschriftung von Dialogelementen müssen die *Beschriftungsregeln* aus dem CUA-Styleguide [Ib90] beachtet werden.	MUSS	Die Beschriftungsregeln sind konkrete Anwendungen des Dialoggrundsatzes der Selbsterklärungsfähigkeit.	ja	
/GUI/3/	Bei der Anordnung von Dialogelementen müssen die *Anordnungsregeln* aus dem CUA-Styleguide [Ib90] beachtet werden.	MUSS	Die Anordnungsregeln sind konkrete Anwendungen des Dialoggrundsatzes der Erwartungskonformität.	ja	
/GUI/4/	Bei der Gestaltung des *Seitenlayouts* müssen folgende Regeln beachtet werden.	MUSS		--	--
/GUI/4/1/	Das Seitenlayout darf der übersichtlichen Präsentation von Informationen nicht im Weg stehen.	INV	Ein zu komplexes Design drängt sich in den Vordergrund und lenkt von den eigentlichen Informationen ab; ein zu simples Design vernächlässigt die Möglichkeiten des Mediums und erschwert so die Arbeit mit dem System.	n/a	s.o.
/GUI/4/2/	Das Seitenlayout (Farbe, Position und Funktion von Elementen etc.) muss im ganzen Portal konsistent sein.	MUSS	Ein durchgängiges Gestaltungskonzept gibt dem Benutzer mehr Sicherheit bei der Orientierung und hilft ihm beim Entwickeln eines mentalen Modells der Site.	ja	Dies ist durch die Verwendung von einheitlichen Klassenbibliotheken (GUILib/GUIBase) sichergestellt.
/GUI/4/3/	Die Funktion und Nutzbarkeit von Webseiten darf bei unterschiedlichen Bildschirmauflösungen und Farbtiefen nicht eingeschränkt werden.	INV	Das Portal wird auf unterschiedlichen Zielsystemen (Desktop bis Notebook) laufen.	ja	
/GUI/4/3/1/	Der Seiteninhalt muss auch bei kleinen Auflösungen horizontal vollständig dargestellt werden und sich größeren Auflösungen flexibel anpassen.	MUSS	Bei zu breitem Seitenlayout laufen Inhalte über den rechten Bildschirmrand hinaus und erfordern umständliches horizontales Scrolling. Da der Benutzer dies im Web nicht erwartet, können wichtige Informationen übersehen werden.	ja	
/GUI/4/3/2/	Informationen und Zusammenhänge dürfen nicht allein durch die Farbgebung vermittelt werden.	INV	Auf Displays mit geringer Farbtiefe (bis hin zu monochrom) können sonst Information verlo-	ja	

			ren gehen.		
/GUI/5/	Bei der Formulierung des *Fließtextes* müssen folgende Regeln beachtet werden.	MUSS		--	--
/GUI/5/1/	Der Text muss „überfliegbar“ sein (d.h. aussagekräftige Überschriften wählen, Schlüsselworte hervorheben, Bullet-Listen verwenden, nur einen Gedanken pro Abschnitt formulieren).	MUSS	Da das Lesen am Bildschirm anstrengend ist, lesen viele Benutzer Webseiten nicht Wort für Wort, sondern überfliegen den Text nur auf der Suche nach relevanten Informationen [Ni97].	n/a	Es findet sich nahezu kein Fließtext auf den Seiten
/GUI/5/2/	Wichtige Informationen müssen zuerst genannt werden (sowohl in Fließtexten als auch bei der Gestaltung der Informationshierarchie)	MUSS	Wenn die Kerninformationen am Anfang stehen, muss der Benutzer nicht lange suchen, kann aber weiterlesen, wenn ihn Details und Hintergründe interessieren.	n/a	s.o.
/GUI/5/3/	Das Vokabular muss der Zielgruppe angepasst sein.	MUSS	Die Verwendung von zielgruppenfremdem Vokabular kann die Arbeit mit Funktionen wie Suche und Index massiv beeinträchtigen.	ja	Die PG hat sich bemüht, Versicherungsfachbegriffe zu verwenden
/GUI/5/4/	Die Benutzungsschnittstelle darf nicht beschrieben werden (z.B. nicht „Klicken Sie hier, um...“).	MUSS	Der Benutzer soll sich auf die Arbeit mit den Informationen konzentrieren und nicht durch Hinweise auf die Benutzungsschnittstelle abgelenkt werden – eine gute Schnittstelle sollte er ohne Nachzudenken „allein mit dem Kleinhirn“ bedienen können.		
/GUI/6/	Beim Einsatz von *Grafiken und Multimedia-Objekten* müssen folgende Regeln beachtet werden.	MUSS		--	--
/GUI/6/1/	Reine Schmuckbilder (Bullets, Linien und Hintergrundbilder) dürfen nicht verwendet werden.	INV	Um die Seite übersichtlich und die Ladezeiten gering zu halten, sollten nur Abbildungen mit Informationsgehalt verwendet werden. Hintergrundbilder können die Lesbarkeit von Texten stark reduzieren.	ja	
/GUI/6/2/	Animationen, Laufschriften und „brandneue“ Technologien dürfen nicht eingesetzt werden, wenn es alternative Standard-Darstellungsmöglichkeiten gibt.	INV	Auf einigen Plattformen können die Elemente möglicherweise	ja	Auf diese Technologien wurde konsequent ver-

			nicht dargestellt werden, so dass Informationen verloren gehen. Auf den anderen Plattformen kann die „effektvolle“ Darstellung die Arbeit mit der Information erschweren (Text in Laufschriften ist z.B. schwer lesbar, kopierbar, indizierbar, druckbar...).		zichtet. (s. Abschnitt GUI-Entwurf)
/GUI/6/3/	Nichttextuelle Objekte müssen mit Textbeschreibungen versehen werden.	MUSS	Der Inhalt von Grafiken und Multimedia-Objekten muss auch für Clients, die diese nicht darstellen können, erschließbar sein, damit keine Informationen verloren gehen.	ja	
/GUI/6/4/	Die Dateigröße von Grafiken und Multimedia-Objekten muss auf ein Maß reduziert werden, das in einem vernünftigen Verhältnis zur Darstellungsqualität steht.	MUSS	Die unvermeidbare Unterbrechung des Arbeitsablaufs durch das Laden von Webseiten sollte so kurz wie möglich gehalten werden, um effizientes Arbeiten zu ermöglichen.	ja	
/GUI/7/	Beim Setzen von *Links* müssen folgende Regeln beachtet werden.	MUSS		--	--
/GUI/7/1/	Links müssen im Fließtext sparsam eingesetzt werden.	MUSS	Zuviele Links in einem Textblock stören die Konzentration des Lesers, da er jedesmal den Lesefluss unterbrechen und entscheiden muss, ob er dem Link folgen soll.	n/a	Diese Anforderung ist nicht messbar.
/GUI/7/2/	Links müssen das verknüpfte Element möglichst exakt beschreiben. Text-Links müssen dazu prägnant formuliert sein, Grafik-Links müssen eindeutige Symbole verwenden.	MUSS	Der Benutzer muss wissen, was sich hinter dem Link verbirgt, um entscheiden zu können, ob das Verfolgen für ihn relevante Informationen liefert.	n/a	Diese Anforderung ist nicht messbar.
/GUI/7/3/	Links müssen eindeutig gekennzeichnet sein. Bei Text-Links geschieht dies durch besondere Farbgebung und Unterstreichung, bei Grafik-Links durch farbige Umrahmung oder Button-Optik.	MUSS	Sind Links nicht eindeutig erkennbar, kann der Benutzer sie nur noch im Trial-And-Error-Verfahren finden und nicht effizient mit der Site arbeiten.	ja	

/GUI/7/4/	Grafik-Links und insbesonders Image Maps müssen durch äquivalente Text-Links ergänzt werden.	MUSS	Der Möglichkeit zur Navigation muss auch bei Verwendung von Text-Clients erhalten bleiben.	ja	In der Portal-Site finden sich keine Image-Maps
/GUI/7/5/	„Gelinkte“ Seiten dürfen nicht in neuen Browser-Fenstern geöffnet werden.	INV	Im Vollbildmodus wird meistens nicht deutlich, dass sich ein neues Fenster geöffnet hat. Der Benutzer ist jedoch irritiert über den deaktivierten Back-Button und weiss nicht, wie er zur vorherigen Seite zurückkehren kann.	nein	Das CMS wird in einem neuen Fenster geöffnet. Bei Verwendung einer 1-Frame-Technik verbietet sich diese Anforderung aber auch.
/GUI/7/6/	Die Funktion des Back-Buttons muss garantiert sein.	MUSS	Benutzer verlassen sich darauf, dass diese „Rettungsleine“ sie immer auf bekanntes Gebiet zurückführt. Die Deaktivierung blockiert daher die am zweithäufigsten genutzte Navigationsfunktion [CaPi94] und verletzt die Grundsätze der Steuerbarkeit, Erwartungskonformität und Erlernbarkeit.	ja	Dies ist durch die Verwendung der 1-Frame-Technik sichergestellt.
/GUI/7/7/	Einmal publizierte Seiten dürfen, soweit möglich, nicht mehr verschoben werden.	INV	Durch Verschiebung einer internen Seite können Bookmarks und interne Links ins Leere zeigen.	n/a	
/GUI/7/8/	Links auf externe Seiten müssen regelmäßig überprüft werden.	MUSS	Durch Verschiebung einer externen Seite können die Links ins Leere zeigen.	n/a	Eine Auslieferung fand noch nicht statt
/GUI/8/	Beim Anlegen des *Navigationssystems* müssen folgende Regeln beachtet werden.	MUSS		--	--
/GUI/8/1/	Webseiten müssen hierarchisch strukturiert werden.	MUSS	Eine Hierarchie ist eine einfache und erweiterbare Struktur, die vom Benutzer leicht und unmittelbar als mentales Modell der Site übernommen werden kann.	ja	
/GUI/8/1/1/	Die Navigation innerhalb der Hierarchie muss auf allen Seiten in konsistenter Form möglich sein.	MUSS	Nur auf diese Weise kann der Benutzer die Seiten korrekt in die Hierarchie seines mentalen Modells einordnen. Zudem wird	ja	

			garantiert, dass der Benutzer alle Seiten direkt oder indirekt erreichen kann, auch wenn er mitten in die Site springt.		
/GUI/8/1/2/	Querverbindungen zwischen Elementen der Hierarchie müssen deutlich von der primären, hierarchischen Organisationsstruktur getrennt sein (z.B. Darstellung der Hierarchie in einer Navigationsleiste, Setzen von Querverbindungen im Fließtext)	MUSS	Andernfalls kann der Benutzer nicht erkennen, wie die Hierarchie strukturiert ist.	ja	
/GUI/8/1/3/	Die aktuelle Position des Benutzers in der Hierarchie muss jederzeit erkennbar sein.	MUSS	Nur so ist die Orientierung des Benutzers in der Site gewährleistet.	ja	Diese Information wird dem Benutzer in der Titelzeile gegeben.

6.9 Anforderungen Legacy

Neben den o.g. Anforderungen wurden weitere Features erfüllt, die zuvor nicht spezifiziert worden sind, die aber im weiteren Verlauf der PG für sinnvoll gehalten wurden.

Im Wesentlichen ist dies die komplette Funktionalität des Legacy-Subsystems. Dieses System bindet eine Partnerdatenbank beispielhaft an das Portalsystem an. Da die Anforderungen an dieses Subsystem erst im späteren Verlauf des Projektes gestellt wurden, sind keine Tabellen wie in den vorangegangenen Kapiteln erstellt worden. Statt dessen wurden die Anforderungen formlos dargestellt. Diese Anforderungen sind – gegliedert in funktionale und nichtfunktionale Anforderungen – nachfolgend beschrieben. Die genauen Funktionen dieses Systems werden in Abschnitt 7.8 beschrieben.

6.9.1 Funktionale Anforderungen

1. Der VAD muss sich zu einem bestimmten Kunden die wichtigsten (harten) Daten aus der Partnerdatenbank der Versicherung anzeigen lassen können. Dazu gehören die allgemeinen Personendaten, die technischen Daten, die Kontoverbindung und die Vertragsbeziehungen.

2. Der VAD muss sich zu einem bestimmten Kunden das Geschichtsbuch aus der Partnerdatenbank der Versicherung anzeigen lassen können. Zum Geschichtsbuch gehören die Versicherungsnummer, das Datum, die Referenz, der Grund für einen Eintrag im Geschichtsbuch, der Sachbearbeiter, usw. Die Vertragsumstellungen müssen ersichtlich sein.

3. Der VAD muss nach Einträgen/Daten in der Partnerdatenbank suchen können, z.B. Name, Vorname, Stadt, usw.

4. Der VAD muss die Möglichkeit haben, ausgehend von den harten Daten eines Kunden, die weichen Daten einzusehen. Dies allerdings nur insofern er für diesen Kunden zuständig ist und diese bei ihm lokal abfragbar sind.

5. Der Benutzer sollte die Möglichkeit haben folgende Abfragen über die Partnerdatenbank zu starten und somit Statistiken zu erhalten:

   - QueryByEnteredDate - Anfrage: Welche Kunden sind seit dem „dd.MM.yyyy“ hinzugefügt worden?
   - QueryByPostalCode - Anfrage: Geographische Verteilung der Kunden
   - QueryByContractCategory - Anfrage: Wie viele Kunden/Verträge sind in der Datenbank mit welchen Vertragskategorien gespeichert?
   - QueryByTimePeriod - Anfrage: Wie viele neue Kunden haben wie viele Verträge zwischen Start- und Enddatum abgeschlossen?

6.9.2 Nichtfunktionale Anforderungen

6. Die Daten in der Partnerdatenbank müssen vor unerlaubtem Lesen geschützt werden. D.h. ein VAD darf nur seine ihm zugeordneten Daten einsehen. Analog darf ein Agenturleiter nur die Daten seiner VADs einsehen. Das Versicherungsunternehmen hat uneingeschränkten Zugriff.

7. Es erfolgt nur ein lesender Zugriff auf das Subsystem Legacy.

7 Entwurf und Systemarchitektur

Nachdem die Anforderungen an das Portal gefunden waren, galt es eine Systemarchitektur zu finden, die diese Anforderungen auch technisch umsetzen konnte. Die Systemarchitektur musste so offen gestalten werden, dass es möglich war, jedes identifizierte Subsystem zu integrieren. Als Schwierigkeit erwies sich die Wahl der Software für jedes Subsystem. Zusätzlich musste es auch möglich sein, dass alle Subsysteme untereinander kommunizieren können.

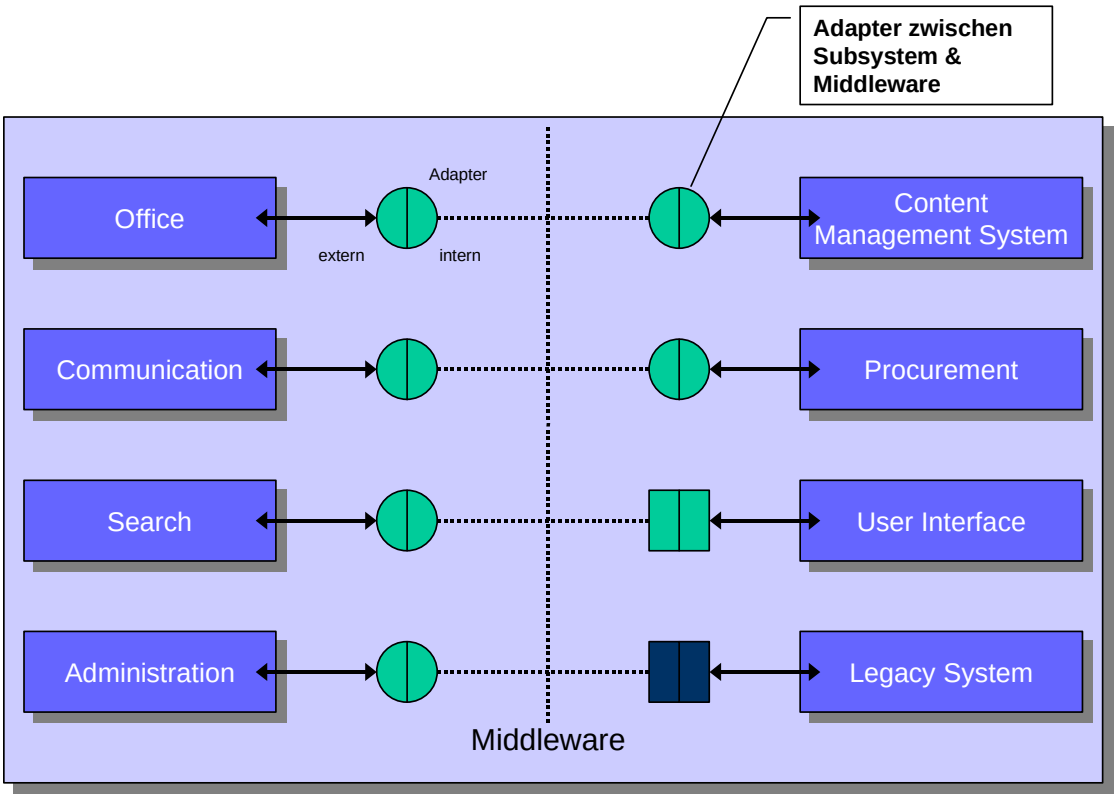


Abbildung 14 - Systemarchitektur

Das IPSI-Portal integriert verschiedene Anwendungen und Dienste miteinander. Dieser integrative Aspekt schlägt sich auch in der Architektur des Systems wider. Jeder integrierten Anwendung bzw. Dienstleistung entspricht ein Subsystem in der Systemarchitektur. Diese Subsysteme sind in Abbildung 14 - Systemarchitektur dargestellt.

Dabei kapselt jedes Subsystem alle Funktionalitäten, die nötig sind, um die entsprechenden Funktionalitäten (die in den Anforderungen für dieses Subsystem festgehalten sind) zu realisieren.

Das Office-Subsystem bietet sämtliche Dienstleistungen, die mit Terminen, Kontakten und Emails zu tun haben.

Das Shop-Subsystem ist für sämtliche Shop-Aktionen zuständig. Alle Abläufe des User-Managements sind im Administration-Subsystem zusammengefasst. Im Legacy-Subsystem sind die Funktionalitäten einer bestehenden Partner-Datenbank gekapselt. Das CMS-Subsystem ist für die Interaktionen mit dem Contentmanagementsystem zuständig und das Search-Subsystem dient der Subsystemübergreifenden Suche.

Die Architektur der einzelnen Subsysteme ist Gegenstand separater Unterkapitel dieses Kapitels.

Die Subsysteme sind die passiven Elemente des Systems, d.h. sie reagieren lediglich auf Aufrufe durch aktive Systemelemente, die den in Kapitel 3 dargestellten fachlichen Workflow in koordinierte Aufrufe auf den hiermit involvierten Subsystemen umsetzen. Diese aktiven Elemente des Systems sind die sog. Controller (sie kontrollieren die Abläufe im System). Controller werden über einen Aufruf des Benutzers im User-Interface aufgerufen und benutzen eine Reihe von Subsystemen, die nötig sind, um die jeweilige Anfrage zu beantworten. Ein Controller ist dabei zuständig für einen UseCase.

Die Controller kommunizieren dabei mit den Subsystemen nur über eine dünne Schnittstelle, die in Form einer sogenannten Fassaden-Klasse realisiert ist (hier wurde das Facade Entwurfsmuster verwendet [GaHe96]).

D.h. die Fassaden-Klasse eines Subsystems kapselt alle Dienstleistungen (in Form von Methoden), die dieses Subsystem nach außen (den Controllern) zur Verfügung stellt. Dies hat zur Folge, dass die interne Struktur des Subsystems nach außen versteckt wird (Prinzip des information hiding) und somit eine geringe Kopplung besteht. Diese Fassadenklassen sind in Abbildung 14 - Systemarchitektur als Adapter dargestellt. Die Adapter sind zweiteilig, da sie zum einen als Schnittstelle zu den Controllern dienen, zum andern als Kanal zu den einzelnen Subsystemen und den dahinterliegenden Software-Systemen.

Die systemtechnische Architektur sieht eine Verteilung der einzelnen Systemelemente vor. D.h. dass sich jedes der Subsysteme zur Ausführungszeit in einem eigenen Adressraum befindet. Der Grund für diese Anforderung ist die bessere Skalierbarkeit des Gesamtsystems: bei Ansteigen des Benutzervolumens können die Subsysteme einfach auf andere Rechner verteilt werden, wodurch es zu einer Lastverteilung kommt.

Diese erhöhte Flexibilität erkauft man sich durch eine steigende Komplexität in der Entwicklung. Das tritt insbesondere bei der Einbindung und Berücksichtigung einer Middleware als Substrat für verteilte System zu Tage. Die Middleware stellt eine Kommunikationsinfrastruktur für die Komponenten eines verteilten Systems zur Verfügung.

Da sich die Projektgruppe frühzeitig für die Programmiersprache JAVA entschied (Gründe hierfür waren die Affinität zu verteilten Systemen durch die umfangreiche objektorientierte Klassenbibliothek [DoDi99], die freie Verfügbarkeit, sowie die Zukunftsträchtigkeit dieser Technologie über die PG hinaus), standen als Middleware-Produkte CORBA und RMI zur Auswahl.

Die Erstellung eines Middleware-Prototypen (s. Kapitel 4.2.6.1) gab Aussage über die generelle Realisierbarkeit und die Handhabung der Middleware. Schließlich entschied sich das Performanzkriterium (RMI-Aufrufe sind fast doppelt so schnell wie CORBA-Aufrufe) für RMI als zu verwendende Middleware.

Die Basis der dargelegten Systemarchitektur ist in Form eines Frameworks realisiert worden. Dieses Framework bietet eine Grundlage für die Erstellung der Subsysteme und der Controller. Des weiteren stellt es geeignete Systemelemente zur Verfügung, um die Kommunikation zwischen diesen Elementen und die schließlich notwendige Anbindung an das User Interface zu ermöglichen. Dieses Framework ist das sogenannte Core-System und wird ebenfalls in einem der folgenden Abschnitte im Detail dargestellt.

Hiermit ist ein grober Überblick über die oberste Abstraktionsebene der Systemarchitektur gegeben. Nach der Beschreibung der für die einzelnen Subsysteme verwendeten Software sollen die einzelnen Systemelemente (die passiven Subsysteme und die aktiven Systemelemente) in den folgenden Abschnitten detaillierter dargestellt werden.

7.1 Eingesetzte Software der Subsysteme

Im folgenden werden die für die Subsysteme eingesetzten Systeme erläutert. Diese sind in Abbildung 15 dargestellt.

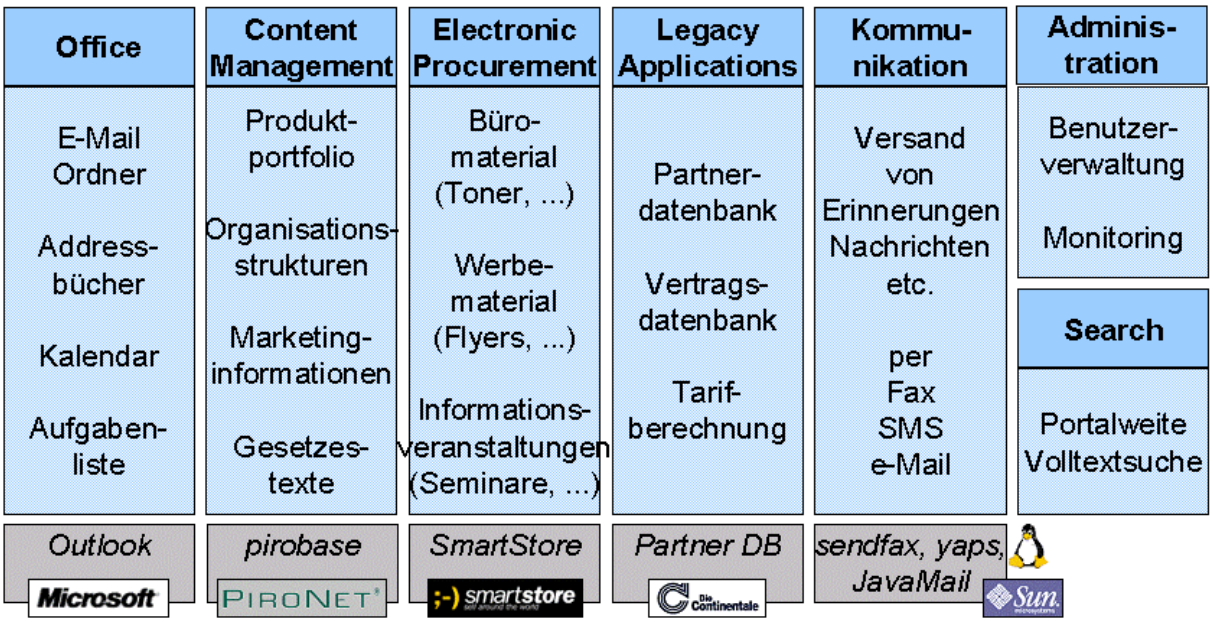


Abbildung 15 – Mapping fachlicher und technischer Systeme

- *Office*
  Als Software für das Subsystem Office haben wir uns für das Produkt Outlook 2000 der Firma Microsoft entschieden, da es alle Eigenschaften, die wir durch unsere Anforderungen definiert haben, zur Verfügung stellt [Mi00]. Outlook ist am Markt akzeptiert und sehr weit verbreitet. Es kann sowohl Emails (für die Kommunikation) als auch Termine (für die Terminverwaltung) und Kontakte (für das Adressbuch) verwalten. Eine JAVA Eigenentwicklung wurde zwar gleichwertig von uns gewichtet, allerdings würde der Aufwand nicht in Relation zu dem Nutzen stehen. Man hätte in dem vordefinierten Zeitraum einer Projektgruppe ein Produkt wie Outlook nicht selbst implementieren können. Aus diesem Grund wollen wir unsere Energie in die Integration stecken, anstatt in die Softwarewareentwicklung eines Produktes, das es auf dem Markt schon gibt.
- *CMS*
  Das Contentmanagementsystem (CMS) Pirobase 4 kommt von der Firma Pironet [Pi00]. Mit dieser Software ist es möglich, Dokumente mit dem Browser in ein System einzustellen und die komplette Dokumentenverwaltung vorzunehmen. Durch Schulungen und die sehr guten Kontakte zu Pironet stand der Einsatz dieser Software nie in Frage. Pirobase 4 läßt sich durch die offene Architektur ideal in unser Portal integrieren. Es ist in JAVA programmiert und realisiert die Kommunikation der Komponenten innerhalb von Pirobase mit CORBA [Su00]. Da CORBA für die PG ebenfalls eine Alternative (oder auch zusätzliche Möglichkeit zur Anbindung des CMS parallel zu RMI) bei der Wahl der Middleware darstellt, ist eine Integration durch die offenen Schnittstellen von CORBA in das Portal hinein einfacher zu bewerkstelligen, als wenn man ein abgeschlossenes System eingesetzt hätte.
- *Procurement*
  Als Procurement bzw. Shopsystem haben wir SmartStore der Firma SmartStore AG eingesetzt [Sm00]. Eine Prüfung der Vor- und Nachteile hat ergeben, dass SmartStore das beste Produkt für die Anforderungen der Projektgruppe bietet. Als Alternativen wurden der Microsoft Site Server und eine JAVA Eigenentwicklung untersucht [Mi00]. Die Eigenentwicklung schied auf Grund des immensen Aufwandes von vornherein aus. Der Site Server hatte zwar einige Vorteile, jedoch ist SmartStore in den entscheidenden Punkten (z.B. geringere Komplexität, Änderbarkeit und weniger hohe Hardwareanforderungen) besser nutzbar.
- *Communication*
  Das Subsystem Communication kann auf bestehende LINUX Freeware Software zurückgreifen [Su00b]. D.h. als Emaildienst wird ein SMTP Server (z.B. basierend auf Sendmail) und als Faxdienst Sendfax eingesetzt. Sendfax greift dabei auf Ghostscript und TEX zurück, um die G3-Vorlage für ein Fax zu generieren. Die Software zum Versenden von SMS heißt yaps (Yet-Another-Pager-Software) und ist in der Lage, an nahezu alle Mobilfunk- und Pager-Anbieter Nachrichten zu verschicken. Alle Produkte sind hinreichend getestet und werden so oder so ähnlich bereits eingesetzt. Für das Zusammenspiel dieser Dienste werden wir JAVA Programme selbst schreiben.
- *Administration*
  Die Administration bzw. das Reporting und das Monitoring setzt auf Freeware JAVA Applets zur Erstellung

von graphischen Managementauswertungen [De00a]. Dabei werden die JAVA Applets mit Parametern aufgerufen und stellen die Ergebnisse im Browser dar. Für das Zusammenspiel und die notwendigen Anpassungen werden wir ebenfalls eigene JAVA Programme schreiben.

- *Legacy-System*
  Auf das Legacy-System wird erst später genauer eingegangen. Es ist nicht geplant, ein eigenes System zu entwerfen, sondern exemplarisch ein bestehendes System einer Versicherung an unser Portal anzubinden, wie z.B. eine Partnerdatenbank oder eine Tarifberechnung.
- *User Interface*
  Das User Interface stellt eine Verallgemeinerung des Graphical-User-Interface (GUI) und z.B. einem Wireless-Application-Protocol Interface (WAP) dar. Es soll verschiedenste Dienste zur Eingabe und Ausgabe von Daten zur Verfügung stellen. Das User Interface muss eine Trennung zwischen Präsentations- und Applikationslogik bieten, um die Benutzungsoberfläche von den Subsystemen unabhängiger zu machen. D.h. es sollte keine hart implementierte Verbindung zwischen diesen beiden Ebenen geben. Dies hat den Vorteil, dass z.B. die GUI eine rein funktionale Sicht auf die Subsysteme hat und keine Methodennamen direkt kennt. Lediglich müssen die Region (z.B. SearchRegion), die Action (z.B. „Search“) und der Content (z.B. „Meier“) über die GUI bekannt sein, aber nicht, wie auf eine Subsystemmethode verzweigt wird.

Im folgenden werden die Teilarchitekturen der jeweiligen Subsysteme als Ergebnis des Feinentwurfs beschrieben.

Das Vorgehen in der Entwurfsphase ist in Kapitel 4.2.5 Feinentwurf dargestellt. Verwendet wurde die Standardsprache zur Beschreibung von objektorientierten Systemmodellen, UML [OeHr99].

7.2 Das Core-System

Das Core-System realisiert das Grundgerüst der in den vorangegangenen Abschnitten erwähnten Architektur, indem es ein Framework für die Entwicklung der Subsysteme, Controller und Formatter bietet.

Ziele bei der Entwicklung des Core-Systems waren die folgenden:

- Unabhängigkeit von Ein- und Ausgabemedien (HTML, WML, ...)
- möglichst weitgehende Trennung der Anwendungslogik von der Präsentationslogik
- Kapselung von Middleware-Details vor den Controllern

Insgesamt sollte also eine wiederverwendbare Plattform entstehen, die eine möglichst einfache Erweiterbarkeit durch neue Controller gestattet.

Im folgenden wird die Statik und die Dynamik des Core-Systems (also die Architektur und die Funktionsweise) beschrieben, um aufzuzeigen, wie die o.g. Ziele umgesetzt wurden.

Die Architektur des Core-Systems in Form eines UML-Klassendiagramms ist in Abbildung 16 - Klassendiagramm Core dargestellt.

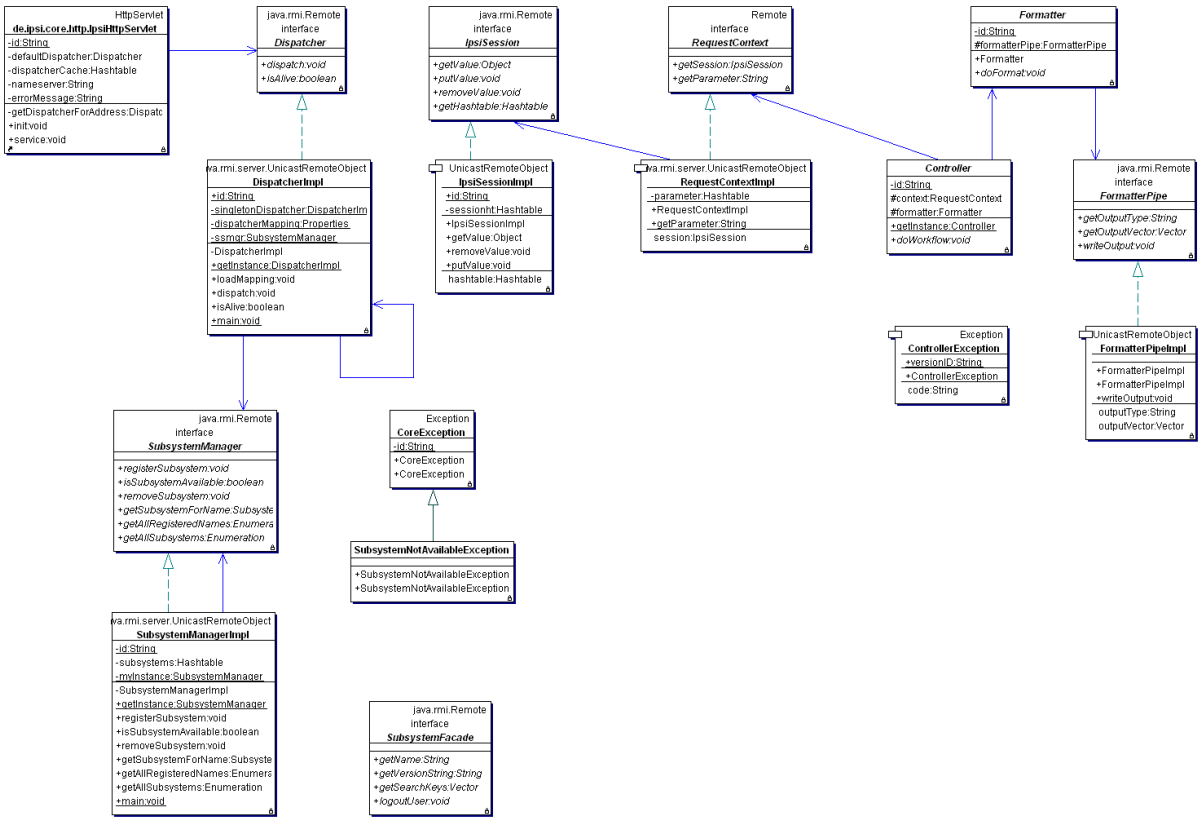


Abbildung 16 - Klassendiagramm Core

Zur Laufzeit werden eine Reihe von Schritten durchlaufen, die die Benutzung des Core-Systems verdeutlichen:

1. Der Subsystemmanager wird als eigenständig ausführbares Programm gestartet. Er dient als Registrierungszentrale für die einzelnen Subsysteme, die Controller bei ihm nachfragen können, um spezifische Methoden auf ihnen aufzurufen. Hierzu gibt der Subsystemmanager ein Objekt vom Typ Subsystemmanager über RMI frei.

2. Der Dispatcher wird ebenfalls als eigener Prozess ausgeführt. Er ist für das Instanziieren der für eine bestimmte Anfrage passenden Controller und Formatter und das Anstoßen der Bearbeitung der Anfrage durch die Controller zuständig. Er fragt daher das Subsystemmanager-Objekt über RMI nach.

3. Die einzelnen Subsysteme sind selber ausführbare Programme und werden daher auch getrennt gestartet. In einem ersten Schritt erzeugt jedes Subsystem ein Objekt vom Typ der subsystemspezifischen Subsystem-Fassade und meldet diese beim entfernten Subsystemmanager-Objekt an.

Bei einer Anfrage eines Benutzers, z.B. um sich bestimmte Artikel des Shops anzusehen, wird das folgende Szenario durchlaufen:

1. Die Anfrage wird über ein medienspezifisches Objekt entgegengenommen. Im Falle von HTTP-Anfragen über das WWW handelt es sich hierbei um ein JAVA-Servlet. Es sei bemerkt, dass nahezu alle anderen Bestandteile des Portals unabhängig davon sind, ob ein Servlet oder ein anderes Objekt die Anfrage entgegengenommen hat. Dieses Servlet erfragt nun den entfernten Subsystemmanager und den entfernten Dispatcher über die RMI Middleware und beauftragt den Dispatcher damit, die Anfrage weiter zu bearbeiten. Zuvor werden noch alle medienspezifischen Inhalte wie die Parameter der HTTP-Anfrage in eine medienunabhängige Form, den RequestContext verpackt. Ebenso wird ein medienunabhängiges Objekt zur Steuerung der Ausgabe (FormatterPipe) erzeugt. Diese beiden Objekte zusammen mit dem Subsystemmanager werden an den Dispatcher geleitet.

2. Der Dispatcher übernimmt die Bearbeitung des Aufrufes. Dazu ermittelt er in einem ersten Schritt den Controller samt Formatter, der für die spezifische Anfrage zuständig ist. Dies geschieht über ein sogenanntes

Dispatchermapping, das separat vom Dispatcher, z.B. in einer Textdatei oder Datenbank hinterlegt ist. Um die Zuordnung von Anfragen zu Controller/Formatter zu realisieren, muss eine Anfrage einem bestimmten Format entsprechen, indem es Parameter enthält, die den Aufruf kennzeichnen. Im HTTP-Beispiel müssen HTTP-Parameter „Region“ und „Action“ enthalten sein. Ein typischer HTTP-Aufruf sähe also z.B. so aus:
http://www.ipsi.de/servlets/ipsi?region=shop&action=getAllArticles
Wenn der Dispatcher so den Namen der Controller- und Formatter-Klasse ermittelt hat, werden über das JAVA-Reflection API [Fl96, GoJo97] den Klassennamen entsprechende Objekte erzeugt. Der instanziierte Controller wird nun mit der Abarbeitung der Anfrage betraut. Dazu wird ihm das Subsystemmanager-Objekt, die FormatterPipe (die dem Dispatcher beide vom Servlet übergeben wurden) sowie der gerade ermittelte Formatter übergeben.

3. Im nächsten Schritt arbeitet der Controller den Aufruf ab. Controller sind Klassen, die für die Bearbeitung genau einer Art von Anfrage zuständig sind. Sie enthalten die gesamte Anwendungslogik, die für die Abarbeitung dieser Art von Anfrage nötig ist, nicht jedoch die Präsentationslogik zur Darstellung des Ergebnisses der Anfragebearbeitung (diese ist in den Formattern bzw. nachgelagerten Klassen) enthalten.
   Um ihre Aufgabe zu erledigen, müssen die Controller i.d.R. verschiedene Subsysteme ansprechen, die die Funktionalitäten der einzelnen Backend-Systeme kapseln. Die benötigten Subsystem-Fassade-Objekte, über die die Kommunikation mit den Subsystemen abläuft (die Subsystem-Fassaden sind also sozusagen die oben beschriebenen Adaptoren), können vom Controller über den ihm übermittelten SubsystemManager nachgefragt werden. Die Kommunikation mit den Subsystem-Fassaden findet ausschließlich über sog. Business-Objekte statt, die Entitäten der Subsysteme darstellen (z.B. Artikel, Benutzer, Nachrichten, etc.). Nachdem alle Subsysteme kontaktiert wurden und diese die benötigten Ergebnisse (z.B. Artikel) in Form von Business-Objekten an die Controller geliefert haben, wird der dem Controller durch den Dispatcher übermittelte Formatter mit der Aufbereitung dieser Ergebnisse für das entsprechende Ausgabemedium beauftragt. Dabei werden ihm alle benötigten Ergebnisse und Steuerinformationen als Parameter übermittelt.

4. Ein Formatter ist für die medienspezifische Aufbereitung der Ergebnisse für eine Menge von Controllern zuständig. Er wertet die Steuerinformationen des Controllers aus und kann so feststellen, was genau darzustellen ist. Allerdings nimmt nicht der Formatter selbst die grafische Aufbereitung der Ergebnisse vor, sondern beauftragt sog. Maskenobjekte damit, dies zu tun. Welches Maskenobjekt nun benutzt werden muss, ist hart im Formatter kodiert und wird über die Controller-Steuerinformationen festgelegt. Die Maskenobjekte wiederum nutzen die ausgabespezifischen GUI-Bibliotheken (GUILib für komplexere Darstellungseinheiten wie z.B. spezielle Tabellen und GUIBase für atomare Darstellungseinheiten wie Textfelder oder Buttons). Die so in den Maskenklassen generierte Ausgabe wird dann den gesamten Weg zurück an das Servlet geleitet und dort an die Anfrage zurückgegeben, wo die Ergebnisse in dem jeweiligen Ausgabetyp dargestellt werden kann (im Browser im HTML-Fall).

7.3 Subsystem Office

Das Subsystem Office integriert MS Outlook in die Benutzungsoberfläche des IPSI-Portals. Dabei werden die Bereiche Termine, Aufgaben, Kontakte und eingeschränkt Emails unterstützt. Für Termine, Aufgaben und Kontakte werden Einfügen-, Ändern-, Löschen- und Suchen-Funktionalitäten angeboten. Emails können nur gesucht und angezeigt werden. Dabei werden nur die Emails des Posteingangs-(Inbox-) Ordners verwendet.

MS Outlook kann mit Hilfe der OLE-Technologie automatisiert werden. Allerdings besteht in JAVA keine direkte Möglichkeit, OLE zu verwenden (eine Ausnahme ist MS Visual J++, das aber nicht dem JAVA-Standard entspricht und somit nicht von allen JVMs unterstützt wird). Abhilfe verschafft das JAVA Native Interface (JNI), mit dem aus JAVA heraus C- bzw. C++-Bibliotheken verwendet werden können [Su97, St98]. Die Funktionen, die über diese Bibliotheken exportiert werden, müssen einer bestimmten JAVA-spezifischen Syntax entsprechen, so dass nicht einfach beliebige Bibliotheken verwendet werden können. Es muss vielmehr eine Wrapper-Schicht erstellt werden, die die Verbindung zwischen JAVA und OLE herstellt (s. Abbildung 17 - Zugriff von JAVA auf MS Outlook). In dieser Wrapper-Schicht wird der Zugriff auf die OLE-Funktionalitäten mit Unterstützung der Microsoft Foundation Classes (MFC) hergestellt.

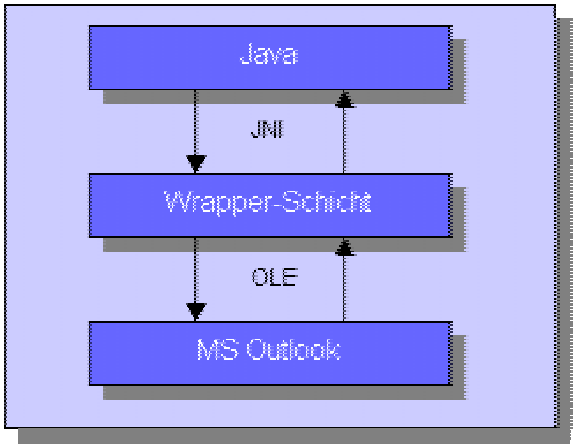


Abbildung 17 - Zugriff von JAVA auf MS Outlook

Der Zugriff auf das Subsystem Office wird einheitlich über die Fassade des Subsystems durchgeführt. Diese Fassade, repräsentiert durch die Klasse *OfficeSubsystemFacade*, wird lokal auf dem Client gestartet. Das bedeutet, dass jeder Benutzer sein eigenes Office-Subsystem hat, denn schließlich wird auch das lokal installierte MS Outlook verwendet. Die Fassade meldet sich beim Subsystem-Manager mit dem Namen des Benutzers an. Über den Subsystem-Manager kann dann jederzeit die Office-Subsystem-Fassade eines Benutzers ermittelt und verwendet werden.

Die Klasse OfficeSubsystemFassade ist die Schnittstelle zwischen dem Office-Subsystem und dem IPSI-Portal. Deshalb führt sie die Operationen nicht selbst aus, sondern reicht sie vielmehr an die Business-Objekte bzw. an das Objekt der Klasse Outlook weiter.

Die Klasse Outlook ist als Singleton realisiert, d.h. es gibt nur ein Objekt dieser Klasse. Sie enthält Methoden zum An- und Abmelden von MS Outlook, sowie die Suchmethoden für Termine, Kontakte, Aufgaben und E-mails. Des weiteren wird eine Methode zur Verfügung gestellt, über die eine Email in einem MS-Outlook-Fenster angezeigt werden kann.

Alle diese Methoden werden in der Klasse Outlook nur deklariert, um schließlich in der Wrapper-Schicht implementiert zu werden.

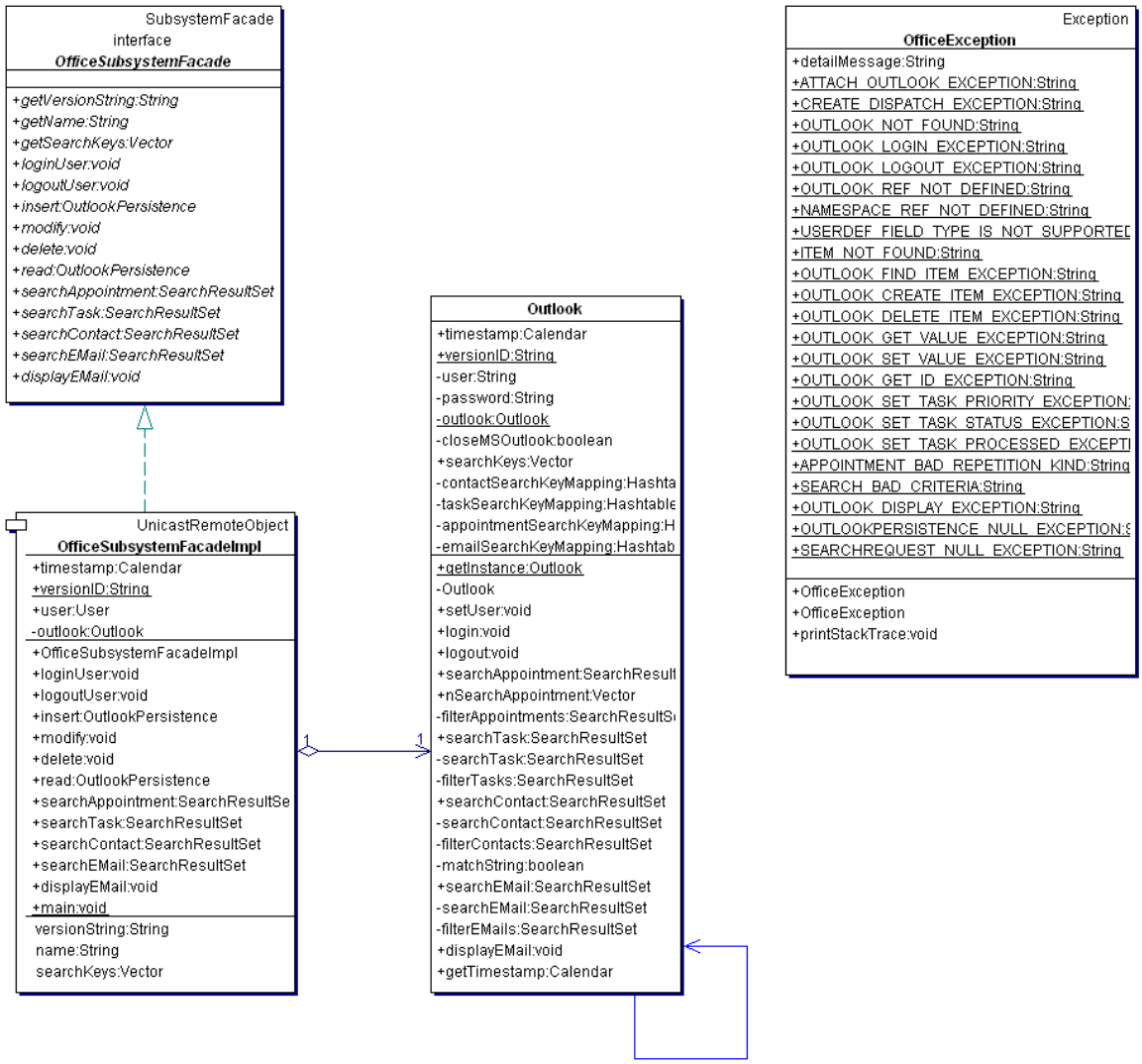


Abbildung 18 - Klassendiagramm der Office-Subsystem-Fassade

Wenn z.B. nach einem Kontakt gesucht werden soll, wird vom IPSI-Portal aus die entsprechende Methode der Fassade aufgerufen. Diese Methode ruft wiederum die entsprechende Methode in der Klasse Outlook auf, die über JNI in der Wrapper-Schicht realisiert ist. Sie verwendet die entsprechenden MS Outlook-Funktionen, die über OLE zur Verfügung gestellt werden, um die gesuchten Kontakte zu finden. Diese Kontakte werden innerhalb der Wrapper-Schicht in JAVA-Objekte der Klasse Contact gepackt, so dass die Kontakt-Objekte im Portal genutzt werden können (s. Abbildung 19 - Beispiel: Suche nach Kontakten).

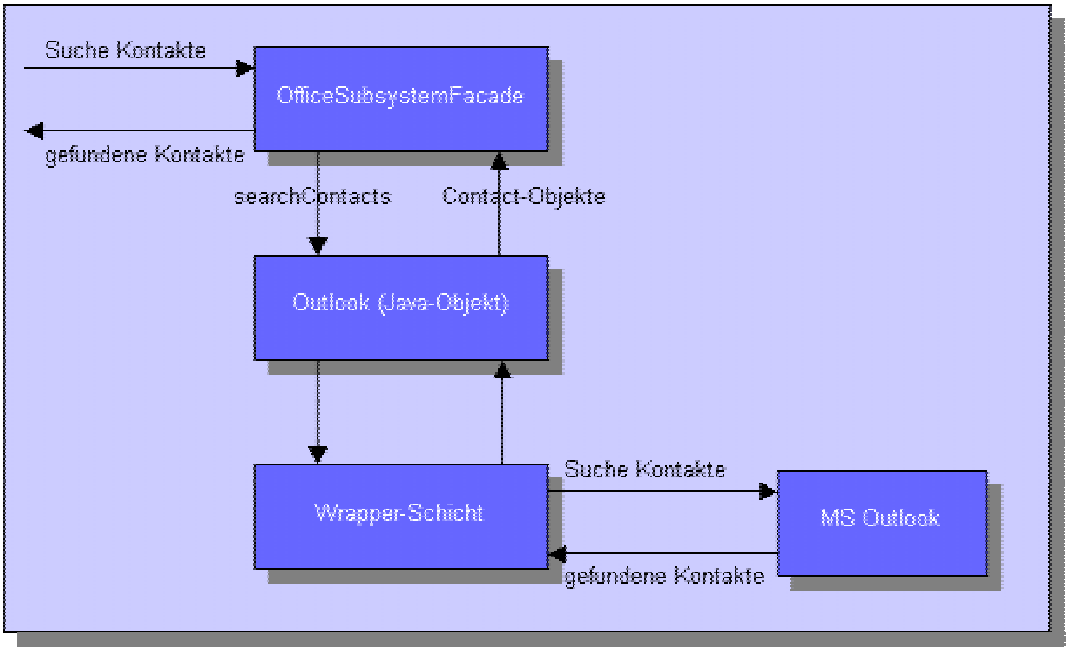


Abbildung 19 - Beispiel: Suche nach Kontakten

Die Einfügen-, Ändern- und Löschen-Operationen werden durch die Business-Objekt-Klassen zur Verfügung gestellt. Damit ein einheitlicher Zugriff auf diese Operationen stattfinden kann, wurde eine abstrakte Klasse OutlookPersistence eingeführt. Diese Klasse stellt zwar die Operationen zur Verfügung, implementiert sie aber nicht. Die Implementierung der Operationen finden ausschließlich in den Subklassen, den Business-Objekt-Klassen, statt. Diese Business-Objekt-Klassen deklarieren ebenfalls, wie die Klasse Outlook, nur die Methoden, während sie in der Wrapper-Schicht implementiert werden.

Analog zu den vier unterstützen Bereichen von MS Outlook gibt es vier verschiedene Typen von Business-Objekten, das sind Appointment, Task, Contact und EmailHeader (s. Abbildung 20 - Klassendiagramm Office-Business-Objekte).

Diese Klassen enthalten ausser diesen Zugriffsoperationen noch die Attribute der Business-Objekte. Dabei handelt es sich lediglich um eine Auswahl der durch MS Outlook zur Verfügung gestellten Eigenschaften. Eine Unterstützung aller Eigenschaften, die MS Outlook zur Verfügung stellt, wäre viel zu aufwendig gewesen und auch nicht wünschenswert. Der Benutzer kann schließlich immer noch diese Eigenschaften in MS Outlook selbst ergänzen / modifizieren.

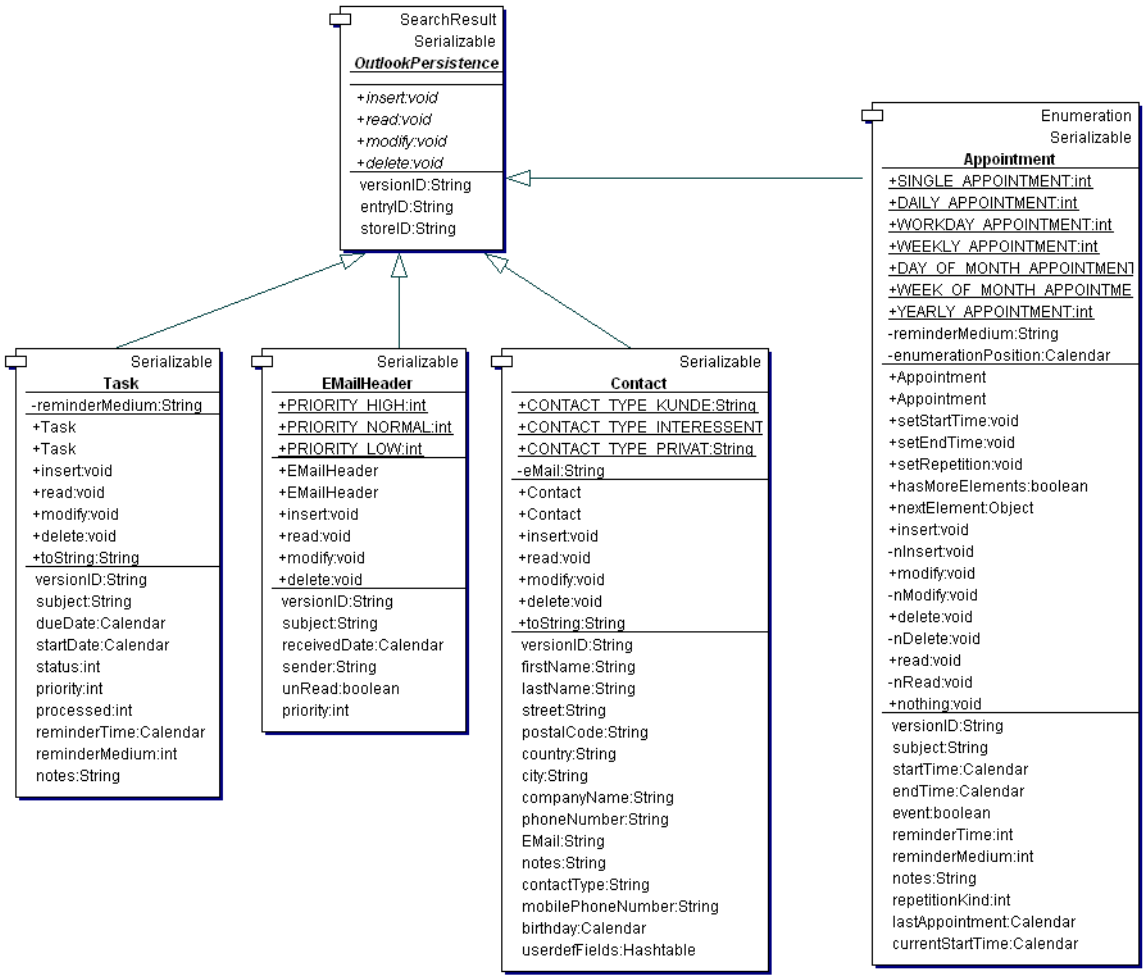


Abbildung 20 - Klassendiagramm Office-Business-Objekte

Wenn Fehler bei Operationen des Subsystems auftreten, werden OfficeExceptions geworfen.

Im Gegensatz zum initialen Design des Systems sind die Klassen AppointmentRepeater und UserdefField entfernt worden. Die Klasse AppointmentRepeater ging in die die Klasse Appointment über, da eine Trennung dieser beiden Klassen aus implementierungstechnischer Sicht keinen Sinn machte. Die Klasse UserdefField wurde gestrichen, da die benutzerdefinierten Felder eines Kontakts besser in einer Hashtable gespeichert werden konnten, wobei der Schlüssel gleich dem Namen des benutzerdefinierten Feldes ist und der Wert dem Wert des benutzerdefinierten Feldes entspricht. In der vorliegenden Realisierung muss dieser Wert eine Zeichenkette sein, aber es sollte ohne erheblichen Aufwand möglich sein, auch Zahlen und Datumswerte zuzulassen.

7.4 Subsystem Administration

Es gibt vier zentrale Klassen im Entwurf des Subsystems Administration: das Buisness-Objekt User, das einen Benutzer des Portals repräsentiert, das Business-Objekt Agency, das eine Agentur im Versicherungsunternehmen abbildet, die Fassadenklasse AdminSubsystemFacade und die Klasse PortalUserDB, die als einzige Klasse Kontakt zur Datenbank hat. Im folgenden werden diese vier Klassen näher betrachtet. Um die Weitergabe mittels RMI zu ermöglichen, gehört zu jedem Buisness-Objekt ein Interface XXX und eine implementierende Klasse XXXImpl. Die Interfaces erben von java.rmi.Remote, die Impl-Klassen erben von java.rmi.UnicastRemoteObject.

7.4.1 User

Das Interface User wird von der Klasse UserImpl implementiert. Diese Klasse repräsentiert einen Benutzer des Portals. Es wurde darauf geachtet, den Entwurf so flexibel wie möglich zu gestalten. Deshalb besitzt ein User eine Hashtable Account mit mehreren *Account-Objekten*. Ein Account-Objekt steht dabei für den Account eines Subsystems; die Schlüssel, unter denen die Accounts in der Hashtable abgelegt sind, bestimmen, für welchen Teilbereich die jeweiligen Accounts gelten. Diese Schlüssel werden als Konstanten in der Klasse Constants abgelegt; in diesem Fall wurden vier Schlüssel eingetragen: Portal, Shop, CMS und Office. Ein Account-Objekt besitzt ein Password und ein Login.

Ein ähnliches Prinzip wurde für die Rechte eines Benutzers gewählt. Ein User besitzt eine Hashtable Right mit einem oder mehreren *Right-Objekten*. Ein Right-Objekt repräsentiert ein bestimmtes Recht; das Schlüsselwort dieses Rechts ist als Keyword im Right-Objekt abgelegt. Außerdem ist dieses Schlüsselwort auch zugleich der Schlüssel, unter dem das Recht in der Hashtable abgelegt ist. Ein Benutzer könnte eine unendliche Anzahl von Rechten besitzen; das dazugehörige Rechtesystem kann also so flexibel wie gewünscht gestaltet werden. Der Einfachheit halber wurde hier einem Benutzer aber nur ein Recht gegeben: entweder VAD, Agenturleiter oder VU. Diese Rechte repräsentieren also die *Rolle* des Benutzers im Versicherungsunternehmen.

Mit den Interessen eines Benutzers verhält es sich genauso. Ein User hat eine Hashtable Interest mit mehreren *Interest-Objekten*. Sie werden benötigt, um im Startbildschirm die Artikel darzustellen, die den Benutzer möglicherweise interessieren. Das Schlüsselwort des Interesses ist als Keyword im Interest-Objekt abgelegt; zugleich ist dieses Schlüsselwort der Schlüssel, unter dem das Interesse in der Hashtable abgelegt ist. Auch hier kann ein User eine beliebige Anzahl von Interessen besitzen.

Die *Adressdaten* eines Benutzers werden in der Klasse Address abgelegt. Zu den Adressdaten gehören Name des Users, seine Anschrift sowie Telefon-, Fax-, Mobilfunknummer und Emailadresse. Ein User besitzt genau ein Address-Objekt.

Kunden eines VADs haben die Möglichkeit, über das Internet ihre Meinung über den VAD kundzutun. Diese Meinung wird in der Klasse Feedback gespeichert. Ein *Feedback* besteht aus einer Bewertung von 1 bis 6 (Note), einem Kommentar (Comment), dem Datum (Date) und dem User-Objekt, das den VAD repräsentiert (Victim). Man beachte, dass das Feedback-Objekt den User kennt, nicht aber umgekehrt. Die Feedbacks werden also nicht durch das User-Objekt ermittelt, sondern von der Fassadenklasse. Jeder Benutzer ist in einer *Agentur* beschäftigt. Diese Agentur wird durch die Klasse Agency repräsentiert. Ein User-Objekt ist genau einem Agency-Objekt zugeordnet.

Es wurde die Klasse UserData entwickelt, um das Eintragen eines neuen Benutzers in die Datenbank zu erleichtern. Diese Klasse beinhaltet alle Daten des Benutzers; sie bietet Methoden, die Daten zu setzen oder auszulesen. Gleichzeitig hat die Klasse UserImpl einen Konstruktor, der ein UserData-Objekt verlangt. Man beachte, dass die Klasse UserImpl keine set-Methoden bietet, um die Daten nachträglich zu manipulieren. Die einzige Möglichkeit, die Daten zu setzen, ist eben jener Konstruktor.

Die beschriebenen Kernklassen und deren Beziehungen sind in Abbildung 21 - User und Agency dargestellt. Abbildung 22 - UserData und AgencyData enthält die erwähnten Hilfsklassen.

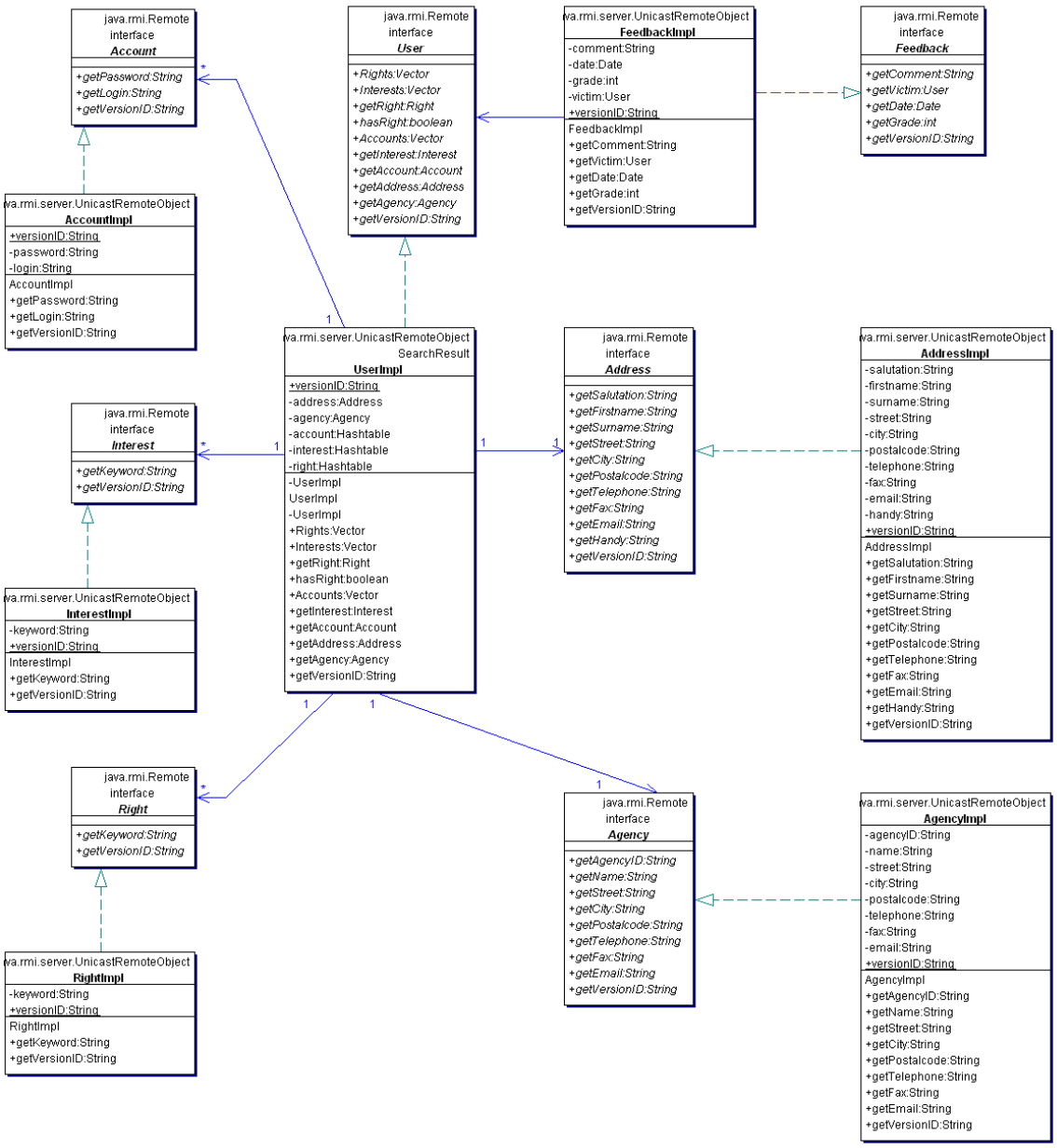


Abbildung 21 - User und Agency

7.4.2 Agency

Das Interface Agency wird von der Klasse AgencyImpl implementiert. Diese Klasse repräsentiert eine Agentur des Versicherungsunternehmens (s. Abbildung 21 - User und Agency). Ein Agentur-Objekt besitzt dabei bestimmte Adress-Attribute sowie eine eindeutige AgenturID. Es wurde eine Klasse AgencyData entworfen (s. Abbildung 22 - UserData und AgencyData). Diese Klasse beinhaltet alle Daten der Agentur und bietet die Möglichkeit, über set- und get-Methoden diese Daten zu manipulieren. Wie auch schon bei User verlangt der Konstruktor von AgencyImpl ein AgencyData-Objekt. Die AgencyImpl-Klasse bietet keine Möglichkeit, Daten nachträglich zu ändern.

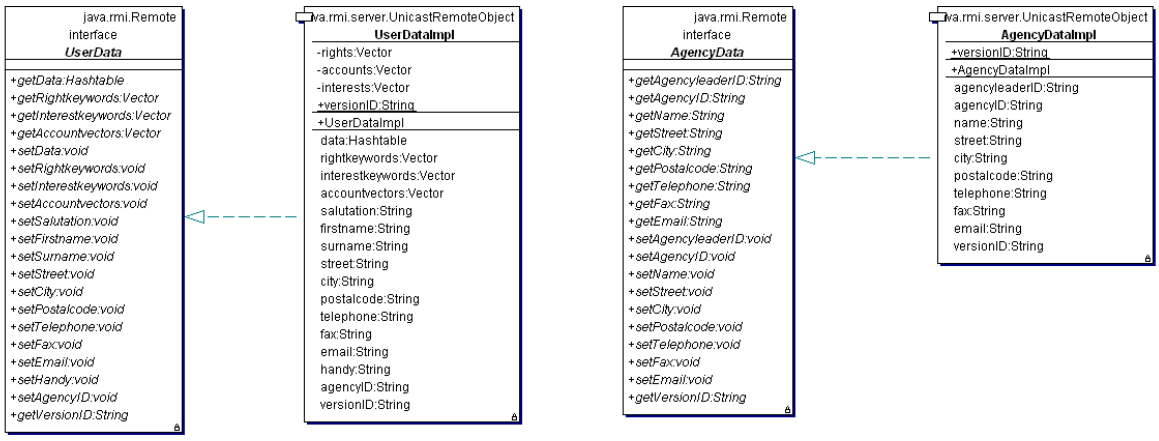


Abbildung 22 - UserData und AgencyData

7.4.3 AdminSubsystemFacade

Das Interface AdminSubsystemFacade wird von der Klasse AdminSubsystemFacadeImpl implementiert. Sie bietet Methoden an, User-Objekte anzufordern, Benutzer anzulegen, zu ändern oder zu löschen, Agentur-Objekte anzufordern, Agenturen anzulegen, zu ändern oder zu löschen, Feedbacks anzufordern oder zu löschen usw. Außerdem bietet sie eine Reihe von Methoden an, die den Controllern die Arbeit erleichtern, wie z.B. das Überprüfen auf existierende IDs oder das Ermitteln aller Benutzer einer Agentur oder aller Agenturleiter.

Wie schon oben erwähnt, besitzt ein Benutzer ein oder mehrere Rechte. Die Überprüfung dieser Rechte geschieht ausschließlich in dieser Fassadenklasse. Deshalb erfordern die meisten Methoden die Übergabe des User-Objekts, das den Akteur der jeweiligen Aktion repräsentiert. Somit können die Rechte / das Recht des Akteurs ausgelesen werden und mit den erforderlichen Rechten verglichen werden. Hat ein Akteur das erforderliche Recht nicht, wird eine NoRightException geworfen. Der Grund, weshalb die Rechteüberprüfung nur in der Fassadenklasse geschieht, ist, dass so das Rechtemanagement leicht geändert werden kann, ohne dass die Controller geändert werden müssen. Außerdem gehört das Rechtemanagement auch „logisch“ zum Subsystem Administration.

Die Klassen sind in Abbildung 23 - AdminSubsystemFacade und PortalUserDB dargestellt.

7.4.4 PortalUserDB

Die Klasse PortalUserDB (s. auch Abbildung 23 - AdminSubsystemFacade und PortalUserDB) bietet im Prinzip die selben Methoden wie die Fassadenklasse an, nur ohne Rechteüberprüfung. Sie hat als einzige Klasse Zugriff auf die Datenbank, um die Datenbankabfragen übersichtlich an einem „Ort“ zu haben und sie so besser pflegen zu können.

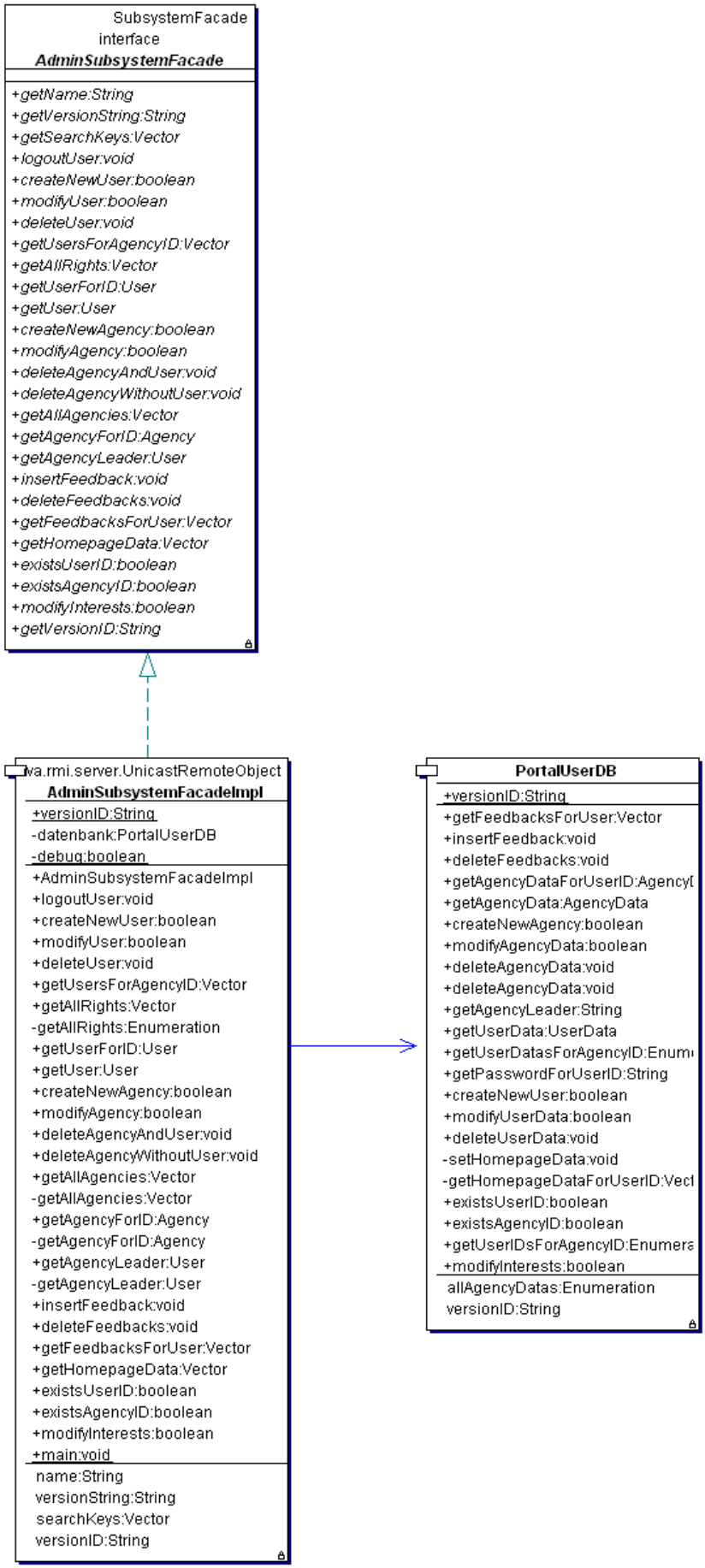

SubsystemFacade
interface
AdminSubsystemFacade
+getName:String
+getVersionString:String
+getSearchKeys:Vector
+logoutUser:void
+createNewUser:boolean
+modifyUser:boolean
+deleteUser:void
+getUsersForAgencyID:Vector
+getAllRights:Vector
+getUserForID:User
+getUser:User
+createNewAgency:boolean
+modifyAgency:boolean
+deleteAgencyAndUser:void
+deleteAgencyWithoutUser:void
+getAllAgencies:Vector
+getAgencyForID:Agency
+getAgencyLeader:User
+insertFeedback:void
+deleteFeedbacks:void
+getFeedbacksForUser:Vector
+getHomepageData:Vector
+existsUserID:boolean
+existsAgencyID:boolean
+modifyInterests:boolean
+getVersionID:String
va.rmi.server.UnicastRemoteObject
AdminSubsystemFacadeImpl
+versionID:String
-datenbank:PortalUserDB
-debug:boolean
+AdminSubsystemFacadeImpl
+logoutUser:void
+createNewUser:boolean
+modifyUser:boolean
+deleteUser:void
+getUsersForAgencyID:Vector
+getAllRights:Vector
-getAllRights:Enumeration
+getUserForID:User
+getUser:User
+createNewAgency:boolean
+modifyAgency:boolean
+deleteAgencyAndUser:void
+deleteAgencyWithoutUser:void
+getAllAgencies:Vector
-getAllAgencies:Vector
+getAgencyForID:Agency
-getAgencyForID:Agency
+getAgencyLeader:User
-getAgencyLeader:User
+insertFeedback:void
+deleteFeedbacks:void
+getFeedbacksForUser:Vector
+getHomepageData:Vector
+existsUserID:boolean
+existsAgencyID:boolean
+modifyInterests:boolean
+main:void
name:String
versionString:String
searchKeys:Vector
versionID:String
PortalUserDB
+versionID:String
+getFeedbacksForUser:Vector
+insertFeedback:void
+deleteFeedbacks:void
+getAgencyDataForUserID:Agency(
+getAgencyData:AgencyData
+createNewAgency:boolean
+modifyAgencyData:boolean
+deleteAgencyData:void
+deleteAgencyData:void
+getAgencyLeader:String
+getUserData:UserData
+getUserDatasForAgencyID:Enum
+getPasswordForUserID:String
+createNewUser:boolean
+modifyUserData:boolean
+deleteUserData:void
-setHomepageData:void
-getHomepageDataForUserID:Vect
+existsUserID:boolean
+existsAgencyID:boolean
+getUserIDsForAgencyID:Enumera
+modifyInterests:boolean
allAgencyDatas:Enumeration
versionID:String

Abbildung 23 - AdminSubsystemFacade und PortalUserDB

7.4.5 Exceptions

Es wurde eine Reihe von Exceptions (s. Abbildung 24 - Exceptions und Constants-Klasse) entwickelt, um die verschiedenen Fehlerfälle besser unterscheiden zu können:

- NoRightException: Diese Exception wird, wie schon oben erwähnt, geworfen, wenn ein Akteur das erforderliche Recht, eine Aktion auszuführen, nicht hat. Sie wird nur in den Methoden der Fassadenklasse geworfen.
- InvalidSubsystemkeyException: Diese Exception wird geworfen, falls der übergebene Schlüssel nicht Schlüssel in der Hashtable Account des User-Objekts ist.
- UserNotFoundException: Diese Exception wird von der Klasse PortalUserDB geworfen, falls die übergebene UserID der Datenbank nicht bekannt ist. Analog dazu gibt es die AgencyNotFoundException.
- AdminSQLException und WrongDriverException: Diese Exceptions können bei einer Datenbankabfrage auftreten, wobei AdminSQLException einfach die bekannte SQLException ersetzt, und WrongDriverException auftritt, wenn der Datenbanktreiber nicht funktioniert bzw. nicht ansprechbar ist.
- AdministrationException: Diese Exception hat momentan keine Funktion. Sie wurde eingeführt, um die vielen Exceptions zu ersetzen bzw. diese Exceptions davon erben zu lassen. Dies wurde jedoch nicht umgesetzt.

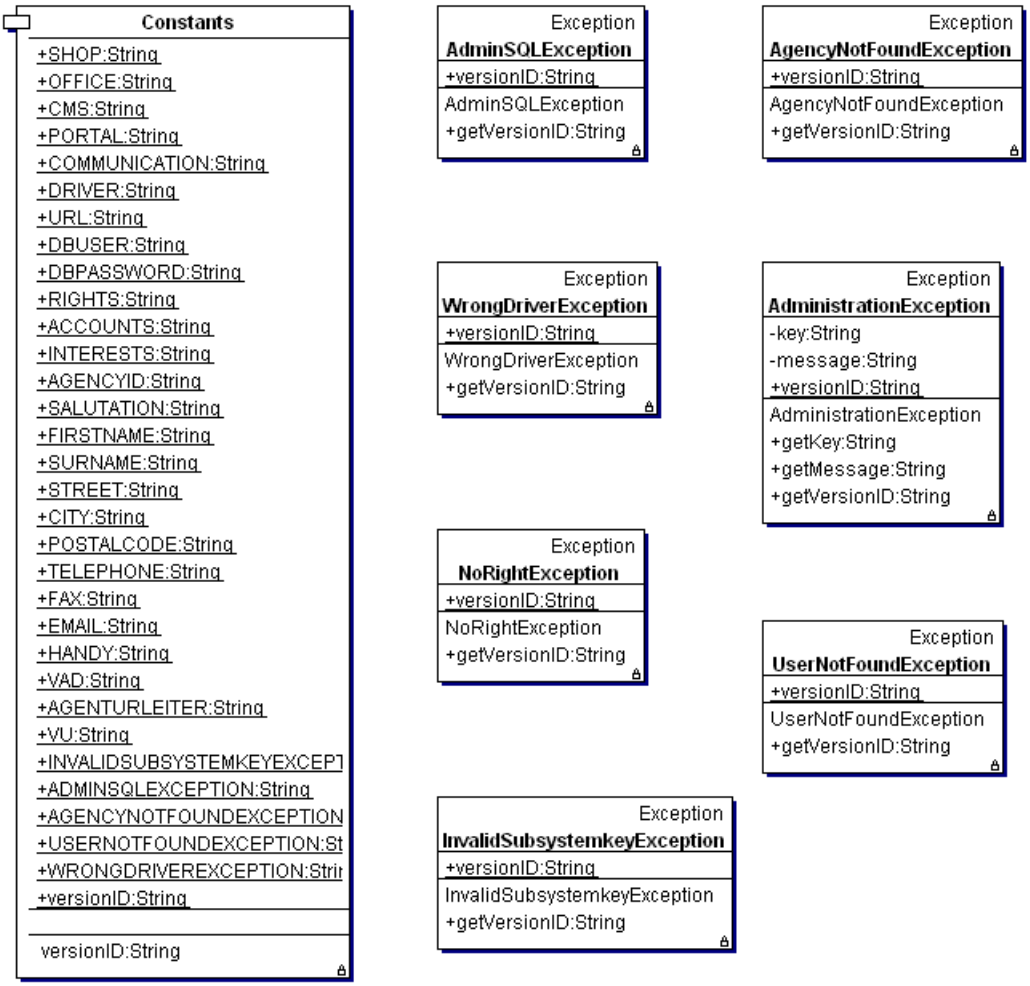


Abbildung 24 - Exceptions und Constants-Klasse

7.5 Subsystem Search

Das Subsystem Search soll die Klassen zur Verfügung stellen, die benötigt werden, um Objekte aus den verschiedenen Subsystemen zu suchen. Wichtig war dabei ein einheitlicher Ansatz für alle Subsysteme. So können später auch noch weitere Subsysteme hinzugefügt werden und die Suche mit wenig Aufwand darauf übertragen werden.

Alle suchbaren Objekte implementieren das Interface *SearchResult*. Die Klasse *SearchResultSet* ist ein Container für Suchergebnisse. Sie kann nur Objekte vom Typ *SearchResult* aufnehmen. Das garantiert Typsicherheit.

Suchanfragen in IPSI sind zunächst immer UND-Verkettungen von Bedingungen. Die Klasse *SearchRequest* ist ein Container für solche Bedingungen. Es wurden ODER-Verkettungen oder beliebige boolesche Ausdrücke für nicht erforderlich gehalten und wo doch nötig, dann durch mehrere Suchanfragen nachgebaut.

Eine Bedingung wird modelliert durch *SearchCriteria*. Diese Klasse hat zwei Attribute, eins für den Namen des Kriteriums (z.B. „Office_Contact_PLZ“) und eins für den gesuchten Wert (z.B. „44141“).

Jede Subsystem-Fassade bietet also eine oder mehrere Methoden der Art

```
public SearchResultSet searchXXXXXX(SearchRequest request);
```

an. Je nach suchbaren Objekten werden andere Suchkriterien (*SearchCriteria*) unterstützt. Als besonderen Service für die Controller zum Aufbau von Suchmasken lassen sich die möglichen Suchkriterien der Subsysteme abfragen. Das wird besonders beim Shop ausgenutzt, dessen Suchkriterien die Warengruppen sind, die sich durchaus ändern können. Die Subsysteme übergeben eine Liste von möglichen Suchkriterien, allerdings ohne das Attribut *value*. Es wurde entschieden, dass statt *value* ohne Wert zu übergeben, eine neue Klasse *SearchCriteriaKey* implementiert wird, von der *SearchCriteria* erbt und sie nur um das Attribut *value* erweitert.

Das nachfolgend abgebildete Klassendiagramm veranschaulicht diese Zusammenhänge.

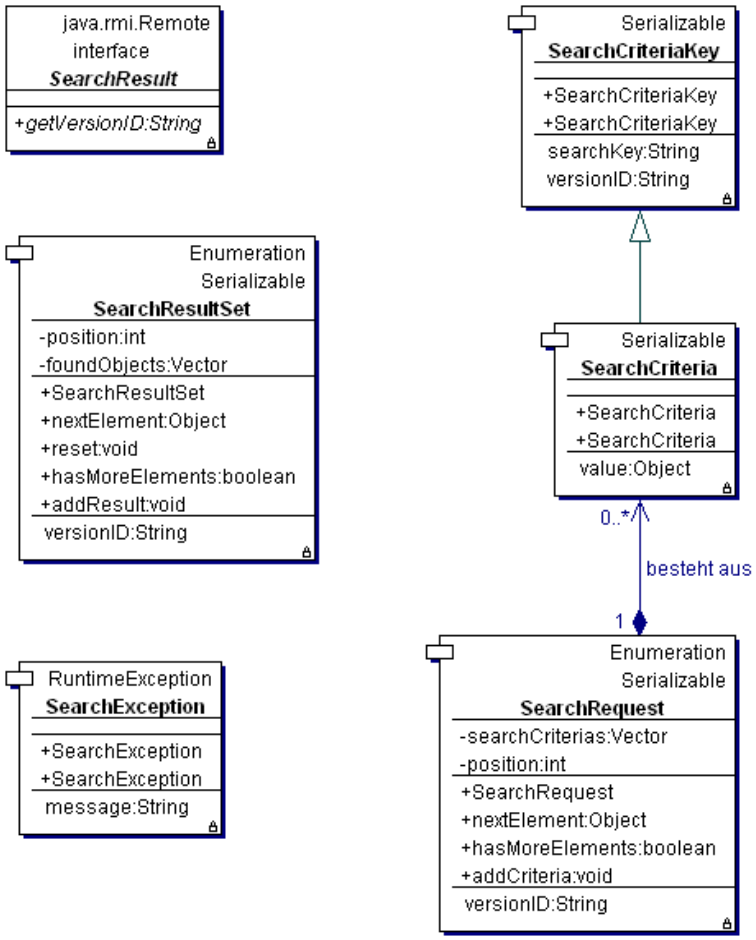


Abbildung 25 - Klassendiagramm des Subsystems Search

7.6 Subsystem Procurement

Beim Entwurf des Subsystems Procurement ging es darum, die Klassen zur Verfügung zu stellen, die benötigt werden, um aus dem Portal in das Shopsystem „SmartStore“ zu gelangen und den Shop so weit wie möglich im Portal zu integrieren. Die zentrale Klasse ist *ShopSubsystemFacade*, die alle benötigten Methoden anderen Subsystemen zur Verfügung stellt.

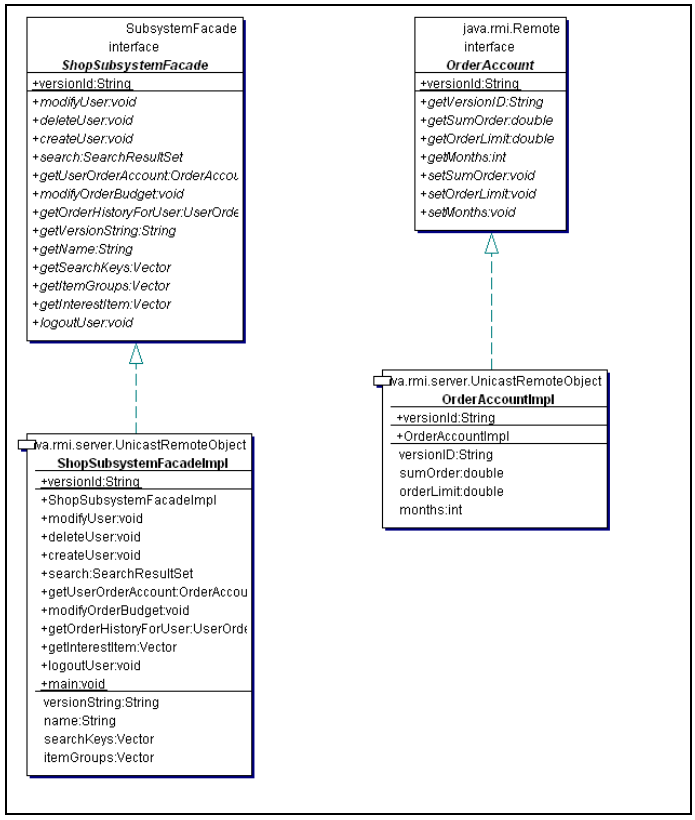


Abbildung 26 - Klassendiagramm des Shop-Subsystems

Die Klasse *OrderAccount* stellt das Budgetkonto eines Benutzers dar. Die Budgetdaten werden in der Methode *getUserOrderAccount* innerhalb der *ShopSubsystemFacade* aus der Shop-Datenbank gelesen und den Attributen des erzeugten *OrderAccount*-Objektes zugewiesen.

Die Bestellhistorien eines Benutzers werden in zwei Arten aufgelistet. Einerseits werden sie als Bestellübersicht nach Warengruppen sortiert aufgelistet. Diese Informationen werden durch ein Objekt der Klasse *UserOrderHistory* transportiert. Anderseits werden die einzelnen Bestellwaren innerhalb einer angegebenen Warengruppe ausgegeben. Die Daten einzelner Bestellungen sind in Objekten der Klasse *ItemGroupHistory* enthalten. Die einzelnen Bestellungen sind Waren, die als Objekte der Klasse *Item* definiert werden. Jede Warengruppe wird durch die Klasse *ItemGroup* dargestellt.

Um ein personalisiertes Angebot an Waren für jeden Benutzer zusammenstellen zu können, hat jeder Benutzer im Portal die Möglichkeit, die für ihn interessanten Warengruppen zu bestimmen. Solche personalisierten Angebote werden durch Objekte der Klasse *ItemsForUser* dargestellt, die eine Reihe von *Item*-Objekten enthalten. Diese *Item*-Objekte beinhalten die Informationen über Waren, die als Top-Produkte vom Shop-Betreiber definiert wurden und zu den für Benutzer interessanten Warengruppen gehören.

Das Klassendiagramm zu diesen textuellen Beschreibungen findet sich in Abbildung 26 - Klassendiagramm des Shop-Subsystems und Abbildung 27 - Klassendiagramm des Shop-Subsystems.

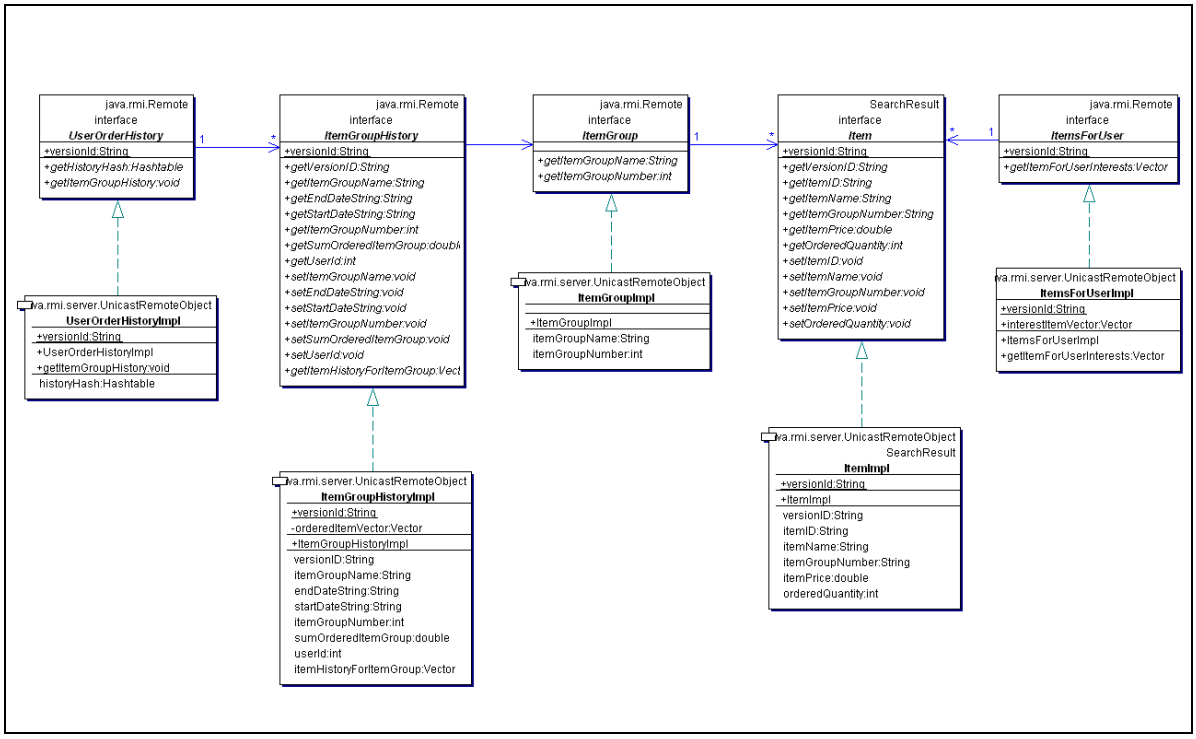


Abbildung 27 - Klassendiagramm des Shop-Subsystems

7.7 Subsystem Kommunikation

In der Anforderungsanalyse (s. Kapitel 6 Anforderungsanalyse) haben wir bestimmte Kriterien festgelegt, denen das IPSI-Subsystem Kommunikation genügen muss. Unter anderem wurden dort folgende Anforderungen ermittelt:

- VADs müssen über verschiedene Medien Nachrichten erhalten können, z.B. über Email, SMS oder FAX.
- Über diese Medien müssen unterschiedliche Nachrichtentypen verschickt werden können.
- Der Zeitpunkt des Versands muss definierbar sein.
- Neue Medien und Nachrichtentypen müssen einfach in das System integrierbar sein.
- Die Ergebnisse aller Versandaktionen müssen protokolliert werden, damit der Systemadministrator bei der Suche nach eventuellen Fehlern unterstützt wird.
- Das Kommunikationssystem muss den anderen Subsystemen eine einheitliche Schnittstelle bieten, über die eine Kommunikation initiiert werden kann.

Zum Einen musste also eine Architektur gefunden werden, die es ermöglicht, unterschiedliche Nachrichtentypen über diverse Medien zu versenden. Zum Anderen musste die Architektur so flexibel erweiterbar sein, dass neue Nachrichtentypen bzw. Versandmedien einfach in das System integriert werden können.

In dem Subsystem Kommunikation des aktuellen IPSI-Portals existiert eine abstrakte Oberklasse Message, von der alle Nachrichtenklassen abgeleitet sind. Zur Zeit sind drei Nachrichtenklassen realisiert: ClientComment, TaskReminder und AppointmentReminder. Dieses sind Business-Objekte, die über die Middleware mit anderen Subsystemen ausgetauscht werden. Der AppointmentReminder und der TaskReminder enthalten Business-Objekte aus dem Subsystem Office, Appointment bzw. Task, während der ClientComment aus primitiven Datentypen besteht. Darüber hinaus verfügt eine Message über einen Fallback-Mechanismus. Dieser tritt in Kraft, wenn eine Nachricht über das bevorzugte Medium nicht versendet werden konnte und daher für den Versand über ein anderes Medium vorbereitet werden muss. Die Abbildung 28 - Klassendiagramm Kommunikation mit dem Teilbereich „Nachrichtentypen“ zeigt die Klassenhierarchie der Nachrichtentypen:

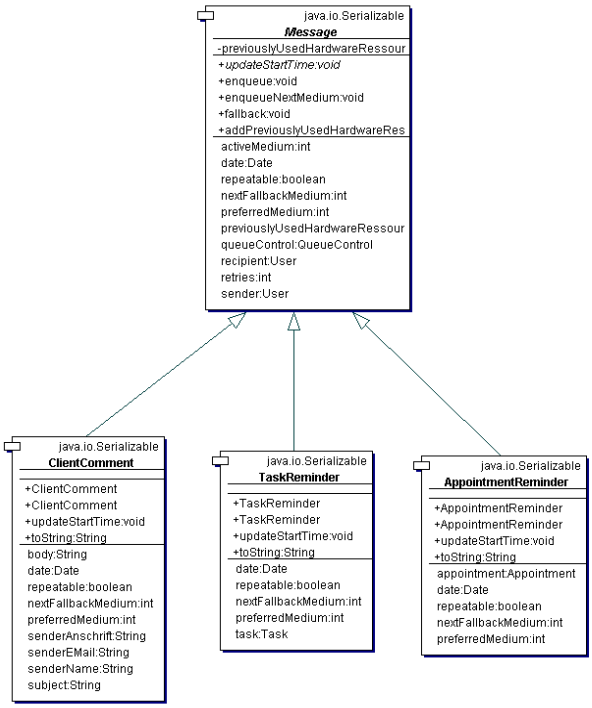


Abbildung 28 - Klassendiagramm Kommunikation mit dem Teilbereich „Nachrichtentypen“

Entsprechend den Nachrichtentypen existiert für die Versandmedien eine abstrakte Oberklasse Versand, von der die drei realisierten Versandklassen FAXVersand, SMSVersand und EmailVersand abgeleitet sind. Diese Versandklassen stellen die Anbindung an die physikalischen Versandkanäle dar, z.B. an einen Emailserver oder an ein Faxmodem. Einerseits ist es dabei möglich, dass mehrere Versandkanäle für ein Medium existieren, z.B. wenn zwei Faxmodems vorhanden sind. Andererseits ist es auch möglich, dass ein Versandkanal von mehreren Versandklassen verwendet wird, z.B. wenn sich ein FAXVersand-Objekt und ein SMSVersand-Objekt ein gemeinsames Modem teilen. Auf die Probleme, die sich dadurch ergeben wird im Zusammenhang mit der Klasse VersandControl näher eingegangen. Jedes Versandmedium benötigt für den Versand ein eigenes Datenformat, (z.B. darf eine SMS-Nachricht nur 160 Zeichen enthalten). Im Entwurf wurde daher für jedes Versandmedium eine Klasse vorgesehen, die die Daten für dieses Medium enthält. Aus Gründen der Typsicherheit implementieren die Datenklassen FaxData, SMSData und EmailData das Interface Data. Schließlich wird noch ein Mechanismus benötigt, der aus einem Nachrichtenobjekt das entsprechende Datenobjekt für dasjenige Versandmedium erzeugt, über das diese Nachricht versendet werden soll. Dieser Mechanismus wird von den Klassen bereitgestellt, die das Interface Formatter implementieren: FAXFormatter, SMSFormatter und EmailFormatter. Ebenso wie von den Datenklassen wird von den Formatter-Klassen je eine Instanz pro Versandmedium benötigt. In Abbildung 29 sind alle Klassen und deren Beziehungen untereinander dargestellt, die unmittelbar mit dem Versand von Nachrichten zusammenhängen.

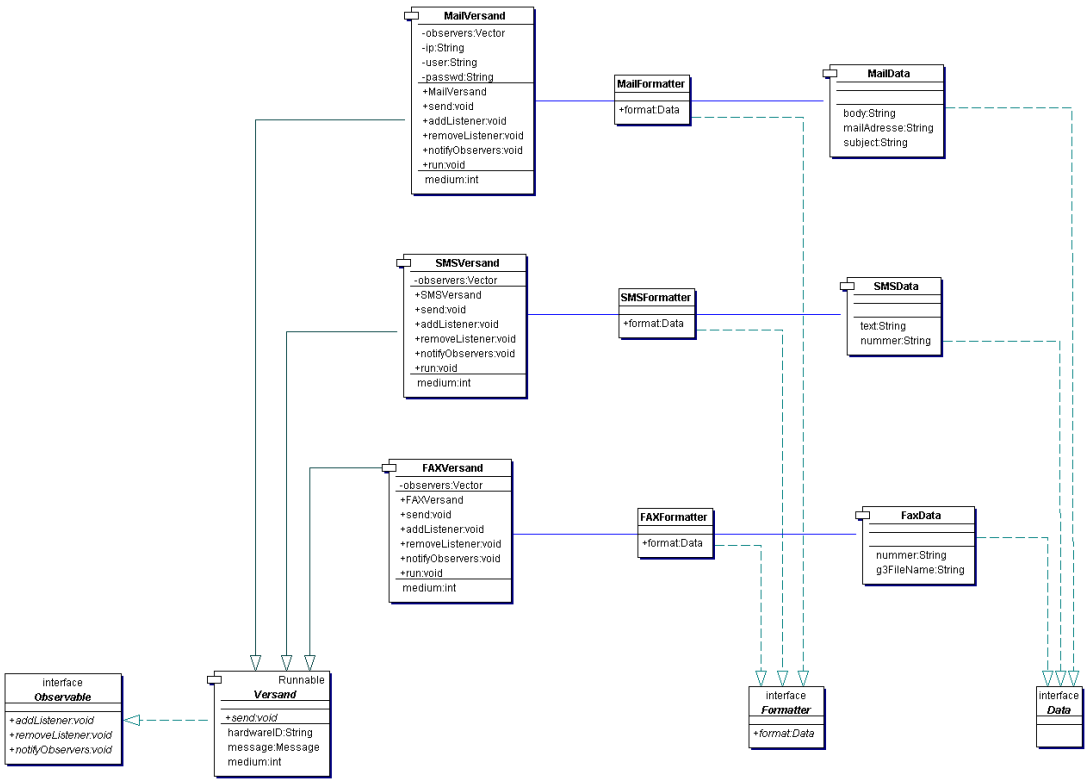


Abbildung 29 - Klassendiagramm Kommunikation mit dem Teilbereich „Versand“

Die bisher beschriebenen Klassen stellen die Basis für das Subsystem Kommunikation dar. Nun werden noch Klassen benötigt, die die Koordination verschiedener Aufgaben übernehmen. Die folgende Abbildung 30 zeigt eine Übersicht über diese Klassen.

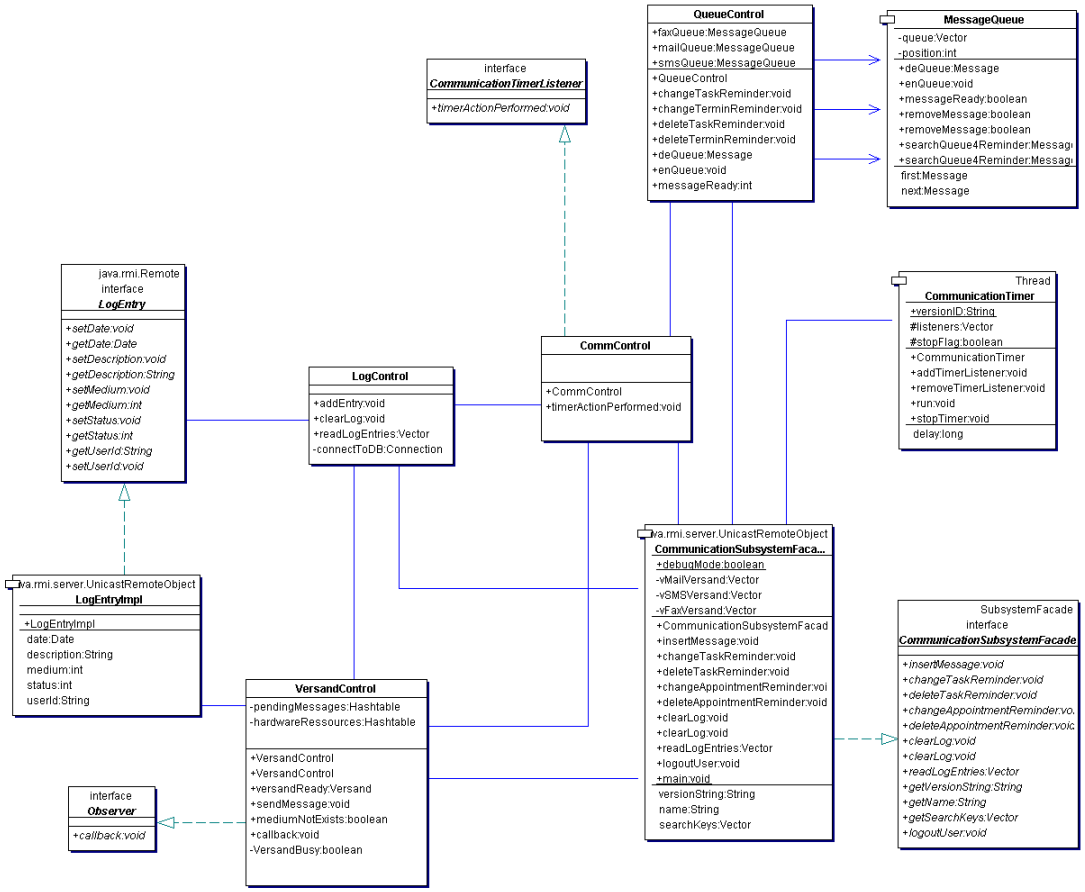


Abbildung 30 - Klassendiagramm Kommunikation mit dem Teilbereich „Koordination“

Eine dieser Aufgaben, die zusätzlichen Koordinationsaufwand erfordert, ist die in den Anforderungen beschriebene Möglichkeit, Nachrichten zu einem bestimmten Termin versenden zu können. Zu diesem Zweck haben wir in der Klasse MessageQueue eine Priority-Queue implementiert, die beliebige Nachrichtentypen aufnehmen kann, unter der Bedingung, dass diese von Message abgeleitet sind. Innerhalb der Queue werden Nachrichten nach dem Versandzeitpunkt sortiert. Neben den Standardfunktionen, die eine Queue ausmacht, bietet die MessageQueue Methoden an, mit denen nach bestimmten Nachrichten anhand eines Business-Objects aus dem Subsystem Office gesucht werden kann. Das ist notwendig, falls bestimmte Nachrichten im Nachhinein geändert werden sollen. Zwischen Message-Queues und Versandmedien besteht eine 1:1-Beziehung. Für die Verwaltung aller Message-Queues ist die QueueControl verantwortlich (vgl. Abbildung 30). Über diese Klasse können Nachrichten in die Queue eines Mediums eingefügt, aus dieser gelöscht oder in dieser verändert werden.

Für die regelmäßige Überprüfung, ob Nachrichten für den Versand bereit sind, ist die Klasse CommControl verantwortlich (vgl. Abbildung 30). Diese Klasse implementiert das Interface CommunicationTimerListener und wird beim Starten des Subsystems beim CommunicationTimer angemeldet. Das hat zur Folge, dass die Methode timerActionPerformed() in bestimmten, vordefinierten Intervallen aufgerufen wird. Innerhalb dieser Methode prüft die CommControl, ob Nachrichten für einen bestimmten Medientyp zum Versand bereitliegen. Ist dieses der Fall, so muss zusätzlich noch geprüft werden, ob ein physikalischer Versandkanal frei und eine Instanz der Versandklasse verfügbar ist. Erst wenn diese Bedingungen erfüllt sind, wird die Nachricht der entsprechenden Queue entnommen und der VersandControl zum Versenden übergeben (vgl. Abbildung 30).

Die VersandControl hat vielfältige Aufgaben zu bewältigen:

1. Sie muss dafür Sorge tragen, dass eine Versandinstanz einen Versandvorgang vollständig beendet hat, bevor diese mit einem neuen Versandauftrag belastet wird.

2. Da es möglich ist, dass mehrere Versandinstanzen ein und dieselbe Hardwareressource benutzen (z.B. können eine FaxVersand-Instanz und eine SMSVersand-Instanz sich ein Modem teilen), muss die VersandControl darauf achten, dass zu jedem Zeitpunkt nur eine der Versandinstanzen aktiv ist, die auf derselben Hardwareressource arbeiten.

3. Das Subsystem Kommunikation darf nicht blockieren, wenn der Versand einer Nachricht einen längeren Zeitraum beansprucht, wie dies z.B. beim Faxversand durch ein längeres Fax möglich wäre.

Zur Vermeidung der im ersten Punkt beschriebenen Problematik, verfügt die VersandControl über eine Hashtabelle (pendingMessages), in der diejenigen Versandinstanzen vermerkt werden, die gerade mit dem Versand eines konkreten Nachrichtenobjektes beschäftigt sind. Wird eine neue Versandaktion initiiert, so wird zunächst die involvierte Versandinstanz als Schlüssel in diese Hashtabelle eingetragen. Den Wert dieses Schlüssels bildet das Nachrichtenobjekt selbst. Anschließend wird die ID der Hardwareressource dieser Versandinstanz ermittelt und in eine weitere Hashtabelle eingetragen (hardwareRessources). Über die Hashtabelle hardwareRessources wird dem oben unter Punkt 2 aufgeführten Sachverhalt Rechnung getragen. Um das unter Punkt 3 beschriebene Problem, dass eine Blockade des gesamten Subsystem bei langen Nachrichten auftreten könnte, zu umgehen, implementiert die Oberklasse Versand das Interface Runnable (vgl. Abbildung 30). Damit kann für jeden Versandvorgang ein eigener Thread gestartet werden. Zur Kommunikation zwischen einer Versandklasse in einem Thread und der VersandControl in einem anderen Thread, wurde eine Observer/Observable-Architektur gewählt (vgl. Abbildung 29 und Abbildung 30). Nach Abschluss eines erfolgreichen oder auch nicht erfolgreichen Versandvorganges wird die VersandControl von einer Versandinstanz hierüber in Kenntnis gesetzt. Ein erfolgreicher Versandvorgang hat zur Folge, dass die Versandklasse und das versendete Nachrichtenobjekt aus der Hashtabelle pendingMessages entfernt werden. Aus der Hashtabelle hardwareRessources wird die von der Versandklasse verwendete ID der Hardwareressource ebenfalls entfernt. Somit ist diese Versandklasse wieder als frei markiert und kann für den Versand weiterer Objekte benutzt werden. Ist während des Versandes ein Fehler aufgetreten, so wird das Nachrichtenobjekt aus der Hashtabelle pendingMessages nicht verworfen, sondern wieder über die QueueControl in die entsprechende Queue eingefügt. Abschließend muss ein Log-Eintrag geschrieben werden, der den Status des Versandvorganges dokumentiert. Zu diesem Zweck bietet die Klasse LogControl aus Abbildung 30 eine Methode zum Speichern eines Log-Eintrags in einer SQL-Datenbank an. Ebenfalls über diese Klasse können Log-Einträge gelesen und gelöscht werden. Um sicherzustellen, dass nur berechtigte Benutzer des Systems die Log-Einträge lesen können, erwartet die Methode readLogEntries() als Eingabe einen Vektor mit User-Objekten, über den das Rechtemanagement realisiert wird.

Die CommunicationSubsystemFacade, ebenfalls in Abbildung 30 abgebildet, stellt eine Schnittstelle dar, über die andere JAVA-Klassen alle Dienste dieses Subsystems ansprechen können. Beim Starten der Subsystem-Fassade müssen einige Parameter zur Konfiguration der einzelnen Versandkanäle eingestellt werden. Die genauen Parameter und deren Bedeutung sind dem HowTo zum Installieren und Konfigurieren des IPSI-Portals (vgl. Abschnitt 11.3) zu entnehmen.

Der Vollständigkeit halber sind in der Abbildung 31 die Hilfsklassen des Subsystems Kommunikation dargestellt. Dazu zählen eine Konstantenklasse sowie einige Exceptions, über die die Fehlerbehandlung gesteuert wird.

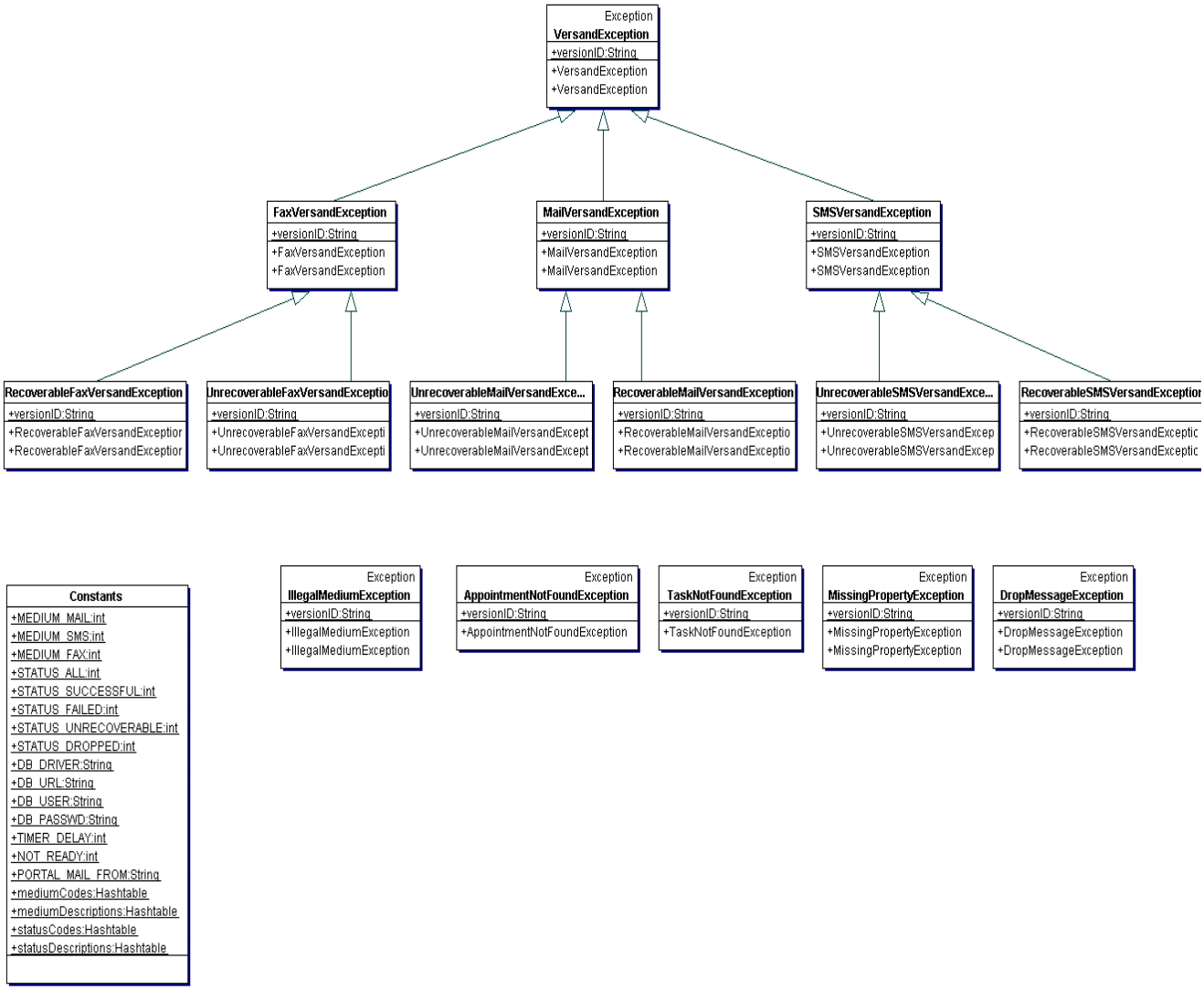


Abbildung 31 - Klassendiagramm Kommunikation mit dem Teilbereich „Hilfsklassen“

7.8 Subsystem Legacy

Bei der exemplarischen Anbindung eines Legacy-Systems haben wir die Unterstützung der Continentalen Versicherung [Co00] erhalten. Die Continentale Versicherung hat ihr Legacy-System, in diesem Fall die Partnerdatenbank, über eine XML-Schnittstelle [W300] angebunden.

7.8.1 Woher kommen die Daten aus der Partnerdatenbank?

Bei der Continentalen hat der VAD somit die Möglichkeit, Daten aus der Partnerdatenbank wie z.B. Kundendaten inklusive Vertragsdaten und Geschichtsbücher abzufragen. Der Ablauf ist in Abbildung 32 dargestellt:

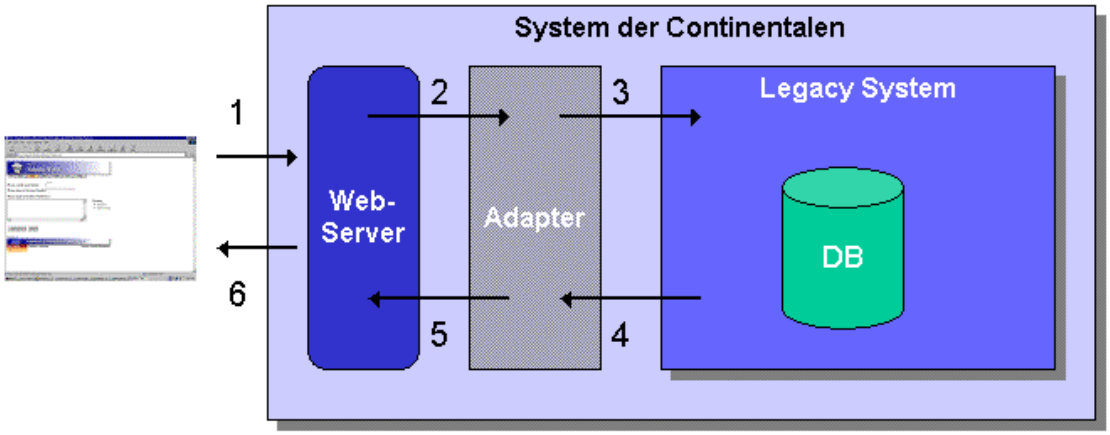


Abbildung 32 - Ablauf der Legacy-Anbindung bei der Continentalen Versicherung

1. Vom Client (das ist ein Stück Software die jeder VAD von der Continentalen gestellt bekommt) wird ein URL mit einem Query String an den Webserver gesendet.

2. Ein Servlet leitet diesen Query String an den entsprechenden Adapter weiter.

3. Der Adapter wandelt den Query String um und benutzt eine COBOL-Schnittstelle zur Abfrage des Datenbankmanagementsystems.

4. Das Resultat der Datenbankabfrage wird vom Adapter empfangen und in ein XML-Dokument gepackt.

5. Der Adapter liefert ein XML-Dokument an das Servlet zurück.

6. Der Webserver sendet das XML-Dokument an den Client zurück. Zur Darstellung des XML-Dokuments werden ein XSL- und ein CSS-Dokument inklusive übertragen. Diese beiden Typen von Dokumenten dienen der Darstellung in der Client-Software.

Da wir in IPSI ein Legacy-System integrieren wollten, war die Möglichkeit der Anbindung an die XML-Daten der Continentalen ideal. Wir haben aus diesem Grund von der Continentalen den XML-Datenstrom erhalten. Damit hatten wir die korrekte Syntax der XML-Tags und der verfügbaren bzw. abfragbaren Daten, die auch in der realen Anwendung verwendet werden. Dieser XML-Datenstrom liegt in Form einer Datei vor, da es nicht gewünscht war, eine physikalische Verbindung zwischen der Universität Dortmund und der Continentalen aufzubauen. Außerdem handelt es sich bei der Integration um eine Beispielanwendung und darf den Produktionsbetrieb nicht gefährden. Es bleibt aber zu betonen, dass dieses durch die Architektur möglich ist.

Um die Integration zu gewährleisten, haben wir uns folgende grobe Architektur, wie in Abbildung 33 dargestellt, überlegt.

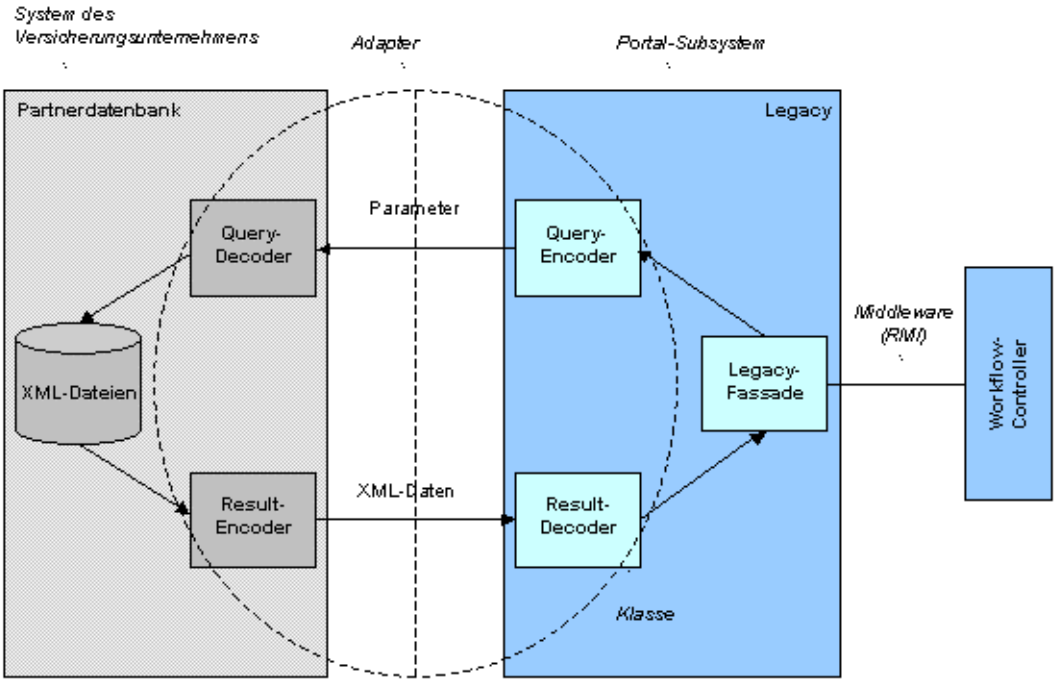


Abbildung 33 - Ablauf der Legacy-Anbindung im IPSI-Portal

Über den Workflow-Controller, der die Business-Logik beinhaltet, wird die richtige Methode in der Legacy-Fassade angesprochen. Ein Workflow-Controller ist beispielsweise für die Aktion „zeige alle Daten eines Kunden“ zuständig. Die Fassade stellt über einen Query-Encoder die Datenbankabfrage, die von der Partnerdatenbank wieder decodiert wird. Innerhalb der Partnerdatenbank werden dann die richtigen XML-Daten bzw. in unserem Fall die richtigen XML-Dateien ermittelt und über den Result-Encoder und –Decoder an die Fassade zurückübergeben. Danach werden die XML-Daten weiterverarbeitet (Instanziierung der Business-Objekte) und an den Workflow-Controller zur Ausgabe übermittelt.

7.8.2 Klassendiagramm

Das Klassendiagramm zum Legacy-Subsystem ist durch die Eigenheiten der Anbindung der Legacy Partnerdatenbank geprägt. D.h. es gibt Klassen zum parsen der XML-Daten und Klassen, die die Business-Objekte repräsentieren. Ferner haben wir einige Statistikklassen kreiert, die die Ergebnisse einer Datenbankanfrage (auf die vorhandenen XML-Daten) beinhalten.

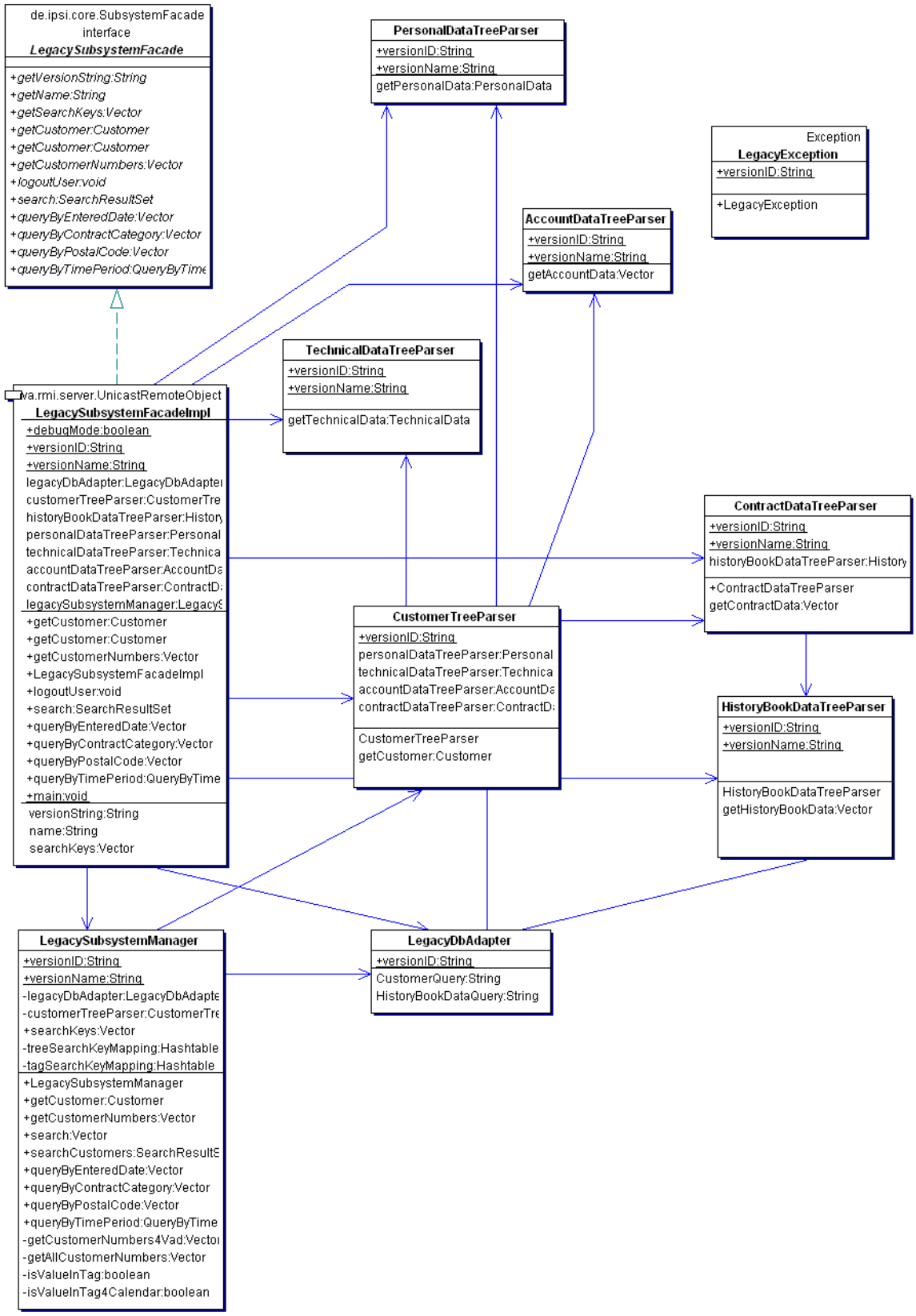


Abbildung 34 - Legacy Klassendiagramm mit der Subsystem Facade

Die LegacySubsystemFacade stellt Methoden zur Verfügung, die von einem Controller aufgerufen werden können (s. Abbildung 34). An diese Methoden wird jeweils ein Vektor von User-Objekten übergeben. Über diese User-Objekte wird das Rechtemanagement gesteuert. Ein VAD darf nur seine eigenen Kunden einsehen und ein Agenturleiter alle Kunden seiner VADs. In dem Fall des Agenturleiters ist der Vektor mit allen User-Objekten der VADs gefüllt.

Einem Objekt der Klasse CustomerTreeWalker wird eine Kundenversicherungsnummer übergeben. Mit Hilfe der Klasse LegacyDbAdapter kann über diese Nummer die richtige XML-Datei ermitteln werden (s. Abbildung 34). Nach dem erfolgreichem Parsen der XML-Datei und dem Anlegen des TreeWalker kann das Business-Objekt belegt werden. Dazu wird der jeweilige Unterbaum des TreeWalkers an die zuständigen Objekte der Klassen PersonalData, TechnicalData, usw. übergeben. Jedes dieser Objekte hat Methoden, die aus dem Unterbaum die Inhalte auslesen und die Attribute belegen. Dabei werden die Attribute über die bekannten XML-Tags identifiziert. Sollte ein XML-Tag mehrmals vorkommen, wird dies in einer Schleife berücksichtigt. Für alle Attribute müssen natürlich die original Datenbankbezeichner für die Unterbäume und für die Tags bekannt sein.

Wir haben dieses Verfahren gewählt, da sich ein Customer somit selbst mit Werten belegen kann. Er bekommt nur eine Kundenversicherungsnummer und ermittelt darüber alle Attribute, sowohl für die persönlichen Daten, die technischen Daten, die Kontoverbindungen, als auch für die Verträge mit den jeweiligen Geschichtsbüchern (s. Abbildung 35).

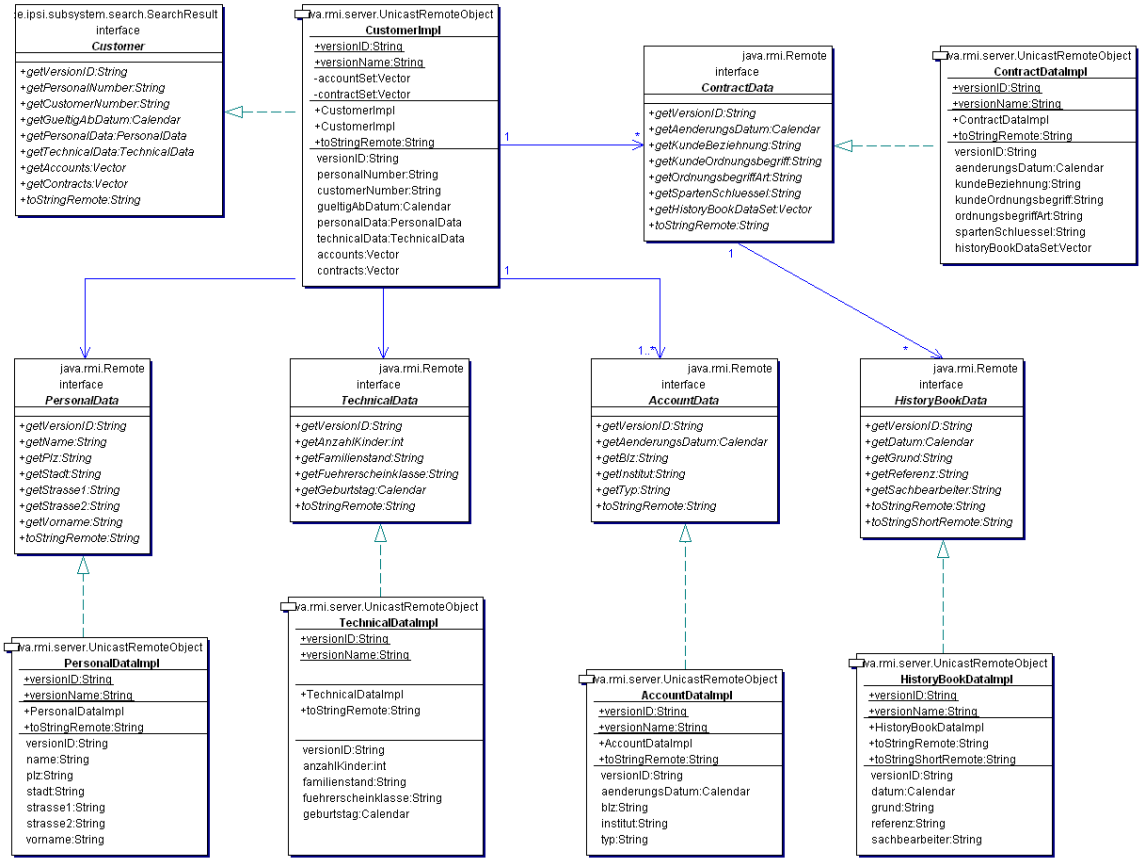


Abbildung 35 - Legacy Klassendiagramm mit der Customer-Klasse

Um die Daten aus der Partnerdatenbank transparenter zu gestalten, haben wir uns sinnvolle Statistikauswertungen überlegt. Diese Auswertungen werden immer nach einem ähnlichen Muster berechnet: Über den LegacySubsystemManager wird eine Anfrage an die XML-Daten formuliert. Die XML-Daten werden dabei eingelesen und auf bestimmte Kriterien hin untersucht (analog zu einer Datenbankanfrage). Die Ergebnisse dieser Anfrage werden dann in Business-Objekten gespeichert.

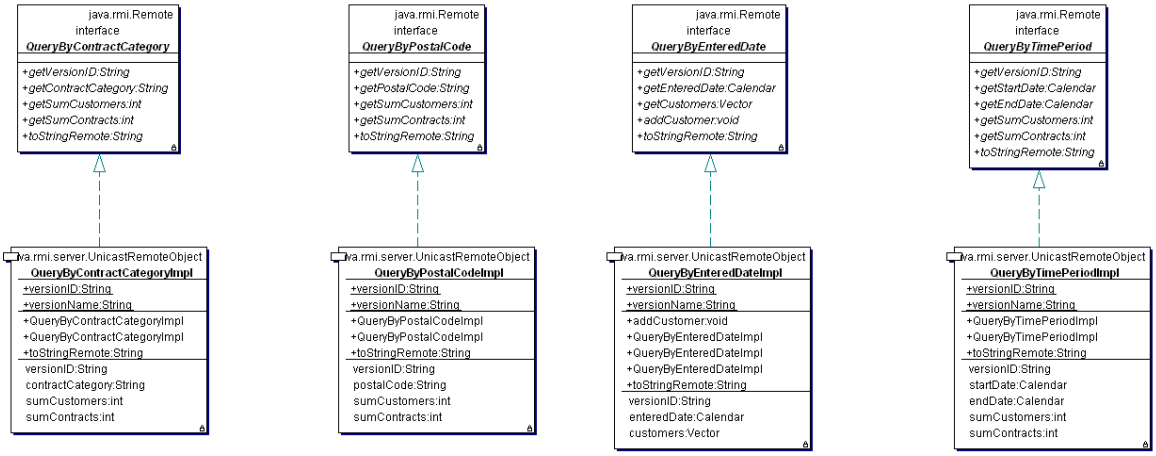


Abbildung 36 - Legacy Klassendiagramm mit den Statistik-Klassen

In Abbildung 36 sind diese Business-Objekte in Form von Klassen dargestellt. Die Bedeutung der einzelnen Anfragen zu Statistikauswertungen der XML-Daten ist wie folgt:

- Business-Objekt: QueryByContractCategory
  Anfrage: Wie viele Kunden/Verträge sind in der Datenbank mit welchen Vertragskategorien?
- Business-Objekt: QueryByPostalCode
  Anfrage: Geographische Verteilung der Kunden
- Business-Objekt: QueryByEnteredDate
  Anfrage: Welche Kunden sind seit dem „dd.MM.yyyy“ hinzugefügt worden?
- Business-Objekt: QueryByTimePeriod
  Anfrage: Wie viele neue Kunden haben wie viele Verträge zwischen Start- und Enddatum abgeschlossen?

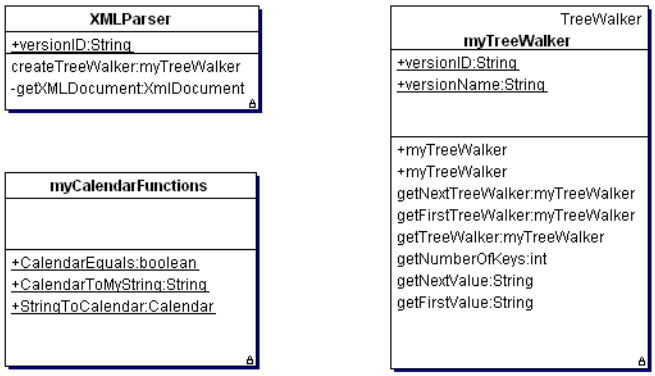


Abbildung 37 - Legacy Klassendiagramm mit den Hilfsklassen

In Abbildung 37 werden die verwendeten Hilfsklassen dargestellt. XMLParser und der myTreeWalker sind zum Parsen und zum Erstellen der Baumstruktur notwendig. Sie beinhalten die speziellen Methoden dafür. Die Klasse myCalendarFunctions ist notwendig, um z.B. Strings in Calendar-Objekte und umgekehrt zu überführen.

7.8.3 Was wurde vom ersten Entwurf bis zur Finalversion geändert?

Der erste Entwurf musste an einigen Stellen verändert werden, da sich durch die Struktur der XML-Daten neue Erkenntnisse ergeben hatten.

Jeder Kunden hat für jeden Vertrag ein Geschichtsbuch. In dem Geschichtsbuch sind dann wiederum 1 bis n Einträge enthalten. D.h., dass die Klasse HistoryBookData eine eins zu eins Beziehung zur Klasse ContractData hat. Die 1 zu 1..* Beziehung zwischen Customer und ContractData bleibt aber so bestehen.

Desweiteren fällt die Klasse Xml2JAVAParser weg, da zum Parsen der XML-Daten ein etwas anderer Mechanismus, wie bereits weiter oben beschrieben über den TreeWalker, verwendet wurde.

7.9 GUI-Entwurf

Neben dem Entwurf der Systemarchitektur musste ein Entwurf der Benutzungsoberfläche durchgeführt werden, der im folgenden dargelegt werden soll. Der erste Abschnitt fasst dabei die konzeptionellen Entwurfsentscheidungen zusammen, der zweite stellt die konkreten Umsetzungen in Form einer Klassenbibliothek dar.

7.9.1 Entwurfsentscheidungen für die GUI

Dieser Abschnitt stellt die Entwurfsentscheidungen vor, die bezüglich der Gestaltung der Benutzeroberfläche getroffen wurden. Der Abschnitt gliedert sich in die allgemeinen Anforderungen und diejenigen an das Grundlayout und die wichtigsten Bestandteile eines WWW-Portals, der Navigationsleiste und dem Dialogbereich.

7.9.1.1 Anforderungen

Bei der Gestaltung der Benutzungsoberfläche für das Portal wurden die Grundsätze der Dialoggestaltung [Di88, Is93] sowie die Anforderungen und Qualifikationen der Zielgruppe „Versicherungsaußendienstmitarbeiter“ als Leitlinien verwendet. So konnte zwar versicherungsspezifisches Fachwissen bei den Benutzern vorausgesetzt werden, nicht aber Erfahrung im Umgang mit einer Portal-Site oder dem WorldWideWeb. Demzufolge sollte die Benutzungsoberfläche leicht erlernbar sein, jedoch auch effizienten Umgang mit dem Fachmaterial ermöglichen. Eine weitere Rahmenbedingung war, dass das System auf einer breiten Palette von Endgeräten und besonders im mobilen Einsatz nutzbar sein sollte. Bei der Gestaltung der HTML-Seiten war also auf die geringe Bandbreite und eingeschränkten Darstellungsmöglichkeiten von Notebooks Rücksicht zu nehmen.

7.9.1.2 Grundlayout

Das Grundlayout ist daher bewusst schlicht gehalten und kommt ohne Skriptsprachen wie JavaScript oder Multimedia-Techniken wie Macromedia Flash aus. Lediglich einige grafische Symbole werden verwendet, um bestimmte Zusammenhänge zu verdeutlichen. Das Layout orientiert sich grob an klassischen, nicht web-basierten Softwareanwendungen, um dem VAD die Einarbeitung zu erleichtern: Im Seitenkopf steht eine Navigationsleiste, die das Pendant zur klassischen Menüleiste darstellt. Darunter folgt der Dialogbereich mit einer oder mehreren gerahmten Boxen, die Inhalte oder Dialogelemente enthalten und optisch an Bildschirmfenster erinnern. Die Seite wird durch eine Fußzeile abgeschlossen, die Statusmeldungen oder einen Copyrighthinweis enthalten kann.

Alle drei Bereiche (Navigationsleiste, Dialogbereich und Fußzeile) werden auf der selben Webseite dargestellt. Auf Frames wurde verzichtet, da sie eine Reihe von Problemen nach sich ziehen: Das fundamentale Konzept des WorldWideWeb ist die *Seite* – sie verbindet Quelltext, Benutzeransicht, Objektadresse und Navigationsaktion in einem atomaren Element und trägt so wesentlich zur einfachen Bedienung des WWW bei. Frames durchbrechen dieses Konzept jedoch durch die Trennung von Objektadresse und Benutzeransicht – mit schwerwiegenden Konsequenzen [Ni96]: Da der aktuelle Status eines Framesets nicht durch die URL ausdrückbar ist, können keine Bookmarks gesetzt, Seiten gespeichert, Adressen weitergegeben werden etc. Komplette Framesets können nicht vernünftig ausgedruckt werden; unerfahrene Benutzer wissen nicht, wie sie einzelne Seiten aus dem Frameset drucken können. Bei älteren Browsern funktioniert der äußerst wichtige Backbutton nicht mehr. Auch aktuelle nicht fensterbasierte oder mobile Clients müssen vor Framesets kapitulieren, da es kaum sinnvolle Äquivalente für reintextuelle oder geringauflösende Displays gibt.

Desweiteren wurde entschieden, dass alle Dialoge im gleichen Browser erscheinen sollen; es werden keine neuen Browser-Fenster (z.B. für Sicherheitsabfragen) geöffnet. Diese Entscheidung wurde aus mehreren technischen und ergonomischen Gründen getroffen: Zum Öffnen neuer Fenster mit einer bestimmten Größe und bestimmten Attributen (Skalierbarkeit, Scrollbarkeit, etc.) ist clientseitig JavaScript erforderlich. Das Funktionieren des Systems wäre also davon abhängig, dass der Browser JavaScript unterstützt und der Benutzer diese Funktion nicht (z.B. aus Sicherheitsgründen) deaktiviert hat. Die gesamte Systemarchitektur ist auf möglichst weitgehende Plattformunabhängigkeit (bis hin zu WAP-Endgeräten) ausgerichtet. Der Verlass auf JavaScript würde das Spektrum der möglichen Plattformen jedoch deutlich einschränken. Zudem ist der Benutzer gewohnt, dass beim Klicken auf Links und Buttons die neuen Webseiten im gleichen Fenster erscheinen. Das Öffnen neuer Browser-Fenster würde somit den Dialoggrundsatz der Erwartungskonformität verletzen.

7.9.1.3 Navigationsleiste

Als Konsequenz arbeitet der Benutzer also mit Dialogmasken anstelle von Fenstern. In dieser Oberfläche, die für das Web zwar typisch, für Anwender klassischer Software jedoch zunächst ungewohnt ist, ist es wichtig, dass das System dem Benutzer jederzeit Antwort auf die folgenden vier Fragen geben kann [Ni83]:

1. Wo bin ich hier?
2. Was kann ich hier tun?
3. Wie kam ich hier hin?
4. Wie komme ich wieder zurück?

Ohne dieses Wissen könnte der Benutzer sich nicht im Portal orientieren; der Dialoggrundsatz der Selbstbeschreibungsfähigkeit würde verletzt.

Die Navigationsleiste als zentraler Orientierungspunkt ist aus diesem Grund in zwei Bereiche aufgeteilt: Die erste Zeile stellt in Form von Links den Pfad dar, der in der Maskenhierarchie von der Homepage zur aktuellen Maske führt. Auf diese Weise sind die Fragen 1, 3 und 4 des Benutzers auf einen Blick beantwortet. Frage 2 wird durch die zweite Zeile der Navigationsleiste beantwortet, die alle Masken, die von diesem Punkt aus erreichbar sind, als Links auflistet.

Beispiel:

Home > **Aufgaben**
Übersicht | Erstellen | Suchen

7.9.1.4 Dialogbereich

Die Gestaltung der Dialoge lehnt sich an die übliche Gestaltung von Dialogboxen in klassischen Softwareanwendungen an – so wurden z.B. die Anordnungs- und Beschriftungsregeln für Dialogelemente aus dem CUA-Styleguide [Ib90] übernommen.

Da die Länge einer Webseite im Gegensatz zur Fläche eines klassischen Dialogfensters jedoch nicht begrenzt ist, wurden umfangreichere Dialoge (z.B. zur Eingabe der persönlichen Daten eines Benutzers) auf einer einzigen Webseite untergebracht, anstatt sie auf mehrere Schritte aufzuteilen. Diese Entscheidung wurde getroffen, da die Aufteilung aufgrund des mehrfachen Verbindungsaufbaus zwischen Client und Server unnötige Verzögerungen im Arbeitsfluss verursacht hätte.

Sämtliche Überprüfungen der Benutzereingaben auf Fehler oder Inkonsistenzen finden serverseitig statt. Dies bedeutet zwar eine kurze Verzögerung im Gegensatz zur clientseitigen Auswertung, wird jedoch durch zwei Gründe gerechtfertigt: Zum einen lassen sich bestimmte Fehlersituationen wie inkonsistente Eingaben nur durch Vergleich mit den auf dem Server gespeicherten Daten erkennen, zum anderen würde die Überprüfung durch den Client auf seiner Seite über HTML hinausgehende Techniken (z.B. JavaScript) erfordern, die nicht auf allen Plattformen verfügbar sind.

7.9.2 Der Entwurf der GUI-Klassenbibliothek

Neben dem eigentlichen Benutzungsoberflächen-Layout wurde eine JAVA-Klassenbibliothek erstellt, die sämtliche grafischen Details des Ausgabemediums (in unserem Beispiel HTML) in einzelnen Klassen kapseln sollte und damit folgende Ziele verfolgte:

- keine Voraussetzung von HTML-Kenntnissen bei den Programmierern der Controller und Formatter: diese Entwickler sollten nur JAVA können müssen
- Sicherstellung eines weitestgehend einheitlichen Look&Feel ohne Verwendung von Templates
- zentrale Änderbarkeit von GUI-Design-Aspekten wie Farben, Aussehen der Tabellen und Portlets, usw.

Die Realisierung dieser Ziele erfolgt durch das Erstellen zweier JAVA-Packages, der GUI-Base (Abbildung 38 - Klassendiagramm GUIBase) und der GUILib (Abbildung 39 und Abbildung 40), die aus einer Menge von Klassen bestehen und als UML Klassendiagramm dargestellt sind.

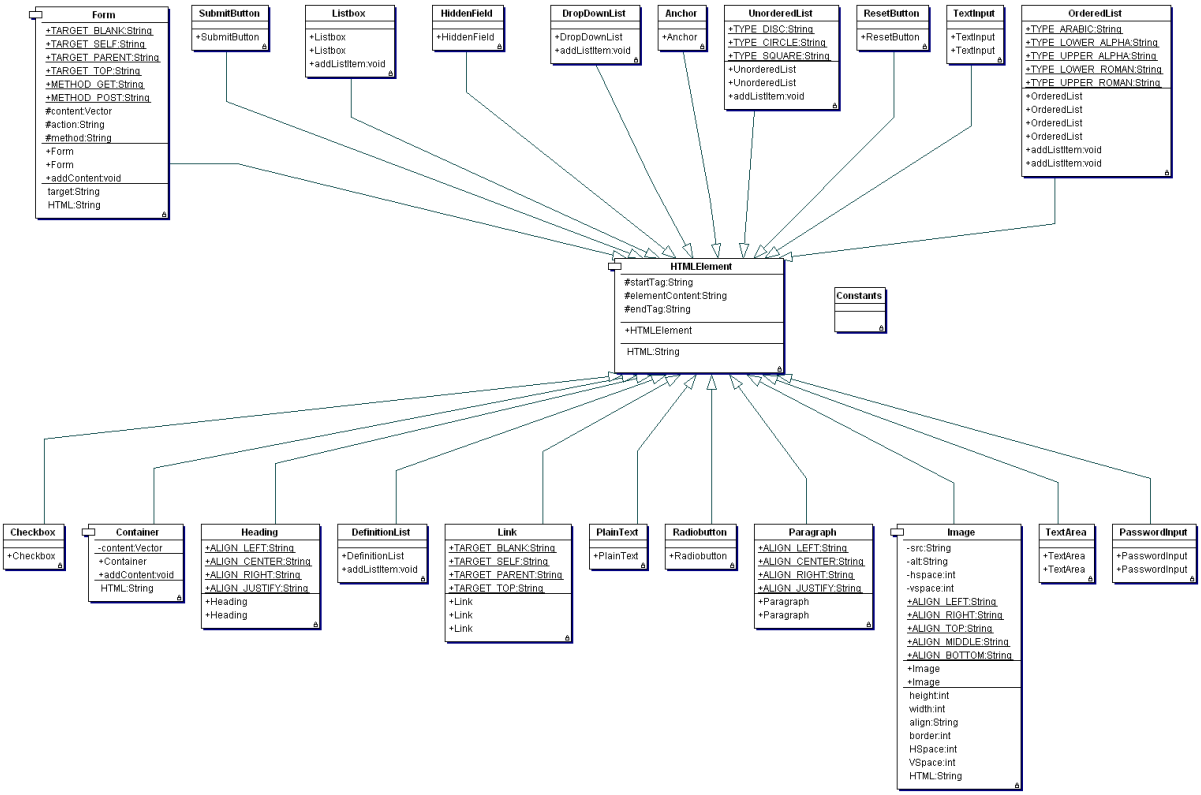


Abbildung 38 - Klassendiagramm GUIBase

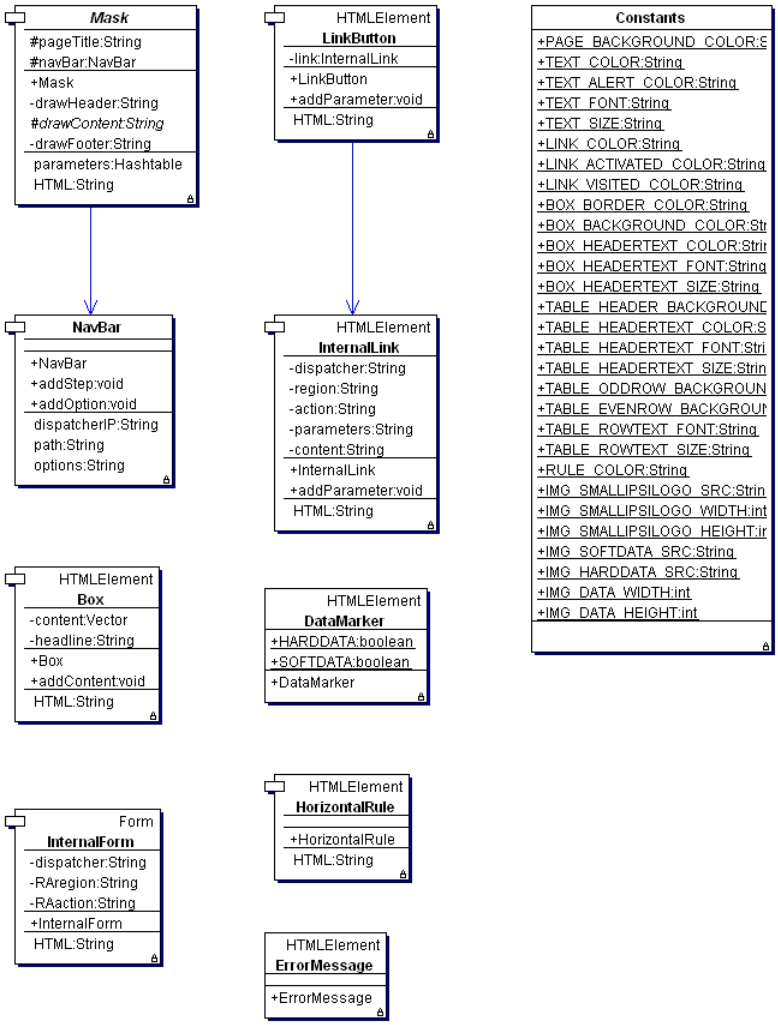


Abbildung 39 - Klassendiagramm GUILib (Teil 1)

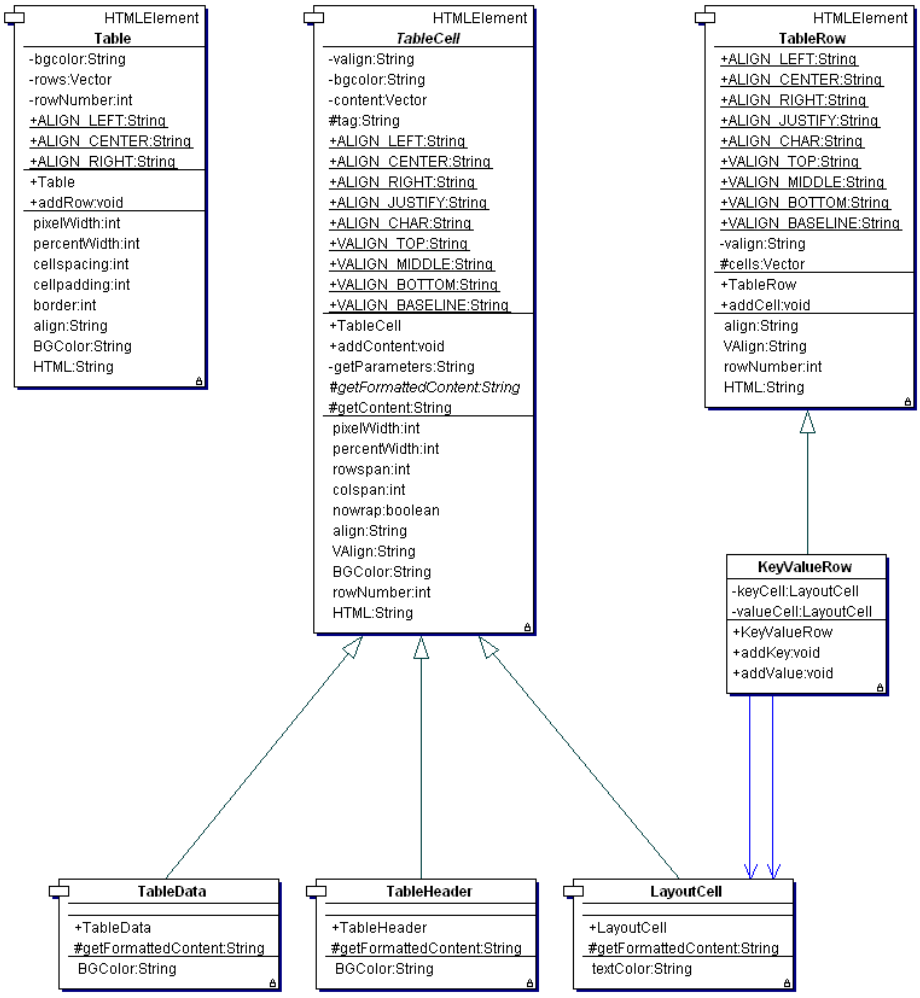


Abbildung 40 - Klassenbibliothek GUILib (Teil 2)

Die Klassen des GUI-Base Packages repräsentieren einfache Elemente der HTML-Syntax wie Links, Textfelder, Listen, etc. Jede dieser GUI-Base Klassen erbt von der abstrakten Klasse HTMLElement. Insgesamt werden atomare Elemente (wie Links oder Textfelder) und Container (nichtatomare Elemente wie Tabellen, Tabellenzeilen, Formulare, etc., die also wieder Elemente enthalten können) unterschieden.

Während in der GUIBase eben eher einfache HTML-Elemente beschrieben werden, die auch noch kein IPSI-spezifisches Aussehen haben, beinhaltet die GUILib komplexe Elemente wie Boxen (Portlets), Tabellen, etc. die auf das Look&Feel des IPSI-Portals zugeschnitten sind.

Die folgenden Tabellen geben eine kurze Übersicht über die einzelnen Klassen der GUI-Base bzw. GUILib:

Die GUIBase enthält einfache HTML-Elemente und ist komplett unabhängig vom IPSI-Design gehalten:

Klasse	Beschreibung	HTML-Tag	Art
Anchor	seiteninternes Verweisziel	...	Atom
Checkbox	Checkbox (m aus n-Auswahl)	<input type=checkbox>	Atom
Constants	leer		
Container	keine Ausgabe, dient zur Kombination mehrerer von HTMLElement abgeleiteter Elemente		Container
DefinitionList	Definitionsliste	<dl>...</dl>	Atom
DropDownList	Dropdown-Liste (1 aus n-Auswahl)	<select>...</select>	Atom
Form	Formular	<form>...</form>	Container
Heading	Überschrift	<h...>...</h...>	Atom

HiddenField	verstecktes Formularfeld	<input type=hidden>	Atom
Image	Bild	<img>	Atom
Link	Verweis	...	Atom
Listbox	Listbox (m aus n-Auswahl)	<select multiple>...</select>	Atom
OrderedList	geordnete Liste	...	Atom
Paragraph	Textabsatz	...	Atom
PasswordInput	Texteingabefeld ohne Klartextanzeige	<input type=password>	Atom
PlainText	unformatierter Text oder roher HTML-Code	...	Atom
Radiobutton	Radiobutton (1 aus n-Auswahl)	<input type=radio>	Atom
ResetButton	Button zum Löschen von Formularen	<input type=reset>	Atom
SubmitButton	Button zum Absenden von Formularen	<input type=submit>	Atom
TextArea	mehrzeiliges Texteingabefeld	<textarea>...</textarea>	Atom
TextInput	einzeiliges Texteingabefeld	<input type=text>	Atom
UnorderedList	ungeordnete Liste	...	Atom

Die GUILib enthält komplexere HTML-Konstrukte im IPSI-Design:

Klasse	Beschreibung	Art
Box	Gerahmter Kasten mit Überschrift (in IPSI müssen alle Dialogelemente in einer Box stehen)	Container
Constants	Farb- und Schriftdefinitionen, Bildinformationen	
DataMarker	Markierung für harte bzw. weiche Daten (Bild)	Atom
ErrorMessage	Fehlermeldung	Atom
HorizontalRule	horizontale Linie	Atom
KeyValueRow	Zeile für zweispaltige Tabellen, in denen z.B. Labels und Formularfelder nebeneinander stehen	Container
LayoutCell	Tabellenzelle für Layout-Zwecke (ohne Farbgebung)	Container
Mask	abstrakte Oberklasse aller Masken	
NavBar	Inhalte der Navigationsleiste (wird nur von Mask verwendet)	
Table	Tabelle	Container
TableCell	abstrakte Tabellenzelle	Container
TableData	Datenzelle für Datentabellen (Zeilen werden wechselweise eingefärbt)	Container
TableHeader	Überschriftenzelle für Datentabellen (automatisch eingefärbt)	Container
TableRow	Tabellenzeile	Container

Die Benutzung der Klassen ist nun folgendermaßen vorgesehen:

Angenommen ein Request wird über das Core-System (s. Kapitel 7.2) an einen Controller weitergeleitet.

Diesem Controller wurde vom Dispatcher ein dem Ausgabemedium (z.B. HTML) entsprechender Formatter übergeben. Der Controller übergibt dem Formatter Steuercodes (definiert in Konstanten) in einer Hashtable als Parameter, die dem Formatter mitteilen, welche Ausgabe (z.B. Maske mit Erfolgsmeldung oder Maske mit Fehlermeldung) zu produzieren ist.

Ein Formatter selbst macht jedoch keinerlei GUI-relevante Ausgaben, sondern instanziiert je nach Steuercode des Controllers ein entsprechendes Maskenobjekt. Maskenobjekte sind Objekte einer Maskenklasse, die wiederum von der Klasse Mask erbt. Ein solches Maskenobjekt übernimmt in der Methode drawContent() die komplette Darstellung einer Bildschirmmaske, die auch genau einer Maske in der MVD-Beschreibung (s. Kapitel 7.10) entspricht.

D.h. in den Maskenobjekten werden nun die Klassen der GUI-Base und der GUILib benutzt, um die Bildschirmmaske, die dem Benutzer angezeigt werden soll, zusammenzusetzen. Dazu werden die benötigten GUI-Klassen instanziiert (in der Regel immer eine Tabelle, die ein NavBar-Objekt und ein oder mehrere Box-Objekte enthält, die dann mit entsprechenden weiteren Elementen wie Listen, Links, Texten usw. gefüllt werden) und aggregiert (z.B. Tabellenzellen-Objekte in Tabellen einsetzen etc.). Schließlich wird das Maskenobjekt die Methode getHTML() in dem obersten Containerobjekt (der übergeordneten Tabelle, in der alle anderen Objekte angeordnet sind) aufrufen, um den gesamten HTML-Code für die Maske zu produzieren. Dies geschieht durch rekursives Aufrufen dieser Methode getHTML (die in der abstrakten Klasse HTMLElement definiert ist und somit von jedem Element zur Verfügung gestellt wird) entlang der Aggregationshierarchie der Objekte in der darzustellenden Maske. Die HTMLElement-Objekte (also Links, Tabellen, usw.) „wissen“, wie sie sich in

HTML-Form darzustellen haben. Diese Darstellung kann über Zugriffsmethoden beeinflusst werden; insgesamt soll aber sichergestellt werden, dass das Standardverhalten der Objekte zu einer IPSI-konformen Ausgabe führt.

Somit erhält der Controller mit einem Aufruf von getHTML() in der instanziierten Maske auch den gesamten HTML-Code, der zur Darstellung einer Bildschirmmaske nötig ist (denn Masken erben auch von HTMLElement), als String und kann diesen entlang der Aufrufhierarchie zurück über den Controller und Dispatcher zurück zum Servlet leiten, wo schließlich der HTML-Code an den aufrufenden Browser zurückgegeben wird.

Der Aufbau eines Maskenobjektes und die Benutzung der GUIBase/GUILib-Klassen soll durch ein Codebeispiel, das eine Maskenklasse darstellt, verdeutlicht werden. Die umfangreichen Erläuterungen finden sich in den Quelltextkommentaren:

Verwendung von GUIBase/GUILib am Beispiel der Prototyp-Maske „Termin eintragen“:

```
package de.ipsi.formatter.html.mask;
import de.ipsi.guibase.*;
import de.ipsi.guilib.*;
/**
 * BEISPIEL: Generierung der Maske „Termin eintragen“ aus dem Prototyp
 * mit Hilfe der GUIBase- und GUILib-Methoden
 */
public class TerminEintragen extends Mask {

  /**
   * Konstruktor: Seitentitel und Navigationsleiste definieren
   */
  public TerminEintragen() {
    // Seitentitel definieren
    pageTitle = „IPSI: Termin eintragen“;

    // Navigationsleiste definieren: 1. Zeile (Pfad zur aktuellen Maske)
    navBar.addStep(„Terminkalender“, „„);
    // Navigationsleiste definieren: 2. Zeile (von hier erreichbare Masken)
    navBar.addOption(„Monatsansicht“, „<URL – to be discussed>„);
    navBar.addOption(„Tagesansicht“, „<URL – to be discussed>„);
    navBar.addOption(„Termin eintragen“, „„);
    navBar.addOption(„Termin löschen“, „<URL – to be discussed>„);
    navBar.addOption(„ToDo-Liste“, „<URL – to be discussed>„);
  };

  /**
   * Maskeninhalt erzeugen
   */
  protected String drawContent() {
    // Box
    Box box = new Box(„Neuen Termin eintragen“);
      // Formular
      Form form = new Form („servlet“, Form.METHOD_POST);
        // Tabelle für Formular-Layout
        Table table = new Table ();
          // Zeile für Titel-Eingabe
          KeyValueRow titelZeile = new KeyValueRow (false);
            PlainText titelLabel = new PlainText („Titel:“);
            titelZeile.addKey(titelLabel);
            TextInput titelFeld = new TextInput („titel“, „„, 30);
            titelZeile.addValue(titelFeld);
          table.addRow(titelZeile);
          // Zeile für Person-Link
          KeyValueRow personZeile = new KeyValueRow (false);
            PlainText personLabel = new PlainText („Person:“);
            personZeile.addKey(personLabel);
            Link personLink = new Link („<URL – to be discussed>„,
                                        parameers.get(„person“).toString());
            personZeile.addValue(personLink);
          table.addRow(personZeile);
          // Zeile für Datum-Eingabe
          KeyValueRow datumZeile = new KeyValueRow (false);
            PlainText datumLabel = new PlainText („Datum:“);
            datumZeile.addKey(datumLabel);
            TextInput datumFeld = new TextInput („datum“, „„, 10, 10);
            datumZeile.addValue(datumFeld);
          table.addRow(datumZeile);
```

```
          // Zeile für Beginnzeit-Eingabe
          KeyValueRow beginnZeile = new KeyValueRow (false);
            PlainText beginnLabel = new PlainText („Beginn:“);
            beginnZeile.addKey(beginnLabel);
            TextInput beginnFeld = new TextInput („beginn“, „„, 5, 5);
            beginnZeile.addValue(beginnFeld);
          table.addRow(beginnZeile);
          // Zeile für Endzeit-Eingabe
          KeyValueRow endeZeile = new KeyValueRow (false);
            PlainText endeLabel = new PlainText („Ende:“);
            endeZeile.addKey(endeLabel);
            TextInput endeFeld = new TextInput („ende“, „„, 5, 5);
            endeZeile.addValue(endeFeld);
          table.addRow(endeZeile);
          // Zeile für Reminder-Eingabe
          KeyValueRow reminderZeile = new KeyValueRow (false);
            PlainText reminderLabel = new PlainText („Benachrichtigung:“);
            reminderZeile.addKey(reminderLabel);
            Checkbox reminderCheckbox = new Checkbox („reminderactive“,
checked“, false);
            reminderZeile.addValue(reminderCheckbox);
            PlainText reminderAktivieren = new PlainText („aktivieren: „);
            reminderZeile.addValue(reminderAktivieren);
            DropDownList reminderTimeDropDown = new DropDownList („reminder
ime“);
              reminderTimeDropDown.addListItem(„0“, true, „am Termin“);
             reminderTimeDropDown.addListItem(„5“, false, „5 Minuten vor-
her“);
              reminderTimeDropDown.addListItem(„10“, false, „10 Minuten vor-
her“);
              reminderTimeDropDown.addListItem(„15“, false, „15 Minuten vor-
her“);
              reminderTimeDropDown.addListItem(„30“, false, „30 Minuten vor-
her“);
              reminderTimeDropDown.addListItem(„60“, false, „1 Stunde vor-
her“);
              reminderTimeDropDown.addListItem(„1440“, false, „1 Tag vor-
her“);
            reminderZeile.addValue(reminderTimeDropDown);
            PlainText reminderVia = new PlainText („ via „);
            reminderZeile.addValue(reminderVia);
            DropDownList reminderMediumDropDown = new DropDownList („remin-
dermedium“);
              reminderMediumDropDown.addListItem(„screen“, true, „Bild-
schirm“);
              reminderMediumDropDown.addListItem(„fax“, true, „Telefax“);
              reminderMediumDropDown.addListItem(„email“, true, „Email“);
              reminderMediumDropDown.addListItem(„sms“, true, „SMS“);
            reminderZeile.addValue(reminderMediumDropDown);
          table.addRow(reminderZeile);
          // Zeile für Notizen-Eingabe
          KeyValueRow notizenZeile = new KeyValueRow (false);
            PlainText notizenLabel = new PlainText („Notizen:“);
            notizenZeile.addKey(notizenLabel);
            TextArea notizenFeld = new TextArea („remindernotes“, 30, 5,
„„);
            notizenZeile.addValue(notizenFeld);
          table.addRow(notizenZeile);
          // Zeile für Buttons
          KeyValueRow buttonZeile = new KeyValueRow (false);
            PlainText buttonLabel = new PlainText („Änderungen:“);
            buttonZeile.addKey(buttonLabel);
            SubmitButton buttonSubmit = new SubmitButton („submit“, „OK“);
            buttonZeile.addValue(buttonSubmit);
            ResetButton buttonReset = new ResetButton („Zurücksetzen“);
            buttonZeile.addValue(buttonReset);
            SubmitButton buttonBack = new SubmitButton („submit“, „Abbre-
chen“);
            buttonZeile.addValue(buttonBack);
          table.addRow(buttonZeile);
        form.addContent(table);
      box.addContent(form);

    String output = box.getHTML();

    return output;
  };
```

```
}
```

Aufruf im Formatter:

```
// Maske anlegen
TerminEintragen terminEintragen = new TerminEintragen();
// Parameter wie BOs etc. in der Hashtable parameters übergeben
terminEintragen.setParameters(parameters);
// HTML-Code der Maske auslesen
String output = terminEintragen.getHTML();
```

Auf diese Weise, nämlich dadurch, dass nun Formatter und auch Maskenobjekte selber im Wesentlichen keinen eigenen HTML-Code produzieren müssen, sondern diesen von den entsprechenden HTMLElement-Objekten geliefert bekommen (sprich diese HTMLElement-Objekte kapseln den HTML-Code), werden genau die drei o.g. Ziele erreicht.

7.10 Masken-Verarbeitung-Datenfluss-Diagramme (MVDs)

Neben den oben dargestellten Klassendiagrammen wurden im Entwurf sogenannte Masken-Verarbeitung-Datenfluss-Diagramme (MVDs) erzeugt, die auf einer eigenen semiformalen Notation beruhen.

Zweck der MVDs ist es, den Inhalt jeder Bildschirmmaske darzustellen und die Verarbeitungsschritte, die zwischen der Darstellung zweier Masken liegen. Weiterhin kann der Datenfluss zwischen Masken und Verarbeitungsschritten abgebildet werden.

Diese Informationen waren sehr hilfreich in der Implementierungsphase, denn aus ihnen konnten die Entwickler entnehmen

- welchen Inhalt eine Bildschirmmaske hatte; d.h. welche Klassen der GUILib bzw. GUIBase-Bibliotheken in den Maskenklassen (s. 7.9.2) zu benutzen waren,
- welche Parameter von der Maske bei bestimmten Aktionen an einen Controller geliefert werden müssen; dabei wird ein Verarbeitungsschritt als ein Controller umgesetzt,
- welche Aktionen ein Controller durchzuführen hat (beschrieben in den Verarbeitungsschritten),
- welche Masken ein Formatter darstellen können muss (nämlich alle Masken, die von einem Verarbeitungsschritt aus über Kanten erreichbar sind).

Die MVDs wurden als Anleitung für die Entwickler genutzt.

Die Syntax der MVDs sieht folgende Elemente vor:

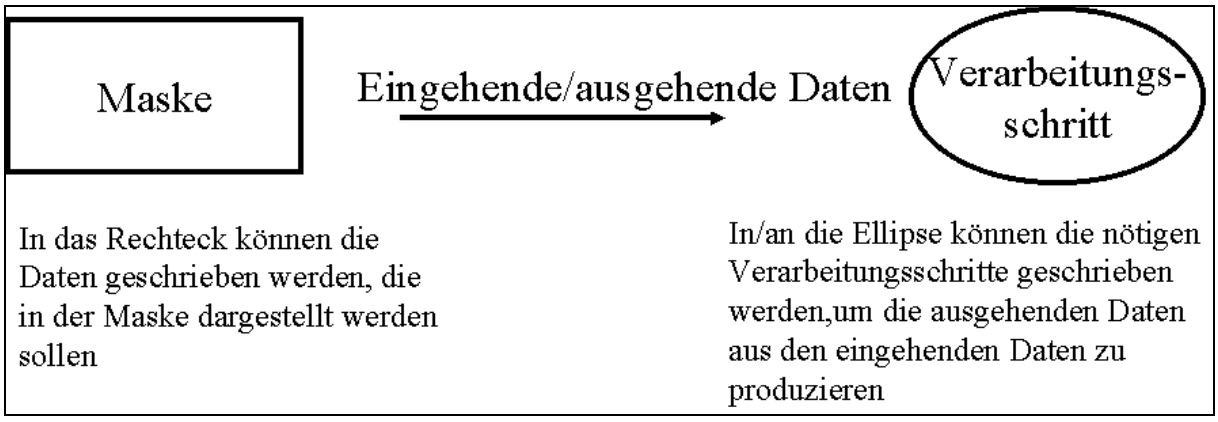


Abbildung 41 - Notationselemente der MVDs

Eine Maske wird durch ein Rechteck dargestellt, in das alle diejenigen Daten geschrieben werden, die in der Maske dargestellt werden. Jede Maske wird später in der Implementierung zu einer Maskenklasse.

Ein Verarbeitungsschritt wird durch eine Ellipse dargestellt. In die Ellipse werden alle Aktionen geschrieben, die nötig sind, um von den in den Verarbeitungsschritt eingehenden Masken zu einer von dem Verarbeitungsschritt ausgehenden Maske zu gelangen. Ein Verarbeitungsschritt wird in genau einen Controller in der Implementierung umgesetzt.

Die Masken werden mit den Verarbeitungsschritten über gerichtete Kanten verbunden. An die Kanten werden die Daten/Parameter geschrieben, die von der Maske an den Verarbeitungsschritt gereicht werden bzw. vom Verarbeitungsschritt an die Maske. Da es sich bei MVDs um bipartite Graphen handelt, können keine Masken mit Masken oder Verarbeitungsschritte mit Verarbeitungsschritten direkt verbunden werden. Wenn eine Maske mit einem Verarbeitungsschritt verbunden ist, so bedeutet dies, dass man von der Maske genau den durch den Verarbeitungsschritt beschriebenen Controller ansprechen muss und ihm die entsprechenden Daten als HTTP-Parameter übergeben muss.

Ist ein Verarbeitungsschritt mit einer oder mehreren Masken verbunden, so bedeutet dies, dass der durch den Verarbeitungsschritt spezifizierte Controller an den mit ihm verbundenen Formatter (s. Kapitel 7.2) die an die jeweilige Kante geschriebenen Parameter liefern muss. Weiterhin muss dieser Formatter je nach Steuercode des Controllers eine der Maskenklassen erzeugen und ansprechen, die durch Kanten mit dem Verarbeitungsschritt im MVD verbunden sind. D.h. ein Formatter muss all die Maskenobjekte erzeugen können, die im MVD von dem jeweiligen Controller aus erreichbar sind. Welche Maske nun angesprochen wird, wird einem Formatter immer durch einen Steuercode mitgeteilt.

Im Folgenden werden die MVDs der einzelnen Subsysteme dargestellt, die den o.g. abstrakten Sachverhalt verdeutlichen werden. Zu jedem Diagramm wird dazu eine überblicksmäßige Beschreibung gegeben.

7.10.1 MVDs zum Subsystem Administration

Das Subsystem Administration enthält Funktionen zur Benutzerauthentifizierung, Benutzer-, Agentur- und Feedbackverwaltung sowie zur Statistik, deren Abläufe in den folgenden MVDs dargestellt sind.

Das Gesamtdiagramm wurde in 4 Teildiagramme aus Gründen der Übersichtlichkeit aufgeteilt. Das erste Diagramm beschreibt den Vorgang der Benutzerauthentifizierung und den Aufbau des Startbildschirms (aus Sicht des Subsystems Administration). Das zweite Diagramm zeigt die Benutzerverwaltung, das dritte die Vorgänge zur Statistik und Feedbackverwaltung, und das letzte Diagramm beschreibt die Agenturverwaltung.

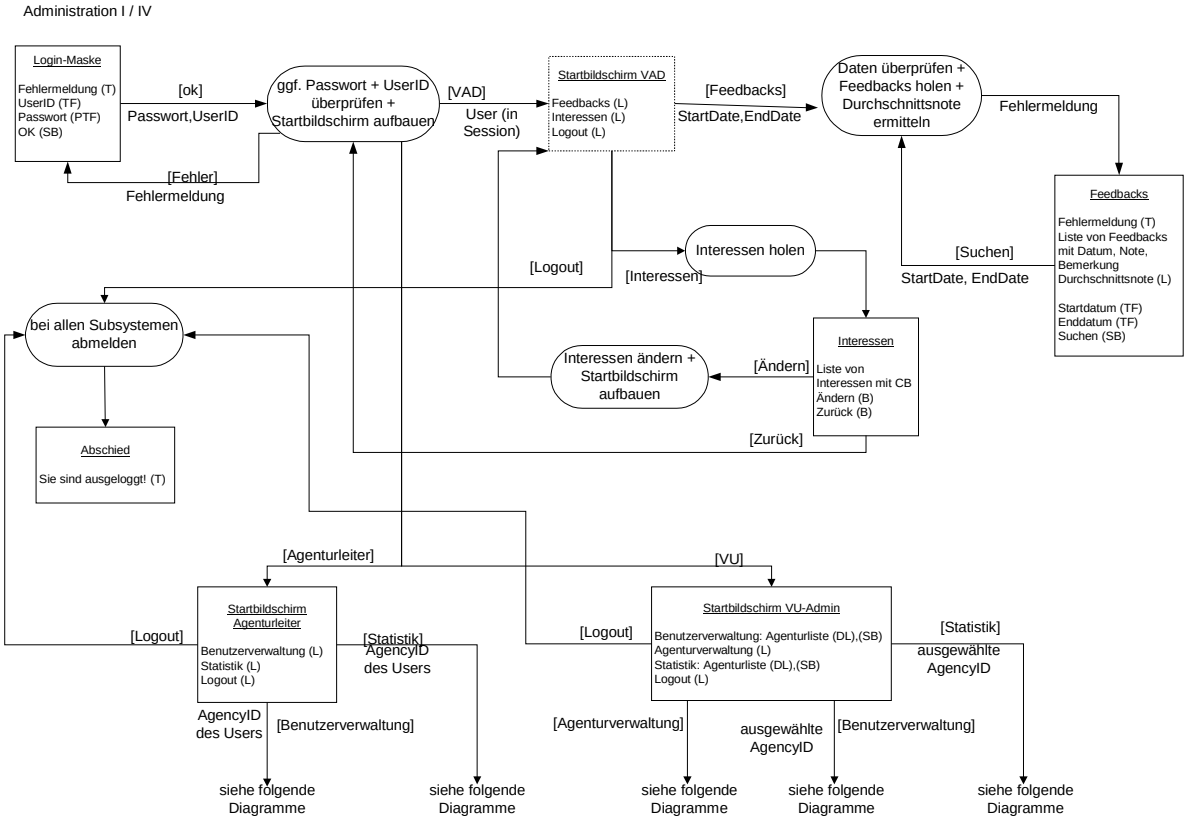


Abbildung 42 - MVD Subsystem Admin (1)

In der Login-Maske muss sich der Benutzer mit Benutzername und Passwort anmelden. Nach erfolgreicher Authentifizierung gelangt er in den Startbildschirm. Dieser unterscheidet sich nach den Rechten des Benutzers: VADs gelangen in den Startbildschirm VAD, Agenturleiter in den Startbildschirm Agenturleiter und VU-Admins in den Startbildschirm VU-Admin.

Der VAD hat die Möglichkeit, vom Startbildschirm aus in die Feedback- oder auch in die Interessensverwaltung zu gelangen. Außerdem kann er sich über den Link „Abmelden“ wieder vom System abmelden. Bei Auswahl des Links „Feedbacks“ gelangt der VAD in die Feedbackverwaltung. Hier findet er eine Liste mit Feedbacks vor, sowie ein Suchformular zur Suche und Filterung bestimmter Feedbacks. Diese werden dann wiederum auf dieser Seite dargestellt. In der Interessensverwaltung kann sich der VAD seine Interessen anschauen und ggf. ändern.

Ein Agenturleiter kann über den Startbildschirm in die Benutzerverwaltung sowie in die Statistik gelangen. Diese Funktionen werden in den Diagrammen 2 und 3 beschrieben. Auch der VU-Admin kann über seinen Startbildschirm die Benutzerverwaltung und die Statistik erreichen, allerdings muss er dazu eine Agentur aus der Liste auswählen. Außerdem existiert ein Link zur Agenturverwaltung. Diese wird in Diagramm 4 beschrieben.

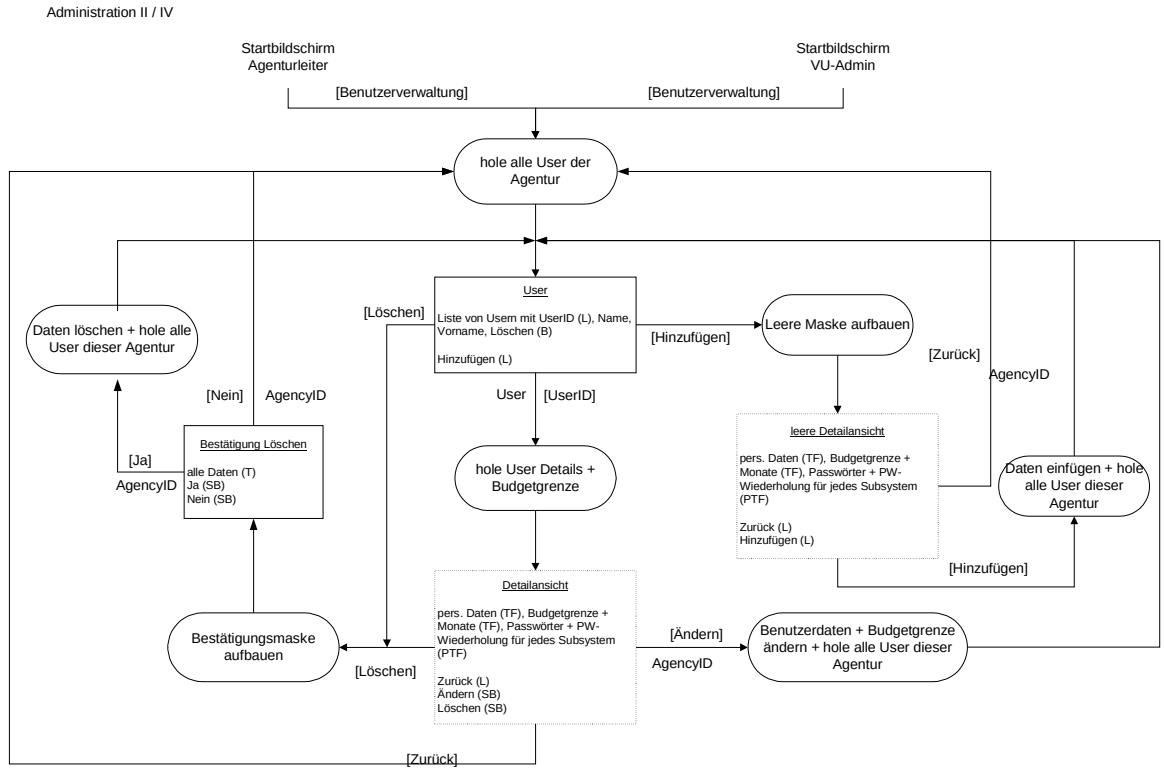


Abbildung 43 - MVD Subsystem Admin (2)

Der zentrale Bestandteil der Benutzerverwaltung ist die User-Maske. In dieser werden alle Benutzer der Agentur mit UserID, Name und Vorname in einer Liste dargestellt. Die UserID ist zugleich ein Link zur Detailansicht. Hier kann man die persönlichen Daten, die Budgetgrenze, die Benutzernamen und Passwörter einsehen und ggf. auch ändern. In der User-Maske befindet sich hinter jedem Benutzer auch noch ein Löschen-Button, mit dem man diesen User löschen kann. Es folgt allerdings noch eine Bestätigungs-Maske. Nach erfolgreichem Löschen gelangt man wieder in die User-Maske. Hier existiert auch noch ein Link zum Hinzufügen von Benutzern, über den man in eine leere Detailansicht gelangt. In dieser Maske kann man alle Daten zu einem User eintragen, also persönliche Daten, Budgetgrenze, Passwörter und Benutzernamen (für jedes Subsystem). Nach erfolgreichem Hinzufügen erscheint wieder die User-Maske.

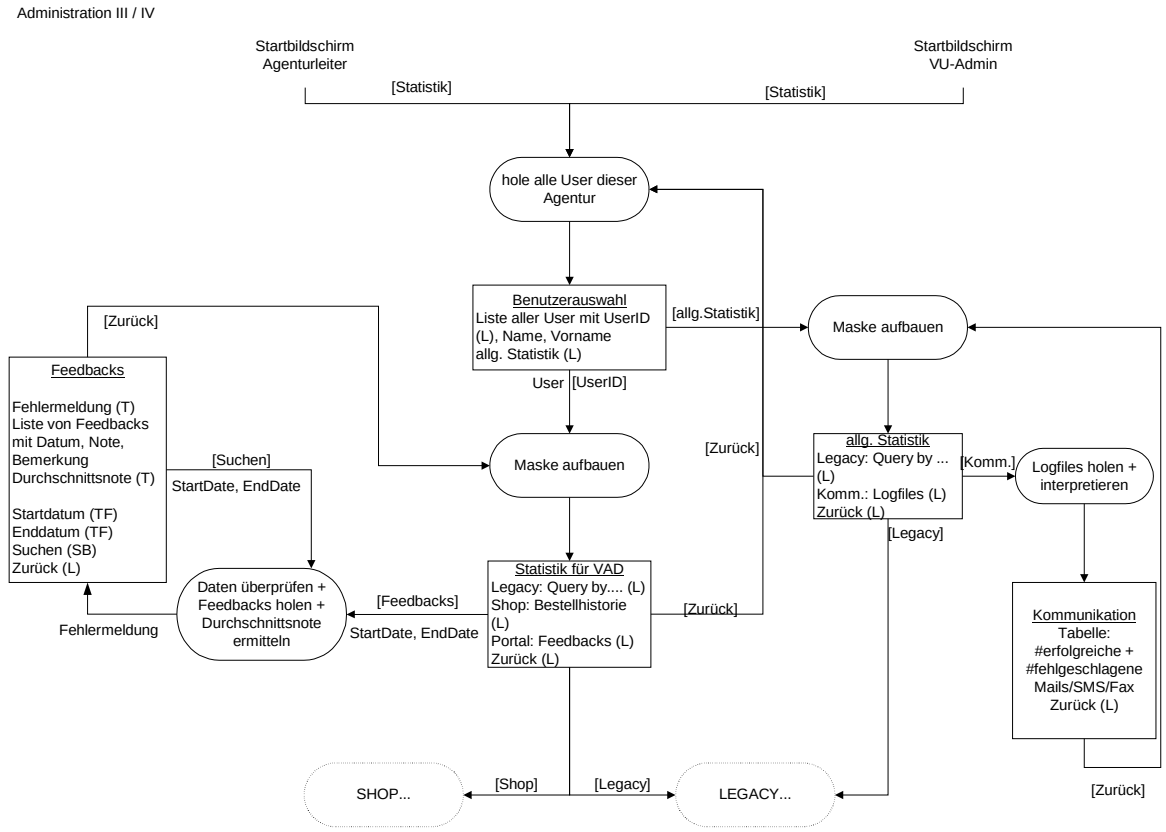


Abbildung 44 - MVD Subsystem Admin (3)

In der Statistik gibt es zwei Zweige: die Einzelstatistik für einen bestimmten Benutzer und die allgemeine Statistik für die gewählte Agentur. In der Maske „Benutzerauswahl“ kann man entweder einen Benutzer aus der Liste aller User der Agentur auswählen und gelangt so in den Bildschirm für die Einzelstatistik, oder man aktiviert den Link „Allgemeine Statistik“, über den man in die Statistik für die ausgewählte Agentur gelangt.

In der Einzelstatistik existieren verschiedene Links. Die „Query by...”-Links führen in das Subsystem Legacy. Diese Masken sind in dem entsprechenden Abschnitt für die Legacy-MVDs beschrieben. Der Link „Bestellhistorie“ führt in das Subsystem Procurement. Hier kann man die Bestellhistorie des Benutzers einsehen. Auch hierfür schlage man in dem entsprechenden Abschnitt nach. Über den Link „Feedbacks“ erreicht man die Feedback-Verwaltung, die schon weiter oben bei der Beschreibung von Diagramm 1 behandelt wurde.

Auch in der Maske „Allgemeine Statistik“ existieren einige Links. Die „Query by...“-Links führen wieder ins Subsystem Legacy, und über den Link „Logfiles“ kann man ins Subsystem Kommunikation gelangen, wo der Benutzer sich alle Logfiles, die dieses Subsystem für die Agentur angelegt hat, anschauen kann.

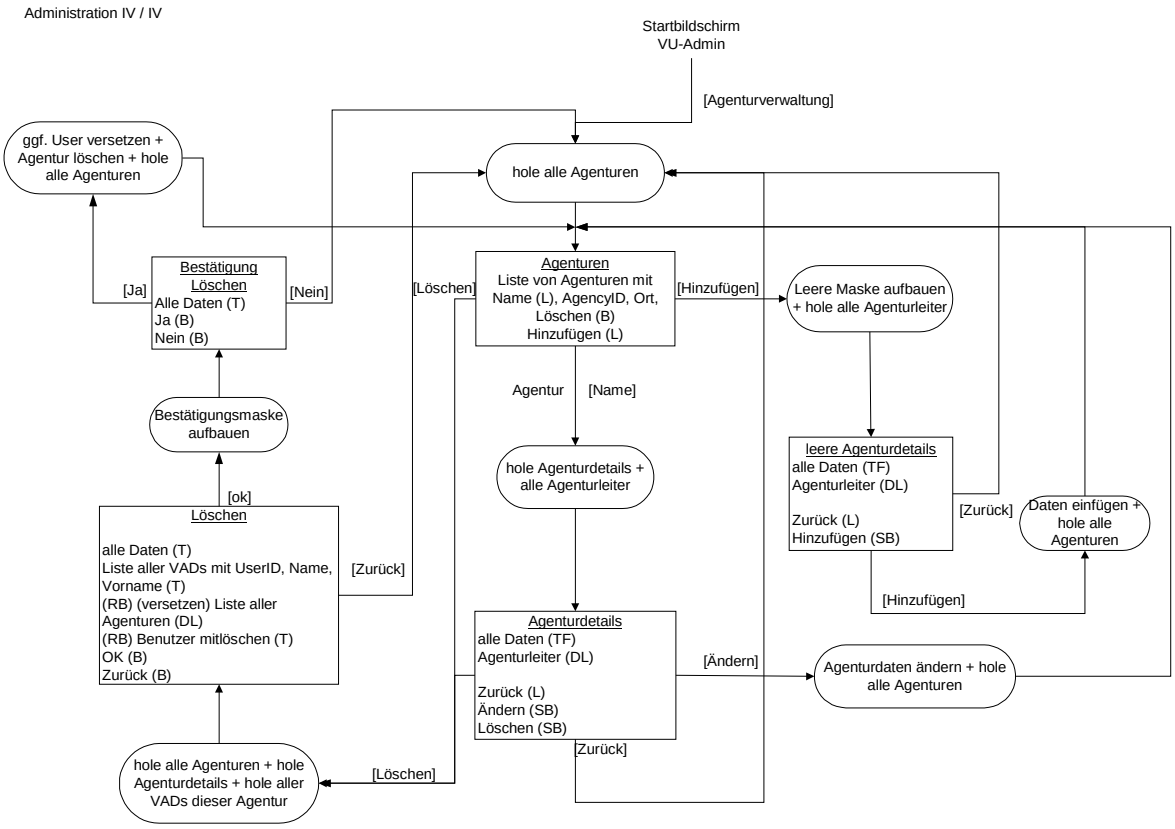


Abbildung 45 - MVD Subsystem Admin (4)

Die Agenturverwaltung ist analog zur Benutzerverwaltung aufgebaut. Der zentrale Bestandteil ist die Agenturen-Maske. Hier sind alle Agenturen in einer Liste aufgeführt. Mittels Klick auf den Agenturnamen gelangt man in die Detailansicht, in der alle Daten der Agentur eingesehen und ggf. auch geändert werden können. Man kann auch den Agenturleiter ändern, indem man einen Leiter aus der Liste aller Agenturleiter auswählt. Es existiert auch noch ein Löschen-Button, über den man in den Löschen-Dialog gelangt. Diesen erreicht man auch über die Agenturen-Maske. In der Löschen-Maske sind noch mal alle Daten sowie alle Benutzer der Agentur aufgeführt. Man kann nun entscheiden, ob die Benutzer auch gelöscht oder in eine andere Agentur versetzt werden sollen. Diese Agentur kann man aus einer Liste auswählen. Hat man die Auswahl getroffen, wird durch den Klick auf OK die Aktion durchgeführt. Man gelangt dann wieder in die Agenturen-Maske. Der Benutzer kann auch eine neue Agentur anlegen. Über den Link „Hinzufügen“ erreicht man eine leere Detailansicht, in der er dann alle Daten eintragen und einen Agenturleiter bestimmen kann. Nach dem Hinzufügen wird dann wieder die Agenturen-Maske dargestellt.

7.10.2 MVDs zum Subsystem Office

Das Subsystem Office enthält Funktionen zur Verwaltung von Kontakten, Terminen, Aufgaben und e-Mails, deren Abläufe in den folgenden MVDs dargestellt sind. Das Gesamtdiagramm wurde in 5 Teildiagramme aus Gründen der Übersichtlichkeit aufgeteilt. Diese Aufteilung entspricht den vier wesentlichen Geschäftsobjekten des Subsystems Office: Kontakte (1 und 2), Termine (3), Aufgaben (4) und e-Mails (5).

Grundsätzlich bietet bereits die Startseite eine gewisse Übersicht über eine kleine Anzahl aktueller e-Mails, Aufgaben und Termine, so dass sich jeweils zwei Einstiegsmöglichkeiten bieten: Entweder per Klick auf das Geschäftsobjekt einen direkten Übergang zu dessen Detailansicht oder der Weg über die Gesamtübersicht des Bereiches. Nur beim Bereich „Kontakte“ kann ausschließlich der Weg über die Gesamtübersicht gegangen werden, da eine Auswahl von Kontakten auf der Startseite nicht sinnvoll ist.

Wegen der (möglichen) Menge der Daten kann in den Gesamtübersichten für Kontakte und Termine nicht alles auf einmal angezeigt werden. Bei Kontakten wird ein Buchstabenbereich angezeigt, bei Terminen eine Monatsübersicht. Von dort aus kann dann von Monat zu Monat geblättert oder jede beliebige Tagesansicht angesteuert werden.

Die Gesamtübersichten bieten einen Überblick über die Geschäftsobjekte und von hier aus geht es auch zu den entsprechenden Detailansichten. Ausgehend von der Gesamtübersicht bieten die Bereiche „Aufgaben“ und „e-Mails“ dazu noch einen Link zu einer speziellen Suchmaske. Von den Ergebnissen der Suche kann natürlich auch die Detailansicht erreicht werden.

In allen Bereichen ist es selbstverständlich möglich Objekte neu anzulegen, sie zu ändern oder zu löschen (außer e-Mails: nur Ansicht). Das Löschen führt im Arbeitsablauf immer über eine Sicherheitsabfrage. Die Gesamtübersichten fungieren dabei als Knotenpunkt. Von hier aus kann zur Maske „Erstellen“, oder „Ändern“ gelangt werden. Von den Detailansichten aus sind „Löschen“ und „Ändern“ erreichbar.

Eine Besonderheit gilt es noch beim Bereich „Kontakte“ zu beachten. Zu einem Kontakt können noch sogenannte „Benutzerdefinierte Felder“ definiert werden. Dafür gibt es eigene Masken, die von der Maske „Kontakt ändern“ bzw. der Maske „Kontakte erstellen“ erreichbar sind.

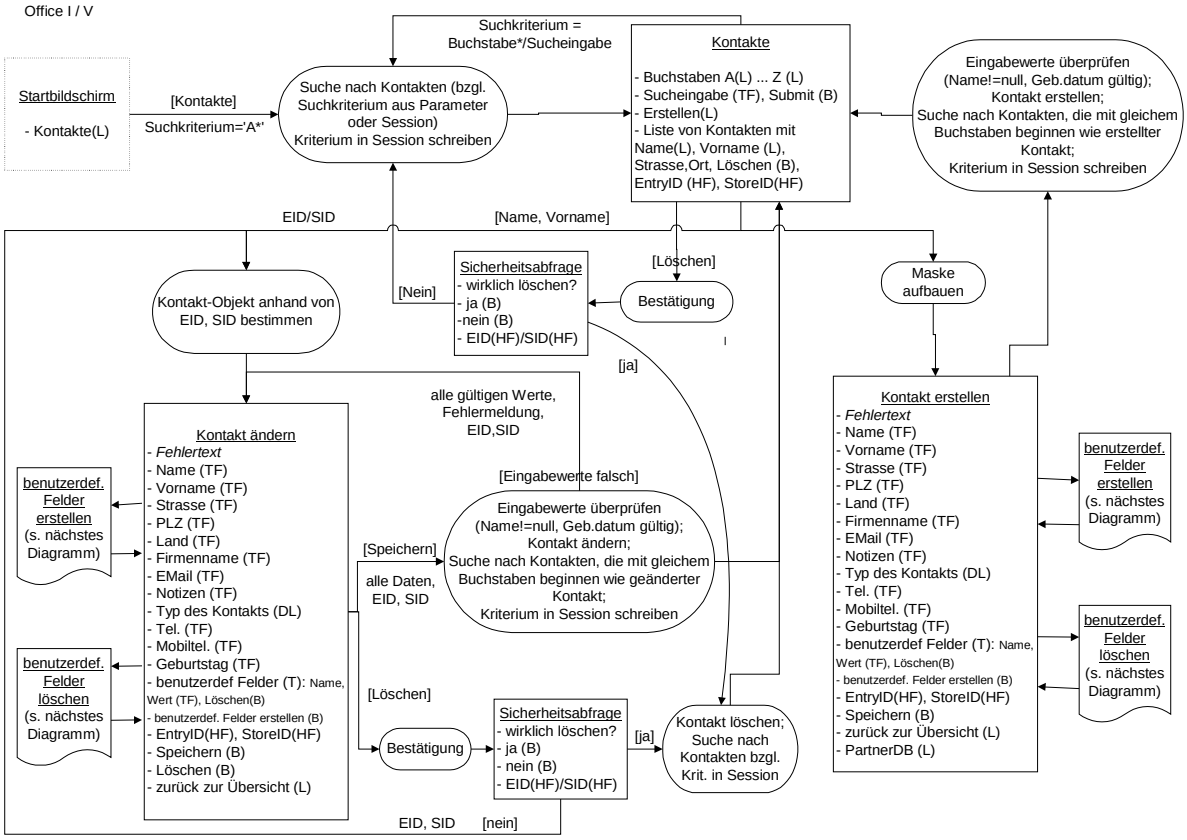


Abbildung 46 - MVD Subsystem Office (1)

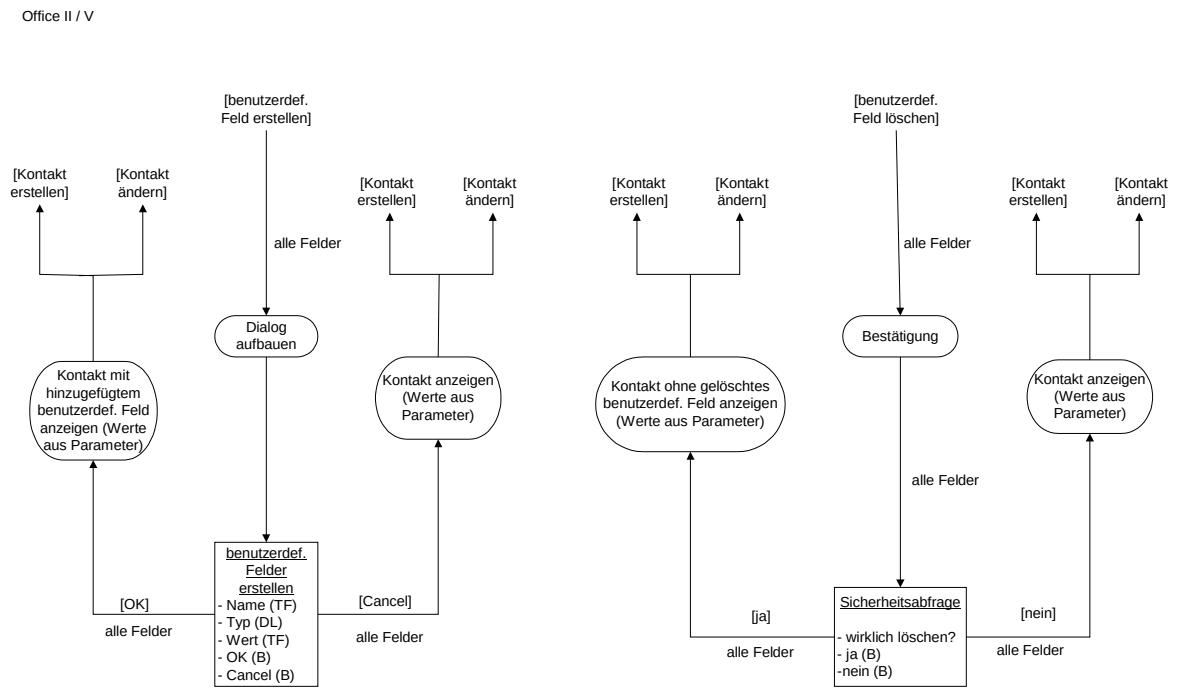


Abbildung 47 - MVD Subsystem Office (2)

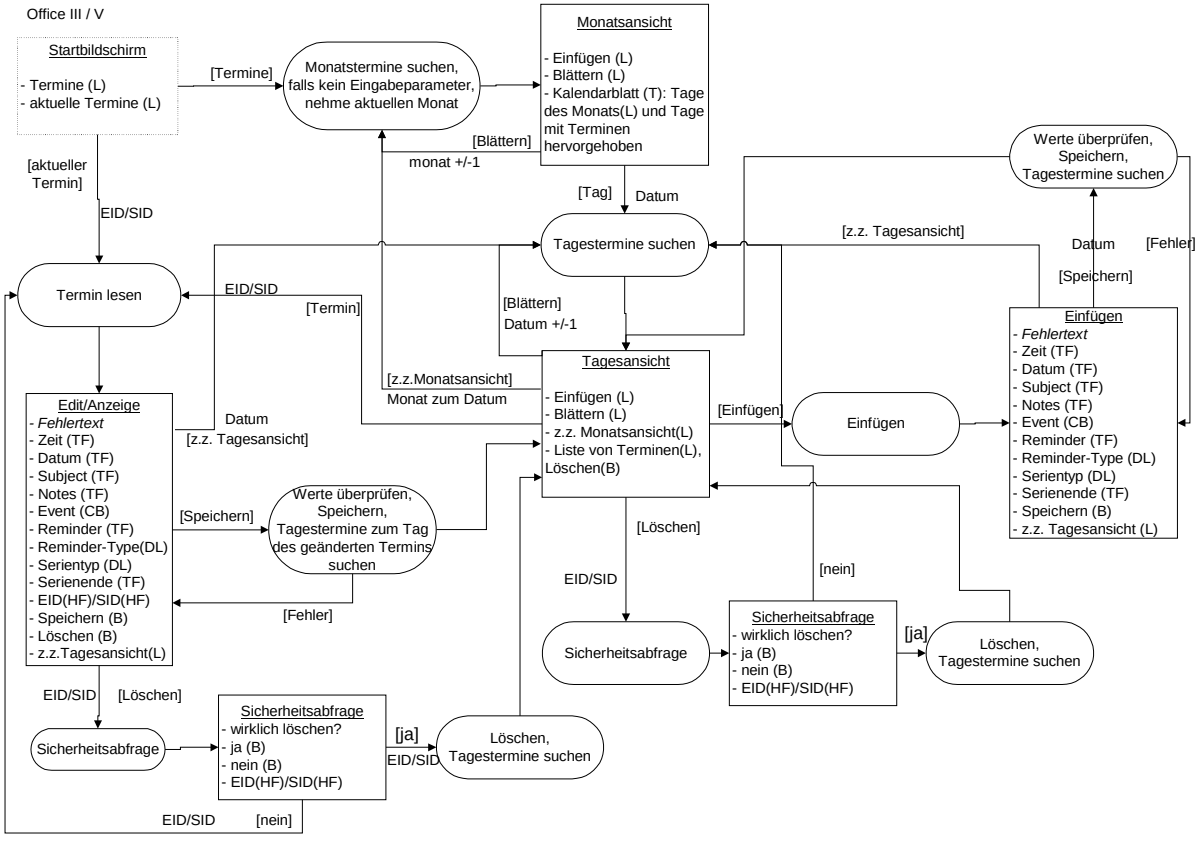


Abbildung 48 - MVD Subsystem Office (3)

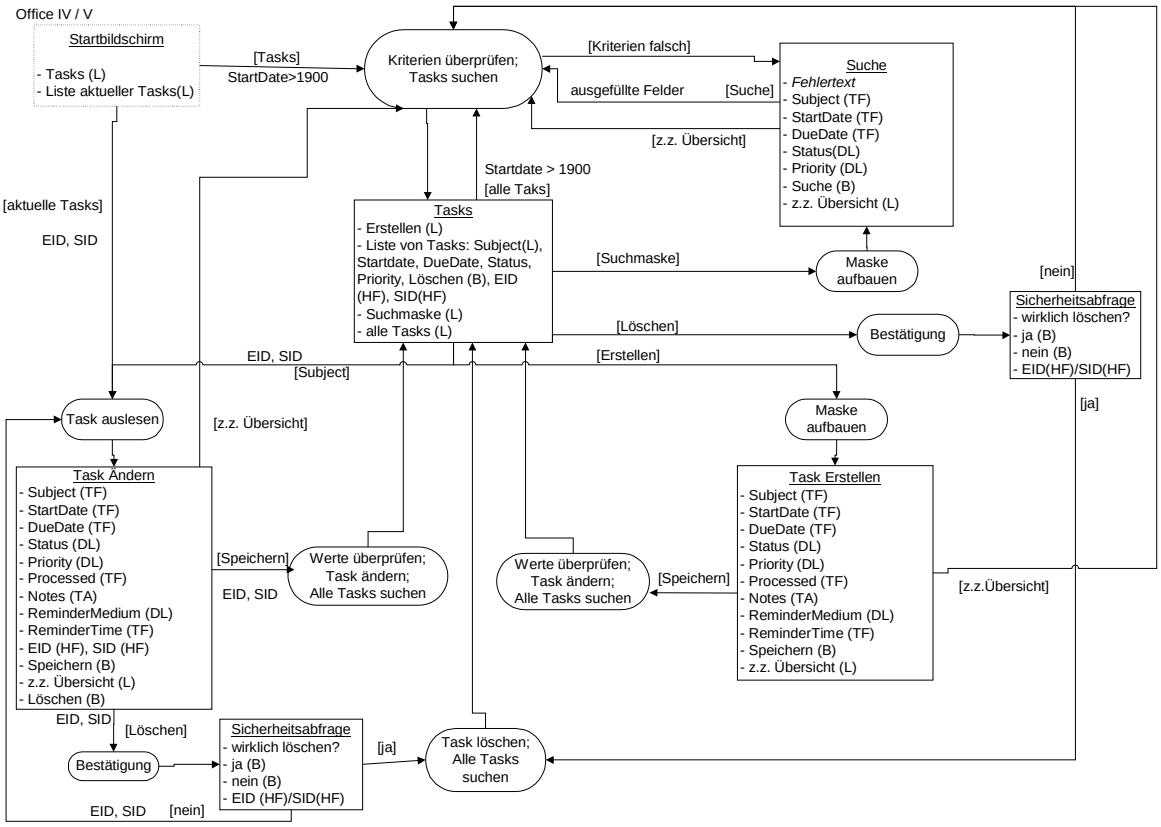


Abbildung 49 - MVD Subsystem Office (4)

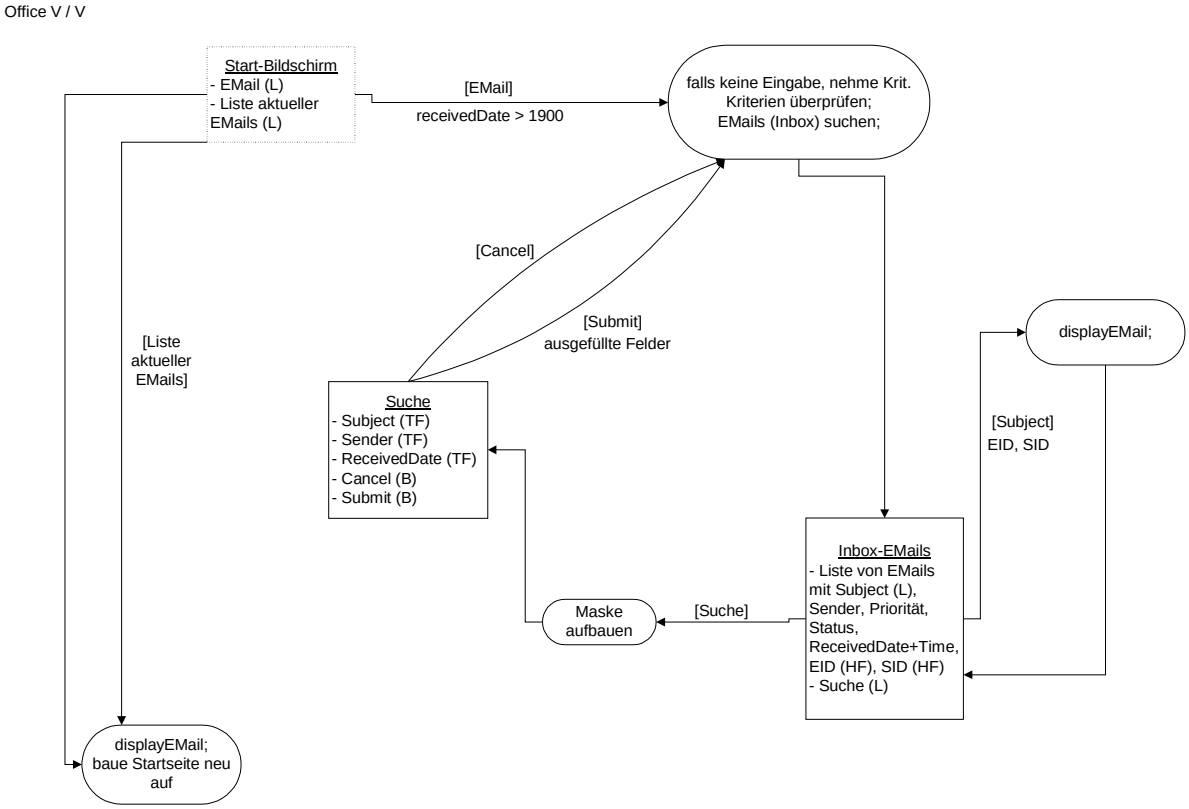


Abbildung 50 - MVD Subsystem Office (5)

7.10.3 MVDs zum Subsystem Procurement

Das Subsystem Procurement stellt dem Benutzer die Shop-Funktionalitäten des Portals zur Verfügung.

Die nachfolgenden MVDs stellen die Funktionsweise des Shopsystems und die möglichen Interaktionen mit dem Benutzer dar. Das gesamte MVD wurde in drei Teile aufgeteilt. Das erste Diagramm stellt die Kernfunktionalität, das zweite die Suchfunktionalität und das dritte schließlich die Funktionalitäten des Statistikbereichs dar.

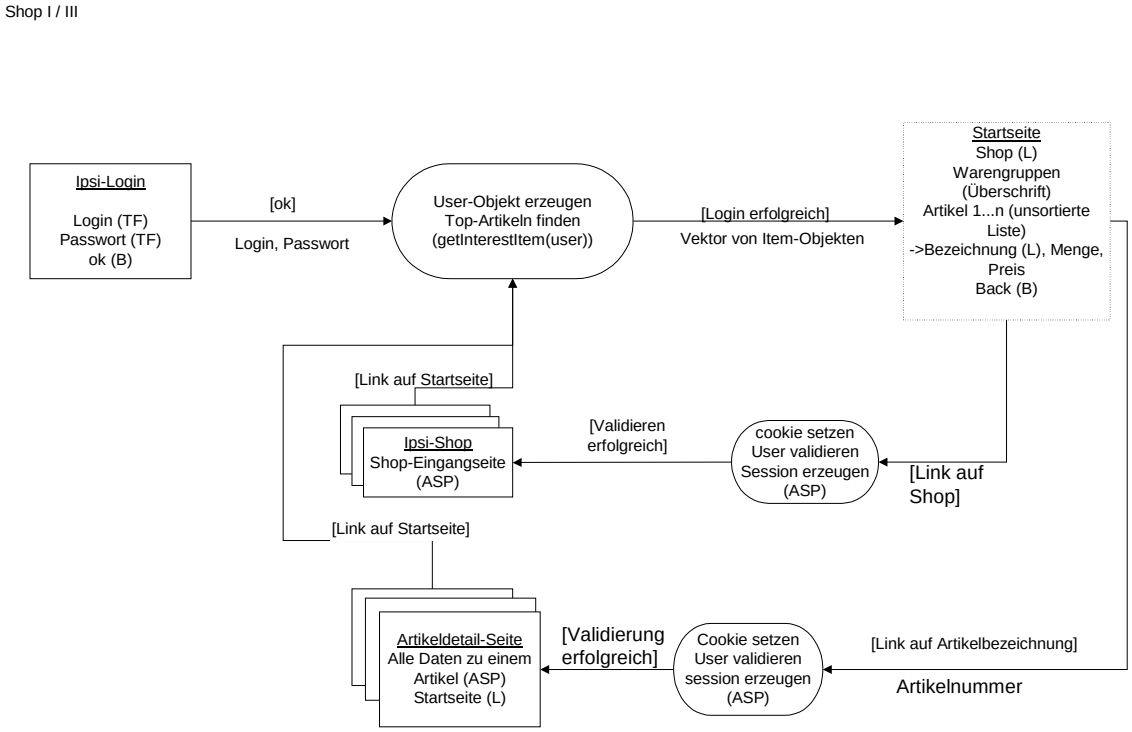


Abbildung 51 - MVD Subsystem Procurement (1)

Um den Shop überhaupt nutzen zu können, ist (anders als bei anderen Subsystemen) schon ein erfolgreiches Login beim IPSI-Portal nötig, da hier gleichzeitig im Shop wichtige Vorarbeiten geleistet werden (Anmeldung beim Shop etc.). Auf der Startseite wird dem Benutzer dann die Möglichkeit gegeben den Shop zu betreten. Alternativ kann der Nutzer auch auf einen Artikel klicken, der auf der Startseite dargestellt ist (z.B. neue Artikel werden so dem Nutzer bekanntgegeben).

Wenn der Nutzer den Shop betritt, werden wiederum einige interne Arbeiten notwendig, die es ermöglichen das System in das Gesamtportal so zu integrieren, dass dem Benutzer nicht auffällt, dass er nun ein völlig anderes System benutzt. Hierzu gehört u.a. das Setzen eines Cookies, dass dem Shopsystem anzeigt, dass der Benutzer sich angemeldet hat. Die weitere Funktionalität wird komplett vom fertigen Shopsystem SmartStore realisiert, das jedoch an vielen Stellen angepasst werden musste, um den integrativen Charakter zu wahren. Diese Maßnahmen sind in Abschnitt 11.5 nachzulesen und hier nicht weiter zu beschreiben.

Auf die gleiche Weise wird die Detailansicht zu Artikeln angezeigt, die der Nutzer von der Startseite aus wählen kann, die aufgerufene ASP-Seite ist selbstredend eine andere (s. Abbildung 51).

Shop II / III

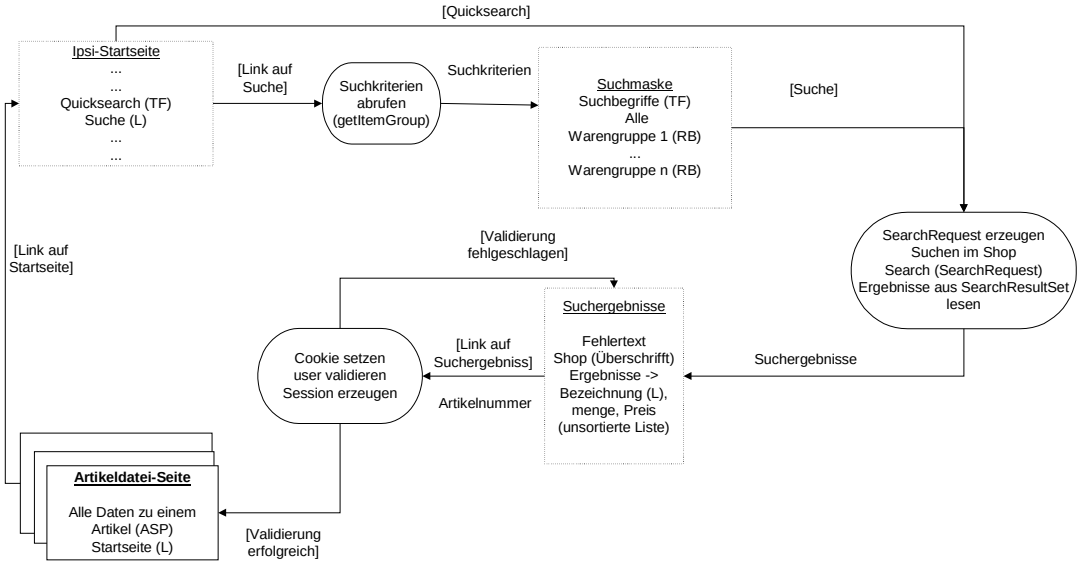


Abbildung 52 - MVD Subsystem Procurement (2)

Die Suchfunktionalitäten (s. Abbildung 52) des Shops werden analog zu den anderen Subsystemen genutzt bzw. zur Verfügung gestellt. D.h. zur Generierung der Suchkriterien der Startseite bzw. des Suchmenüs werden Suchkriterien über einen Controller beim Subsystem Procurement abgefragt und zurückgeliefert. Die Suche wird von diesen Seiten angestoßen, über ein SearchRequest an das Subsystem Procurement geliefert, das daraufhin die entsprechende ASP-Seite des Shops aufruft (nachdem die nötigen Vorarbeiten zur Validierung des Logins vorgenommen wurden). Die Darstellung der Artikel und die weiterführende Navigation sind dann wieder im Shopsystem SmartStore selbst implementiert.

Shop III / III

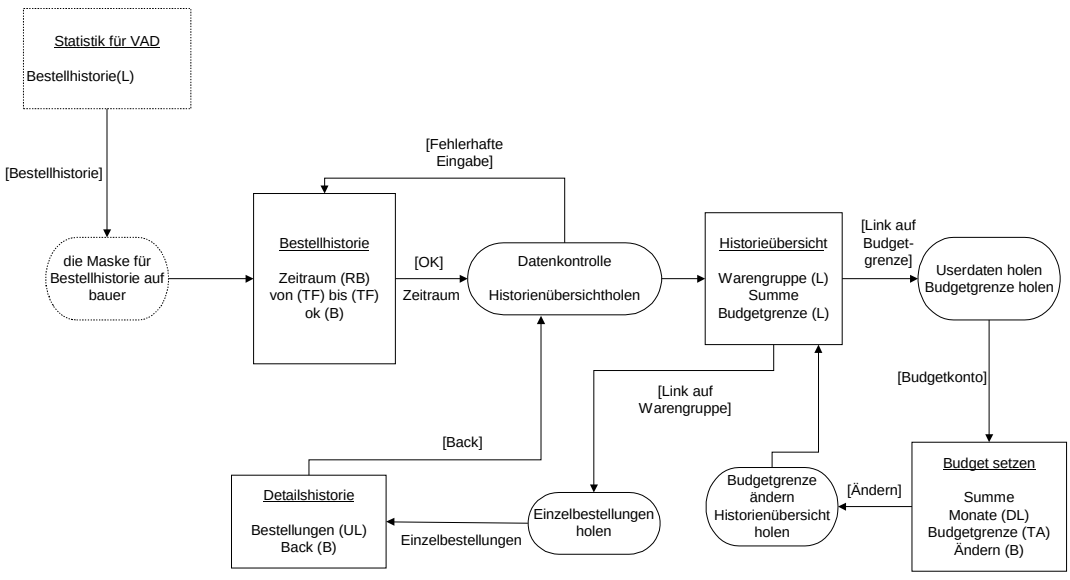


Abbildung 53 - MVD Subsystem Procurement (3)

Das Subsystem Procurement stellt weiterhin für den VAD-Statistikbereich (Abbildung 53) einige Daten zur Verfügung. Der Agenturleiter (das ist die Rolle des Nutzers, der die statistischen Daten für den VAD abruft) kann von der Statistikseite auf den Teilbereich Bestellhistorie verzweigen und dort die bestellten Artikel eines VADs einsehen.

Dazu gibt er zunächst den gewünschten Betrachtungszeitraum an; diese Vorselektion ist sinnvoll, um die Ergebnismenge nicht zu groß werden zu lassen.

Dem AL wird dann die Historie zunächst in einer Übersicht dargestellt, die nach Warengruppen geordnet ist.

Zu jeder Warengruppe kann in die Detailbestellungen des VAD in dieser Gruppe verzeigt werden. Außerdem kann die Budgetgrenze des VAD modifiziert werden, die ein Einkaufslimit darstellt.

7.10.4 MVDs zum Subsystem Legacy

Das Legacy Subsystem ist vom Startbildschirm über die Quicksuche oder über den Link Partnerdatenbank zu erreichen. Über die eigentliche Maske der Partnerdatenbank können die Kernfunktionen zur Suche und zu einzelnen vorgefertigten Abfragen gestartet werden. Die Abläufe sind beginnend in Abbildung 54 - MVD Subsystem Legacy (1) dargestellt.

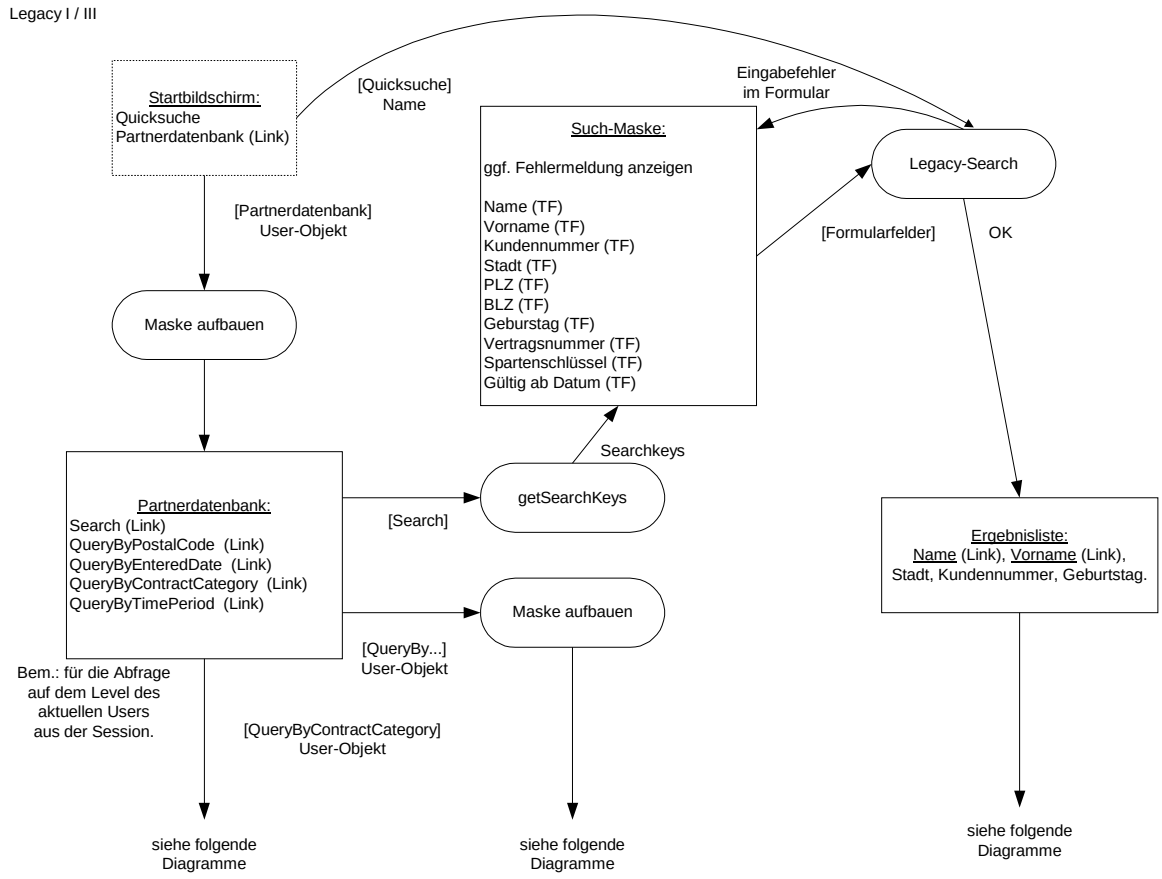


Abbildung 54 - MVD Subsystem Legacy (1)

Die Suche ruft über den Controller getSearchKeys die Suchmaske auf. In dieser Suchmaske kann der Benutzer seine Suchkriterien eingeben. Dabei können auch Kombinationen zwischen einzelnen Kriterien, wie z.B. Name und Stadt verwendet werden. Die ausgefüllten Formularfelder werden an den Legacy-Search Controller zur Verarbeitung übergeben. Dieser Controller startet die Suche in der Partnerdatenbank und gibt die Suchergebnisse an die Maske Ergebnisliste weiter. Die Quicksuche vom Startbildschirm ist eine Untermenge der Suche und

übergibt an den Legacy-Search Controller lediglich den Namen der zu suchenden Person. Innerhalb der Ergebnisliste sind sowohl der Name als auch der Vorname Links zu den Detailinformationen einer Person, d.h. einer Kombination von harten und weichen Daten. Um von einer gefundenen Person die Detailinformationen einsehen zu können, wird in das Subsystem Office übergegangen, vgl. Abbildung 56 - MVD Subsystem Legacy (3).

Von der Maske Partnerdatenbank aus Abbildung 54 - MVD Subsystem Legacy (1) können die Abfragen auf das Legacy System gestartet werden. In der Maske der Abfrage QueryByPostalCode kann der Benutzer die Postleitzahlengrenzen eingeben. Diese von-bis-Grenzen werden an den gleichnamigen Controller übergeben, der wiederum die Abfrage durchführt. Das Ergebnis wird dem Benutzer in Form einer Tabelle angezeigt (aufgeteilt nach der Summe der Kunden und nach der Summe der Verträge). Dieser Ablauf ist in Abbildung 55 - MVD Subsystem Legacy (2) detailliert dargestellt. Ähnlich ist der Ablauf bei den Abfragen QueryByEnteredDate und QueryByTimePeriod. Der Benutzer muss immer seine Abfragekriterien in die Maske eingeben und erhält das Ergebnis nach der Ausführung des jeweiligen Controllers in einer Ergebnismaske angezeigt. Die Abfrage QueryByContractCategory kommt dagegen ohne Eingabe des Benutzers aus, da jede Vertragskategorie mit den Summen als Ergebnis aus der Partnerdatenbank selektiert wird. Das Ergebnis ist ebenfalls eine Tabelle.

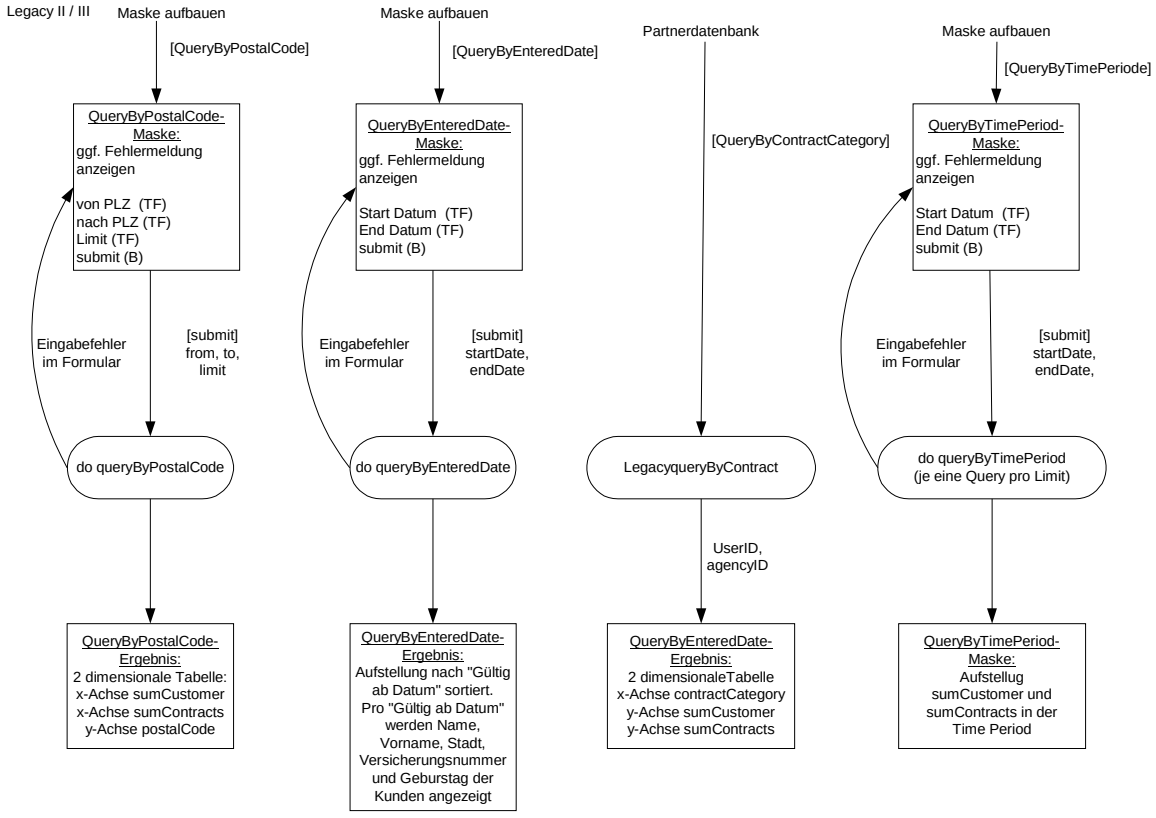


Abbildung 55 - MVD Subsystem Legacy (2)

Alle Abfragen berücksichtigten die Rechte des aktuell angemeldeten Benutzers. Ein VAD kann nur seine eigenen Daten aus der Partnerdatenbank einsehen. Ein Agenturleiter sieht hingegen alle Daten der ihm unterstellten Benutzer. Beim Versicherungsunternehmen ist dies analog. Aus diesem Grund wird den Abfragen ein User-Objekt übergeben. In diesem User-Objekt sind die Rechte verankert.

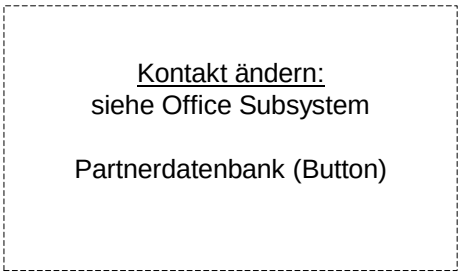


Abbildung 56 - MVD Subsystem Legacy (3)

7.10.5 MVDs zum Subsystem Search

Das Subsystem Search realisiert die integrative Suchfunktionalität über viele Portalkomponenten hinweg und stellt diese Funktionalität dem Nutzer zur Verfügung. Das nachfolgende MVD zeigt die Arbeitsweise des Subsystems.

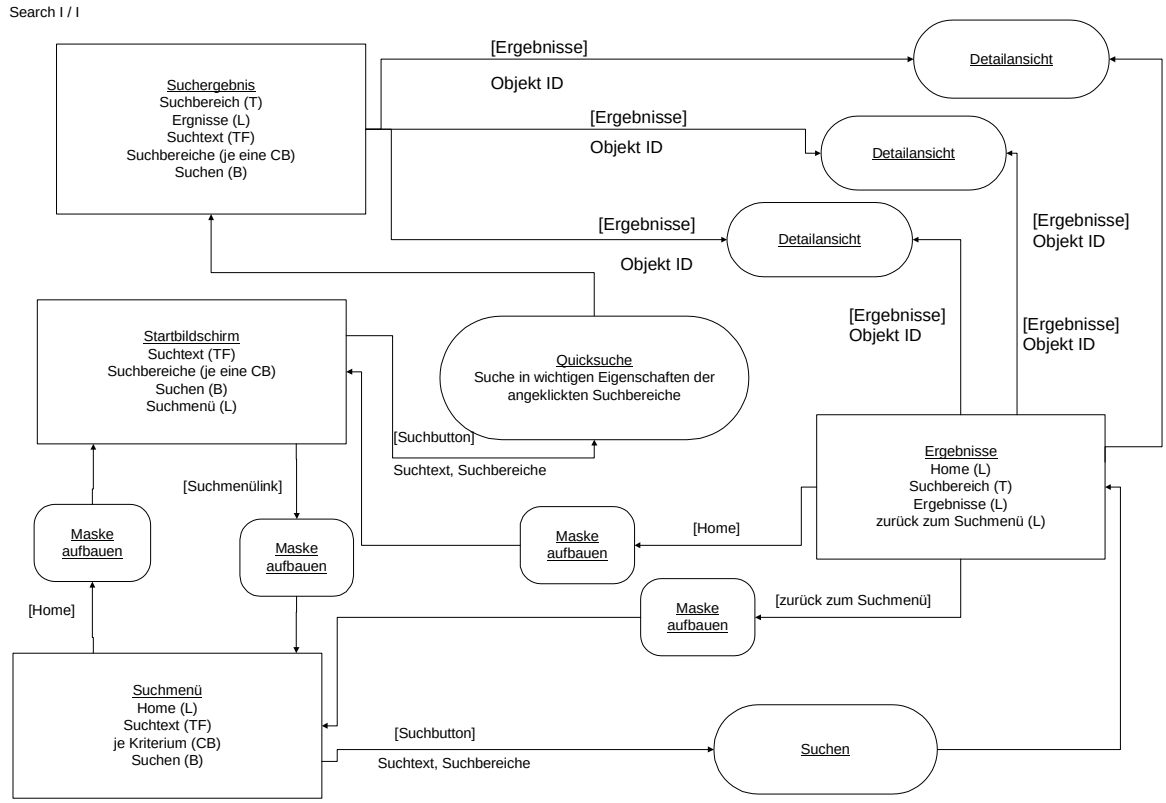


Abbildung 57 - MVD Subsystem Search

Auf dem Startbildschirm hat der Nutzer die Möglichkeit direkt über eine Quicksuche zu suchen oder dies über ein detaillierteres Suchmenü zu tun.

Für die Quicksuche wird lediglich ein Suchtext eingegeben und bestimmte Suchbereiche (Portalbereiche) angewählt. Als Ergebnis werden dem Benutzer die Suchresultate in einer neuen Maske als Liste von Einträgen (wie von üblichen Suchmaschinen gewohnt) präsentiert. Durch einen Klick auf ein Suchergebnis kann zu der jeweiligen Fundstelle verzweigt werden.

Im Suchmenü (das dynamisch über einen Controller/Formatter aufgebaut wird indem jedes registrierte Subsystem nach seinen Suchkriterien befragt wird) stehen dem Benutzer eine größere Anzahl an Suchkriterien zur Verfügung, die abhängig von den zu durchsuchenden Subsystemen sind. Das Ergebnis der Suche über das Suchme-

nü erscheint analog zum Ergebnis der Quicksuche, lediglich die Suche selber wird aufgrund der verschiedenartigen Parameter anders als bei der Quicksuche realisiert (daher ein eigener Verarbeitungsschritt).

8 Implementierung

Das Kapitel Implementierung soll nach der Darstellung der im Entwurf erstellten Architekturen darlegen, wie diese Architekturen in der Programmierphase umgesetzt wurden. Hierbei wird auf Besonderheiten eingegangen, und es werden teilweise Verbesserungsvorschläge gegeben.

Auch in diesem Kapitel soll die Strukturierung nach Subsystemen beibehalten werden.

8.1 Subsystem Office

Den PG-Teilnehmern wurde früh klar, dass bei der Implementierung des Subsystems Office eine ganze Reihe von Schwierigkeiten auftauchen würden. Um die damit verbundenen Unsicherheiten zu minimieren, wurden ja auch bereits Durchstichprototypen angefertigt (s. Kapitel 5).

8.1.1 Zugriff auf Outlook

Der direkte Zugriff auf Microsoft Outlook wird durch eine DLL realisiert, die mit dem C++-Builder von Microsoft erstellt wurde [Kr97]. Es besteht direkter Zugriff auf die Outlook-Objekt-Hierachie. Als größtes Problem erwies sich dabei die mangelhafte Dokumentation dieser Objekthierarchie. Statt einer vollständigen API finden sich dort lediglich eine Reihe von Beispielen, die für die meisten Anwendungen ausreichen und hilfreicher sind als Syntaxdefinitionen (zumal diese nicht auf C/C++ sondern auf Visual-Basic zutrifft), aber für die Lösung einiger, speziellerer Probleme nichts beitragen. So läßt sie jegliche Dokumentation über Objektsuche mit Wildcards vermissen, genauso wie klare Aussagen zum Finden von Terminserien-Objekten. Es gäbe noch mehr Beispiele. Auch eine Recherche im Internet ergab, dass die meisten anderen Entwickler auch nur das fertig bekommen haben, was in den Beispielen der Dokumentation vorexerziert wird.

Daraus ergeben sich letztlich auch die Lücken der Realisierung:

- Terminserien können zwar angelegt werden, wenn auch nur sechs bestimmte von uns definierte Typen (täglich, werktäglich, wöchentlich, Datum des Monats, Woche des Monats, jährlich), aber die Suchfunktion findet jeweils nur den sogenannten Mastertermin, d.h. den ersten Termin der Serie. Dadurch wird es letztendlich unsinnig diese Funktion in IPSI zu verwenden.
- Suche mit Wildcards funktioniert grundsätzlich nicht, obwohl dies in Outlook offensichtlich möglich ist. Dies muss mühsam in JAVA nachprogrammiert werden, da wir darauf nicht verzichten können. (Bsp.: Wer weiß schon den vollständigen Titel des Termins, den er sucht?)

8.1.2 JNI

JNI (JAVA-Native-Interface) ist die Technik, mit deren Hilfe wir aus JAVA-Methoden heraus die DLL-Funktionen aufrufen. Für die PG-Teilnehmer war dies eine völlig neue Technik, deren Erlernen einige Zeit kostete und die Entwicklung am Anfang verlangsamte.

Im erstellten Prototyp war die Behandlung von Fehlern noch ausgeklammert. Es ist aber einfach, per JNI von C aus JAVA-Exceptions zu werfen, und davon wurde rege Gebrauch gemacht.

8.1.3 RMI

Erst gegen Ende der Implementierung wurde die Integration von RMI (Remote-Method-Invocation), der verwendeten Middleware, durchgeführt. Da einige Teilnehmer bereits Erfahrungen mit dieser Technologie gesammelt hatten und eine klar verständliche Dokumentation zur Umstellung des Subsystems vorlag, war eine Umsetzung schnell möglich.

Leider versagten einige der schon erstellten Methoden danach den Dienst und warfen überraschende Laufzeitfehler. Das lag daran, dass die RMI-Architektur beim Entwurf zu wenig berücksichtigt wurde. Im Subsystem Office werden die BO's (Business-Objects) nicht wie in den meisten anderen Subsystemen *by-reference*, sondern *by-*

value übergeben und so mussten wir einige Wertübergaben nachtragen und so den Entwurf anpassen, was aber nur wenig Zeit in Anspruch nahm.

8.1.4 Die Anwendung

Jeder VAD benötigt ein eigenes unabhängiges Subsystem Office, da er auch ein eigenes Outlook verwendet. Er muss dieses Subsystem selber starten, bevor er mit IPSI darauf zugreifen kann. Die PG entschloss sich dafür, das Subsystem durch eine *JAVA-Application* zu realisieren. Die Alternative wäre ein Applet gewesen, was den Vorteil gehabt hätte, dass der VAD nicht selber noch einen zusätzlichen Programmaufruf gehabt hätte. Ein Applet ist jedoch durch sehr strenge Sicherheitsvorschriften eingeschränkt, und ein Zugriff auf die Festplatte des VADs (da befindet sich nun mal Outlook) ist ohne weiteres nicht zulässig. Ein Workaround wie *Trusted-Applets* scheiterte an der dafür notwendigen Einarbeitungszeit. Für die Zukunft sei empfohlen, sich doch die Mühe zu machen und auf Applets umzustellen. Abgesehen vom Aufwand, der mit den Sicherheitsmechanismen zusammenhängt, ist eine Überführung von der *JAVA-Application* in ein *JAVA Applet* sehr schnell zu bewerkstelligen.

8.2 Subsystem Administration

8.2.1 Allgemeines

Jeder Benutzer des Portals hat mehrere IDs: (Portal)UserID, ShopID, CmsID, OfficeID. Diese IDs sind nicht lebenslang gültig, das heißt ein *Administrator* kann diese IDs bei Bedarf verändern. Das bedeutet aber, dass bei einer Änderung einer ID alle anderen Daten (Interessen, Feedbacks, etc.) verändert werden müssen, um auch noch nach der Änderung erreichbar zu sein. Außerdem müssen die anderen Subsysteme davon unterrichtet werden. Dieser Aufwand ist relativ hoch, so dass es wohl einfacher (aber nicht komfortabler für den Benutzer) gewesen wäre, die IDs als unveränderbar festzulegen.

Die Rechte, die vergeben werden, sind nur Rollen (VAD, Agenturleiter, VU). Der Entwurf gibt aber sehr viel mehr her; theoretisch kann ein Benutzer unendlich viele Rechte besitzen, hier hat er aber nur ein Recht, das genau seiner Rolle im Versicherungsunternehmen entspricht (siehe Kapitel 7.4.1). Der Einfachheit halber wurde diese Version gewählt. Es ist anzumerken, dass die Rechteüberprüfung *ausschließlich* in der Fassadenklasse stattfindet, weil so das Rechtemanagement leicht geändert werden kann, ohne dass die Controller geändert werden müssen. Außerdem gehört das Rechtemanagement auch „logisch“ zum Subsystem Administration (siehe Kapitel 7.4.3).

8.2.2 Datenbank

Die einzige Klasse, die Zugriff auf die Datenbank hat, in der alle Daten der Benutzer und Agenturen verwaltet werden, ist die Klasse PortalUserDB. Es wird die Datenbank mySQL benutzt. Die benutzten Datenbankabfragen sind nicht immer effizient gestaltet, aber dafür einfach und für die Belange der PG auch ausreichend (nur relativ wenig Daten). Außerdem wird mySQL nicht voll ausgereizt, so könnte man z.B. noch Primary-Keys, NOT-NULL-VALUES und UNIQUE-Spalten verwenden. Ein Transaktionsmanagement hingegen bietet mySQL in der verwendeten Version nicht an und wurde auch nicht nachprogrammiert, da der Aufwand als zu hoch eingeschätzt wurde. Insgesamt kann man sagen, dass das Zusammenwirken mit der Datenbank etwas effizienter hätte gestaltet werden können, vielleicht auch durch den Einsatz eines mächtigeren Datenbankmanagementsystems.

8.2.3 Verbesserungs- und Erweiterungsvorschläge

Man könnte eine neue Exception AdminException einführen, die alle bisherigen Exceptions in diesem Subsystem ersetzt. Dadurch könnte man die hohe Anzahl der Exceptions verringern, müsste in den Controllern dann weniger catch-Blöcke benutzen.

Anstatt bei jedem Datenbankzugriff eine eigene Connection aufzubauen, könnte man ein Connection-Pooling aufbauen. Dies würde zu einem verbesserten Performanzverhalten führen.

8.3 Subsystem Procurement

Die Implementierung beim Subsystem „Procurement“ bestand hauptsächlich darin, das Shopsystem „SmartStore Standard Edition 2.0.2“ so weit wie möglich im Portal zu integrieren. SmartStore basiert auf Active-Server-Pages, deren Funktionen mit den Skriptsprachen VBScript und JavaScript implementiert sind, während das Portal im wesentlichen auf JAVA aufbaut. Für diese Integration musste man in der Lage sein, innerhalb von JAVA Klassen einerseits die ASP-Funktionen vom Shop ausführen und anderseits direkte Anfragen an die Shop-Datenbank stellen zu können. Darüber hinaus sollten die Identifikationsdaten des Benutzers beim Eintritt in den Shop aus dem Portal übernommen werden, damit der Benutzer sich nicht zusätzlich beim Shop identifizieren muss. Diese Besonderheiten werden nun näher erläutert:

- Zugriff auf ASP-Funktionen mit JAVA:
  Diese Lösung wurde bei der Implementierung von externer Suche nach Shop-Artikeln eingesetzt. Dabei ist die im Shop als ASP-Funktion vorhandene Suche als eine Methode in der Subsystem-Fassade zur Verfügung gestellt. Die Ausführung der vorhandenen Suchfunktion, die in einer ASP-Datei beinhaltet war, ist mit Hilfe der Klasse java.net.URL realisiert worden. Die Suchfunktion wurde so geändert, dass die Ausgabe keine HTML-Steuerelemente, sondern nur die Artikelattribute beinhaltet. Aus diesen Attributen wurden in der Subsystem-Fassade entsprechende „Business-Objects“ erzeugt und an die aufrufenden Klassen weitergeleitet [Mi99, Hi97].
- Direkter Zugriff mit JAVA auf Shop-Datenbank:
  Mit dieser Lösung wurde der größte Anteil der Shop-Integration im Portal realisiert. Damit der Portalbenutzer auch im Shop identifiziert wird, mussten die Benutzerdaten in der Portaldatenbank und Shop-Datenbank konsistent sein. Dazu musste jede Änderung in der Benutzerdatenbank des Portals auch in der Shop-Datenbank vermerkt werden. SmartStore speichert alle Daten in einer Microsoft Access Datenbank. Von den Methoden in der Subsystem-Fassade wurde mit Hilfe der JDBC-ODBC-Bridge direkt auf der Shop-Datenbank zugegriffen.
- Automatische Authentifizierung des Benutzers:
  SmartStore fordert den Benutzer beim Shop-Eintritt auf, sich zu identifizieren. Um dies zu automatisieren, der Benutzer sich nur einmal im Portal anmelden muss, war es notwendig, die Anmeldedaten des Benutzers zum Shop weiterzuleiten, sobald der Benutzer den Shop betritt. Zu diesem Zweck wurde eine ASP-Funktion als Zwischenstufe eingesetzt, die den Login-Namen des Benutzers aus dem URL liest, diese Information in ein Cookie schreibt und erst dann die Eingangsseite des Shops aufruft. Die Eingangsseite liest die Anmeldedaten des Benutzers nicht aus dem Anmeldeformular, sondern aus dem Cookie, und identifiziert den Benutzer. Diese Prozedur wird außer bei dem standardmäßigen Shop-Eingang auch in zwei weiteren Situationen durchgeführt. Erstens wenn ein Benutzer aus den Ergebnissen der Portalsuche ein Artikel wählt, um im Shop direkt auf die Artikeldetailseite zu gelangen. Zweitens wenn im Bereich Benutzerinteressen Artikel gewählt werden, die auch zu den entsprechenden Artikeldetailseiten im Shop führen. Auch bei diesen beiden Situationen muss eine automatische Authentifizierung des Benutzers im Shop durchgeführt werden, bevor die gewünschten Shop-Seiten betreten werden können.

Eine weitere Besonderheit war die Realisierung eines Budgetkontos für jeden Benutzer. In SmartStore wurde ursprünglich eine Bestellgrenze berücksichtigt, die aber den für das Portal formulierten Anforderungen nicht entsprach. Zu diesem Zweck wurde die Shop-Datenbank um das Budgetkonto erweitert, in dem für jeden Benutzer eine Budgetgrenze, ein Zeitraum, für den die Budgetgrenze gelten sollte und die Summe der Bestellungen innerhalb der bestimmten Zeitraum eingetragen werden. Diese Attribute wurden berücksichtigt, bevor eine Bestellung betätigt werden kann.

Weiterhin ist zu erwähnen, dass SmartStore Standard Edition den Einsatz vom Microsoft Internet Information Server und Microsoft Access Datenbank voraussetzt. Von daher musste das Shop-Subsystem auf einer Microsoft-Plattform (Windows NT 4.0) betrieben werden, während das Portal selbst in einer Umgebung mit Linux und Apache-Webserver läuft. Für die Integration des Subsystems im Portal musste die Shop-Subsystem-Fassadenklasse im Gegensatz zu den anderen Subsystemen auf einem separaten Shop-Server gestartet werden, wodurch das Subsystem dann mittels RMI mit den anderen Subsystemen im Portal kommunizieren konnte.

8.3.1 Verbesserungs- und Erweiterungsvorschläge

Als Verbesserung wird in erster Linie der Einsatz einer fähigeren Datenbank an Stelle von MS-Access empfohlen. Vor allem sollte die Datenbank Transaktionen unterstützen.

Die Verwaltung von SmartStore geschieht standardmäßig durch eine Client-Software, die nicht webfähig ist. Im Rahmen des Projektes wurde eine webbasierte Benutzerverwaltung realisiert, die unabhängig von dem o. g. Client-Programm ist. Ein Erweiterungsvorschlag wäre die Realisierung einer webbasierten Produktverwaltung für den Shop, womit die Administratoren unabhängig vom Vorhandensein der Client-Software die Shopinhalte verwalten könnten.

8.4 Subsystem Kommunikation

8.4.1 Änderungen des Entwurfs gegenüber der ersten Version

Beim Entwurf des Subsystems Kommunikation stand eine offene Architektur im Vordergrund. Dieses zeigt sich darin, dass es jetzt problemlos möglich ist, neue Nachrichtentypen zu definieren, die über die Versandmedien verschickt werden können. Ebenso einfach ist es möglich, neue Versandmedien in das bestehende System zu integrieren. Der erste Entwurf sah vor, dass der Versand der Nachrichten über drei verschiedene Medien, nämlich Email, SMS und Fax, realisiert wird. In den Cut-Through-Prototypen wurde der Versand von Fax und SMS über das Telefonnetz getestet. Da aber im Serverraum kein Telefonanschluss zur Verfügung steht, war es nicht möglich, diese Versandmedien im Rahmen der Projektgruppe bereitzustellen. Aus diesem Grund ist als einziges Versandmedium die Email vollständig implementiert. Zur Veranschaulichung der Erweiterbarkeit des Subsystems sind die Versandmedien SMS und Fax teilweise realisiert worden. D.h., dass diese Versandkanäle im System logisch vorhanden und wählbar sind, aber die Klasse, die den technischen Versand übernimmt, nicht implementiert ist. So erzeugt ein Versuch, eine Nachricht über einen dieser Kanäle zu verschicken einen internen Fehler. Als Folge davon wird diese Nachricht an das nächste Versandmedium in der Fallback-Reihenfolge weitergereicht. Die Fallback-Reihenfolge ist in jedem Nachrichtentyp definiert. Im Endeffekt landen also alle Nachrichtentypen früher oder später beim Versand per Email.

Weiterhin sieht der erste Entwurf vor, dass die „MessageQueue“ eine abstrakte Oberklasse ist, von der drei Queues erben: eine SMSQueue, eine FaxQueue und eine EmailQueue. Gegen Ende des Entwurfs zeigte sich aber, dass die drei Queue-Arten eine identische Funktionalität bereitstellen würden. Daher wurden diese drei abgeleiteten Klassen durch drei Instanzen der „MessageQueue“ ersetzt.

Die letzte Änderung des Entwurfs betrifft die Verteilung des IPSI-Portals via RMI. Die einzelnen Subsysteme von IPSI wurden zunächst entworfen, ohne dass die verteilte Umgebung berücksichtigt wurde. Durch die nachträglich eingeführte verteilte Laufzeitumgebung waren Änderungen an den Business-Objekten notwendig, auf die von entfernten Computern zugegriffen wird. RMI bietet zwei Möglichkeiten des Zugriffs auf entfernte Objekte: *call-by-value* und *call-by-reference*. Während beim *call-by-reference* direkt auf das entfernte Objekt zugegriffen wird, wird bei *call-by-value* eine Kopie des Objektes erstellt und auf diese Kopie zugegriffen. Standardmäßig werden in IPSI Objekte mittels *call-by-reference* angesprochen. Im Gegensatz dazu stellt das Subsystem Kommunikation Business-Objekte bereit, auf die mit *call-by-value* zugegriffen wird. Im einzelnen sind das „ClientComment“, „TaskReminder“ und „AppointmentReminder“. Die Abweichung vom IPSI-Standard ist notwendig, da die betroffenen Objekte von den Office-Controllern erzeugt und der „CommunicationSubsystemFacade“ als Parameter übergeben werden. Diese Objekte werden im Subsystem Kommunikation nach der Parameterübergabe im Speicher gehalten, bis der Zeitpunkt zum Versenden erreicht ist. Würden die Objekte *by-reference* übergeben und würde in der Zwischenzeit das Subsystem Office abgeschaltet werden, so wären auch alle Objekte nicht mehr verfügbar, die dieses Subsystem erzeugt hat. Folglich könnte das Subsystem Kommunikation nicht mehr auf diese Objekte zugreifen und keine Benachrichtigungen mehr versenden. Dies käme nicht zum Tragen, wenn Office durchgehend auf einem Server laufen würde. Da aber dieses Subsystem auf dem Client beim VAD läuft und immer wieder abgestellt wird, dürfen diese Objekte nicht *by-reference* übergeben werden. Eine andere Möglichkeit dieses Problem zu umgehen, wäre, die Objekte nicht nur im Speicher zu halten, sondern persistent zu sichern. Ein weiterer Vorteil der persistenten Datenhaltung im Subsystem Kommunikation wäre die Möglichkeit, den Server kurzzeitig herunterfahren zu können, ohne das alle Reminder verloren gingen.

8.4.2 Probleme während der Implementierung

- Emailversand über SMTP mit JAVAEmail:
  Während der Implementierung der Klasse „EmailVersand“ haben wir uns zunächst in jedem Detail an die in der JAVAEmail-Dokumentation [Su98] beschriebenen Schritte gehalten, um Emails zu versenden. Unter anderem haben wir dort MIME-Typen der Email und Formate für die Empfängeradresse(n) gesetzt. Aus für uns unverständlichen Gründen war das Ergebnis, dass keine Emails versendet wurden. Schließlich haben wir

den Quellcode der HowTo-Dokumentation zum Emailversand über SMTP mittels copy-and-paste in die „EmailVersand“-Klasse übernommen. In diesem Code wird darauf verzichtet, irgendwelche MIME-Typen o.ä. zu setzen. Nach dieser Änderung funktionierte der Emailversand problemlos.

- Regelmäßige Aufrufe der „CommControl“-Klasse über einen Timer:
  Es wird eine Methode benötigt, die regelmäßig prüft, ob neue Benachrichtigungen verschickt werden sollen. Im IPSI-Portal ist dazu die Methode „timerActionPerformed()“ der Klasse „CommControl“ vorgesehen, die in bestimmten Intervallen aufgerufen wird. Als Timer, der die regelmäßigen Aufrufe tätigt, haben wir zunächst die Klasse „JAVAx.swing.Timer“ verwendet. Auf unseren Entwicklungsrechnern, die unter dem Betriebsystem Windows NT betrieben werden, funktionierte das einwandfrei. Für die Integration in das IPSI-Portal muss das Subsystem Kommunikation jedoch auf unserem Server ausgeführt werden, der unter dem Betriebsystem Linux arbeitet. Dabei trat das Problem auf, dass die Timer-Klasse einen X-Server benötigt, dieser unter Linux aber nicht dauerhaft zur Verfügung steht. Folglich konnte das Subsystem in dieser Variante nicht auf dem Server genutzt werden. Als Lösung haben wir eine äquivalente Timerklasse („CommunicationTimer“) und ein Interface („CommunicationTimerListener“) geschrieben. Seit dem Einsatz dieser Klassen läuft das Subsystem Kommunikation sowohl unter Windows NT als auch unter Linux problemlos.

8.4.3 Verbesserungs- und Erweiterungsvorschläge

Zunächst einmal sei die oben erwähnte Verbesserungsmöglichkeit bzgl. der Persistenz der Benachrichtigungsobjekte hier noch einmal aufgeführt. Es wird sich aus Wartungsaspekten sicherlich nicht vermeiden lassen, den Server hin und wieder herunterzufahren. Da dabei alle Daten verloren gingen, wäre dies sicherlich ein großes Problem. Würde das System also praktisch eingesetzt, wäre dieses sicherlich ein Punkt, der frühestmöglich umgesetzt werden sollte.

Weiterhin ist die Fallback-Reihenfolge in den Nachrichtenobjekten zur Zeit fest kodiert. Lediglich das bevorzugte Versandmedium kann vom Benutzer ausgewählt werden. Wünschenswert aus Sicht des Benutzers wäre sicherlich eine Möglichkeit, die Fallback-Reihenfolge individuell festzulegen.

Schließlich wäre auch eine Untersuchung der Performance des Subsystems Kommunikation unter starker Belastung interessant. Beim Entwurf und Design wurde zwar so geplant, dass keine Störungen während der Laufzeit bei einem größeren Datenvolumen auftreten, aber diese Mechanismen wurden bisher noch nicht gezielt getestet. Eventuell könnten dort noch Schwierigkeiten auftreten, bei denen geeignete Gegenmaßnahmen ergriffen werden müssten.

8.5 Subsystem Legacy

8.5.1 Was ist XML?

XML – Extensible-Markup-Language

ist eine Metasprache und eignet sich optimal zur Beschreibung von dokumentenorientierten und hierarchisch strukturierten Informationen mit großer Bandbreite. XML ist in der Lage Dokumente nicht nur nach formalen Kriterien zu gliedern, sondern auch nach inhaltlichen Gesichtspunkten. Es gibt im Gegensatz zu HTML eine klare Trennung von Inhalt und Darstellung [Ju99].

Der größte Vorteil von XML liegt in der Möglichkeit, selbst definierte Tags einzusetzen. Die Continentale nimmt aus diesem Grund einfach die Datenbankbezeichner als Tags, z.B. <KuGeburtsDatum> 11.01.1974 </KuGeburtsDatum>.

Zusätzlich ist es möglich, eine DTD – Document-Type-Definition zu verwenden.

Eine DTD definiert Klassen von Dokumenttypen [BoUl99]. Festgelegt ist damit, dass die Tags in einer bestimmten Struktur in den betreffenden XML-Dokumenten enthalten sein müssen. Die DTD benutzt XML, um die strukturelle Integrität eines Dokumentes zu überprüfen. Da die Daten der Continentalen allerdings aus einer Legacy-Datenbank kommen und unverändert sind, ist eine DTD ist diesem Zusammenhang nicht notwendig.

8.5.2 Wie kann man XML-Dateien parsen?

Sun stellt auf seiner Website eine „JAVA API for XML Parsing“ zur Verfügung [Su00c]. Die aktuelle Version 1.01 wird von uns verwendet und wird auch in unserer Library eingesetzt.

Das Prinzip ist folgendes: Man übergibt der API den Namen einer XML-Datei, die dann eingelesen, auf Richtigkeit hin analysiert und in einer Dokumentenstruktur (Objekt der Klasse XmlDocument) zurückgegeben wird. Um mit dieser Struktur arbeiten zu können, haben wir das Dokument in eine Baumähnliche Struktur mit Hilfe der Klasse TreeWalker überführt. Ein TreeWalker kann ähnlich wie ein Suchbaum durchlaufen werden, und die XML-Daten sind als „key-value-Paare“ (der „key“ entspricht dann einem XML-Tag) darin abgelegt. D.h., dass man z.B. mittels eines TreeWalkers nach einem XML-Tag „KuGeburtsDatum“ (dieser Bezeichner steht in den XML-Daten für das Kunden-Geburtsdatum) im gesamten Baum oder aber auch nur in einem Teilbaum des TreeWalker suchen kann. Wird das XML-Tag gefunden gibt es Methoden, die das „value“ zurückliefern.

Als wir zum ersten Mal bei der Erstellung des Prototypen mit der API gearbeitet hatten war uns der TreeWalker noch nicht bekannt, sodass wir erst andere Wege gesucht hatten. Nur durch Zufall sind wir auf diese Lösung gestoßen. Der TreeWalker gibt uns genau die Möglichkeit, XML-Tags und deren Inhalt zu ermitteln und dies sogar schnell und effizient. Ein Zugriff auf Teilbäume ist dabei mit der größte Vorteil, da man so geschickt die XML-Struktur ausnutzen kann, z.B. wenn sich XML-Tags wiederholen.

8.5.3 Welche Probleme können beim Parsen auftreten?

Man muss sich immer darüber im Klaren sein, dass beim Parsen auf die Korrektheit der Inhalte zu achten ist, z.B. muss beim Einlesen des XML-Tag „KuGeburtsDatum“ der Inhalt ein gültiges Datum enthalten. Sobald sich das Format ändert muss ebenfalls darauf reagiert werden. Wir haben bei all unseren Annahmen die Referenzdaten und -Formate der Continentalen benutzt.

Eine weitere Schwierigkeit ergibt sich daraus, dass die „JAVA API for XML Parsing“ in unserem Fall nicht in der Lage ist, korrekt Umlaute einzulesen. Sobald ein „ä“, „ö“ oder „ü“ auftaucht, bricht der Parser mit einer Fehlermeldung ab. Aus diesem Grund haben wir in unseren Testfällen auf Sonderzeichen verzichtet.

8.5.4 Wie werden die XML-Daten in Business-Objekten gespeichert?

Das Prinzip zum Speichern der XML-Daten in Business-Objekte wurde bereits in Kapitel 7.8 dargestellt. Der folgende Ausschnitt soll einen Einblick in den Quellcode geben.

```
  /**
   * Die Methode getCustomer holt sich die Kunden-Xml-Files
   * vom LegacyDbAdapter und generiert daraus das Customer-Objekt eines
Kunden mitsamt
   * aller Verträge und deren Geschichstbücher.
   * @param String: Die Kundennummer, dessen Customer-Objekt erzeugt werden
soll.
   * @return Customer: Ein Customer-Objekt, dass alle Daten eines Kunden
enthält
   * @throws LegacyException: Wird geschmissen, wenn die Kundennumer nicht
existiert oder etwas mit dem zugehörigen Xml-Datenstrom nicht stimmt.
   */

  Customer getCustomer( String versnr ) throws LegacyException, ja-
va.rmi.RemoteException {

    XMLParser xmlParser = new XMLParser();
    String filename = lnkLegacyDbAdapter.CustomerQuery( versnr );
    myTreeWalker tree = xmlParser.createTreeWalker( filename );
    if ( tree == null ) {
      throw new LegacyException( „Legacy: CustomerTreeParser: getCustomer:
RootTree ist null!“ );
    }

    // relevante Trees aus dem „root-Tree“ erstellen:
    myTreeWalker KuKeys = tree.getFirstTreeWalker( „KuKeys“ );
    myTreeWalker KuPersonenDaten = tree.getFirstTreeWalker( „KuPersonenDa-
```

```
ten“ );
    myTreeWalker KuAdressDaten = tree.getFirstTreeWalker(„KuAdressDaten“ );

    if ( KuKeys == null ) {
      throw new LegacyException( „Legacy: CustomerTreeParser: getCustomer:
SubTree <KuKeys> nicht gefunden!“ );
    }
    if ( KuPersonenDaten == null ) {
      throw new LegacyException( „Legacy: CustomerTreeParser: getCustomer:
SubTree <KuPersonenDaten> nicht gefunden!“ );
    }
    if ( KuAdressDaten == null ) {
      throw new LegacyException( „Legacy: CustomerTreeParser: getCustomer:
SubTree <KuAdressDaten> nicht gefunden!“ );
    }

    // CustomerNumber aus dem Tree <KuKeys> auslesen:
    String customerNumber = KuKeys.getFirstValue( „KuPersonenNr“ );
    // PersonalNumber aus dem Tree <KuPersonenDaten> auslesen.
    String personalNumber = KuPersonenDaten.getFirstValue( „KuPersonalNr“ );
    String gueltigAbDatumStr = KuPersonenDaten.getFirstValue( „KuGueltigAb-
Datum“ );
    Calendar gueltigAbDatum;
    SimpleDateFormat gueltigAbDatumDate = new SimpleDateFormat( „dd.MM.yyyy“
);
    try {
      gueltigAbDatumDate.parse( gueltigAbDatumStr );
    } catch ( ParseException pe ) {
      throw new LegacyException( „Legacy: CustomerTreeParser: getCustomer:
Gueltig ab Datum nicht korrekt: usage dd.MM.yyyy“ );
    }
    gueltigAbDatum = gueltigAbDatumDate.getCalendar();
    PersonalData personalData = personalDataTreeParser.getPersonalData( Ku-
PersonenDaten, KuAdressDaten );
    TechnicalData technicalData = technicalDataTreeParser.getTechnicalData(
KuPersonenDaten );
    Vector accountSet = accountDataTreeParser.getAccountData( tree );
    Vector contractSet = contractDataTreeParser.getContractData( tree );
    Customer customer = new CustomerImpl( accountSet,
                                          contractSet,
                                          customerNumber,
                                          gueltigAbDatum,
                                          personalData,
                                          personalNumber,
                                          technicalData );
    return customer;
  }
}
```

Am Anfang der Methode wird über die Versicherungsnummer die jeweilige XML-Datei ermittelt. Diese XML-Datei wird dann geparst und das Ergebnis wird in einen TreeWalker überführt. Anhand des TreeWalker ist es möglich, gezielt XML-Tags oder Unterbäume im TreeWalker anzuspringen und die Inhalte auszulesen. Damit können nach und nach die Attribute gefüllt und das Customer-Objekt instanziiert werden.

9 Qualitätsmanagement

Neben der Einhaltung des Zeit- und Budgetrahmens (letzterer kann bei Projektgruppen nur in der Anzahl investierter Personentagen berücksichtigt werden) ist die Erfüllung der gestellten Anforderungen das dritte Ziel eines jeden Projektes. Die Übereinstimmung eines Softwaresystems mit seinen Anforderungen wird auch als Qualität bezeichnet (DIN 55350, Teil 11). Die Sicherstellung der Qualität des entwickelten Systems war ein wichtiger Aspekt der Projektgruppe, was sich nicht zuletzt in der Besetzung der Rolle des Qualitätsmanagers ausdrückte. Daher wurden alle End- und viele Zwischenergebnisse Qualitätssicherungsmaßnahmen unterworfen.

Das Vorgehen der Qualitätssicherung (QS) und die Ergebnisse dieser Maßnahmen in Form von Dokumenten ist Gegenstand dieses Abschnittes. Die Qualitätssicherung lässt sich grob in konstruktive und analytische Maßnahmen unterteilen.

Konstruktive Maßnahmen versuchen vor der Erstellung eines Produktes dessen Qualität sicherzustellen, z.B. durch die Erstellung von Richtlinien, Checklisten und Verfahrensanweisungen. So war es die Aufgabe des Qualitätsmanagers, Dokumentenvorlagen für sämtliche zu erstellende Dokument vorzuschlagen, die dann in den

litätsmanagers, Dokumentenvorlagen für sämtliche zu erstellende Dokument vorzuschlagen, die dann in den Projektgruppensitzungen verabschiedet wurden. Bezüglich der Richtlinien wurde ein bereits existierender Coding-Style ausgewählt und der Quellcode-Entwicklung aller Programmierer zugrundegelegt. Arbeitsanweisungen konnten aus dem Prozessmodell (s. Kapitel 4 Prozessleitfaden) abgeleitet werden. Dies geschah zumeist in den Sitzungen des Project Management Boards (Gremium bestehend aus dem Projektmanager, dem Qualitätsmanager und dem Chefarchitekten) und wurde dann entweder mündlich in den Projektgruppensitzungen vorgetragen oder aber in eigenen How-To-Dokumenten festgehalten. Ebenso wurden Prototypen für die Erstellung von Subsystemen, Controller und Formatter erstellt, die ebenfalls als eine Art von Richtlinie angesehen werden können.

Analytische Maßnahmen haben dagegen eher diagnostischen Charakter, indem sie nach der Erstellung eines Produktes dieses überprüfen. Bei den analytischen Maßnahmen wurden in der Projektgruppe vorrangig testende Methoden eingesetzt. Durchgeführt wurden Modultests (im Folgenden Klassentests), Komponententests, Integrationstests und schließlich der Systemtest.

Die Klassentests bestanden darin, die Methoden einer Klasse mit frei gewählten Testdaten über einen Testtreiber zu testen. Bei der Auswahl der Testdaten sollten clevere Grenzdaten gewählt werden. Aufgrund der zur Verfügung stehenden Zeit wurde auf eine über den Quelltext des Testtreibers hinausgehende Dokumentation des Klassentests verzichtet.

Gegenstand der Komponententests war ein komplettes Subsystem. Für diesen Test wurden Testdokumente erstellt, für die zuvor eine Vorlage von Seiten des Qualitätsmanagements vorgegeben wurde. In diesen Dokumenten wurden die einzelnen Testfälle dokumentiert. Die Dokumente sind im Anschluss an dieses Kapitel abgedruckt. Genau wie beim Klassentest handelte es sich beim Komponententest um einen funktionalen Black-Box-Test, d.h. die Testdaten wurden nicht aufgrund der Struktur des Programms gewählt, um eine Überdeckung des Kontroll- oder Datenflusses zu erreichen, sondern in Bezug auf die Erfüllung oder Nichterfüllung der zuvor definierten funktionalen Anforderungen.

Dieses Testerfahren wurde auch im Integrationstest angewendet, in dem die Funktionsweise der erstellten Controller untersucht wurde, die ja bereits mehrere Subsysteme bei ihrer Ausführung kontaktieren und somit integrativen Charakter besitzen. Auch hier wurde auf eine Dokumentation aus zeitlichen Gründen verzichtet.

Dokumentiert wurde dagegen der Systemtest, bei dem alle spezifizierten Use Cases durch die Benutzung des fertiggestellten Systems gestestet wurden. Die Abnahme erfolgte durch einen Betreuer der Projektgruppe, der stichprobenartig Systemfunktionalitäten testete. Auch diese Dokumentation findet sich im Anschluss an dieses Kapitel.

Die bisher genannten Testverfahren sind allesamt dynamische Verfahren, da sie die Ausführbarkeit des Quellcodes voraussetzen. Neben dieser Art von Testverfahren fanden auch statische Testverfahren in der Projektgruppe Anwendung. Hierzu zählen u.a. Reviews, Walkthrougs und Inspektionen, die sich hauptsächlich in ihrem Grad an Formalität bei der Durchführung und der Rolle des Moderators (Autor vs. unabhängiger Prüfer) unterscheiden. Während der Projektgruppe wurden die weniger formalen Methode des Reviews und Walkthroughs eingesetzt, was wiederum pragmatische Gründe (Zeit, Aufwand) hatte. Bei diesen Vorgehensweisen führt der Autor des zu prüfenden Gegenstandes die Gruppe der Prüfer (in unserem Fall waren dies stets zwei bis drei andere Projektgruppenmitglieder) durch sein Werk und erläutert dessen Funktionsweise oder Aufbau.

Diesem Review wurden in der Projektgruppe insbesondere die Subsysteme im Quelltext als auch die zugehörigen Komponententests unterzogen. Ergebnis eines solchen Reviews war ein Review-Bericht, der Empfehlungen an den Autor des Prüflings zur Verbesserung oder Fehlerbeseitigung beinhaltete. Die Überprüfung erfolgte anhand mehrerer Kriterien. Zum einen wurde die durch den Coding-Styleguide vorgegeben Richtlinie zur Verfassung des Quelltextes überprüft. Zum anderen wurde die Erfüllung des Designs anhand der vorhandenen Klassendiagramme getestet. Ebenso wurden die Use-Cases in den Review mit einbezogen, um sicherzustellen, dass alle Funktionalitäten auch abgedeckt wurden. Schließlich wurden auch Empfehlungen zur Verbesserung des Quelltextes im Hinblick auf den Grad der Umsetzung des Paradigmas der Objektorientierung gegeben, die auf den persönlichen Erfahrungen der Prüfer basierten.

Die Entscheidung über die Umsetzung dieser Empfehlungen lag in der Verantwortung des Autors selber. Dies erschien insbesondere vor dem Hintergrund der Tatsache, dass es innerhalb einer Projektgruppe keine eindeutige Befehlshierarchie gibt, ein gangbarer Weg zu sein.

Zusammenfassend ist zu bemerken, dass eine ganze Reihe von sowohl konstruktiven als auch analytisch-testenden Verfahren der Qualitätssicherung eingesetzt wurden, um die Einhaltung der gestellten Anforderungen sicherzustellen, auch wenn aufgrund der zeitlichen Begrenzung oft die weniger formalen Verfahren gewählt wurden.

Wie bereits im obigen Text angekündigt, folgen nun die Testdokumente zum Komponententest und zum Systemtest. Die Testdokumente sind den einzelnen Subsystemen unterteilt (auch beim Systemtest, da die Use Cases sehr gut auf die Subsysteme zugeordenet werden konnten, was ein arbeitsteiliges Testen begünstigte). Die Darstellung folgt der gewohnten Tabellendarstellung in mehreren Spalten und Zeilen. Jede Zeile enthält einen Testfall. Zur Erklärung seien die Spaltenbezeichnungen und deren Zweck nun noch einmal aufgeführt:

- Laufende Nummer: dient der Identifizierung des Testfalls
- Betroffene Methoden: listet alle Objektmethoden auf, die durch den jeweiligen Testfall getestet werden
- Eingabe/Vorgehen: beschreibt das Vorgehen zum durchführen des Testfalls
- Erwartete Ausgabe: stellt die erwartete Ausgabe dar, mit der die tatsächliche Ausgabe verglichen wird
- Tatsächliche Ausgabe: s.o. (ok in dieser Spalte gibt an, dass erwartete und tatsächliche Ausgabe übereinstimmten)
- Kommentar: ermöglicht die Erläuterung der Ausgabe bzw. eingeleiteter Maßnahmen zur Behebung des Fehlers

Die Tabellen zum Systemtest enthalten im wesentlichen die gleichen Spalten mit folgenden Abweichungen und Ergänzungen:

- statt der Spalte „Methode“ gibt es eine Spalte „Aktion“, da sich der Systemtest nicht an Objektmethoden, sondern an Use Cases orientiert (das von der PG verwendete Region/Action-Paradigma orientiert die Actions an Use Cases)
- OK?: diese Spalte enthält ein „X“, wenn der Testfall positiv verlief (keine Fehler)
- Priorität: enthält einen Wert zwischen 1 und 3, der angibt, ob der beobachtete Fehler wesentlich oder weniger wichtig für die Portalfunktionalität ist; entsprechend dieser Priorität wurden die Fehler behandelt
- Testgrundlage: sollte die Nummer des Builds enthalten, der dem Systemtest zu Grunde lag (dient der Versionierung)

9.1.1 Komponententest Subsystem Office Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Subsystem):	Office		Testteilnehmer:	Chris, Traugott,
Testzeitraum: (Datum)	25.05.00			

Lfd.Nr	betroffene Methode(n)	Eingabe / Vorgehen	erwartete Ausgabe	tatsächliche Ausgabe / Beobachtung	Kommentar
1	login, logout	Outlook per Hand öffnen Verbindung zur Fassade per RMI herstellen (im folgenden immer notwendig) Etwas in Outlook per JAVA tun (z.B. Kontakt einfügen) Prozess beenden Prozess wiederholen	In Outlook: zwei neue Kontakte	ok	dieser Testfall funktionierte erst nach erheblichen Anpassungen im C++-Code
2	login, logout	• Outlook falls nötig schließen • Etwas in Outlook per JAVA tun (Kontakt einfügen) • Prozess beenden	• Outlook öffnet sich • Neuer Kontakt zu sehen • Outlook schließt sich		
3	Insert	• null-Pointer übergeben	• Exception wird geworfen	ok	
4	Insert, Appointment-Konstruktoren, Appointment.Read	• Termin in JAVA anlegen und in Outlook einfügen • Termin in JAVA verändern • Termin aus Outlook lesen	• In Outlook: neuer Termin • In JAVA: Veränderungen sind futsch	ok	
4a		Besondere Termine • leerer Termin	• Defaultwerte in Outlook	ok (Termin zum Erstellungszeitpunkt wird angelegt)	
4b		Besondere Termine • Terminserie mit Ende vor Beginn des Ursprungstermins (3.6.2000-7.4.2000)	• Nur Ursprungstermin	nicht ok	Terminserie mit Ende 30.4.2001 wird angelegt
5	SearchAppointment, modify	• Termin in Outlook per Hand anlegen • Termin per JAVA suchen • Termin per JAVA verändern und in Outlook speichern	• In Outlook: veränderter Termin	ok	

6	Insert, Appoint-ment.setStartdate	• Termin mit Datum im Dezember per JAVA anlegen (Grund: JAVA-Dezember = 11, C-Dezember=12)	• In Outlook: Termin im Dezember	ok	
7	Insert, modify	• Termin per JAVA in Outlook anlegen • Diesen Termin per Hand in Outlook löschen, während JAVA-Prozess noch läuft • Termin per JAVA in Outlook verändern	• OfficeException wird geworfen	ok	die Exception wird erst dann geworfen, wenn der Termin auch aus dem Papierkorb gelöscht wor-den ist; sonst wird der Termin im Papierkorb geändert
8	Insert, Read	• Termin per JAVA in Outlook anlegen • Diesen Termin per Hand in Outlook ändern, während JAVA-Prozess noch läuft • Termin per JAVA aus Outlook lesen	• In JAVA: geänderter Termin	ok	
9	Insert, Delete	• Termin per JAVA in Outlook anlegen • Termin in Outlook öffnen • Termin per JAVA in Outlook löschen	• OfficeException	Ergebnis ist besser als erwartet	Termin wird gelöscht; Outlook-Fenster des Ter-mins wird geschlossen
10	Insert, Modify	• Termin per JAVA in Outlook anlegen • Diesen Termin per Hand in Outlook löschen, während JAVA-Prozess noch läuft • Diesen Termin per Hand in Outlook wieder aus dem Papierkorb holen, während JAVA-Prozess noch läuft • Termin per JAVA in Outlook verändern	• In Outlook: geänderter Termin	ok	
11	Insert, Contact-Konstruktoren, Contact.Read	• Kontakt in JAVA anlegen und in Outlook einfügen • Kontakt in JAVA verändern • Kontakt aus Outlook lesen	• In Outlook: neuer Kontakt • In JAVA: Veränderungen sind futsch	ok	
11a		Besondere Kontakte • leerer Contact	• Defaultwerte in Outlook	ok	
12	SearchContact, modify	• Kontakt in Outlook per Hand anlegen • Kontakt per JAVA suchen • Kontakt per JAVA verändern • Kontakt per JAVA in Outlook speichern	• In Outlook: veränderter Kontakt	nicht ok	Country kann nicht geän-dert werden

13	Insert, Contact.setBirthday	• Kontakt mit Geburtstag im Dezember per JAVA anlegen (Grund: JAVA-Dezember = 11, C-Dezember=12)	• In Outlook: Kontakt mit Geburts tag im Dezember	nicht ok	November statt Dezember
14	Insert, modify	• Kontakt per JAVA in Outlook anlegen • Diesen Kontakt per Hand in Outlook löschen, während JAVA-Prozess noch läuft • Kontakt per JAVA in Outlook verändern	• OfficeException wird geworfen	ok	
15	Insert, Read	• Kontakt per JAVA in Outlook anlegen • Diesen Kontakt per Hand in Outlook ändern, während JAVA-Prozess noch läuft • Kontakt per JAVA aus Outlook lesen	• In JAVA: geänderter Kontakt	ok	
16	Insert, Delete	• Kontakt per JAVA in Outlook anlegen • Kontakt in Outlook öffnen • Kontakt per JAVA in Outlook löschen	• Outlook-Fenster wird geschlossen	ok	
17	Insert, Modify	• Aufgabe per JAVA in Outlook anlegen • Diese Aufgabe per Hand in Outlook löschen, während JAVA-Prozess noch läuft • Diese Aufgabe per Hand in Outlook wieder aus dem Papierkorb holen, während JAVA-Prozess noch läuft • Aufgabe per JAVA in Outlook verändern	• In Outlook: geänderte Aufgabe	ok	
18	Insert, Task-Konstruktoren, Task.Read	• Aufgabe in JAVA anlegen und in Outlook einfügen • Aufgabe in JAVA verändern • Aufgabe aus Outlook lesen	• In Outlook: neue Aufgabe • In JAVA: Veränderungen sind futsch	ok, aber: wie kann Reminder in Outlook aktiviert werden? (ReminderTime ja auch in Outlook gesetzt)	
19		Besondere Aufgaben • leere Aufgabe anlegen	• Defaultwerte in Outlook	ok	
20	SearchTask, modify	• Aufgabe in Outlook per Hand anlegen • Aufgabe per JAVA suchen • Aufgabe per JAVA verändern • Aufgabe per JAVA in Outlook speichern	• In Outlook: veränderte Aufgabe	ok	Bei der Ausgabe des Tasks in JAVA wird das Startdate mit ausgegeben. Dieses war im Testfall aber falsch (1.0.4501???).

21	Insert, modify	• Aufgabe per JAVA in Outlook anlegen • Diese Aufgabe per Hand in Outlook löschen, während JAVA-Prozess noch läuft • Aufgabe per JAVA in Outlook verändern	• OfficeException wird geworfen	ok	
22	Insert, Read	• Aufgabe per JAVA in Outlook anlegen • Diese Aufgabe per Hand in Outlook ändern, während JAVA-Prozess noch läuft • Aufgabe per JAVA aus Outlook lesen	• In JAVA: geänderte Aufgabe	ok	
23	Insert, Delete	• Aufgabe per JAVA in Outlook anlegen • Aufgabe in Outlook öffnen • Aufgabe per JAVA in Outlook löschen	• Fenster in der die geöffnete Aufgabe angezeigt wird schliessen	ok	
23	Insert, Modify	• Aufgabe per JAVA in Outlook anlegen • Diesen Aufgabe per Hand in Outlook löschen, während JAVA-Prozess noch läuft • Diesen Aufgabe per Hand in Outlook wieder aus dem Papierkorb holen, während JAVA-Prozess noch läuft • Aufgabe per JAVA in Outlook verändern	• In Outlook: geänderte Aufgabe	ok	
24	SearchEmail	• Emails in Outlook erstellen • Per JAVA alle Emails suchen / nur nach einer ganz bestimmten suchen und anzeigen	• In JAVA: Köpfe der Emails	ok	
25	DisplayEmail	• Emails in Outlook erstellen • Per JAVA suchen und anzeigen • Per JAVA Email aufpoppen lassen	• Outlookfenster geht auf.	ok	

9.1.2 Komponententest Subsystem Administration Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Subsystem):	Administration	Testteilnehmer:	Stefan Göbel
Testzeitraum: (Datum)	8.5.2000 - 11.5.2000		

Lfd.Nr	betroffene Methode(n)	Eingabe / Vorgehen	erwartete Ausgabe	tatsächliche Ausgabe / Beobachtung	Kommentar
1	getUser	Gültige UserID und richtiges Portalpasswort, hier: „goebel01“, „testipsip“	Ein Userobjekt, das den User mit der UserID „goebel01“ repräsentiert, mit allen Daten gefüllt.	Stimmt mit der erwarteten Ausgabe überein.	

2	getUser	Gültige UserID und falsches Portalpasswort, hier: „goebel01“, „testipsis“	Null	Stimmt mit der erwarteten Ausgabe überein.	Inkonsistent? Bei falschem User gibt es eine Exception, bei falschem Pwd NULL ??
3	getUser	Nicht vorhandene UserID, irgendein Passwort, hier: „goebel00“, „hallo“	UserNotFoundException	Stimmt mit der erwarteten Ausgabe überein.	
4	getUsersForAgencyID	Gültiger User Actor und gültige AgencyID, hier: das Userobjekt, das den User mit der UserID „goebel01“ und dem Recht „Constants.VU“ repräsentiert und die AgencyID „0001“	Alle Userobjekte der Agentur „0001“, also: ajeche00, bidaum00, hmeier00 (alphabetisch sortiert)	Stimmt mit der erwarteten Ausgabe überein.	
5	getUsersForAgencyID	Gültiger User Actor und nicht vorhandene AgencyID, hier: das Userobjekt, das den User mit der UserID „goebel01“ und dem Recht „Constants.VU“ repräsentiert und die AgencyID „0005“	AgencyNotFoundException	Stimmt mit der erwarteten Ausgabe überein.	
6	getUsersForAgencyID	Leeres Userobjekt Actor und gültige AgencyID, hier „0001“	Keine Ahnung	NullPointerException in User.hasRight	Kann der Fall überhaupt eintreten? Hier muss ggf. nachgebessert werden.
7	getUsersForAgenyID	gültiger User Actor und gültige AgencyID, hier: das Userobjekt, das den User mit der UserID „bidaum00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und die AgencyID „0001“	Alle Userobjekte der Agentur „0001“, also: ajeche00, bidaum00, hmeier00 (alphabetisch sortiert)	NoRightException	Fehler entdeckt und behoben (2 Strings werden verglichen mit .equals, nicht mit ==
7a	getUsersForAgencyID	gültiger User Actor und gültige AgencyID, hier: das Userobjekt, das den User mit der ID „bidaum00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und die AgencyID „0001“	Alle Userobjekte der Agentur „0001“, also: ajeche00, bidaum00, hmeier00 (alphabetisch sortiert)	Stimmt mit der erwarteten Ausgabe überein.	Wiederholung von 7
8	getUsersForAgencyID	Nicht gültiger User Actor und gültige AgencyID, hier: das Userobjekt, das den User mit der UserID „hmeier00“ und dem Recht „Constants.VAD“ repräsentiert und die AgencyID „0001“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
9	getUserForID	gültiger User Actor und gültige UserID, hier: das Userobjekt, das den User mit der UserID „goebel01“ und dem Recht „Constants.VU“ repräsentiert und die UserID „ajeche00“	Ein Userobjekt, das den User mit der UserID „ajeche00“ repräsentiert, mit allen Daten gefüllt.	Stimmt mit der erwarteten Ausgabe überein.	

10	getUserForID	gültiger User Actor und nicht vorhandene UserID, hier: das Userobjekt, das den User mit der UserID „goebel01“ und dem Recht „Constants.VU“ repräsentiert und die UserID „ajeche01“	UserNotFoundException	Stimmt mit der erwarteten Ausgabe überein.	
11	getUserForID	gültiger User Actor und gültige UserID, hier: das Userobjekt, das den User mit der UserID „bidaum00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und die UserID „ajeche00“	Ein Userobjekt, das den User mit der UserID „ajeche00“ repräsentiert, mit allen Daten gefüllt.	Stimmt mit der erwarteten Ausgabe überein.	
12	getUserForID	Nicht gültiger User Actor und gültige UserID, hier: das Userobjekt, das den User mit der UserID „hmeier00“ und dem Recht „Constants.VAD“ repräsentiert und die UserID „ajeche00“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
13	getUserForID	Nicht gültiger User Actor und gültige UserID, hier: das Userobjekt, das den User mit der ID „becker00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und die UserID „ajeche00“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
14	getAgencyForID	gültige AgencyID, hier: „0001“	Agenturobjekt, das die Agentur mit der AgencyID „0001“ repräsentiert.	Stimmt mit der erwarteten Ausgabe überein.	
15	getAgencyForID	Nicht vorhandene AgencyID, hier: „0005“	AgencyNotFoundException	Stimmt mit der erwarteten Ausgabe überein	
16	getAgencyForID	gültiger User Actor und gültige AgencyID, hier: das Userobjekt, das den User mit der UserID „goebel01“ und dem Recht „Constants.VU“ repräsentiert und die AgencyID „0001“	Agenturobjekt, das die Agentur mit der AgencyID „0001“ repräsentiert.	Stimmt mit der erwarteten Ausgabe überein	
17	getAgencyForID	gültiger User Actor und gültige UserID, hier: das Userobjekt, das den User mit der userID „bidaum00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und die AgencyID „0001“	Agenturobjekt, das die Agentur mit der AgencyID „0001“ repräsentiert	Stimmt mit der erwarteten Ausgabe überein.	
18	getAgencyForID	Nicht gültiger User Actor, gültige AgencyID, hier: das Userobjekt, das den User mit der userID „hmeier00“ und dem Recht „Constants.VAD“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	

		repräsentiert und die AgencyID „0001“			
19	getAgencyForID	Nicht gültiger User Actor und gültige AgencyID, hier: das Userobjekt, das den User mit der userID „becker00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und die AgencyID „0001“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
20	getAgencyLeader	gültige Agency agency, hier: das Agenturobjekt, das die Agentur mit der agencyID „0001“ repräsentiert	Das Userobjekt, das den User mit der ID „bidaum00“ repräsentiert	Stimmt mit der erwarteten Ausgabe überein.	
21	getAgencyLeader	Nicht vorhandene Agency agency, hier: das Agenturobjekt, das die Agentur mit der agencyID „0005“ repräsentiert	AgencyNotFoundException	Stimmt mit der erwarteten Ausgabe überein.	
22	getAgencyLeader	gültiger User actor und gültige Agency agency, hier: das Userobjekt, das den User mit der userID „goebel01“ und dem Recht „Constants.VU“ repräsentiert und das Agenturobjekt mit der AgencyID „0001“	Das Userobjekt, das den User mit der ID „bidaum00“ repräsentiert	Stimmt mit der erwarteten Ausgabe überein.	
23	getAgencyLeader	gültiger User Actor und gültige Agency agency hier: das Userobjekt, das den User mit der userID „bidaum00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und das Agenturobjekt mit der AgencyID „0001“	Das Userobjekt, das den User mit der ID „bidaum00“ repräsentiert	Stimmt mit der erwarteten Ausgabe überein.	
24	getAgencyLeader	Nicht gültiger User Actor und gültige Agency agency, hier: das Userobjekt, das den User mit der userID „hmeier00“ und dem Recht „Constants.VAD“ repräsentiert und das Agenturobjekt mit der AgencyID „0001“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
25	getAgencyLeader	Nicht gültiger User Actor und gültige Agency agency, hier: das Userobjekt, das den User mit der userID „becker00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert und das Agenturobjekt mit der AgencyID „0001“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
26	getAllAgencies	keine	Alle Agenturobjekte in einer Enumeration of Agency, alphabetisch sortiert.	Stimmt mit der erwarteten Ausgabe überein.	

27	getAllAgencies	gültiger User Actor, hier: das Userobjekt, das den User mit der userID „goebel01“ und dem Recht „Constants.VU“ repräsentiert	Alle Agenturobjekte in einer Enumeration of Agency, alphabetisch sortiert.	Stimmt mit der erwarteten Ausgabe überein.	
28	getAllAgencies	Nicht gültiger User Actor, hier: das Userobjekt, das den User mit der userID „becker00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
29	getAllRights	keine	Alle Rechteobjekte in einer Enumeration of Right	Stimmt mit der erwarteten Ausgabe überein.	
30	getAllRights	gültiger User Actor, hier: das Userobjekt, das den User mit der UserID „goebel01“ und dem Recht „Constants.VU“ repräsentiert	Alle Rechteobjekte in einer Enumeration of Right	Stimmt mit der erwarteten Ausgabe überein.	
31	getAllRights	gültiger User Actor, hier: das Userobjekt, das den User mit der userID „becker00“ und dem Recht „Constants.AGENTURLEITER“ repräsentiert	Das Rechteobjekt „vad“ in einer Enumeration of Right	Stimmt mit der erwarteten Ausgabe überein.	
32	getAllRights	Nicht gültiger User Actor, hier: das Userobjekt, das den User mit der userID „hmeier00“ und dem Recht „Constants.VAD“ repräsentiert	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
33	getFeedbacksForUser	gültiger User Actor, gültiger User user, Date startdate, Date enddate hier: actor mit UserID „goebel01“ und Recht „Constants.VU“, user mit UserID „hmeier00“, startdate „10000“, enddate das aktuelle Datum	Die entsprechenden Feedbackobjekte in einer Enumeration of Feedback	Stimmt mit der erwarteten Ausgabe überein.	
34	getFeedbacksForUser	gültiger User Actor, gültiger User user, Date startdate, Date enddate hier: actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“, user mit UserID „hmeier00“, startdate „10000“, enddate das aktuelle Datum	Die entsprechenden Feedbackobjekte in einer Enumeration of Feedback	Stimmt mit der erwarteten Ausgabe überein.	
35	getFeedbacksForUser	gültiger User Actor, gültiger User user, Date startdate, Date enddate hier: actor mit UserID „hmeier00“ und Recht „Constants.VAD“, user mit UserID „hmeier00“, startdate „10000“, enddate das aktuelle Datum	Die entsprechenden Feedbackobjekte in einer Enumeration of Feedback	Stimmt mit der erwarteten Ausgabe überein.	
36	getFeedbacksForUser	Nicht gültiger User Actor, gültiger User user, Date startdate, Date enddate hier: actor mit UserID „becker00“ und Recht „Con-	NoRightException	Leere Enumeration	Fehler entdeckt und behoben: 1. equals() statt ==

		„Constants.AGENTURLEITER“, user mit UserID „hmeier00“, startdate „10000“, enddate das aktuelle Datum			2. NoRightException wurde nie geworfen
36a	getFeedbacksForUser	Nicht gültiger User Actor, gültiger User user, Date startdate, Date enddate hier: actor mit UserID „becker00“ und Recht „Constants.AGENTURLEITER“, user mit UserID „hmeier00“, startdate „10000“, enddate das aktuelle Datum	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
37	getHomepageData	User user, hier: user mit der UserID „goebel01“	Alle Homepagedaten in einem Vector	Stimmt mit der erwarteten Ausgabe überein.	
38	createNewAgency	gültiger User actor, Vector data, hier: actor mit UserID „goebel01“ und Recht „Constants.VU“ und data, Inhalt: („0009“,“hmeier00“)	true	Ausgabe ist true, Daten sind in der Datenbank eingetragen	
39	createNewAgency	gültiger User actor, Vector data, hier: actor mit UserID „goebel01“ und Recht „Constants.VU“ und data, Inhalt: („0009“,“hmeier00“) (schon vorhanden !)	false	Stimmt mit der erwarteten Ausgabe überein.	
40	createNewAgency	Nicht gültiger User actor, Vector data, hier: actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“ und data, Inhalt: („0008“,“becker01“)	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
41	createNewUser	gültiger User actor und UserData data, hier: actor mit UserID „goebel01“ und Recht „Constants.VU“ und neue data	true	Stimmt mit der erwarteten Ausgabe überein.	
42	createNewUser	gültiger User actor und UserData data, hier: actor mit UserID „goebel01“ und Recht „Constants.VU“ und schon vorhandene data	false	Stimmt mit der erwarteten Ausgabe überein.	
43	createNewUser	gültiger User actor und UserData data, hier: actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“ und neue data, User soll in Agency „0001“	true	Stimmt mit der erwarteten Ausgabe überein.	
44	createNewUser	Nicht gültiger User actor und UserData data, hier: actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“ und neue data, User soll in Agency „0002“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	

45	createNewUser	Nicht gültiger User actor und UserData data, hier: actor mit UserID „hmeier00“ und Recht „Constants.VAD“ und neue data	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
46	modifyUser	gültiger User actor und UserData data, hier: actor mit UserID „goebel01“ und Recht „Constants.VU“ und veränderte data (für UserID=„muelle01“ ist AgencyID jetzt „0009“	nichts	Keine Ausgabe, Daten in der Datenbank verändert.	
47	modifyUser	gültige User actor und UserData data, hier: actor mit UserID „goebel01“ und Recht „Constants.VU“ und veränderte data (für UserID=„muelle01“ soll AgencyID jetzt „0006“ sein, diese gibt’s aber nicht)	AgencyNotFoundException	Stimmt mit der erwarteten Ausgabe überein.	
48	modifyUser	gültiger User actor und UserData data, hier: actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“ und veränderte data (für UserID=„muelle00“ ist AgencyID jetzt „0009“	nichts	Keine Ausgabe, Daten in der Datenbank verändert.	
49	modifyUser	Nicht gültiger User actor und UserData data, hier: actor mit UserID „becker00“ und Recht „Constants.AGENTURLEITER“ und veränderte data (für UserID=„muelle00“ ist AgencyID jetzt „0003“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
50	modifyUser	gültiger User actor und UserData data, hier: actor mit UserID „muelle00“ und Recht „Constants.VAD“ und veränderte data (für UserID=„muelle00“ ist AgencyID jetzt „0009“	nichts	Keine Ausgabe, Daten in der Datenbank verändert.	
51	modifyAgency	gültiger User actor, gültige Agency agency, Vector data, hier: Actor mit UserID „goebel01“ und Recht „Constants.VU“, Agency mit AgencyID „0003“ und neue data	nichts	Keine Ausgabe, Daten in der Datenbank verändert.	
52	modifyAgency	gültiger User actor, gültige Agency agency, Vector data, hier: actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“, Agency mit AgencyID „0001“ und neue data	nichts	Keine Ausgabe, Daten in der Datenbank verändert.	
53	modifyAgency	Nicht gültiger User actor, gültige Agency agency, Vector data, hier: actor mit UserID „muelle00“ und Recht „Constants.VAD“,Agency mit AgencyID „0001“ und neue data	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	

54	modifyAgency	Nicht gültige User actor, gültige Agency agency, Vector data, hier: actor mit UserID „becker00“ und Recht „Constants.AGENTURLEITER“, Agency mit AgencyID „0001“ und neue data	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
55	modifyAgency	gültiger User actor, gültige Agency agency, Vector data, hier: Actor mit UserID „goebel01“ und Recht „Constants.VU“, Agency mit AgencyID „0001“ und neue data mit falscher UserID	UserNotFoundException	Stimmt mit der erwarteten Ausgabe überein.	
56	insertFeedback	gültige UserID, grade, date, comment, hier: „becker01“,“3“,heutiges Datum, „Durchschnittlich“	nichts	Keine Ausgabe, Daten in die Datenbank eingetragen.	
57	insertFeedback	Nicht vorhandene UserID, grade, date, comment, hier: „becker02“,“3“,heutiges Datum, „Durchschnittlich“	UserNotFoundException	Stimmt mit der erwarteten Ausgabe überein.	
58	existsUserID	UserID, hier: „goebel01“	true	Stimmt mit der erwarteten Ausgabe überein.	
59	existsUserID	Nicht vorhandene UserID, hier: „goebel02“	false	Stimmt mit der erwarteten Ausgabe überein.	
60	deleteAgencyAndUser	gültiger User actor, gültige Agency agency, hier: Actor mit UserID „goebel01“ und Agency mit AgencyID „0009“	nichts	Keine Ausgabe, die entsprechene Agentur und ihre User sind aus der Datenbank entfernt worden.	
61	deleteAgencyAndUser	Nicht gültiger User actor, gültige Agency agency, hier: Actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“ und Agency mit AgencyID „0003“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
62	deleteAgencyWithoutUser	gültiger User actor, gültige Agency agency, gültige Agency newagency, hier: Actor mit UserID „goebel01“ und Agency mit AgencyID „0003“ und newagency mit AgencyID „0002“	nichts	Keine Ausgabe, die entsprechene Agentur ist aus der Datenbank entfernt worden, die zugehörigen User sind in eine andere Agentur versetzt worden.	

63	deleteAgencyWithoutUser	Nicht gültiger User actor, gültige Agency agency, gültige Agency newagency, hier: Actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“ und Agency mit AgencyID „0001“ und newagency mit AgencyID „0002“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
64	deleteUser	gültiger User actor und gültiger User user, hier: Actor mit UserID „goebel01“ und Recht „Constants.VU“ und user mit UserID „muelle01“	nichts	Keine Ausgabe, User ist aus der Datenbank entfernt worden.	
65	deleteUser	gültiger User actor und gültiger User user, hier: Actor mit UserID „bidaum00“ und Recht „Constants.AGENTURLEITER“ und user mit UserID „muelle00“	nichts	Keine Ausgabe, User ist aus der Datenbank entfernt worden.	
66	deleteUser	Nicht gültiger User actor und User user, hier: Actor mit UserID „becker00“ und Recht „Constants.AGENTURLEITER“ und user mit UserID „muelle00“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
67	deleteUser	Nicht gültiger User actor und gültiger User user, hier: Actor mit UserID „becker01“ und Recht „Constants.VAD“ und user mit UserID „muelle00“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	
68	deleteFeedbacks	gültiger User actor, gültiger User user, startdate, enddate, hier: : Actor mit UserID „goebel01“ und Recht „Constants.VU“ und user mit UserID „becker00“, startdate „10000“ und enddate=„jetzt“	nichts	Keine Ausgabe, Feedbacks sind aus der Datenbank entfernt worden.	
69	deleteFeedbacks	gültiger User actor, gültiger User user, startdate, enddate, hier: : Actor mit UserID „becker00“ und Recht „Constants.AGENTURLEITER“ und user mit UserID „becker01“, startdate „10000“ und enddate=„jetzt“	nichts	Keine Ausgabe, Feedbacks sind aus der Datenbank entfernt worden.	
70	deleteFeedbacks	gültiger User actor, gültiger User user, startdate, enddate, hier: : Actor mit UserID „ajeche00“ und Recht „Constants.VAD“ und user mit UserID „ajeche00“, startdate „10000“ und enddate=„jetzt“	nichts	Keine Ausgabe, Feedbacks sind aus der Datenbank entfernt worden.	
71	deleteFeedbacks	Nicht gültiger User actor, gültiger User user, startdate, enddate, hier: : Actor mit UserID „becker00“ und Recht „Constants.AGENTURLEITER“ und user mit UserID „bidaum00“, startdate „10000“ und enddate=„jetzt“	NoRightException	Stimmt mit der erwarteten Ausgabe überein.	

72	getName	nichts	„Administration“	Stimmt mit der erwarteten Ausgabe überein.	
73	getVersionString	nichts	VersionString	Stimmt mit der erwarteten Ausgabe überein.	

9.1.3 Komponententest Subsystem Procurement Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Subsystem):	Procurement	Testteilnehmer:	Christian Leifkes
Testzeitraum: (Datum)	8.5.2000 – 12.5.2000		Wahid Bashirazad

Lfd.Nr	betroffene Methode(n)	Eingabe / Vorgehen	erwartete Ausgabe	tatsächliche Ausgabe / Beobachtung	Kommentar
1	createUser	Gültiges User-Objekt, Budgetgrenze und Budgetzeitraum. SmartStoredatenbank ist vor dem Einfügen leer.	kein Rückgabewert User-Objekt wird in der SmartStoredatenbank in die Tabellen „adress“ und „bestellkonto“ eingetragen.	Alles ok.	
2	createUser	Gültiges User-Objekt, Budgetgrenze und Budgetzeitraum. SmartStoredatenbank enthält bereits Einträge.	kein Rückgabewert User-Objekt wird in der SmartStoredatenbank in die Tabellen „adress“ und „bestellkonto“ eingetragen.	Alles ok.	
3	createUser	Gültiges User-Objekt, Budgetgrenze und Budgetzeitraum. SmartStoredatenbank enthält bereits einen Eintrag für diesen User.	UserExistsException wird geschmissen. SmartStoredatenbank wird nicht verändert.	Nach kleiner Änderung ok.	
4	createUser	null als User-Objekt	NullPointerException wird geschmissen.	Nach Anpassung wurde die Exception geschmissen.	Ungültige Werte von orderLimit und month werden bereits im Controller abgefangen.
5	modifyUser	Gültiges User-Objekt User existiert in der SmartStoredatenbank.	Kein Rückgabewert Die Änderungen in der Tabelle „adress“ der SmartStoredatenbank werden durchgeführt.	Alles ok.	
6	modifyUser	Gültiges User-Objekt User existiert nicht in der SmartStoredatenbank.	UserNotFoundException wird geschmissen. Die Tabelle „adress“ der SmartSto-	Alles ok.	

			redatenbank wird nicht verändert.		
7	modifyUser	null als User-Objekt	NullPointerException wird geschmissen	Nach kleiner Änderung ok.	
8	deleteUser	Gültiges User-Objekt User existiert in der SmartStoredatenbank.	Kein Rückgabewert Die Änderungen in der Tabelle „adress“ der SmartStoredatenbank werden durchgeführt.	Alles ok.	
9	deleteUser	Gültiges User-Objekt User existiert nicht in der SmartStoredatenbank.	UserNotFoundException wird geschmissen. Die Tabelle „adress“ der SmartStoredatenbank wird nicht verändert.	Alles ok.	
10	deleteUser	null als User-Objekt	NullPointerException wird geschmissen	Nach kleiner Änderung ok.	
11	search	SearchRequest mit gültigen SearchCriteria und SearchString Passende Artikeln in der Datenbank vorhanden.	SearchResultSet beinhaltet die passenden Artikeln.	Alles ok.	
12	search	SearchRequest mit gültigen SearchCriteria und SearchString Passende Artikeln in der Datenbank nicht vorhanden.	SearchResultSet ist leer.	Alles ok.	
13	search	null als SearchRequest	NullPointerException wird geschmissen.	Nach kleiner Änderung ok.	
14	getUserOrderAccount	Gültiger User User Existiert in der Datenbank.	OrderAccount-Objekt mit den korrekten Daten des Bestellkontos.	Alles ok.	
15	getUserOrderAccount	Gültiger User User Existiert nicht in der Datenbank.	UserNotFoundException wird geschmissen.	Alles ok.	
16	getUserOrderAccount	null als User	NullPointerException wird geschmissen.	Nach kleiner Änderung ok.	

17	setOrderBudget	Gültige User-Objekt, orderLimit und month User existiert in der Datenbank.	Kein Rückgabewert Die Tabelle „bestellkonto“ wird aktualisiert.	Alles ok.	
18	setOrderBudget	Gültige User-Objekt, orderLimit und month User existiert nicht in der Datenbank	UserNotFoundException wird geschmissen.	Alles ok.	
19	setOrderBudget	User-Objekt ist null.	NullPointerException wird geschmissen.	Nach kleiner Änderung ok.	Ungültige Werte von orderLimit und month werden bereits im Controller abgefangen.
20	getOrderHistoryForUser	Gültige User-Objekt, Startdatum und Enddatum. User existiert in der Datenbank. Es gab Bestellungen in der Zeitspanne.	Objekt vom Typ „UserOrderHistory“ mit den Bestelldaten des Benutzers wird zurückgegeben.	Alles ok.	
21	getOrderHistoryForUser	Gültige User-Objekt, Startdatum und Enddatum. User existiert in der Datenbank. Es gab keine Bestellungen in der Zeitspanne.	UserOrderHistory ist leer.	Alles ok.	
22	getOrderHistoryForUser	Gültige User-Objekt, Startdatum und Enddatum. User existiert in der Datenbank. User hat überhaupt keine Bestellungen.	UserOrderHistory ist leer.	Alles ok.	
23	getOrderHistoryForUser	Gültige User-Objekt, Startdatum und Enddatum. User existiert nicht in der Datenbank.	UserNotFoundException wird geschmissen.	Alles ok.	
24	getOrderHistoryForUser	Gültige User-Objekt. Startdatum und Enddatum ergeben ungültigen Zeitraum.	UserOrderHistory ist leer.	UserOrderHistory ist nicht leer, da Calendar ein ungültiges Datum korrigiert, indem z.B. der 40. Monat als der 4. Monat im 3. Jahr behandelt wird. Daher sollte der Controller die eingegebenen Daten überprüfen (vgl. Bemerkung)	Überprüfung des Datums muss von dem Controller vorgenommen werden. Die Reihenfolge von Startdatum und Enddatum ist egal.

25	getOrderHistoryForUser	User-Objekt, Startdatum oder Enddatum ist „null“	NullPointerException wird geschmissen.	Nach kleiner Änderung ok.	
26	ItemGroupHistory.getItemHistoryForItemGroup	Keine Eingabeparameter.	Ein Vector bestehend aus Item-Objekten wird ausgegeben.	Alles ok.	
27	getSearchKeys	Keine Eingabeparameter.	Ein Vector bestehend aus Warengruppennamen wird ausgegeben.	Alles ok.	Als Search-Key wird zusätzlich „alle“ ausgegeben. Bei Verbindungsfehler wird SQLException geschmissen.
28	Orderform.asp	Kreditlimit für 3, 6 und 12-monatige Budgetgrenze testen. Bestellungen finden innerhalb desselben Budgetzeitraumes statt.	Kreditlimit wird bei jeder Bestellung berücksichtigt.	Nach kleiner Änderung ok	
29	Orderform.asp	Kreditlimit für 3, 6 und 12-monatige Budgetgrenze testen. Bestellungen finden in mehreren Budgetzeiträumen statt.	Kreditlimit wird bei jeder Bestellung richtig berücksichtigt und dabei zurückgesetzt.	Nach kleiner Änderung ok	
30	ipsi_cust_auth.asp	Es wird ein cookie mit korrekten Anmeldedaten gesetzt.	Die Anmeldeseite des Shops wird übersprungen und der Benutzer wird gegenüber dem Shop identifiziert.	Alles ok	
31	ipsi_cust_auth.asp	Es wird ein cookie mit ungültigen Login gesetzt.	Die Login-Seite des Shops wird Aufgerufen.	Alles ok	Nach der Integration des Shops im Portal, soll statt Login-Seite des Shops eine Fehlermeldung ausgegeben werden. Es ist zur Zeit nicht möglich, weil es davon anhängig ist, wo ipsi_cust_auth.asp aufgerufen wird. (Client-Browser, Controller oder Shop-Subsystemfacade)
32	ipsi_cust_auth.asp	Es wird ein cookie mit falschem Passwort gesetzt.	Die Login-Seite des Shops wird Aufgerufen.	Alles ok	Siehe 31

9.1.4 Komponententest Subsystem Kommunikation Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Subsystem):	Communication	Testteilnehmer:	Hassan, Matthias
Testzeitraum: (Datum)	09.05.2000		

Lfd.Nr	betroffene Methode(n)	Eingabe / Vorgehen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung	Kommentar
1	CommunicationSubSystemFacade	null statt Properties-Feld	Fehlermeldung	NullPointerException	FEHLER: Wurde zunächst nicht von der Methode abgefangen. Methode nach dem Test um Code ergänzt, der entsprechende Exception mit Hinweistext ausgibt.
2	CommunicationSubsystemFacade	Properties-Feld ohne Werte	zu versendende Nachrichten werden verworfen	Nachrichten durchlaufen alle Fallback-Medien und werden dann verworfen.	Es existieren keine Versandkanäle → kein Versand möglich.
3	CommunicationSubsystemFacade	Properties-Feld mit ungültiger Kanal-ID: BrieftaubenChannel0: Taube0 Taube0_ID: Erna	zu versendende Nachrichten werden verworfen	Nachrichten durchlaufen alle Fallback-Medien und werden dann verworfen.	Da der BrieftaubenChannel dem System nicht bekannt ist, werden diese Properties überhaupt nicht abgefragt, eine Fehlermeldung ist daher nicht nötig.
4	CommunicationSubsystemFacade	Properties-Feld mit fehlenden Werten (z.B. SMTP0_ID) EmailChannel0: SMTP0 SMTP0_USER: SMTP0_PASSWD:	Fehlermeldung, da wichtige Information (hier: ID) fehlt.	keine Ausgabe	FEHLER: Methode fragt nicht ab, ob eine benötigte Information fehlt → entsprechende Exception muss ergänzt werden.
5	CommunicationSubsystemFacade (korrigiert)	Properties-Feld mit fehlenden Werten (z.B. SMTP0_ID) EmailChannel0: SMTP0 SMTP0_USER: SMTP0_PASSWD:	Fehlermeldung, da wichtige Information (hier: ID) fehlt.	MissingPropertyException	Fehler wird jetzt wie erwartet abgefangen.
6	CommunicationSubsystemFacade	Properties-Feld mit gültigen Werten: EmailChannel0: SMTP0 SMTP0_ID: homer SMTP0_USER: SMTP0_PASSWD:	keine Ausgabe, Emailkanal wird initialisiert. Nachrichten werden normal versandt.	Nachrichten werden normal versandt.	Ob der Kanal tatsächlich initialisiert wurde und die Parameter korrekt sind, merkt man beim Versand einer Email (die Facade kann die Korrektheit der Parameter nicht prüfen).
7	CommunicationSubsystemFacade	Properties-Feld mit falschen Werten: EmailChannel0: SMTP0 SMTP0_ID: marge	keine Ausgabe, Emailkanal wird initialisiert. Nachrichten können nicht versandt werden.	Nachrichten werden an Fallback-Medien weitergegeben	Die Facade kann die Korrektheit der Parameter nicht prüfen, daher wird der Fehler erst beim Versand bemerkt.

		SMTP0_USER: SMTP0_PASSWD:			
8	insertMessage	null statt Message-Objekt	Fehlermeldung	NullPointerException	Fehler wie erwartet abgefangen.
9	insertMessage	TaskReminder sofort als E-mail versenden	sofortiger Emailempfang	Email empfangen.	Emailversand funktioniert.
10	insertMessage	TaskReminder sofort als Fax versenden	sofortiger Emailempfang, da Fax und SMS nicht eingerichtet sind	Email empfangen	Fallback-Mechanismus funktioniert
11	insertMessage	TaskReminder mit ungültigem Mediencode (1234)	Fehlermeldung	IllegalMediumException	Fehler wie erwartet abgefangen.
12	insertMessage	TaskReminder ohne gesetzten Mediencode	Fehlermeldung	IllegalMediumException	Fehler wie erwartet abgefangen.
13	insertMessage	TaskReminder in einer Minute per Email verschicken	Emailempfang nach einer Minute	Email zur erwarteten Zeit emfpangen	Priority-Queue funktioniert.
14	insertMessage	TaskReminder mit fehlender Emailadresse	Ausweichen auf Fallback-Medien, Fehlermeldung im Log	System schreibt Fehlermeldung und weicht aus. Da keine Fallback-Kanäle existieren, wird die Nachricht verworfen.	Mechanismus funktioniert.
15	insertMessage	TaskReminder mit ungültiger Emailadresse m@book@ipsi.de	Fehlermeldung, Ausweichen auf Fallback-Medien	System schreibt Fehlermeldung und weicht aus. Da keine Fallback-Kanäle existieren, wird die Nachricht6verworfen.	Mechanismus funktioniert.
16	insertMessage	TaskReminder ohne Task-Objekt	Fehlermeldung	nicht abgefangene NullPointer-Exception	FEHLER: ungültiges Objekt wird nicht abgefangen.
17	insertMessage (korrigiert)	TaskReminder ohne Task-Objekt	Fehlermeldung	NullPointerException mit Erläuterung	Fehler jetzt abgefangen.

18	insertMessage	TaskReminder ohne Sender- und Empfänger-Objekt	Fehlermeldung	NullPointerException mit Erläuterung	Fehler abgefangen.
19	insertMessage	AppointReminder als Email versenden	Reminder wird versandt.	Reminder wird versandt.	Versand funktioniert.
20	insertMessage	AppointmentReminder mit ungültiger Adresse m@book@ipsi.de versenden	Fehlermeldung, Ausweichen auf Fallback-Medien	System schreibt Fehlermeldung und weicht aus. Da keine Fallback-Kanäle existieren, wird die Nachricht verworfen.	Fallback-Mechanismus funktioniert.
21	insertMessage	AppointmentReminder mit falscher Adresse mbook@ipsi.de versenden	3maliger Retry, dann Fallback mit entsprechenden Log-Fehlermeldungen	3maliger Retry, dann Fallback mit entsprechenden Log-Fehlermeldungen	Retry-Mechanismus funktioniert.
22	insertMessage	ClientComment als Email verschicken	Email wird versandt	Email wird versandt, Subject wird jedoch nicht ausgegeben	FEHLER: Versand funktioniert, Subject-Ausgabe wird ergänzt.
23	insertMessage	ClientComment als Fax verschicken	kein Faxkanal → Versand per Email	Email wird versandt, Subject jetzt auch ausgegeben.	Fallback funktioniert.
24	insertMessage	ClientComment als SMS verschicken	ungültiger Medientyp → Nachricht verwerfen	Exception wird geworfen, Nachricht verworfen	Da SMS nicht als Versand zugelassen ist, gibt's auch keine Fallback-Strategie, die Nachricht wird vielmehr sofort verworfen.
25	insertMessage	ClientComment als Fax verschicken und falsche Email-Adresse angegeben	Fallback auf Email, 3x Retry, Nachricht verwerfen und Vorgänge protokollieren	Fallback auf Email, 3x Retry, Nachricht verwerfen und Vorgänge protokollieren	Mechanismus funktioniert.
26	insertMessage	ClientComment mit fehlenden Feldern verschicken	Fehlendes Feld fehlt auch in der Email	An Stelle des fehlenden Felds wird „null" angezeigt	Fehlende Felder beeinträchtigen Verarbeitung nicht.
27	changeAppointmentReminder	Uhrzeit eines Reminders vorverlegen und Subject ändern	Email mit neuem Subject wird früher verschickt.	Email mit neuem Subject wird früher verschickt.	Verschiebung funktioniert. Der fragliche Queueeintrag wird auch unter mehreren Einträgen gefunden.
28	changeAppointmentReminder	Uhrzeit eines Reminders zurückverlegen und Subject ändern	Email mit neuem Subject wird später verschickt	Email mit neuem Subject wird später verschickt	Verschiebung funktioniert.
29	changeAppointmentReminder	Termin verschieben, der nicht in der Queue ist	Fehlermeldung	AppointmentNotFoundException	Fehler abgefangen.

30	changeAppointmentReminder	null-Termin übergeben	Fehlermeldung	NullPointerException mit Erläuterung	Fehler wird abgefangen.
31	deleteAppointmentReminder	in der Queue enthaltenen Reminder löschen	Reminder wird gelöscht	Reminder wird gelöscht	Mechanismus funktioniert.
32	deleteAppointmentReminder	nicht in der Queue enthaltenen Reminder löschen	Fehlermeldung	AppointmentNotFoundException	Fehler abgefangen.
33	deleteAppointmentReminder	null-Termin übergeben	Fehlermeldung	NullPointerException mit Erläuterung	Fehler wird abgefangen.
34	changeTaskReminder	Uhrzeit eines Reminders vorverlegen und Subject ändern	Email mit neuem Subject wird früher verschickt.	Email mit neuem Subject wird früher verschickt.	Verschiebung funktioniert. Der fragliche Queueeintrag wird auch unter mehreren Einträgen gefunden.
35	changeTaskReminder	null-Task übergeben	Fehlermeldung	NullPointerException mit Erläuterung	Fehler wird abgefangen.
36	changeTaskReminder	Task verschieben, der nicht in der Queue ist	Fehlermeldung	TaskNotFoundException	Fehler abgefangen.
37	deleteTaskReminder	nicht in der Queue enthaltenen Task löschen	Fehlermeldung	TaskNotFoundException	Fehler abgefangen.
38	deleteTaskReminder	in der Queue enthaltenen Reminder löschen	Reminder wird gelöscht	Reminder wird gelöscht	Mechanismus funktioniert.
39	deleteTaskReminder	null-Termin übergeben	Fehlermeldung	NullPointerException mit Erläuterung	Fehler wird abgefangen.
40	readLogEntries	null, STATUS_ALL, MEDIUM_MAIL	alle Email-Ereignisse aller User	alle Email-Ereignisse aller User	Filter funktioniert (keine User- und Status-Einschränkung)
41	readLogEntries	[hghane], STATUS_ALL, MEDIUM_MAIL	alle Email-Ereignisse von Hassan	alle Email-Ereignisse von Hassan	Filter funktioniert (User-Einschränkung)

42	readLogEntries	[hghane], STATUS_ALL, MEDIUM_FAX	keine Ereignisse	keine Ereignisse	hat nicht gefaxt ☺
43	readLogEntries	unbekannter Status- bzw. Mediencode	keine Ereignisse	keine Ereignisse	wird nicht als Fehler zurückgegeben, da readLogEntries so unabhängig von den zur Zeit existierenden Stati bzw. Medien bleibt.
44	clearLog	bekannter User	nur Ereignisse des Users werden gelöscht	nur Ereignisse des Users werden gelöscht	Filter funktioniert.
45	clearLog	nicht existierender User	keine Ereignisse werden gelöscht	keine Ereignisse werden gelöscht	keine Fehlermeldung, da der Datenbank-Rückgabewert nicht erkennen läßt, wieviele Zeilen gelöscht wurden (außerdem liegt nicht wirklich ein Fehler vor).
46	clearLog	null	alle Ereignisse werden gelöscht	alle Ereignisse werden gelöscht	Filter funktioniert.

9.1.5 Komponententest Subsystem Legacy Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Subsystem):	Legacy	Testteilnehmer:	Christian, Stefan P.
Testzeitraum: (Datum)	26.06.00		

Lfd.Nr	betroffene Methode(n)	Eingabe / Vorgehen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung	Kommentar
1	getCustomer (String, Vector)	versnr = 10002 (gültig), users = leerer Vector	kein Kunde gefunden -> return null	ok	
2	getCustomer (String, Vector)	versnr = 10002 (gültig), users = null	LegacyException wird geschmissen	return null	Geändert: Jetzt wird die LegacyException geschmissen.
3	getCustomer (String, Vector)	versnr = 10002 (gültig), users = „bidaum00“ – der User hat aber keine Rechte, auf diesen Kunden zuzugreifen.	kein Kunde gefunden -> return null	ok	

3	getCustomer (String, Vector)	versnr = 10002 (gültig), users = „goebel01“- der User hat das Recht, auf diesen Kunden zuzugreifen.	Der Kunde mit entsprechender Versicherungsnummer wird zurückgegeben.	ok	
4	getCustomer (String, Vector)	versnr = 10002 (gültig), users = „bidaum00“ + „goebel01“ – Einer der User hat das Recht auf den Kunden zuzugreifen.	Der Kunde mit entsprechender Versicherungsnummer wird zurückgegeben.	ok	
5	getCustomer (String, Vector)	versnr = 10003 (ungültig), users = „goebel01“	LegacyException wird geschmissen	ok	
6	getCustomer (String, Vector)	versnr = null, users = „goebel01“	LegacyException wird geschmissen	ok	
7	getCustomer (String, Vector)	versnr = null, users = null	LegacyException wird geschmissen	ok	
8	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = 01.01.1950, users = „goebel01“ – der User hat das Recht, auf diesen Kunden zuzugreifen	Das User-Objekt für Margot Schmidt wird zurückgegeben	ok	
9	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = 01.01.1950, users = „bidaum00“ – der User hat nicht das Recht, auf diesen Kunden zuzugreifen	Kunde wird nicht gefunden: return null	ok	
10	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = 01.01.1950,	Das User-Objekt für Margot Schmidt wird zurückgegeben	ok	

	tor)	users = „goebel01“ + „becker01“ – einer der User hat das Recht, auf diesen Kunden zuzugreifen			
11	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = 01.01.1950, users = „bidaum00“ + „becker01“ – keiner der User hat das Recht, auf diesen Kunden zuzugreifen	Kunde wird nicht gefunden: return null	ok	
12	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = 01.01.1950, users = leerer Vector	Kunde wird nicht gefunden: return null	ok	
13	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = 01.01.1950, users = null	LegacyException wird geschmissen	ok	
14	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = 01.01.1955, users = „goebel01“	Der Kunde existiert nicht in der Kundendatenbank (falsches Geb.-Datum) -> return null	ok	
15	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Julia, birthday = 01.01.1950, users = „goebel01“	Der Kunde existiert nicht in der Kundendatenbank (falscher Vorname) -> return null	ok	
16	getCustomer (String, String, Calendar, Vector)	name = Schmitte, vorname = Margot, birthday = 01.01.1950, users = „goebel01“	Der Kunde existiert nicht in der Kundendatenbank (falscher Name) -> return null	ok	

17	getCustomer (String, String, Calendar, Vector)	name = null, vorname = Margot, birthday = 01.01.1950, users = „goebel01“	LegacyException wird geschmissen	ok	
18	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = null, birthday = 01.01.1950, users = „goebel01“	LegacyException wird geschmissen	ok	
19	getCustomer (String, String, Calendar, Vector)	name = Schmidt, vorname = Margot, birthday = null, users = „goebel01“	LegacyException wird geschmissen	ok	
20	getCustomer-Numbers	users = null	LegacyException wird geschmissen	ok	
21	getCustomer-Numbers	users = leerer Vector	Ein leerer Vector wird zurückgegeben.	ok	
22	getCustomer-Numbers	users = „goebel01“	return „10002“ + „10112“	ok	
23	getCustomer-Numbers	users = „bidaum00“	return „10111“	ok	
24	getCustomer-Numbers	users = „becker01“	return leerer Vector	ok	
25	getCustomer-Numbers	users = „goebel01“ + „bidaum00“	return „10002“ + „10112“ + „10111“	ok	

26	getName	./.	„Legacy“	ok	
27	getSearchKeys	./.	Die verfügbare SearchKeys werden zurückgegeben.	ok	
28	queryByContractCategory	users = „becker01“	becker01 hat keine Kunden.	ok	
29	queryByContractCategory	users = „goebel01“	Category 71: 8* Category 77: 2*	ok	
30	queryByContractCategory	users = null	LegacyException wird geschmissen	ok	
31	queryByEnteredDate	startDate = 10.03.1998 endDate = 18.03.1998 users = „goebel01“	Margot Schmidt muss gefunden werden.	ok	
32	queryByEnteredDate	startDate = 18.03.1996 endDate = 22.03.1996 users = „goebel01“	Otti Mersch muss gefunden werden	ok	
33	queryByEnteredDate	startDate = 16.02.1996 endDate = 22.06.1998 users = „goebel01“	Otti Mersch und Margot Schmidt müssen gefunden werden; Eva Jücker darf nicht gefunden werden.	ok	
34	queryByEnteredDate	startDate = 10.03.1998 endDate = 18.03.1998 users = „bidaum00“	Es darf kein Kunde gefunden werden, da „bidaum00“ keine Rechte hat, diesen Kunden zu betrachten.	ok	
35	queryByEnteredDate	startDate = 16.02.1996 endDate = 22.06.1998 users = null	LegacyException wird geschmissen	ok	

36	queryByEnteredDate	startDate = 16.02.1996 endDate = 22.06.1998 users = leerer Vector	Es wird kein Kunde gefunden	ok	
37	queryByEnteredDate	startDate = null endDate = 01.01.90 users = „goebel01“	LegacyException wird geschmissen	ok	
38	queryByEnteredDate	startDate = 01.01.90 endDate = null users = „goebel01“	LegacyException wird geschmissen	ok	
39	queryByEnteredDate	startDate = 22.06.1998 endDate = 16.02.1996 users = „goebel01“	Otti Mersch und Margot Schmidt müssen gefunden werden; Eva Jücker darf nicht gefunden werden. Start- und Enddatum sind vertauscht.	ok	
40	queryByPostalCode	from = 00000 to = 99999 limit = 5 users = „goebel01“	Otti Mersch und Margot Schmidt müssen gefunden werden; Eva Jücker darf nicht gefunden werden, da goebel01 keine Rechte hat.	ok	
41	queryByPostalCode	from = 00000 to = 44227 limit = 5 users = „goebel01“	Es darf nur Otti Mersch gefunden werden.	ok	
42	queryByPostalCode	from = 51234 to = 55555 limit = 1 users = „goebel01“ + „bidaum00“	Otti Mersch und Eva Jücker wird gefunden	ok	
43	queryByPostalCode	from = 512 to = 5555 limit = 1 users = „goebel01“ + „bidaum00“	LegacyException wird geschmissen (from und to müssen 5-stellig sein)	JAVA-interne Exception wird geschmissen	Geändert: Jetzt wird die Legacy-Exception geschmisssen.
44	queryByPostalCode	from = 99999 to = 00000 limit = 1 users = „goebel01“ + „bi-	LegacyException wird geschmissen (from und to sind vertauscht)	ok	

		„bidaum00“			
45	queryBy-PostalCode	from = -10000 to = 99999 limit = 5 users = „goebel01“ + „bi-daum00“	LegacyException wird geschmis-sen (from und to müssen im Be-reich von 00000 bis 99999 sein)	ok	
46	queryBy-PostalCode	from = bla to = 55555 limit = 1 users = „goebel01“ + „bi-daum00“	LegacyException wird geschmis-sen	ok	
47	queryBy-PostalCode	from = null to = 55555 limit = 1 users = „goebel01“ + „bi-daum00“	LegacyException wird geschmis-sen	ok	
48	queryBy-PostalCode	from = 51234 to = null limit = 1 users = „goebel01“ + „bi-daum00“	LegacyException wird geschmis-sen	ok	
49	queryBy-PostalCode	from = 51234 to = bla limit = 1 users = „goebel01“ + „bi-daum00“	LegacyException wird geschmis-sen	ok	
50	queryBy-PostalCode	from = 51234 to = 55555 limit = 1 users = null	LegacyException wird geschmis-sen	ok	
51	queryBy-PostalCode	from = 51234 to = 55555 limit = 1 users = leerer Vector	Es wird kein User gefunden	ok	

52	query-ByTimePeriod	startDate = 18.03.1998 endDate = 20.04.1998 users = „goebel01“	Vertragsvolumen für Margot Schmidt wird berechnet.	ok	
53	query-ByTimePeriod	startDate = 20.04.1998 endDate = 18.03.1998 users = „goebel01“	Vertragsvolumen für Margot Schmidt wird trotzdem richtig berechnet. (Start- und Ende ist vertauscht)	ok	
54	query-ByTimePeriod	startDate = 18.03.1998 endDate = 20.04.1998 users = null	LegacyException wird geschmissen	ok	
55	query-ByTimePeriod	startDate = 18.03.1998 endDate = null users = „goebel01“	LegacyException wird geschmissen	ok	
56	query-ByTimePeriod	startDate = null endDate = 20.04.1998 users = „goebel01“	LegacyException wird geschmissen	ok	
57	search	SearchRequest: legacy_customerNumber = 10.002 legacy_gueltigAbDatum_>= 18.03.1998 legacy_gueltigAbDatum_<= 19.03.1998 users = „goebel01“	Margot Schmidt wird gefunden.	ok	
58	search	SearchRequest: legacy_customerNumber = 10.002 legacy_gueltigAbDatum_>= 18.03.1998 legacy_gueltigAbDatum_<= 19.03.1998 users = null	LegacyException wird geschmissen	ok	
59	search	SearchRequest: null users = „goebel01“	LegacyException wird geschmissen	ok	

60	search	SearchRequest: legacy_customerNumber = 10.002 legacy_gueltigAbDatum_>= 18.03.1998 legacy_gueltigAbDatum_<= 19.03.1998 users = „becker00“	Kein Kunde wird gefunden. return null	ok	

9.1.6 Systemtest Subsystem Office (Appointment) Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	Office - Appointment	Testteilnehmer:	Stefan Göbel
Testzeitraum: (Datum)	07.08.2000 bis 07.08.2000		

Lfd.Nr	betroffene Action	Eingabe / Vorgehen beim Testen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung / Kommentar	Ok?	Priorität 1-3 1 = hoch/krit.	Testgrundlage (Build)
	AppoinmentModifyMask	Klick auf einen Termin im Startbildschirm	Genau dieser Termin wird korrekt mit allen Details angezeigt				
	AppoinmentList	Klick auf „Termine“ im Startbildschirm	Es wird die Monatsansicht des aktuellen Monats gezeigt, und zwar so wie im MVD vorgeschrieben	• Es erscheint eine Monatsübersicht, allerdings nicht als Kalenderblatt; • die Termine sind unsortiert; • Termine am letzten Tag des Monats werden nicht dargestellt		• 3 • 1 • 1	
	AppointmentList	Überprüfe alle Links der NavBar	„Erstellen“, „Home“, „Suchen“	Alle Links funktionieren einwandfrei	X		
	AppoinmentList	Klick auf „Blättern +“	Es wird die Monatsansicht des nächsten Monats gezeigt, und zwar so wie im MVD vorgeschrieben	Siehe oben		• 3 • 1 • 1	
	AppoinmentList	Klick auf „Blättern -“	Es wird die Monatsansicht des vorherigen Monats gezeigt, und zwar so wie im MVD vorgeschrieben	Siehe oben		• 3 • 1 • 1	
	AppoinmentList -> AppoinmentDailyList	Klicke auf einen Tag des Monats	Genau dieser Tag wird in der Tagesansicht gemäß MVD dargestellt	• Es erscheint eine Tagessübersicht, allerdings nicht als Kalenderblatt; • Es werden auch Termine von anderen Tagen (+1, -1) dargestellt		• 3 • 1 • 1	

	lyList			deren Tagen (+1,-1) dargestellt • Die Ausgabe ist unsortiert			
	AppointmentDaily-List	Überprüfe NavBar	„Home“, „Erstellen“, „Suche“, „Monatsübersicht“	Alle Links funktionieren einwandfrei	X		
	AppointmentDaily-List	Klick auf „Blättern+“	Es wird die Tagesansicht des nächsten Tages gezeigt	Siehe oben		• 3 • 1 • 1	
	AppointmentDaily-List	Klick auf „Blättern-“	Es wird die Tagesansicht des vorherigen Tages gezeigt	Siehe oben		• 3 • 1 • 1	
	AppointmentDaily-List -> Appoinment-ModifyMask	Klicke auf einen Termin	Es wird die Detailansicht des Termins gezeigt		X		
	AppointmentModifyMask	Überprüfe NavBar	„Home“, „Erstellen“, „Suche“, „Monatsübersicht“	Alle Links funktionieren einwandfrei, allerdings ist die Reihenfolge der Links anders als oben		3	
	AppointmentModifyMask -> AppointmentModify	Ungültiges Datum in „StartDatum“ bzw. „Enddatum“	Fehlermeldung mit richtigem Datumsformat, Feld „StartDatum“ bzw. „EndDatum“ ist markiert	Das richtige Feld ist markiert, allerdings wird kein erklärender Fehlertext mit richtigem Datumsformat angezeigt		3	
	AppointmentModifyMask -> AppointmentModify	Fehlendes Datum in „StartDatum“ bzw. „EndDatum“	Fehlermeldung, Feld „StartDatum“ bzw. „EndDatum“ ist markiert	Fehlendes Startdatum: NullPointerException tritt auf Fehlendes Enddatum: Office_Subsystem_Error		1	
	AppointmentModifyMask -> AppointmentModify	Ungültiges Datum in „StartDatum“ UND „Enddatum“	Fehlermeldung mit richtigem Datumsformat, Feld „StartDatum“ UND „EndDatum“ sind markiert	Nur das erste Feld ist markiert, das andere ist unverändert, keine richtige Fehlermeldung (s.o)		3	

	AppointmentModifyMask -> AppointmentModify	Fehlendes Datum in „StartDatum“ UND „Enddatum“	Fehlermeldung, Feld „StartDatum“ UND „EndDatum“ sind markiert	NullPointerException		1	
	AppointmentModifyMask -> AppointmentModify	Feld „Titel“ nicht ausgefüllt	Fehlermeldung, Feld „Titel“ ist markiert	Feld „Titel“ ist markiert, die Fehlermeldung enthält keinen erklärenden Text		3	
	AppointmentModifyMask -> AppointmentModify	Alle Felder korrekt ausgefüllt	Termin wurde geändert, Tagesansicht erscheint		X		
	AppointmentModifyMask -> AppointmentDeleteMask	Klick auf „Löschen“	Sicherheitsabfrage erscheint		X		
	AppointmentDeleteMask -> AppointmentDelete	Klick auf „ja“	Termin wird gelöscht, Tagesansicht erscheint		X		
	AppointmentDeleteMask -> AppointmentModifyMask	Klick auf „nein“	Termin wird nicht gelöscht, Termindetailansicht erscheint		X		
	AppointmentDailyList -> AppointmentDeleteMask	Klick auf „Löschen“	Sicherheitsabfrage erscheint		X		

	AppointmentInsertMask	Überprüfe NavBar	„Home“, „Suchen“, „Monatsübersicht“	Alle Links arbeiten einwandfrei	X		
	AppointmentInsertMask -> AppointmentInsert	Ungültiges Datum in „StartDatum“ bzw. „Enddatum“	Fehlermeldung mit richtigem Datumsformat, Feld „StartDatum“ bzw. „EndDatum“ ist markiert	Das richtige Feld ist markiert, allerdings wird kein erklärender Fehlertext mit richtigem Datumsformat angezeigt		3	
	AppointmentInsertMask -> AppointmentInsert	Fehlendes Datum in „StartDatum“ bzw. „EndDatum“	Fehlermeldung, Feld „StartDatum“ bzw. „EndDatum“ ist markiert	Fehlendes Startdatum: NullPointerException tritt auf Fehlendes Enddatum: Office_Subsystem_Error		1	
	AppointmentInsertMask -> AppointmentInsert	Ungültiges Datum in „StartDatum“ UND „Enddatum“	Fehlermeldung mit richtigem Datumsformat, Feld „StartDatum“ UND „EndDatum“ sind markiert	Nur das erste Feld ist markiert, das andere ist unverändert, keine richtige Fehlermeldung (s.o)		3	
	AppointmentInsertMask -> AppointmentInsert	Fehlendes Datum in „StartDatum“ UND „Enddatum“	Fehlermeldung, Feld „StartDatum“ UND „EndDatum“ sind markiert	NullPointerException		1	
	AppointmentInsertMask -> AppointmentInsert	Feld „Titel“ nicht ausgefüllt	Fehlermeldung, Feld „Titel“ ist markiert	Feld „Titel“ ist markiert, die Fehlermeldung enthält keinen erklärenden Text		3	
	AppointmentInsertMask -> AppointmentInsert	Alle Felder korrekt ausgefüllt	Termin wurde eingefügt, Tagesansicht erscheint	Wurde vorher keine Fehlermeldung provoziert, klappt das Einfügen einwandfrei; sonst wird der Termin mit falschem Datum eingefügt ****BEMERKUNG**** Überhaupt scheinen vorherige Fehlermeldungen das erstellen bzw. ändern von Terminen zu behindern		1++	
	AppointmentSearchMask –> Appoint-	Ungültige Eingabe in „Startdatum“	Fehlermeldung und richtiges Datumsformat	Fehlermeldung, aber kein erklärender Text		3	

	mentSearch						
	AppointmentSearchMask -> AppointmentSearch	Ungültige Eingabe in „Enddatum“	Fehlermeldung und richtiges Datumsformat				
	AppointmentSearchMask -> AppointmentSearch	Feld „Titel“ ausgefüllt, ungültige Eingabe „Startdatum“	Fehlermeldung und richtiges Datumsformat, Feld „Titel“ noch gefüllt	Fehlermeldung ohne erklärenden Text, Feld „Titel“ nicht mehr gefüllt		3	
	AppointmentSearchMask -> AppointmentSearch	Joker (*) im Feld „Subject“ benutzt	Alle Appointments werden gefunden	Alle Termine werden gefunden, allerdings als unsortierte Liste dargestellt		2	
	AppointmentSearchMask -> AppointmentSearch	Alle Felder bleiben leer	Alle Appointments werden gefunden	Alle Termine werden gefunden, allerdings als unsortierte Liste dargestellt		2	
	AppointmentList	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		
	AppointmentDailyList	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		
	TaskInsertMask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn	Erst beim Erstellen kommt die Fehlermeldung		3	
	TaskModifyMask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		

	TaskSearch-Mask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		
	TaskDelete-Mask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		

9.1.7 Systemtest Subsystem Office (Contact) Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	Office	Testteilnehmer:	Leifkes
Testzeitraum: (Datum)	04.08.00		

Lfd.Nr	betroffene Action	Eingabe / Vorgehen beim Testen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung / Kommentar	Ok?	Priorität 1-3 1 = hoch/krit.	Testgrundlage (Build)
1	ContactSearch	Aus der Home-Maske heraus Adressbuch aufrufen	Die ContactSearchMask soll aufgerufen werden und nach „A*“ suchen.		X		aktuelle Sourcen vom 03.08.00
2	ContactSearch	Alle Links der NavBar prüfen in der Maske ContactSearch-Mask (Erstellen, Home)	Die Links sollen auf die angegebenen Seiten führen		X		aktuelle Sourcen vom 03.08.00
3	ContactSearch	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Die Seite bleibt weiß. Absturz durch eine NullpointerException in der getOfficeSubsystem-Methode		2 [korrigiert 06.08.00: CH]	aktuelle Sourcen vom 03.08.00
4	ContactSearch	Click auf einen Buchstaben in der ContactSearchMask	Die passenden Kontakte werden gefunden und in der Contact-SearchMask angezeigt. Zu jedem Kontakt sind 3 Links vorhanden: Ändern (durch click auf den Namen), Ändern (durch click auf den Ändern-Button) und Löschen (durch click auf den Löschen-Button).		X		aktuelle Sourcen vom 03.08.00
5	ContactSearch	Ein leerer Suchstring und click auf „Suchen“	Keine Kontakte werden gefunden.		X		aktuelle Sourcen vom 03.08.00

6	ContactSearch	Suche nach *	Alle Kontakte werden gefunden		X		aktuelle Sourcen vom 03.08.00
7	ContactSearch	Suche nach T*	Alle Kontakte, die mit T beginnen werden aufgelistet		X		aktuelle Sourcen vom 03.08.00
8	ContactSearch	Suche nach t*	Alle Kontakte, die mit T beginnen, werden aufgelistet	Keine Kontakte werden gefunden. (Evtl. sollte man hier Klein-/Großschreibung ignorieren)		3 [korrigiert 06.08.00: CH]	aktuelle Sourcen vom 03.08.00
9	ContactSearch	Suche nach *t*	Alle Kontakte, die ein t enthalten werden aufgelistet		X		aktuelle Sourcen vom 03.08.00
10	ContactSearch	Suche nach *e	Alle Kontakte, die mit e enden werden aufgelistet		X		aktuelle Sourcen vom 03.08.00
11	ContactCreate	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Eine leere Seite erscheint.		2 [korrigiert 06.08.00: CH]	aktuelle Sourcen vom 03.08.00
12	ContactCreate	Ausgangspunkt: ContactCreateMask. Alle Felder leer ->„ ok“	Eingabemaske mit Fehlerbeschreibung wird dargestellt		X		aktuelle Sourcen vom 03.08.00
13	ContactCreate	Ausgangspunkt: ContactCreateMask. Name gültig, beginnt mit einem Großbuchstaben; Geburtstag leer ->„ ok“	Liste aller Kontakte, die mit gleichem Buchstaben beginnen. Kontakt wird korrekt eingetragen.	Der Kontakt wird zwar eingetragen, aber das Geburtsdatum wird auf „01.01.4501“ gesetzt.		1 [korrigiert 06.08.00: CH]	aktuelle Sourcen vom 03.08.00
14	ContactCreate	Ausgangspunkt: ContactCreateMask. Name gültig, beginnt mit einem Kleinbuchstaben; Geburtstag leer ->„ok“	Liste aller Kontakte, die mit gleichem Buchstaben beginnen.	Wenn ein Kleinbuchstabe eingegeben wird, wird nicht nach Kontakten, die mit diesem Buchstaben beginnen gesucht. Der Kontakt wird erst gefunden, wenn explizit nach diesem Buchstaben gesucht wird (per Link).		2 [korrigiert 06.08.00 CH]	aktuelle Sourcen vom 03.08.00

15	ContactCreate	Ausgangspunkt: ContactCreateMask. Name gültig Geburtstag != null, aber ungültig ->„ok“	Eingabemaske mit Fehlerbeschreibung wird dargestellt		X		aktuelle Sourcen vom 03.08.00
16	ContactCreate	Ausgangspunkt: ContactCreateMask. Name null Geburtstag gültig ->„ok“	Eingabemaske mit Fehlerbeschreibung wird dargestellt		X		aktuelle Sourcen vom 03.08.00
17	ContactCreate	Ausgangspunkt: ContactCreateMask. Name gültig Geburtstag gültig ->„ok“	Liste aller Kontakte, die mit gleichem Buchstaben beginnen.		X		aktuelle Sourcen vom 03.08.00
18	ContactCreate	Ausgangspunkt: ContactCreateMask Irgendweche Werte eingeben -> „zurücksetzen“	Alle Felder im Formular werden auf die ursprünglichen Werte zurückgesetzt.		X		aktuelle Sourcen vom 03.08.00
19	ContactCreate	Ausgangspunkt: ContactCreateMask. Werte in allen Feldern eintragen -> „ok“	Alle Felder sollen in den Outlook-Kontakt übernommen werden.		X		aktuelle Sourcen vom 03.08.00
20	ContactCreateDialog -> ContactCreateMask	Alle Links der NavBar prüfen in der ContactCreateMask	Die Links sollen auf die angegebene Seite führen	„Übersicht“: Fehler - Wenn zuvor in der Übersicht kein Buchstabe gewählt war, werden trotzdem alle Kontakte mit A angezeigt. Sonst Ok. „Partner-DB“: Link fehlt „Home“: Ok.		3 [Übersicht: ist ok, da defaultmäßig Kontakte angezeigt werden, die mit A beginnen; Partner-DB: Link soll hier auch gar nicht existieren - CH 07.08.00]	aktuelle Sourcen vom 03.08.00

21	ContactCreateDialog	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Der Controller arbeitet auch ohne User-Objekt. Problem: Die Links sind falsch gesetzt: Home führt zu „Missing-Parameter“, Übersicht führt zu einer NullpointerException.		2 [korrigiert: CH 07.08.00 (Fehler, dass Benutzer nicht eingeloggt)]	aktuelle Sourcen vom 03.08.00
22	ContactCreateUserdefFieldDialog	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ wird dargestellt.	Der Controller arbeitet auch ohne User-Objekt. Problem: Die Links sind falsch gesetzt: Home führt zu „Missing-Parameter“, Übersicht führt zu einer NullpointerException.		2 [korrigiert: CH 07.08.00]	aktuelle Sourcen vom 03.08.00
23	ContactCreateUserdefFieldDialog	Ausgangspunkt: ContactCreateMask Es existiert noch kein Userdef-Field -> „benutzerdef Feld zufügen“	Die Maske „benutzerdef. Felder erstellen“ wird dargestellt.		X		aktuelle Sourcen vom 03.08.00
24	ContactCreateUserdefFieldDialog	Ausgangspunkt: ContactCreateMask Es existierten bereits Userdef-Fields -> „benutzerdef Feld zufügen“	Die Maske „benutzerdef. Felder erstellen“ wird dargestellt.		X		aktuelle Sourcen vom 03.08.00
25	ContactCreateUserdefField	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Der Controller arbeitet auch ohne User-Objekt. Allerdings stimmen die Links in der darauf folgenden KontaktErstellen-Maske nicht mehr: Home führt zu „Missing-Parameter“, Übersicht führt zu einer NullpointerException.		2 [korrigiert: CH 07.08.00]	aktuelle Sourcen vom 03.08.00
26	ContactCreateUserdefField	Ausgangspunkt: CreateUserdefFieldMask „Name“ = null „Wert „ = null -> „OK“	Es wird kein benutzerdef Feld hinzugefügt. Rücksprung zur ContactCreate-Mask mit den vorher eingegebenen Daten.	Ok, allerdings funktioniert anschließend der Button „zurücksetzen“ nicht mehr wie gewünscht, da die zuvor eingegebenen Werte jetzt als Default im Formular vermerkt sind.	?	[lässt sich nicht vermeiden: 06.08.00 CH]	aktuelle Sourcen vom 03.08.00
27	ContactCreateUserdefField	Ausgangspunkt: CreateUserdefFieldMask „Name“ = null „Wert „ = null -> „Abbrechen“	Es wird kein benutzerdef Feld hinzugefügt. Rücksprung zur ContactCreate-Mask.	s.o.	?	[s. 26]	aktuelle Sourcen vom 03.08.00

28	ContactCreateUserdefField	Ausgangspunkt: CreateUserdefFieldMask „Name“ != null „Wert „ != null -> „OK“	Es wird das angegebene benutzerdef Feld hinzugefügt. Rücksprung zur ContactCreateMask.	s.o.	?	[s. 26]	aktuelle Sourcen vom 03.08.00
29	ContactCreateUserdefField	Ausgangspunkt: CreateUserdefFieldMask „Name“ != null „Wert „ != null -> „Abbrechen“	Es wird kein benutzerdef Feld hinzugefügt. Rücksprung zur ContactCreateMask.	s.o.	?	[s. 26]	aktuelle Sourcen vom 03.08.00
30	ContactDeleteUserdefFieldConfirm	Alle Links der NavBar prüfen in der ContactUserdefField-Confirm-Maske	Die Links sollen auf die angegebenen Seiten führen		X		aktuelle Sourcen vom 03.08.00
31	ContactDeleteUserdefFieldConfirm	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Die Maske ContactUserdefFieldConfirm erscheint. Der Link auf Home liefert ein „MissingParameter“ Der Link auf Übersicht liefert eine NullpointerException.		2 [Übersicht - korrigiert: 06.08.00 CH]	aktuelle Sourcen vom 03.08.00
32	ContactDeleteUserdefField	Alle Links der NavBar prüfen in der ContactCreateMask	Die Links sollen auf die angegebene Seite führen		X		aktuelle Sourcen vom 03.08.00
33	ContactDeleteUserdefField	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Die ContactCreateMask erscheint. Der Link auf Home liefert ein „MissingParameter“ Der Link auf Übersicht liefert eine NullpointerException.		[korrigiert: CH 07.08.00]	aktuelle Sourcen vom 03.08.00
34	ContactDeleteConfirm	Alle Links der NavBar prüfen in der ContactDeleteConfirm-Mask	Die Links sollen auf die angegebene Seite führen		X		aktuelle Sourcen vom 03.08.00
35	ContactDeleteConfirm	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Die ContactDeleteConfirmMask erscheint. Die Links auf „Home“ und „Übersicht“ funktionieren nicht.		2 [korrigiert: CH 07.08.00]	aktuelle Sourcen vom 03.08.00

36	ContactDelete	Alle Links der NavBar prüfen in der vom Controller aufgebauten Maske ContactSearch-Mask	Die Links sollen auf die angegebenen Seiten führen		X		aktuelle Sourcen vom 03.08.00
37	ContactDelete	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Egal, ob „Ja“ oder „Nein“ gewählt wird, es wird eine NullpointerException geschmissen und eine leere Seite erscheint.		2 [korrigiert: 06.08.00 CH]	aktuelle Sourcen vom 03.08.00
38	ContactDelete	Der Kontakt wird zwischenzeitlich in Outlook gelöscht. Danach wird versucht, diesen Kontakt aus dem Portal heraus zu löschen.	Evtl. irgendein Hinweis, dass der Kontakt nicht gefunden wurde.	Die ContactSearchMask wird angezeigt und der Kontakt ist weg.		3	aktuelle Sourcen vom 03.08.00
39	ContactDelete	Ausgangspunkt: ContactDeleteConfirmMask Eingabe: „Ja“	Der Kontakt wird gelöscht Die ContactSearchMask wird angezeigt		X		aktuelle Sourcen vom 03.08.00
40	ContactDelete	Ausgangspunkt: ContactDeleteConfirmMask Eingabe: „Nein“	Der Kontakt bleibt bestehen. Die ContactSearchMask wird angezeigt		X		aktuelle Sourcen vom 03.08.00
41	ContactModifyDialog	Alle Links der NavBar prüfen in der ContactModifyMask	Die Links sollen auf die angegebene Seite führen (Home, Übersicht, Erstellen)		X		aktuelle Sourcen vom 03.08.00
42	ContactModifyDialog	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Absturz: NullpointerException		2 [korrigiert : 06.08.00 CH]	aktuelle Sourcen vom 03.08.00
43	ContactModifyDialog	Click auf den Link „Daten aus Partner-DB anzeigen“	Dieser Link sollte nur erscheinen, wenn Der Kontakt vom Typ Kunde ist. Wenn der Kunde existiert, soll er angezeigt werden. Andernfalls soll eine Fehlermeldung erscheinen	(1)Der Link wird immer angezeigt, ungeachtet des Typs des Kontaktes. (2)Existiert der Kunde springt man in das Legacy-Subsystem. (3)Existiert der Kunde nicht gibt es eine Fehlermeldung „Cant Access Legacy“		Beobachtung (1): Prio 2 [korrigiert: 06.08.00 CH] Beobachtung (3): Prio 1 [korrigiert: 04.08.00 CL]	aktuelle Sourcen vom 03.08.00

44	ContactModifyDialog	Kontakt ohne Vornamen erzeugen; Click auf den Link „Daten aus Partner-DB anzeigen“	Fehler: Für die Suche in Legacy werden name, vorname und geburtstag benötigt.	Der Kontakt Otti Mersch wird ohne vornamen trotzdem gefunden?!?	X	2 [korrigiert: 04.08.00 CL]	aktuelle Sourcen vom 03.08.00
45	ContactModifyDialog	Kontakt ohne Geburstagseintrag erzeugen. Click auf den Link „Daten aus Partner-DB anzeigen“	Fehler: Für die Suche in Legacy werden name, vorname und geburtstag benötigt.	CirticalErrorMask: Legacy_Wrong_Birthday_Format		2	aktuelle Sourcen vom 03.08.00
46	ContactModifyDialog	Click auf den Link „Löschen“	Man gelangt zu einer Sicherheitsabfrage, ob der Kontakt wirklich gelöscht werden soll.	(Dieser Pfad wurde weiter oben schon getestet, daher wird hier nur der Link geprüft)	X		aktuelle Sourcen vom 03.08.00
47	ContactModifyDialog	Click auf den Link „benutzerdefiniertes Feld zufügen“	Man gelangt zu der oben bereits getesteten Maske ContactCreateUserdefFieldDialog		X		aktuelle Sourcen vom 03.08.00
48	ContactModifyDialog	Click auf den Link „benutzerdefiniertes Feld löschen“	Man gelangt zu der oben bereits getesteten Maske DeleteUserdefFieldConfirm		X		aktuelle Sourcen vom 03.08.00
49	ContactModify	Alle Links der NavBar prüfen in den Masken ContactSearchMask (korrekte Eingaben) und ContactModifyDialogMask (Fehlerhafte Eingaben)	Die Links sollen auf die angegebenen Seiten führen		X		aktuelle Sourcen vom 03.08.00
50	ContactModify	Aufruf des Controllers, ohne dass ein User in der Session eingetragen ist.	Die Fehlermaske „User not logged in“ soll erscheinen.	Absturz durch eine NullpointerException		2 [korrigiert: 06.08.00 CH]	aktuelle Sourcen vom 03.08.00
51	ContactModify	Werte in allen Feldern eintragen und click auf „ändern“	Der Kontakt soll mit allen Werten gespeichert werden.		X		aktuelle Sourcen vom 03.08.00
52	ContactModify	Kontakt erstellen mit Werten in allen Feldern und 2 benutzerdefinierten Feldern. Danach diesen Kontakt ändern und alle Werte in den Feldern löschen (bis auf Name), und	Der Name bleibt bestehen, alle anderen Werte werden gelöscht.	Die meisten Werte werden auch gelöscht, nur der Geburtstag und die benutzerdefinierten Felder kann man nicht löschen!		1 [korrigiert: 07.08.00 CH]	aktuelle Sourcen vom 03.08.00

		die benutzerdefinierten Felder löschen. Click auf „ändern“.					
52	ContactModify	name = null	Fehlermeldung „name is required“ in der ContactModifyMask und rote Markierung der Zeile		X		aktuelle Sourcen vom 03.08.00
53	ContactModify	name != null, geburtstag ungültig	Fehlermeldung „falsches Datumsformat“ in der ContactModify-Mask und rote Markierung der Zeile		X		aktuelle Sourcen vom 03.08.00

9.1.8 Systemtest Subsystem Office (Email) Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	office	Testteilnehmer:	Leifkes
Testzeitraum: (Datum)	04.08.00		

Lfd.Nr	betroffene Action	Eingabe / Vorgehen beim Testen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung / Kommentar	Ok?	Priorität 1-3 1 = hoch/krit.	Testgrundlage (Build)
1	EmailDisplay	Ausgangspunkt:StartMaske Click auf eine neue Email.	Die Email öffnet sich im Outlook-Fenster. Die Inbox wird im Portal angezeigt. Die Email wird als gelesen markiert.		X		aktuelle Sourcen vom 03.08.00
2	EmailDisplay	Ausgangspunkt: StartMaske Neue Emails werden angezeigt. In Outlook wird eine Email gelöscht und anschließend versucht, diese im Portal anzuzeigen.	Fehlermeldung: Kann Email nicht finden	In Outlook muss eine Email nicht nur gelöscht, sondern auch noch aus dem Papierkorb entfernt werden. Liegt die Email nur im Papierkorb, wird diese trotzdem noch gefunden	X		aktuelle Sourcen vom 03.08.00
3	EmailDisplay	Ausgangspunkt: InboxÜbersichtMaske Click auf eine Email.	Ein Outlook-Fenster soll sich mit der Email öffnen und die Maske aktualisieren.		X		aktuelle Sourcen vom 03.08.00
4	EmailSearch	Ausgangspunkt: StartMaske Click auf „Nachrichten“	Der Inhalt der Inbox wird angezeigt. Nicht gelesene Emails werden fett dargestellt.		X		aktuelle Sourcen vom 03.08.00

5	EmailSearch	Ausgangspunkt: Suchmaske Suche nach Betreff = „*“	Alle Emails werden gefunden und in der InboxÜbersichtMaske dargestellt.		X		aktuelle Sourcen vom 03.08.00
6	EmailSearch	Ausgangspunkt Suchmaske. Suche nach Betreff = „Re:*“ In der Ergebnismaske geht man einen Schritt zurück und drückt auf Reload im Browser.	Da keine Emails mit „Re:*“ existieren, wird eine leere Übersicht angezeigt. Nachdem man im Browser ein Schritt zurück gegangen ist und reload-drückt, wird dieselbe suchanfrage nochmal gestellt, obwohl man wieder in der Inbox-Übersicht ist.	In der NavBar fehlt ein Link, um die gesamte Emailübersicht mit allen E-mails zu erreichen. In der Inbox-ÜbersichtsMaske sollte erkennbar sein, dass grade das Ergebnis zur Suchanfrage „XYZ“ dargestellt wird.		1 [korrigiert: 07.07.00 CH (Übersicht zeigt jetzt immer alle Emails an; nach Suchkriterieneingabe erscheint Suchergebnisliste)]	aktuelle Sourcen vom 03.08.00
7	EmailSearch	Ausgangspunkt Suchmaske. Suche nach Betreff mit verschiedenen Suchanfragen	Die richtige Ergebnismenge wird angezeigt		X		aktuelle Sourcen vom 03.08.00
8	EmailSearch	Ausgangspunkt Suchmaske. Suche nach „erhalten von: [=]“ mit verschiedenen Suchanfragen	Die richtige Ergebnismenge wird angezeigt		X		aktuelle Sourcen vom 03.08.00
9	EmailSearch	Ausgangspunkt Suchmaske. Suche nach „erhalten von: [=]“ mit ungültigem Datum	Erneut die Suchmaske, jetzt allerdings mit Fehlertext und rot markierter Zeile. Die übrigen Sucheingaben sollen noch da sein.		X		aktuelle Sourcen vom 03.08.00
10	EmailSearch	Ausgangspunkt Suchmaske. Suche nach „erhalten von: [=]“ mit verschiedenen Suchanfragen	Die richtige Ergebnismenge wird angezeigt	Es werden keine Emails gefunden.		1 [korrigiert: 07.08.00 CH (muss heissen „erhalten am“ statt „erhalten von“)]	aktuelle Sourcen vom 03.08.00
11	EmailSearch	Ausgangspunkt Suchmaske. Suche nach „erhalten von: [>=]“ mit ungültigem Datum	Erneut die Suchmaske, jetzt allerdings mit Fehlertext und rot markierter Zeile. Die übrigen Sucheingaben sollen noch da sein.		X		aktuelle Sourcen vom 03.08.00

12	EmailSearch	Ausgangspunkt Suchmaske. Suche nach „erhalten von: [>=]“ mit verschiedenen Suchanfragen	Die richtige Ergebnismenge wird angezeigt		X		aktuelle Sourcen vom 03.08.00
13	EmailSearch	Ausgangspunkt Suchmaske. Suche nach „erhalten von: [<=]“ mit ungültigem Datum	Erneut die Suchmaske, jetzt allerdings mit Fehlertext und rot markierter Zeile. Die übrigen Sucheingaben sollen noch da sein.		X		aktuelle Sourcen vom 03.08.00
14	EmailSearch	Ausgangspunkt Suchmaske. Suche nach „erhalten von: [<=]“ mit verschiedenen Suchanfragen	Die richtige Ergebnismenge wird angezeigt	Dieses Kriterium funktionier als „kleiner“, nicht als „kleiner gleich“!		1 [korrigiert: 07.08.00 CH (muss heissen „erhalten am“ statt „erhalten von“)]	aktuelle Sourcen vom 03.08.00
15	EmailSearch-Dialog	Testen der Suchmaske Werte in alle Felder eintragen und auf Zurücksetzten drücken.	Alle Felder werden wieder gelöscht		X		aktuelle Sourcen vom 03.08.00

9.1.9 Systemtest Subsystem Office (Task) Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	Office - Task	Testteilnehmer:	Stefan Göbel
Testzeitraum: (Datum)	03.8.2000 bis 04.8.2000		

Lfd.Nr	betroffene Action	Eingabe / Vorgehen beim Testen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung / Kommentar	Ok?	Priorität 1-3 1 = hoch/krit.	Testgrundlage (Build)
1	TaskList	Alle Links der NavBar überprüfen (Home, Übersicht, Erstellen, Suchen)	Home: IPSI-Today-Seite Übersicht: TaskList mit allen Tasks Erstellen: leere Task-Erstellen-Maske Suchen: Suchmaske	Home: ok Übersicht: ok Erstellen: ok Suchen: ok	X		Head1 + Klassen vom 4.8.2000
2	TaskList -> TaskModifyMask	Name-Link überprüfen	TaskModifyMask mit allen Daten des Tasks		X		

3	TaskList -> TaskModi-fyMask	Ändern-Link überprüfen	TaskModifyMask mit allen Daten des Tasks		X		
4	TaskList -> TaskDelete-Mask	Löschen-Link überprüfen	Sicherheitsabfrage (ja/nein)		X		
5	TaskModi-fyMask	Alle Links der NavBar überprü-fen (Home, Übersicht, Erstellen, Suchen)	Home: IPSI-Today-Seite Übersicht: TaskList mit allen Tasks Erstellen: leere Task-Erstellen-Maske Suchen: Suchmaske	Home: ok Übersicht: ok Erstellen: ok Suchen: ok	X		
6	TaskModi-fyMask -> TaskModify	Ungültiges Datum in „beginnt am“, dann „ändern“	Fehlermeldung mit richtigem Da-tumsformat, Feld „beginnt am“ ist rot markiert		X		
7	TaskModi-fyMask -> TaskModify	Fehlendes Datum in „beginnt am“, dann „ändern“	Übersichtsmaske erscheint, Datum wurde automatisch gesetzt		X		
8	TaskModi-fyMask -> TaskModify	Ungültiges Datum in „fällig am“, dann „ändern“	Fehlermeldung mit richtigem Da-tumsformat, Feld „fällig am“ ist rot markiert		X		
9	TaskModi-fyMask -> TaskModify	Fehlendes Datum in „fällig am“, dann „ändern“	Übersichtsmaske erscheint, Datum wurde automatisch gesetzt		X		
10	TaskModi-fyMask -> TaskModify	Ungültiges Datum in „fällig am“ und in „beginnt am“, dann „än-dern“	Fehlermeldung und beide Felder sind rot markiert	Nur Feld „fällig am“ wurde markiert, das andere Feld ist unverändert SP: Es wird immer nur ein Feld rot markiert. Dies ist gewollt und kann nicht mehr korrigiert werden!	X	3	
11	TaskModi-fyMask -> TaskModify	Ungültiges Datum in „Erinne-rung am“, dann „ändern“	Fehlermeldung mit richtigem Da-tumsformat, Feld „Erinnerung am“ ist rot markiert		X		

12	TaskModifyMask -> TaskModify	Fehlendes Datum in „Erinnerung am, dann „ändern“	Übersichtsmaske erscheint, Datum wurde automatisch gesetzt		X		
13	TaskModifyMask -> TaskModify	Nichts geändert, nur auf „ändern“ geklickt	Task bleibt unverändert, Übersichtsmaske erscheint		X		
14	TaskModifyMask -> TaskModify	Alle Felder geändert, dann auf „ändern“	Task wurde überall geändert, Übersichtsmaske erscheint		X		
15	TaskModifyMask –> TaskModify	Alle Felder korrekt gefüllt, aber Enddatum vor Startdatum	Fehlermeldung in diese Maske, die auf den Mißstand aufmerksam macht	CriticalErrorMask erscheint, Fehlermeldung: „OfficeSubsystemError“ SP: Fehler korrigiert! Es wird jetzt eine ungültige Eingabe angezeigt.	X	1	
16	TaskModifyMask	Einige Felder geändert, klicke dann auf „Zurücksetzen“	Alter Zustand wiederhergestellt		X		
17	TaskInsertMask	Einige Felder ausgefüllt, klicke auf „zurücksetzen“	Alle Felder wieder leer		X		
18	TaskInsertMask –> TaskInsert	Alle Felder freigelassen, dann auf „erstellen“	Fehlermeldung, alle benötigten Felder markiert	Feld „Titel“ wurde markiert	X		
19	TaskInsertMask –> TaskInsert	Alle Felder gefüllt, aber ungültiges Datum in „Beginnt am“	Fehlermeldung mit richtigem Datumsformat, Feld „beginnt am“ ist rot markiert		X		
20	TaskInsertMask –> TaskInsert	Alle Felder gefüllt, aber ungültiges Datum in „Fällig am“	Fehlermeldung mit richtigem Datumsformat, Feld „fällig am“ ist rot markiert		X		
21	TaskInsertMask –> TaskInsert	Alle Felder korrekt gefüllt	Task wurde eingefügt, Übersichtsmaske erscheint		X		

22	TaskInsert-Mask –> TaskInsert	Alle Felder korrekt gefüllt, „Erinnerung am“ aber freigelassen	Task wurde eingefügt, Übersichtsmaske erscheint	Feld „Erinnerung am“ wurde automatisch gefüllt SP: Aus technischen Gründen werden alle Datumsfelder per default mit dem aktuellen Datum versehen.	X		
23	TaskInsert-Mask –> TaskInsert	Alle Felder korrekt gefüllt, aber Enddatum vor Startdatum	Fehlermeldung in diese Maske, die auf den Mißstand aufmerksam macht	CriticalErrorMask erscheint, Fehlermeldung: „OfficeSubsystemError“ SP: Fehler korrigiert! Es wird jetzt eine ungültige Eingabe angezeigt.	X	1	
24	TaskDelete-Mask -> TaskDelete	Klicke auf „nein“	Übersichtsmaske erscheint, Task wurde nicht gelöscht		X		
25	TaskDelete-Mask -> TaskDelete	Klicke auf „ja“	Übersichtsmaske erscheint, Task wurde gelöscht		X		
26	TaskModifyMask -> TaskDelete	Klicke auf „löschen“	Sicherheitsabfrage erscheint		X		
27	TaskList	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		
28	TaskInsert-Mask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		
29	TaskModifyMask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		
30	TaskSearch-Mask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		
31	TaskDelete-Mask	Direkter Zugriff ohne eingeloggten User	Fehlermeldung UserNotLoggedIn		X		

32	TaskSearch-Mask –> TaskSearch	Ungültige Eingabe in „Beginnt am“	Fehlermeldung und richtiges Datumsformat		X		
33	TaskSearch-Mask -> TaskSearch	Ungültige Eingabe in „fällig am“	Fehlermeldung und richtiges Datumsformat		X		
34	Task-SearcgMask -> TaskSearch	Ungültige Eingabe in „erinnern am“	Fehlermeldung und richtiges Datumsformat		X		
35	TaskSearch-Mask -> TaskSearch	Feld „Titel“ ausgefüllt, ungültige Eingabe in „Beginnt am“	Fehlermeldung und richtiges Datumsformat, Feld „Titel“ noch gefüllt	Fehlermeldung ist da, Feld „Titel“ leer SP: Bei der Suchmaske werden die Werte im Gegensatz zur Insert- oder Modify-Maske nicht übernommen.	X	3	
36	TaskSearch-Mask	Joker (*) im Feld „Titel“ benutzt	Alle Tasks werden gefunden		X		
37	TaskSearch-Mask	Alle Felder bleiben leer	Alle Tasks werden gefunden		X		

9.1.10 Systemtest Subsystem Administration Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	Administration und Search	Testteilnehmer:	Chris, Traugott
Testzeitraum: (Datum)	31.07.2000, Ende offen		

Lfd. Nr	betroffene Action	Eingabe / Vorgehen beim Testen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung / Kommentar	Ok?	Priorität 1-3 1 = hoch/krit.	Testgrundlage (Build)
AL1	Als AL admin.modifyuser	Jedes Attribut, das man ausfüllen **muss**, einmal weglassen.	Fehlermeldung oben, Attributbezeichnung in rot		X		Sieben
AL 2	Als AL admin.modifyuser	Negative Budgetgrenze	Fehlermeldung, die dem Nutzer das Problem beschreibt.	Interne Fehlermeldung		3 SG: Fehler korrigiert	Sieben

AL 3	Als AL admin.modifyuser	Office-/Shop-/CMS-ID auf die ID eines anderen Nutzers ändern	Ablehnung dieser Änderung, Erklärung für den Nutzer	Änderung akzeptiert		1 SG: Fehler korrigiert	Sieben
AL 4	Als AL admin.modifyuser	User-ID auf die ID eines anderen Nutzers ändern	Ablehnung dieser Änderung, Erklärung für den Nutzer	Änderung zwar verforfen, aber: MODIFY_ERROR Benutzer wurde wahrscheinlich nicht geändert!		2 SG: Fehler korrigiert	Sieben
AL 5	Als AL admin.modifyuser	User-ID ändern	Änderung passiert		X		Sieben
AL 6	Als AL Admin.createnewuser	Jedes Attribut, das man ausfüllen **muss**, einmal weglassen.	Fehlermeldung oben, Attribut-bezeichnung in rot		X		Sieben
AL 7	Als AL Admin.createnewuser	Normalen Nutzern mit allen Attributen hinzu-fügen	Neuer Benutzer in Übersicht		X		Sieben
AL 8	Als AL Admin.deleteuser, ad-min.confirmdeleteuser	Nutzer löschen, dann aber doch nicht mehr wollen	User bleibt		X		Sieben
AL 9	Als AL Admin.deleteuser, ad-min.confirmdeleteuser, Admin.createnewuser	Nutzer löschen und es auch tun, ihn danach mit gleichem login sofort wieder anlegen	• User wird gelöscht • er wird wieder angezeigt		X		Sieben
AL 10	Als AL Admin.deleteuser, ad-min.confirmdeleteuser, admin.modifyuser	Nutzer löschen, per Back-Button ein paar Mal zurück und versu-chen ihn zu verändern	Fehler: Nutzer existiert nicht.		X		Sieben
AL 11	Als AL Admin.logout	Sich abmelden, versu-chen wieder drauf zu-zugreifen	Erneuter Zugriff schlägt fehlt	Offensichtlich bin ich nicht abgemeldet		1 SG: Konnte nicht nachvollzogen wer-den	Sieben
AL 12	Als AL Emptyuserdetails	Neuen Benutzer anle-gen, verschiedene Rol-len, verschiedene Agen-turen	Sollte nur mit eigener Agentur mit VADs überhaupt machbar sein (schon von der GUI aus)		X		Neun

AL 13	Als AL Userdetail	Jemand zum VU oder AL befördern.	Nicht möglich durch GUI.		X		Neun
AL 14	Als AL Userdetail	Sich selber befördern.	Nicht möglich durch GUI.		X		Neun
AL 15	Als AL Userdetail	Jemand in eine andere Agentur versetzen	Nicht möglich durch GUI.		X		Neun
AL 16	Als AL Userdetail	Versuchen, Information über VU oder VADs anderes Agenturen zu sehen	Nicht möglich durch GUI.		X		Neun
L1	login	Eingabe eines falschen Passworts	Fehlermeldung, dass Passwort falsch	richtige Fehlermeldung, aber kein Link zu erneutem Login		3 SG: Fehler korrigiert	Sourcen vom 07.08.00
L2	login	Eingabe eines falschen Benutzernamens	Fehlermeldung, dass Benutzer nicht existiert	richtige Fehlermeldung, aber kein Link zu erneutem Login		3 SG: Fehler korrigiert	Sourcen vom 07.08.00
L3	login	Eingabe eines richtigen Benutzernamens und Passwort	Einloggen ins Portal		x		
L4	Login	Überprüfen der Links persönliche Nachrichten	Outlook-Fenster wird geöffnet, in dem die persönliche Nachricht steht; Verzweigung in Email-Übersicht		X		
L5	Login	Überprüfen, ob die Termine der nächsten Woche angeziegt werden (max. 5)	Anzeige der Termine der nächsten Woche		X		
L6	Login	Überprüfen der Links auf aktuelle Termine	Detailansicht des ausgewählten Termins		X		

L7	Login	Überprüfen, ob noch nicht erledigte Aufgaben angezeigt werden (max. 5)	Anzeige der noch nicht erledigten Aufgaben	es werden keine Aufgaben angezeigt (weder überfällige, noch Aufgaben die in der aktuellen Woche zu erledigen sind)		1 SG: Fehler korrigiert	
L8	Login	Überprüfen, ob präferierte Shop-Produkte angezeigt werden	Anzeige der präferierten Shop-Produkte	nicht testbar, da Shop nicht verfügbar		SG: ist ok	
L8	Login	Überprüfen, der Links auf Shop-Produkte	Detailansicht des Shop-Produkts	nicht testbar, da Shop nicht verfügbar		SG: ist ok	
L8	Login	Überprüfen des Links auf Nachrichten	Anzeige der Liste aller Emails des Inbox-Folders		X		
L9	Login	Überprüfen des Links auf Online-Shop	Verzweigung zu Shop	nicht testbar, da Shop nicht verfügbar		SG: ist ok	
L10	Login	Überprüfen des Links auf Adressbuch	Verzweigung zu Kontaktübersicht		X		
L11	Login	Überprüfen des Links auf Partner-DB	Verzweigung zu Partner-DB		X		
L12	Login	Überprüfen des Links auf Informationssystem	Verzweigung zu CMS	nicht testbar, da nur im Pool aufrufbar		SG: ist ok	
L13	Login	Überprüfen des Links auf Termine	Verzweigung zur Terminübersicht		X		
L14	Login	Überprüfen des Links Aufgaben	Verzweigung zur Aufgabenübersicht		X		

L15	Login	Überprüfen des Links Suche	Verzweigung zur Expertensuche		X		
L16	Login	Überprüfen des Links Logout	Ausloggen aus dem IPSI-Portal; Ausloggen aus allen Subsystemen	1) beim Ausloggen aus dem Portal und bei anschließendem Anklicken des Home-Buttons erscheint die Fehlermeldung MISSING_PARAMETER; besser: Fehlermeldung, dass Benutzer nicht eingeloggt 2) bei Verwendung des Links „Neu anmelden“ in der Nav.Bar wird nicht der richtige Dispatcher verwendet		3 SG: Fehler korrigiert	
L17	Login	Abmelden vom Portal, wenn Benutzer-Objekt nicht mehr in der Session ist	Fehlermeldung, dass Benutzer nicht mehr eingeloggt	Fehlermeldung MISSING_PARAMETER wird angezeigt; besser: Fehlermeldung Benutzer ist nicht eingeloggt		3 SG: Fehler korrigiert	
VU1	Als VU Usermanagement	Als VU einloggen, eine Agentur auswählen und anzeigen lassen	Liste mit Namen, Adressen, ..		X		Neun
VU2	Als VU Emptyuserdetails	Neuen Benutzer anlegen (VAD), als dieser einloggen und Rechte überprüfen	Neuer Benutzer ist in der gewählten Agentur und sein Handlungsspielraum ist wie gewünscht.	Neuer Benutzer ist in der gewählten Agentur, aber ich kann mich nicht als er einloggen.		1 SG: Das kommt davon, wenn man die UserID nicht korrekt eingibt, also: kein Fehler!	Zehn
VU3	Als VU Emptyuserdetails	Neuen Benutzer anlegen (AL), als dieser einloggen und Rechte überprüfen	Neuer Benutzer ist in der gewählten Agentur und sein Handlungsspielraum ist wie gewünscht.		X		Neun
VU4	Als VU Emptyuserdetails	Neuen Benutzer anlegen (VU), als dieser einloggen und Rechte überprüfen	Neuer Benutzer ist in der gewählten Agentur und sein Handlungsspielraum ist wie gewünscht.		X		Zehn

VU5	Als VU Userdatail	Agentur eines Nutzers ändern	Nutzer in anderer Agentur	White-screen, null-pointer exception in modifyusercontroller		1 SG: Fehler, weil Shop nicht verfügbar war; es wird jetzt abgefangen	Neun
VU6	Als VU Userdetail	Rolle ändern VAD – AL, AL – VU, VAD – VU	Wird gemacht	White-screen, null-pointer exception in modifyusercontroller		1 SG: Fehler, weil Shop nicht verfügbar war; es wird jetzt abgefangen	Neun
VU7	Als VU Userdetail	Eingene Rolle ändern VU – VAD	Sollte nicht erlaubt sein. Fehlermeldung.	White-screen, null-pointer exception in modifyusercontroller		1 SG: Fehler, weil Shop nicht verfügbar war; es wird jetzt abgefangen	Neun
VU8	Als VU Agencymanagement	Auf Agenturverwaltung klicken	Eine Liste mit Agenturen und Operationen darauf.		X		Neun
VU9	Als VU Emptyagencydetail	Neue Agentur anlegen, jedes Attribut einmal weglassen.	Die Attribute, die eingegeben werden **müssen** werden gefordert: Fehlermeldung.		X		Neun
VU10	Als VU deleteagencymask	Eine Agentur weglöschen.	Sie ist weg.	White-Screen		1 SG: Fehler korrigiert	Neun
VU11	Als VU Deleteuser	Sich selber löschen	Sollte nicht gehen. Fehlermeldung.	(dazu möchte ich erstmal einen neuen User haben, als der ich mich dann löschen kann)	?	SG: Fehler korrigiert	Neun
I1	Als VAD getAllInterests	Die eigenen Interessen ansehen	Liste mit möglichen Interessen, tatsächliche sind angekreuzt		X		Zehn
I2	Als VAD ModifyUserInterest	Interessen ändern, anschliessend auf der Homepage Artikelübersicht ansehen	Bestätigung über geänderte Interessen, auf der Homepage zu mir passende Werbung		X		Zehn

9.1.11 Systemtest Subsystem Administration (Statistics) Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	Statistik	Testteilnehmer:	Wahid Bashirazad

Testzeitraum: (Datum)	08.07.2000						
Lfd.Nr	**betroffene Action**	**Eingabe / Vorgehen beim Testen**	**erwartete Ausgabe/Verhalten**	**tatsächliche Ausgabe/Beobachtung / Kommentar**	**Ok?**	**Priorität 1-3 1 = hoch/krit.**	**Testgrundlage (Build)**
1	feedbackmask	Anfrage nach Feedbacks	Eingabemöglichkeit zur Bestimmung des Zeitraums	Maske für die Eingabe von Startdatum und Enddatum	X		Build 8
2	feedbackmask	Eingabe von gültigen Start- und Enddatum	Liste der eingegangenen Feedbacks in in eingegebenen Zeitraum	Liste der Feedbacks, jeweils angegeben durch Datum, Bemerkung und Note.	X		Build 8
3	feedbackmask	Startdatum ist leer, Enddatum gültig	Fehlermeldung	Fehlermeldung über ungültiges Datumsformat	X		Build 8
4	feedbackmask	Startdatum ist gültig, Enddatum leer	Fehlermeldung	Fehlermeldung über ungültiges Datumsformat	X		Build 8
5	feedbackmask	Start- oder Enddatum haben kein gültiges Format	Fehlermeldung	Fehlermeldung über ungültiges Datumsformat	X		Build 8
6	feedbackmask	Die Reihenfolge von Start und Enddatum stimmt nicht.	Fehlermeldung oder Reihenfolge korrigieren und die Feedbacks ausgeben.	Fehlermeldung über die falsche Reihenfolge.	X		Build 8
7	feedbackmask	Das Jahr ist in Start- oder Enddatum zweistellig eingegeben	Fehlermeldung oder automatische Korrektur.	Keine Fehlermeldung oder automatische Korrektur		3	Build 8
8	feedbackmask	In dem eingegebenen Zeitraum sind keine Feedbacks eingetragen.	Entsprechende Meldung über fehlende Feedbacks.	Meldung über fehlende Feedbacks.	X		Build 8
9	CommunicationLogfiles-Mask	Link auf Kommunikationslogfiles	Eingabemöglichkeit zur Bestimmung von gewünschten Logfileinformationen	Maske für die Eingabe von Medium, Status und UserID	X		Build 8
10	CommunicationLogfiles	Abfrage der Emails von allen Benutzern	Liste der Emails von allen Benutzer	Liste aller Emails mit Zeitpunkt**,** Medium, Status, UserId und Beschreibung	X		Build 8

11	CommunicationLogfiles	Abfrage der Emails von einem Benutzer	Liste der Emails von dem gewählten Benutzer	Liste aller Emails der Benutzers mit Zeitpunkt**,** Medium, Status, UserID und Beschreibung	X		Build 8
12	CommunicationLogfiles	Abfrage der Emails von einem Benutzer, der keine Emails versendet hat.	Entsprechende Meldung über fehlende Einträge in Logfile.	Meldung über fehlende Logfileeinträge.	X		Build 8
13	CommunicationLogfiles	Abfrage der Emails. Keine Eingabe im Feld „UserID“	Liste der Emails von allen Benutzern	Liste aller Emails mit Zeitpunkt**,** Medium, Status, UserID und Beschreibung	X		Build 8
14	CommunicationLogfiles	Abfrage der Emails. Eintrag eines nichtexistierenden Benutzer im Feld „UserID“	Fehlermeldung	Fehlermeldung über den nichtexistierenden Benutzer.	X		Build 8
15	getDateForHistory	Link auf Abfrage der Bestellhistorie	Eingabemöglichkeit für die Bestimmung von Historienzeitraum	Eingabemaske mit Möglichkeit der Auswahl von vorgegebenen Zeiträumen und freie Zeitraumeingabe	X		Build 10
16	getItemGroupHistory	Auswahl eines vorgegebenen Zeitraums	Liste der Warengruppe mit Summe der Bestellungen in jeweiligen Warengruppe	Liste der Warengruppe mit Summe der Bestellungen in jeweiliger Warengruppe	X		Build 10
17	getItemGroupHistory	Eingabe eines gültigen Zeitraums	Liste der Warengruppe mit Summe der Bestellungen in jeweiligen Warengruppe	Liste der Warengruppe mit Summe der Bestellungen in jeweiliger Warengruppe	X		Build 10
18	getItemGroupHistory	Eingabe eines ungültigen Startdatums	Fehlermeldung	Fehlermeldung über das fehlerhafte Startdatum	X		Build 10
19	getItemGroupHistory	Eingabe eines ungültigen Enddatums	Fehlermeldung	Fehlermeldung über das fehlerhafte Enddatum	X		Build 10
20	getItemGroupHistory	Startdatum oder Enddatum ist leer.	Fehlermeldung	Fehlermeldung über das fehlerhafte Startdatum	X		Build 10

21	getItem-GroupHistory	Startdatum und Enddatum sind vertauscht.	Die Reihenfolge korrigieren und Liste der Warengruppe mit Summe der Bestellungen in jeweiliger Warengruppe ausgeben.	Liste der Warengruppen mit Summe der Bestellungen in jeweiliger Warengruppe in korrigierten Datumsreihenfolge	X		Build 10
22	getOrdered-Items	Auswahl einer Warengruppe	Liste der einzelnen Artikel, die von der gewählten Warengruppe bestellt worden sind.	Liste der einzelnen Artikel, die von der gewählten Warengruppe bestellt worden sind.	X		Build 10
23	getBudgetAccount	Link auf Budgetkonto	Ausgabe von Budgetgrenze und Budgetzeitraum mit Editiermöglichkeit	Formular mit enthaltenen Informationen bzgl. Budgetgrenze, Budgetzeitraum und Summe der bisherigen Bestellungen.	X		Build 10
24	ModifyOrder-Limit	Änderung der Budgetgrenze mit einem gültigen Wert	Bestätigung der Budgetänderung	Bestätigung und Ausgabe der neuen Budgetdaten	X		Build 10
25	ModifyOrder-Limit	Eingabe eines ungültigen Wertes als Budgetgrenze	Fehlermeldung	Fehlermeldung mit Hinweis auf fehlerhafte Eingabe	X		Build 10
26	ModifyOrder-Limit	Budgetgrenze ist leer	Fehlermeldung	Fehlermeldung mit Hinweis auf fehlerhafte Eingabe	X		Build 10

9.1.12 Systemtest Subsystem Legacy Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	Legacy	Testteilnehmer:	Hassan Ghane
Testzeitraum: (Datum)	06.08.2000		

Lfd.Nr	Betroffene Action	Eingabe / Vorgehen beim Testen	erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung / Kommentar	Ok?	Priorität 1-3 1 = hoch/krit.	Testgrundlage (Build)
1	QueryByTimePeriodMask	Statistik über Neuzugänge in einem bestimmten Zeitraum	Zeitraumabfragemaske	Zeitraumabfragemaske	X		Build8

2	QueryByTimePeriod	Zeitraumeingabe: gültige Start- und Endzeitpunkt (01.01.1990-31.12.2000)	Auflisten der Statistik über Neuzugänge im angegebenen Zeitraum.	Anzahl der Neukunden und der Neuverträge im angegebenen Zeitraum.	X		Build8
3	QueryByTimePeriod	Zeitraumeingabe: gültiger Start- und Endzeitpunkt in vertauschte Reihenfolge (31.12.2000-01.01.1990)	Die Reihenfolge des Datums richtig stellen.	Bei der Ausgabe der Statistik wird die richtige Reihenfolge des Datums berücksichtigt. In der ausgegebenen Maske wird aber das Datum wie eingegeben dargestellt.	X	korrigiert am 09.08.00	Build8
4	QueryByTimePeriod	Zeitraumeingabe: Startzeitpunkt gleich Null	Fehlermeldung über den fehlenden Startzeitpunkt.	Fehlermeldung: Das Format des Datums im Feld „Startzeitpunkt“ ist falsch. Richtiges Format tt.MM.jjjj	X		Build8
5	QueryByTimePeriod	Zeitraumeingabe: Endzeitpunkt gleich Null	Fehlermeldung über den fehlenden Endzeitpunkt.	Fehlermeldung: Das Format des Datums im Feld „Endzeitpunkt“ ist falsch. Richtiges Format tt.MM.jjjj	X		Build8
6	QueryByTimePeriod	Zeitraumeingabe: Startzeitpunkt hat kein richtiges Format.	Fehlermeldung über das falsche Format.	Fehlermeldung: „Das Format des Datums im Feld „Startzeitpunkt“ ist falsch. Richtiges Format tt.MM.jjjj“	X		Build8
7	QueryByContractCategory	Statistik über Kunden und Verträge nach Versicherungssparten sortiert	Statistik nach Versicherungssparten	Tabellarische Ausgabe der Statistik über Vertragssparten, Anzahl der Kunden und Anzahl der Verträge.	X		Build8
8	QueryByPostalCodeMask	Statistik über Kunden und Verträge nach Postleitzahlenbereichen sortiert.	Die Eingabemaske für die einzugebenden Postleitzahlen.	Maske mit Eingabemöglichkeiten: Untere PLZ-Grenze: Obere PLZ-Grenze: Anzahl relevanter Stellen:	X		Build8
9	QueryByPostalCode	Gültige Eingabe: Untere PLZ-Grenze:00000 Obere PLZ-Grenze:99999 Anzahl relevanter Stellen:5	Statistik über Kunden und Verträge in allen Postleitzahlenbereichen sortiert ausgeben.	Tabellarische Ausgabe der Statistik über die Anzahl der Kunden und Verträge sortiert nach allen Postleitzahlen.	X		Build8
10	QueryByPostalCode	Gültige Eingabe: Untere PLZ-Grenze:50000 Obere PLZ-Grenze:99999 Anzahl relevanter Stellen:5	Statistik über Kunden und Verträge in den eingegebenen Postleitzahlenbereichen sortiert ausgeben.	Tabellarische Ausgabe der Statistik über die Anzahl der Kunden und Verträge sortiert nach den eingegebenen Postleitzahlen.	X		Build8

11	QueryByPostalCode	Gültige Eingabe: Untere PLZ-Grenze:50000 Obere PLZ-Grenze:99999 Anzahl relevanter Stellen:2	Statistik über Kunden und Verträge mit Berücksichtigung der Anzahl relevanter Stellen in den eingegebenen Postleitzahlenbereichen sortiert ausgeben.	Tabellarische Ausgabe der Statistik über die Anzahl der Kunden und Verträge sortiert nach Anzahl der relevanten Stellen in den eingegebenen Postleitzahlen	X		Build8
12	QueryByPostalCode	Untere PLZ-Grenze:leer Gültige Eingabe bei Obere PLZ-Grenze und Anzahl relevanter Stellen	Fehlermeldung über die fehlende Eingabe in Untere PLZ-Grenze.	Fehlermeldung: „die untere PLZ-Grenze muss 5-stellig sein.“	X		Build8
13	QueryByPostalCode	Obere PLZ-Grenze: leer Gültige Eingabe bei Untere PLZ-Grenze und Anzahl relevanter Stellen	Fehlermeldung über die fehlende Eingabe in Obere PLZ-Grenze.	Fehlermeldung: „die obere PLZ-Grenze muss 5-stellig sein.“	X		Build8
14	QueryByPostalCode	Anzahl relevanter Stellen: leer Gültige Eingabe bei Obere PLZ-Grenze und Untere PLZ-Grenze.	Fehlermeldung über die fehlende Eingabe in Anzahl relevanter Stellen.	Fehlermeldung: „Das Feld „ Anzahl relevanter Stellen“ enthält keine gültige Zahl.“	X		Build8
15	QueryByPostalCode	„Untere PLZ-Grenze“ und „Obere PLZ-Grenze“ werden in falscher Reihenfolge eingegeben.	Fehlermeldung über die falsche Reihenfolge.	Fehlermeldung: „Die Reihenfolge der PLZ ist vertauscht“	X		Build8
16	QueryByPostalCode	Negative Eingabe bei „Untere PLZ-Grenze.“	Fehlermeldung über die negative Eingabe bei Untere PLZ-Grenze..	Fehlermeldung:“ Die untere PLZ-Grenze liegt nicht im Bereich von 00000 bis 99999.Bitte korrigieren.“	X		Build8
17	QueryByPostalCode	Negative Eingabe bei „Obere PLZ-Grenze“.	Fehlermeldung über die negative Eingabe bei Obere PLZ-Grenze..	Fehlermeldung: „Die obere PLZ-Grenze liegt nicht im Bereich von 00000 bis 99999.Bitte korrigieren.“	X		Build8
18	QueryByPostalCode	Negative Eingabe bei „Anzahl relevanter Stellen“.	Fehlermeldung über die negative Eingabe bei „Anzahl relevanter Stellen“.	Fehlermeldung: „Wählen Sie für die relevanten Stellen eine Zahl zwischen 1 und 5.“	X		Build8

19	QueryByPostalCode	„Obere PLZ-Grenzen“ enthält nicht numerische Zeichen.	Fehlermeldung über die nicht numerische Zeichen bei obere PLZ-Grenze.	Fehlermeldung: „Das Feld „Obere PLZ-Grenzen „ enthält keine gültige Zahl.	X		Build8
20	QueryByPostalCode	„Unter PLZ-Grenzen“ enthält nicht numerische Zeichen.	Fehlermeldung über die nicht numerischen Zeichen bei Untere PLZ-Grenze.	Fehlermeldung: „Das Feld „Untere PLZ-Grenzen „ enthält keine gültige Zahl.	X		Build8
21	QueryByPostalCode	Anzahl relevanter Stellen enthält nicht numerische Zeichen.	Fehlermeldung über die nicht numerischen Zeichen bei „Anzahl relevanter Stellen“	Fehlermeldung: „Das Feld „Anzahl relevanter Stellen „ enthält keine gültige Zahl.“	X		Build8
22	QueryByEnteredDateMask	Abfrage der Kundenneuzugänge	Maske für die Eingabe der Start- und Endzeitpunkte für die Abfrage der Kundenneuzugänge.	Die Datumseingabemaske für „ Abfrage der Kundenneuzugänge“	X		Build8
23	QueryByEnteredDate	Zeitraumeingabe: gültige Start- und Endzeitpunkte (01.01.1990-31.12.2000)	Auflisten der Kundenneuzugänge im angegebenen Zeitraum.	Eine Liste von Neukunden mit Neukundendatum, Name, Stadt, Kundennummer und Geburtstag.	X		Build8
24	QueryByEnteredDate	Zeitraumeingabe: gültige Start- und Endzeitpunkte in vertauschte Reihenfolge (31.12.2000-01.01.1990)	Die Reihenfolge des Datums richtig stellen.	Bei der Ausgabe der Neukundenliste wird die richtige Reihenfolge des Datums berücksichtigt. .	X		Build8
25	QueryByEnteredDate	Zeitraumeingabe: Startzeitpunkt gleich Null	Fehlermeldung über den fehlenden Startzeitpunkt.	Fehlermeldung: Das Format des Datums im Feld „Startzeitpunkt“ ist falsch. Richtiges Format tt.MM.jjjj	X		Build8
26	QueryByEnteredDate	Zeitraumeingabe: Endzeitpunkt gleich Null	Fehlermeldung über den fehlenden Endzeitpunkt.	Fehlermeldung: Das Format des Datums im Feld „Endzeitpunkt“ ist falsch. Richtiges Format tt.MM.jjjj	X		Build8
27	QueryByEnteredDate	Zeitraumeingabe: Startzeitpunkt hat kein richtiges Format.	Fehlermeldung über das falsche Format.	Fehlermeldung: „Das Format des Datums im Feld „Startzeitpunkt“ ist falsch. Richtiges Format tt.MM.jjjj“	X		Build8
28	ShowCustomer	Auswahl eines Kunden aus der Neukundenliste.	Adressdaten und Kundennummer des ausgewählten Kunden anzeigen.	Kontaktdaten von dem ausgewählten Kunden werden angezeigt.	X		Build8

29	ShowDetailedLegacyCustomer	Auswahl der Kundennummer aus der Neukundenliste.	Stammdaten des gewählten Neukunden.	Stammdaten des ausgewählten Kunden werden mit folgenden Punkten angezeigt: Allgemeine Personendaten, technische Daten, Kontoverbindungen und Vertragsbeziehungen	X		Build8
30	HistoryBook	Auswahl einer Versicherungsnummer aus der Liste der Vertragsbeziehungen.	Anzeige aller relevanten Daten zu dem ausgewählten Versicherungsvertrag.	Anzeigen des Geschichtsbuchs des ausgewählten Vertrages.	X		Build8
31	LegacySearchMask	Suche nach bestimmten Kunden	Maske für die Suche in Partnerdatenbank	Maske für die Suche in Partnerdatenbank	X		Build8
32	LegacyDoSearch	Gültige Eingabe für die Suche nach der Kundennummer	Der gesuchte Kunde als Suchergebnis.	Das Suchergebnis enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
33	LegacyDoSearch	Eingabe der eigenen VAD-ID	Anzeige der Liste von eigenen Kunden	Eine Liste von der eigenen Kunden wird angezeigt.	X		Build8
34	LegacyDoSearch	Eingabe einer fremden VAD-ID	Es soll keine Liste von der Kunden der fremden VAD angezeigt werden.	Es wird keine Suchergebnis angezeigt.	X		Build8
35	LegacyDoSearch	Eingabe einer VAD-ID durch einen eingeloggten VU bzw. AL. (Die eingegebene VAD-ID ist dem VU bzw. AL untergeordnet)	Er soll eine Liste der Kunden des eingegebenen VAD angezeigt bekommen, wenn der VAD dem jeweiligen VU bzw. AL untergeordnet ist.	Es wird keine Suchergebnis angezeigt. Bemerkung von CL: Diese Maske ist als VU oder AL gar nicht erreichbar. Daher habe ich in der Suchmaske das Eingabefeld für die VAD-ID ganz herausgenommen (ein VAD kann sowieso nur seine eigenen Kunden sehen)	X	geändert am 09.08.00	Build8
36	LegacyDoSearch	Eine VAD-ID wird durch einen eingeloggten VU bzw. AL eingegeben, die ihm (VU bzw. Al) nicht zugeordnet ist.	Es soll keine Liste angezeigt werden.	Es wird keine Suchergebnis angezeigt	X		Build8

37	LegacyDoSearch	Gültige Eingabe für die Suche nach dem Kundennamen. (Groß- und Kleinschreibung wird berücksichtigt)	Der gesuchte Kunde als Suchergebnis.	Das Suchergebnis enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
38	LegacyDoSearch	Gültige Eingabe für die Suche nach der Stadt. (Groß- und Kleinschreibung wird berücksichtigt)	Anzeige aller Kunden in der eingegebenen Stadt.	Das Suchergebnis enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
39	LegacyDoSearch	Gültige Eingabe für die Suche nach der PLZ.	Anzeige aller Kunden in der eingegeben PLZ.	Das Suchergebnis enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
40	LegacyDoSearch	Gültige Eingabe für die Suche nach Geburtstag.	Anzeige aller Kunden, die an dem eingegebenen Datum ihren Geburtstag haben.	Das Suchergebnis enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
41	LegacyDoSearch	Eingabe eines Geburtsdatums, das kein gültiges Format hat.	Fehlermeldung wegen des ungültigen Datumsformat.	Fehlermeldung: „Das Format des Datums im Feld “Geburtstag“ ist falsch. Richtiges Format: tt.MM.jjjj“	X		Build8
42	LegacyDoSearch	Gültige Eingabe für die Suche nach Vertragsnummer.	Anzeige des Kunden, für die die eingegebene Vertragsnummer gilt.	Das Suchergebnis enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
43	LegacyDoSearch	Auswahl eines Spartenschlüssels.	Anzeige aller Kunden, die in der ausgewählten Sparte Verträge haben.	Das Suchergebnis enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
44	LegacyDoSearch	Eingabe vom gültigen Datum in Felder „Neukunden am“, „Neukunden seit dem“ und „Neukunden vor dem“	Anzeige aller Kunden unter Berücksichtigung des eingegebenen Datums.	Das Suchergebniss enthält Name, Stadt, Kundennummer und Geburtstag des Kunden.	X		Build8
45	LegacyDoSearch	Eingabe vom ungültigen Datum in Felder „Neukunden am“, „Neukunden seit dem“ und „Neukunden vor dem“	Fehlermeldung	Fehlermeldung mit dem Hinweis, dass das Datum ungültig ist.	X		Build8

46	LegacyDoSearch	Eingabe von Teilzeichenketten in den Feldern, Kundennummer, VAD-ID, Name, Stadt, PLZ, Bankleitzahl und Vertragsnummer	Auflisten von Kunden, deren Daten die jeweiligen Teilzeichenketten enthalten.	Auflisten der Kunden unter Berücksichtigung der eingegeben Teilzeichenketten.	X		Build8

9.1.13 Systemtest Subsystem Search Testentwurfsspezifikation / Testfallspezifikation

Testgegenstand (Region):	Search
Testzeitraum: (Datum)	

Testteilnehmer:	Christoph Haase

Anmerkung: Für alle Testfälle stand der Shop nicht zur Verfügung.

Lfd.Nr	betroffene Action	Eingabe / Vorgehen beim Testen	Erwartete Ausgabe/Verhalten	tatsächliche Ausgabe/Beobachtung / Kommentar	Ok?	Priorität 1-3 1 = hoch/krit.	Testgrundlage (Build)
1	QuickSearch	kein Suchkriterium eingegeben, aber alle Bereiche selektiert	für jeden Bereich werden alle Elemente gefunden	Shop nicht verfügbar	?		eigene Klassen vom 09.08.2000
2	QuickSearch	wie (1) und Shop nicht verfügbar	für jeden gültigen Bereich werden alle Elemente gefunden; Meldung, dass Shop nicht verfügbar	soweit ok, allerdings steht in der URL: http://www.ipsi.de/servlets/ipsi/**ipsi**	x		eigene Klassen vom 09.08.2000
3	QuickSearch	wie (1) und Office nicht verfügbar	für jeden Bereich werden alle Elemente gefunden; Meldung, dass Office nicht verfügbar		x		eigene Klassen vom 09.08.2000
4	QuickSearch	Suche nach einer existierenden Email	Auflistung der Email		x		eigene Klassen vom 09.08.2000
5	QuickSearch	Suche nach einem existierenden Kontakt	Anzeige des Kontakts	zunächst: Fehlermeldung, dass Office-Subsystem nicht gefunden; nächste Versuche: Kontakt gefunden; Fehler nicht reproduzierbar, deshalb ok	x		eigene Klassen vom 09.08.2000
6	QuickSearch	Suche nach einem nicht existierenden Kontakt	Meldung, dass Kontakt nicht gefunden wurde		x		eigene Klassen vom 09.08.2000

7	QuickSearch	Suche nach einem bestimmten Kunden in der Partner-DB	Auflistung des Kunden		x		eigene Klassen vom 09.08.2000
8	QuickSearch	Eingabe eines Suchkriteriums, dass in mehreren Bereichen gefunden wird, alle Bereiche selektiert (Beispiel: Kunde, der in Partner-DB und Kontakte gefunden wird)	Auflistung aller gefundenen Elemente		x		eigene Klassen vom 09.08.2000
9	QuickSearch	Suche wenn Benutzer nicht eingeloggt	Fehlermeldung, dass Benutzer nicht eingeloggt	Fehlermeldungen, dass alle Systeme nicht erreichbar sind		2	eigene Klassen vom 09.08.2000
10	QuickSearch	keine Eingabe von Suchkriterium und keine Bereichsauswahl	Anzeige, dass Treffermenge leer	keine Meldung bzgl. Treffermenge erscheint		3	eigene Klassen vom 09.08.2000
11	BuildSearch-Mask	Wechsel nach Expertensuche	Anzeige des Expertensuche-Eingabebildschirm	Anm.: der Button Expertensuche entspricht nicht dem Design eines Link-Buttons	x		eigene Klassen vom 09.08.2000
12	BuildSearch-Mask	Überprüfung des Home-Buttons			x		eigene Klassen vom 09.08.2000
13	Search	keine Eingabe eines Suchkriteriums und alle Felder auswählen	Anzeige aller Elemente der Bereiche	es werden Exceptions geworfen und ein leerer Bildschirm erscheint		1	eigene Klassen vom 09.08.2000
14	Search	Suche nach einer Email und Subsystem Office läuft nicht	Fehlermeldung, dass Subsystem Office nicht erreichbar		x		eigene Klassen vom 09.08.2000
15	Search	Suche nach einem Kontakt mit einer bestimmten Email	Anzeige des Kontakts		x		eigene Klassen vom 09.08.2000

16	Search	Suche nach einem bestimmten Produkt	Anzeige des Produkts	Shop nicht verfügbar	?		eigene Klassen vom 09.08.2000
17	Search	Suche nach einem Task	Anzeige des Tasks		x		eigene Klassen vom 09.08.2000
18	Search	Suche nach einem Task-Titel mit Angabe von Wildcards (z.B. „*offene*“)	Anzeige der Aufgaben, die dem Muster entsprechen	zunächst: Fehlermeldung, dass Office-Subsystem nicht gefunden, obwohl es läuft (vgl. (5)); dann: keine Aufgaben gefunden, obwohl welche existieren (Suche mit Wildcards funktioniert generell nicht) ; Anm. CH: es wird anscheinend generell eine Suche mit Wildcards gemacht! Das sollte irgendwo vermerkt werden oder eingegebene Wildcards gelöscht werden.		1	eigene Klassen vom 09.08.2000
19	Search	Suche nach einem Termin durch Angabe eines Datums	Anzeige des Termins		x		eigene Klassen vom 09.08.2000
20	Search	Suche, wenn Benutzer nicht eingeloggt	Fehlermeldung, dass Benutzer nicht eingeloggt ist	Fehlermeldung, dass Subsysteme nicht verfügbar		2	eigene Klassen vom 09.08.2000
21	Search/QuickSearch	Klick auf Link eines gefundenen Termins	Detailansicht des Termins		x		eigene Klassen vom 09.08.2000
22	Search/QuickSearch	Klick auf Link einer gefundenen Aufgabe	Detailansicht der Aufgabe	zunächst: Internal Server Error dann: Fehlermeldung, dass Benutzer nicht eingeloggt		1	eigene Klassen vom 09.08.2000
23	Search/QuickSearch	Klick auf Link eines gefundenen Kontakts	Detailansicht des Kontakts		x		eigene Klassen vom 09.08.2000

24	Search/QuickSearch	Klick auf Link einer gefundenen Email	Öffnen eines Outlook-Fenster zur Ansicht der Email und Wechsel in Emailbereich		x		eigene Klassen vom 09.08.2000
25	Search/QuickSearch	Klick auf Link eines gefundenen Produkts	Detailansicht des Produkts	Shop nicht verfügbar	?		eigene Klassen vom 09.08.2000
26	Search/QuickSearch	Klick auf Link eines gefundenen Kunden der Partner-DB	Detailansicht des Kunden		x		eigene Klassen vom 09.08.2000

10 Die systemtechnische Architektur des IPSI-Portals

Neben der in Kapitel 7 dargelegten softwaretechnischen Architektur hat jedes technische System eine systemtechnische Architektur, die insbesondere Hardwarekomponenten (Server, Clients, Router etc.) als Elemente identifiziert und in Beziehung zueinander setzt. Auf diese Hardwarekomponenten werden die Softwarkomponenten verteilt und laufen dort in Form von Prozessen ab. Die Dokumentation der systemtechnischen Architektur dient vor allem in Form einer Referenzarchitektur dazu, das entwickelte Portal zu einem späteren Zeitpunkt wieder betreiben zu können. Daher werden in den folgenden Abschnitten dieses Kapitels insbesondere Anmerkungen zum Betrieb einzelner Subsysteme (also zum Start der Prozesse auf bestimmter Hardware) gemacht.

Die Gliederung des Kapitels orientiert sich an den eingesetzten Systemkomponenten, über die im ersten Abschnitt ein zusammenfassender Überblick gegeben wird.

10.1.1 Einführung

Das IPSI-Portal ist als verteiltes System konzipiert (s. Abbildung 58 – Die Verteilung des IPSI-Portals), d.h. die verschiedenen Softwarebestandteile des Systems laufen auf diversen unterschiedlichen Rechnersystemen. Der zentrale Bestandteil von IPSI ist ein Linux-Server, auf dem die *JVMs* (JAVA virtuelle Maschinen) der meisten internen Subsysteme von IPSI laufen, so z.B. das Kommunikationssubsystem oder die Core-Komponenten.

Weiterhin wird mindestens ein Microsoft Windows-NT Server [Mi00] benötigt, auf dem das von IPSI *eingebundene Contentmanagementsystem* (CMS) Pirobase [Pi00] und das *Shopsystem* SmartStore [Sm00] betrieben werden. Aus Gründen der besseren Wartbarkeit, Performance und Übersichtlichkeit ist es allerdings ratsam, für diese beiden Subsysteme getrennte Server einzusetzen, so wie es auch während der Projektphase gehandhabt wurde.

Ein sich am IPSI-Portal anmeldender Client benötigt spezielle Software, um das System nutzen zu können. Unter anderem ist zwingend die Installation von *Microsoft Outlook 2000* [Mi00] notwendig. Die von der PG entwickelte *DLL* (Dynamic-Link-Library), die zum Aufruf der Outlook-Funktionalität via *JNI* (JAVA-Native-Interface) aus einem Webbrowser heraus benötigt wird, und die Office-Subsystem-Fassade, die die *RMI*-Kommunikation zum Server ermöglicht, müssen ebenfalls eingerichtet werden (s. Kapitel 11.1).

Im folgenden Abschnitten sollen nun die Besonderheiten der einzelnen Systembestandteile im Detail beschrieben werden, um einer Person die Möglichkeit zu geben, den Aufbau und die Verteilung der verschiedenen Komponenten soweit nachvollziehen zu können, das sie das IPSI-Portal gegebenenfalls selbst einrichten oder warten kann.

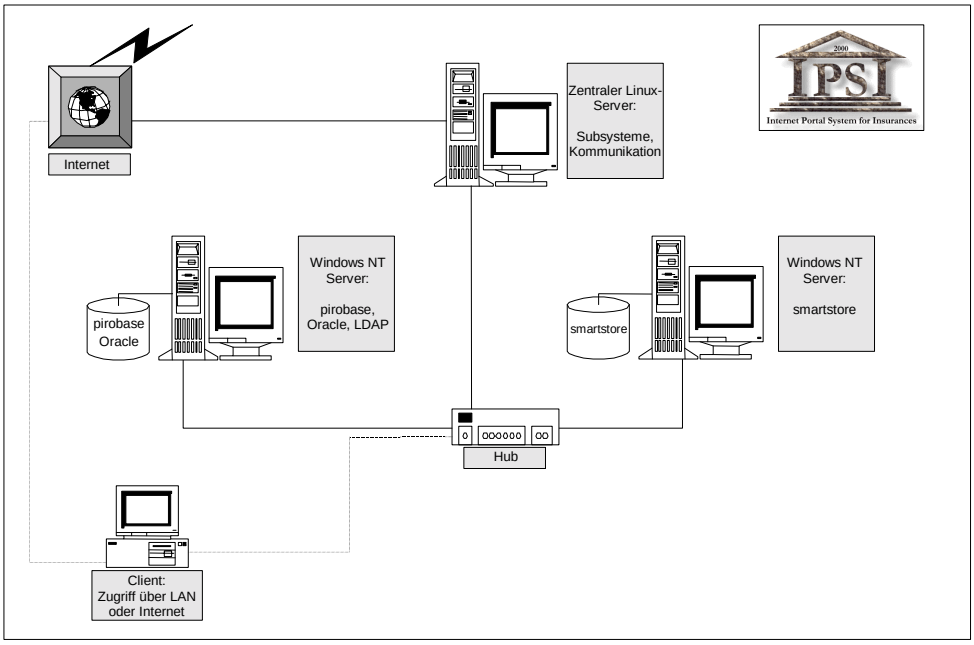


Abbildung 58 – Die Verteilung des IPSI-Portals

10.1.2 Der Pirobase-Server

Das bei IPSI als Subsystem eingebundene Contentmanagementsystem ist Pirobase der Firma Pironet AG. Ursprünglich war geplant, Version 4 zu verwenden, die eine deutlich stärkere externe Steuerung und Anpassung zulässt als Version 3. Aus organisatorischen Gründen musste leider auf Version 4 verzichtet werden, so dass sich die Anpassungen des Systems in engen Grenzen halten.

Es wird hier vorausgesetzt, dass ein Windows NT Server mit vorinstalliertem Pirobase bereitsteht. Die Installation des Pirobase-CMS ist komplex und kann daher nicht als Eigenleistung vorausgesetzt werden. Weiterhin sind die Hardwareanforderungen eines Pirobase-Servers zu beachten. Auch die Zusammenstellung geeigneter Serverhardware ist ein komplexes Gebiet, weshalb hier nur ein kurzer Überblick erfolgen soll.

Ein 450 MHz-Prozessor vom Typ Pentium 2 oder 3 ist Mindestvoraussetzung. Als untere Grenze für den Arbeitsspeicher mögen 256 MB gelten. Als Massenspeicher sollte eine moderne schnelle Festplatte mit mindestens 10 GB zum Einsatz kommen. Die von Pirobase verwendete Oracle-Datenbank profitiert deutlich von SCSI-Systemen, besonders wenn viele Zugriffe zu erwarten sind. Selbstverständlich muss der Server über eine TCP-IP-Netzwerkanbindung verfügen. Diese kann entweder über den lokalen IPSI-Serververbund oder über eine externe Internetanbindung hergestellt werden. Diese Netzwerkanbindung muss natürlich korrekt konfiguriert sein. Weitere Besonderheiten in Bezug auf die Konfiguration des Betriebssystems sind nicht vorhanden.

Als erstes müssen auf diesem System genau die Benutzergruppen und Benutzerkonten angelegt werden, die im IPSI-Portal existieren (s. Abbildung 59 - Die Gruppen in Pirobase). Dabei ist darauf zu achten, dass die Vergabe der Benutzerrechte denen des IPSI-Portals entspricht. So sollte es z.B. nur dem VU möglich sein, die Verzeichnisstruktur des Content-Store zu manipulieren. Einfache Dokumente einstellen kann dagegen (zumindest in bestimmten Bereichen) auch ein VAD.

Abbildung 59 - Die Gruppen in Pirobase

Da das Pirobase-3 System benutzt wurde, ist eine extern gesteuerte Anmeldung eines Benutzers nicht möglich, dieses wäre nur bei Pirobase-4 der Fall gewesen. Ein Benutzer des Systems muss sich also jedesmal zusätzlich am CMS anmelden, wenn er in diesen Bereich von IPSI wechselt. Trotzdem sollten die Benutzernamen und Paßworte exakt denen des IPSI-Portals entsprechen, um Konsistenz zu wahren und die Erwartungskonformität zu erfüllen.

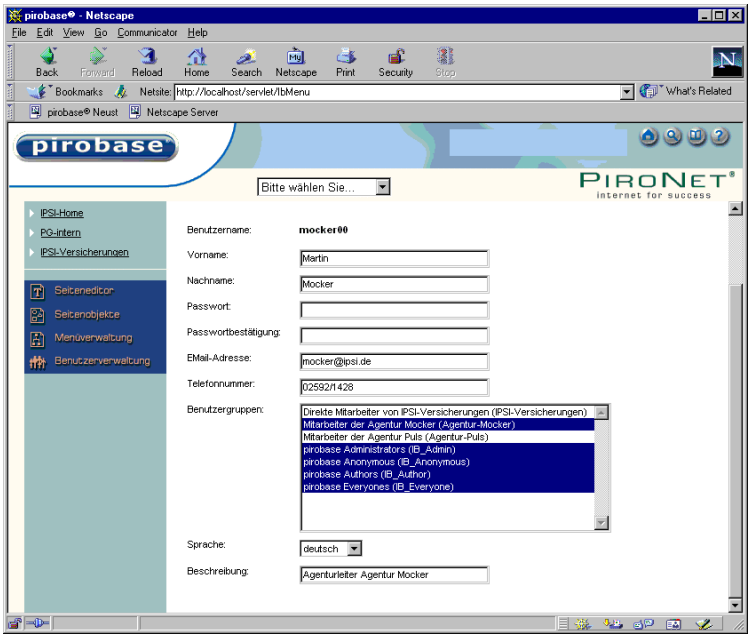


Abbildung 60 - Editieren eines Benutzerkontos

Pirobase hat zwar ein eigenes Werkzeug zur Verwaltung von Benutzerkonten (s. Abbildung 60 - Editieren eines Benutzerkontos), es hatte sich aber herausgestellt, dass aus unerfindlichen Gründen manche Benutzerkonten damit nicht angelegt werden konnten. In diesem Fall kann man die Benutzer direkt über Netscape SuiteSpot (s. Abbildung 61 - Netscape Benutzerverwaltung) anlegen (Webbrowser mit http://localhost:2000 öffnen), da beide Möglichkeiten die selbe systemweite LDAP-Datenbank manipulieren.

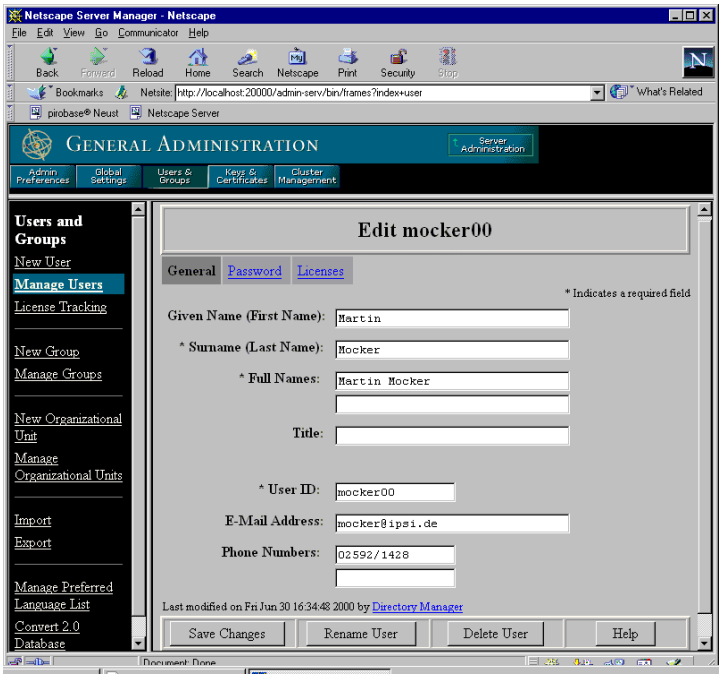


Abbildung 61 - Netscape Benutzerverwaltung

Der Aufbau des Content-Store hängt von den jeweiligen Interessen eines Versicherungsunternehmens ab. In der während des IPSI-Projekts angelegten Testumgebung wurde ein VU simuliert, dem zwei Agenturen angeschlossen sind. Jede Agentur hat mehrere angestellte VADs und jeweils einen Agenturleiter. Der verfügbare Content entspricht realen Bereichen der Versicherungswirtschaft, so gibt es z.B. Bereiche für interne Newsletter und Memos des Unternehmens oder auch Anträge und Schadensanzeigen, die ein VAD beim Kunden herunterladen

und ausdrucken kann. Diese Dokumente können dann direkt vom Kunden ausgefüllt werden, wobei der VAD behilflich sein kann.

10.1.3 Der SmartStore-Server

SmartStore 2.0.2 wurde als Shopsystem für die Realisierung des Subsystems Procurement im Portal eingesetzt. Dieses Shopsystem basiert auf *Active Server Pages* und speichert die Daten standardmäßig in einer Microsoft Access Datenbank [Sm00][HoSu00].

Die Systemvoraussetzungen für den Betrieb des Shopsystems sind:

- Mind. 133 MHz Intel Pentium® Prozessor (empfohlen 200 MHz)
- Windows NT 4.0 (ab Service Pack 4)
- 64 MB Arbeitsspeicher (empfohlen 128 MB)
- Mind. 100 MB freier Festplattenspeicher
- Weitere erforderliche Softwarekomponenten sind auf der Installations-CD enthalten.

Die genaue Installationsprozedur ist in Abschnitt 11.5 dargestellt.

10.1.4 Die Clients

Die Client-Rechner dienen dem Zugriff auf das IPSI-System über einen WWW-Browser. Dieser WWW-Browser ist auch die zentrale Software, die auf dem Client zur Verfügung stehen muss. Der Browser muss JAVA-fähig sein, um Applets ausführen zu können. Getestet wurden die Browser von Netscape (Communicator in der Version 4.72) und Microsoft (Internet Explorer Version 5.0). Empfohlen wird der Einsatz des Microsoft Internet Explorers. Die Software von Netscape zeigte teilweise unerklärliches Verhalten bei einigen Seitenaufrufen, die der Explorer ohne Probleme durchführte.

Weiterhin wird auf jedem Client zur Kommunikation zwischen dem lokalen Office-Subsystem und anderen Portalkomponenten eine JAVA-Applikation und eine Windows DLL benötigt. Die Verwendung von Outlook macht Windows als Betriebssystem unentbehrlich. Diese Aspekte werden detailliert im Abschnitt 11.1 besprochen.

Die genannte benötigte Client-Software impliziert Anforderungen an die auf der Clientseite eingesetzte Hardware. Die PG verwendete folgende Hardware als Client-Rechner

- Prozessoren: P 200 – P II 350
- Arbeitsspeicher: 64 – 128 MB RAM
- Festplatten: 500 MB – 5 GB
- Betriebssystem: Windows NT 3.5

10.1.5 RMI und Firewalls

Es muss noch auf einige wichtige Besonderheiten in Bezug auf RMI-Kommunikation über unterschiedliche Netzwerke eingegangen werden, die sich im Laufe der Implementierung und des Tests gezeigt haben.

Als versucht wurde, das IPSI-Portal aus einem anderen lokalen Netzwerk heraus via Internet anzusprechen, kam zwar eine Kommunikation zustande, es konnten aber keine RMI-Pakete empfangen werden. Dies ließ sich unter anderem darauf zurückführen, dass dieses Netzwerk hinter einer Firewall lag. Die einzelnen Arbeitsstationen hatten wie im IPSI-Netz lokale nicht-routbare IP-Adressen (192.168.10.xxx), der Zugriff auf das Internet wurde von einer Firewall via IP-Masquerading bereitgestellt. Die RMI-Kommunikation erreichte zwar den IPSI-Server, die zurückgesandten RMI-Pakete konnten allerdings nicht wieder dem zugreifenden Rechner zugeteilt werden, da die Kommunikation mit der externen IP-Adresse der Firewall geführt wurde.

Weiterhin wurde während der Suche nach einer Lösung dieses Problems der Hinweis im Usenet [De00] gefunden, dass bei großen RMI-Paketen (im Kilobyte Bereich) der angesprochene RMI-Server automatisch selbst eine Verbindung initiiert und deshalb an der Firewall der Absenderadresse scheiterte.

Die Lösung dieses Problems bestand darin, den Client mit einer ISDN-Karte auszustatten und sich zur Internet-Nutzung über einen normalen Call-by-Call Internet-Provider einzuwählen (wie es z.B. von Mobilcom oder Man-

nesmann Arcor bereitgestellt wird). In diesem Fall bekommt der Client eine eigene IP-Adresse zugewiesen, die während der Dauer der Einwahl gültig und eindeutig ist. Leider ließ sich auch dann keine Verbindung zum IPSI-Server herstellen, die Symptome waren dabei ähnlich wie im zuvor geschilderten Fall.

Dieses Problem ließ sich letztendlich beseitigen, indem die Treiber der im Rechner eingebauten Netzwerkkarte entfernt wurden, so dass kein weiteres Netzwerk Interface mehr vorhanden war. Dies führte schließlich zum Erfolg. Daraus läßt sich feststellen, dass die kommunizierten RMI-Pakete sich automatisch an das Interface der Netzwerkkarte banden. Eine Möglichkeit der Einflußnahme darauf besteht vermutlich nicht. Das Problem wurde in dieser Richtung allerdings nicht weiter untersucht.

Zusammenfassend lassen sich zwei wichtige Forderungen für Clients feststellen, die über das Internet auf das IPSI-Portal zugreifen möchten:

1. Der zugreifende Client darf keine IP-Adresse aus dem reservierten Pool nicht-routbarer Adressen erhalten, da eine Kommunikation über IP-Masquerading mit hoher Wahrscheinlichkeit fehlschlagen wird. Er benötigt zwingend eine eindeutige routbare Adresse. Die einzige Ausnahme besteht, wenn sich wie im IPSI-Netz Server und Client im selben lokalen Netzwerk befinden.

2. Der zugreifende Client sollte nur ein installiertes Netzwerk-Interface für TCP/IP-Zugriff haben. Wenn mehrere Netzwerk-Interfaces vorhanden sind (z.B. Netzwerkkarte und ISDN-Karte oder Modem zur Interneteinwahl) ist nicht vorherzusehen, an welches Netzwerk-Interface sich die RMI-Kommunikation binden wird.

11 Installation und Konfiguration

In diesem Kapitel sollen detaillierte Anweisungen zur Installation der Softwaresystem auf den im vorigen Kapitel beschriebenen Hardwaresystemen dargestellt werden. Die Gliederung erfolgt daher nach den verwendeten Subsystemen und weiteren Komponenten, die einer Installation und Konfiguration bedürfen.

11.1 Subsystem Office

11.1.1 Anforderungen

Zur Verwendung des Office-Subsystems wird eine lokale Installation von MS Outlook 2000 ohne Service Packs benötigt. Es existieren keine Erfahrungen, wenn zusätzlich Service Packs installiert werden. Bei Tests sind Probleme aufgetreten, wenn MS Outlook 2000 über eine ältere Version installiert wurde, so dass das möglichst vermieden werden sollte.

Außerdem wird eine JAVA-Laufzeitumgebung benötigt. Tests waren erfolgreich mit den Laufzeitumgebungen der JDKs 1.2.2 und 1.3 von Sun. Die Verwendung anderer Laufzeitumgebungen sollte aber unproblematisch sein.

Desweiteren müssen die Klassen des Office-Subsystems (momentan über ipsi.jar zu bekommen) und die Dynamic Link Library „office.dll“ auf den Client kopiert werden.

11.1.2 Starten des Subsystems Office

Das Office-Subsystem verwendet eine lokale Installation von MS Outlook, so dass es auch lokal gestartet werden muss. Dies ist momentan über eine JAVA-Application realisiert, die explizit auf dem Client gestartet werden muss (in einer weiteren Ausbaustufe sollte das über ein signiertes Applet realisiert werde, so dass der Benutzer sich nur in das Portal einloggen muss). Dieser Applikation wird der Benutzername (des Portals) und das Passwort als Parameter übergeben.

Zum Starten des Office-Subsystems steht eine Batchdatei zur Verfügung, die allerdings angepasst werden muss.

Sie hat folgendes Aussehen:

```
set path=<Name des Office.dll-Verzeichnisses>;%path%


JAVA  -cp <Name des ipsi.jar-Verzeichnisses>\ipsi.jar
      -DRMINameServer=<IP-Adresse des RMI-Nameservers>
      de.ipsi.subsystem.office.OfficeSubsystemFacadeImpl
      <Benutzer-Login> <Passwort>
```

Diese Batchdatei muss nun wie folgt angepasst werden:

1. Eintragung des Verzeichnisnamens der office.dll-Datei in der set path-Anweisung
   Bsp.: set path=c:\lib;%path% (wobei sich office.dll im Verzeichnis c:\lib befindet)

2. Eintragung des Verzeichnisses, in dem sich ipsi.jar befindet
   Bsp.: ... –cp c:\lib\ipsi.jar (wobei sich die ipsi.jar-Datei im Verzeichnis c:\lib befindet)

3. Eintragung der IP-Adresse des RMI-Nameservers
   hier ist (zur Zeit) die Adresse des PG-Servers einzutragen: 129.217.40.211

4. Eintragung des Benutzerloginnamens und Benutzerpasswort
   z.B.: meier00 meier
   (hierbei ist zu beachten, dass jeder Benutzer seine eigene Kennung verwendet; ansonsten kann es vorkommen, dass mehrere Subsysteme mit gleichem Namen laufen; das kann zu undefiniertem Verhalten führen)

11.2 Subsystem Administration

11.2.1 Allgemein: Zugriff auf mySQL

MySQL unterscheidet nicht zwischen einzelnen Benutzern, sondern zwischen der Kombination aus Benutzer und Host. Diese Kombination ergibt eine eindeutige ID. Das Sicherheitssystem regelt hierbei genauestens, welcher Benutzer von welchem Host welche Rechte auf welcher Datenbank besitzt. Diese Einstellungen werden in den drei Tabellen user, db und host abgelegt.

11.2.1.1 Die user-Tabelle

Die user-Tabelle beinhaltet alle Host-/User-Kombinationen, welche den *MySQL*-Server connecten dürfen. Alle Berechtigungen, die ein Benutzer in dieser Tabelle enthält, gelten für alle Datenbanken, sofern keine erweiterten Berechtigungen für den jeweiligen Benutzer in der Tabelle db definiert wurden. Man kann diese Berechtigungen auch als grundlegende Einstellungen ansehen und ein datenbankabhängiges Feintuning in der Tabelle db festlegen.

Der Tabellenaufbau:

```
Field Type Key Default
----------------------------
Host char(60) PRI
User char(16) PRI
Password char(16) -
Select_priv enum(N,Y) - N
Insert_priv enum(N,Y) - N
Update_priv enum(N,Y) - N
Delete_priv enum(N,Y) - N
Create_priv enum(N,Y) - N
Drop_priv enum(N,Y) - N
Reload_priv enum(N,Y) - N
Shutdown_priv enum(N,Y) - N
Process_priv enum(N,Y) - N
File_priv enum(N,Y) - N
```

11.2.1.2 Die db-Tabelle

In dieser Tabelle wird definiert, welche Datenbank der jeweilige Host/Benutzer mit welchen Berechtigungen verwenden darf. Sie stellt wie bereits erwähnt das Feintuning da.

Der Tabellenaufbau:

```
Field Type Key Default
-------------------------
Host char(60) PRI
Db char(64) PRI
User char(16) PRI
Select_priv enum(N,Y) - N
Insert_priv enum(N,Y) - N
Update_priv enum(N,Y) - N
Delete_priv enum(N,Y) - N
Create_priv enum(N,Y) - N
Drop_priv enum(N,Y) - N
```

11.2.1.3 Die host-Tabelle

Die host-Tabelle ist in großen Netzwerken als *Nachschlage*-Tabelle für leere Host-Einträge in der db-Tabelle sinnvoll. Möchte man, dass ein Benutzer von jedem Host in dem Netzwerk auf den DB-Server zugreifen kann, sollte man den Host-Eintrag in der db-Tabelle auslassen und alle Host des Netzwerkes in der host-Tabelle eintragen.

Der Tabellenaufbau:

```
Field Type Key Default
-------------------------
Host char(60) PRI
Db char(64) PRI
Select_priv enum(N,Y) - N
Insert_priv enum(N,Y) - N
Update_priv enum(N,Y) - N
Delete_priv enum(N,Y) - N
Create_priv enum(N,Y) - N
Drop_priv enum(N,Y) - N
```

11.2.1.4 Definitionen

Die HOST und DB Spalten können Strings mit Wildcards (% und _) beinhalten. Wird für diese Spalten kein Wert eingetragen, entspricht dies dem Wert %.

Ein HOST kann sein:

- localhost
- ein Hostname
- eine IP-Nummer
- ein String mit Wildcards

Ein leerer HOST-Eintrag in der db-Tabelle bedeutet: -kein Host- aus der host-Tabelle. Ein leerer HOST-Eintrag in der host- oder user-Tabelle bedeutet: -kein Host-.

Die Spalte DB beinhaltet den Namen einer Datenbank oder einer SQL Regexp.

Ein leerer Benutzereintrag bedeutet: -kein Benutzer-. In dieser Spalte können keine Wildcards verwendet werden.

Die Berechtigungen der user-Tabelle werden mit den Berechtigungen aus der db-Tabelle Oder-verknüpft. Dies bedeutet, dass ein *Superuser* nur in der Tabelle user mit allen Berechtigungen festgelegt auf Y eingetragen werden muss.

Wenn man den Aufbau der Tabellen näher betrachtet, wird man feststellen, dass die user-Tabelle zusätzlich zu den Zugriffsberechtigungen auf die jeweilige Datenbank auch administrative Berechtigungen regelt. Dadurch sollte klar sein, dass diese Tabelle die grundlegenden Berechtigungen regelt.

11.2.1.5 Festlegen von Berechtigungen

Im folgenden Beispiel soll der Benutzer custom mit dem Paßwort stupid angelegt werden. Der Benutzer soll die Möglichkeit haben sich von den Hosts localhost, server.domain und whitehouse.gov anzumelden. Die Datenbank bankaccount möchte er nur von dem Host localhost und die Datenbank customers von allen Hosts ansprechen können.

```
> mysql -u root mysql

mysql> insert into user (host,user,password)
values(localhost,custom,password(stupid));
mysql> insert into user (host,user,password)
values(server.domain,custom,password(stupid));
mysql> insert into user (host,user,password)
values(whitehouse.gov,custom,password(stupid));
mysql> insert into db(host,db,user,Select_priv,Insert_priv,Update_priv,
Delete_priv,Create_priv,Drop_priv)values
(localhost,bankaccount,custom,Y,Y,Y,Y,Y,Y);
mysql> insert into db(host,db,user,Select_priv,Insert_priv,Update_priv,
Delete_priv,Create_priv,Drop_priv)values
(%,customers,custom,Y,Y,Y,Y,Y,Y);
mysql> quit
> mysqladmin reload
```

11.2.2 Erstellen der Datenbank ipsi

11.2.2.1 Zugriffsrechte

Zuerst führt man ein Skript aus, das die Datenbank ipsi erstellt, die entsprechenden Hosts und User einträgt und sie mit der Datenbank verknüpft. Exemplarisch sei hier nur ein Host (192.168.10.12) aufgeführt. Natürlich müssen hier alle Hosts eingetragen werden, von denen der Zugriff auf die Datenbank möglich sein soll. Außerdem ist hier nur ein beispielhafter User aufgeführt (ipsiuser), der wirklich eingetragene User samt Passwort bleibt aus Gründen der Datensicherheit geheim.

Aufruf:

```
>mysql --user=root --password=admin mysql < sql-dbsetup.txt
```

sql-dbsetup.txt:

```
create database ipsi;

insert into host(Host, Db, Select_priv, Insert_priv, Update_priv,
Delete_priv, Create_priv, Drop_priv)
values ('192.168.10.12', 'ipsi', 'Y', 'Y', 'Y', 'Y', 'N', 'N');

insert into user(Host, User, Password, Select_priv, Insert_priv,
Update_priv,Delete_priv, Create_priv, Drop_priv, Reload_priv,Shutdown_priv,
Process_priv, File_priv)
values('%', 'ipsiuser', password('ipsipassword'), 'Y', 'Y', 'Y','Y',
'N', 'N', 'N', 'N', 'N', 'N');

insert into db(Host, Db, User, Select_priv, Insert_priv, Update_priv,
Delete_priv, Create_priv, Drop_priv)values('%', 'ipsi', 'ipsiuser', 'Y',
'Y', 'Y', 'Y', 'N', 'N');
```

11.2.2.2 Datenbankschema

11.2.2.2.1 Tabelle PORTALUSER

Ein Benutzer besitzt UserIDs und Passwörter, und zwar jeweils ein Paar für jedes Subsystem (Administration, Shop, CMS, Office). Außerdem besitzt er Adressdaten, wie z.B. Anschrift, Telefonnummern oder Email-Adresse. Er ist genau einer Agentur zugeordnet, deren AgenturID auch eingetragen ist.

NAME	VALUE
UserID	varchar(255)
PASSWORD	varchar(255)
SHOPID	varchar(255)
SHOPPASSWORD	varchar(255)
CMSID	varchar(255)
CMSPASSWORD	varchar(255)
OFFICEID	varchar(255)
OFFICEPASSWORD	varchar(255)
SALUTATION	varchar(255)
FIRSTNAME	varchar(255)
SURNAME	varchar(255)
STREET	varchar(255)
CITY	varchar(255)
POSTALCODE	varchar(255)
TELEPHONE	varchar(255)
FAX	varchar(255)
EMAIL	varchar(255)
HANDY	varchar(255)
AGENCYID	varchar(255)

11.2.2.2.2 Tabelle AGENCIES

Eine Agentur besitzt eine eindeutige AgenturID sowie genau einen Agenturleiter. Außerdem müssen die Adressdaten der Agentur eingetragen werden.

NAME	VALUE
AGENCYID	varchar (255)
AGENVYLEADERID	varchar (255)
NAME	varchar (255)
STREET	varchar (255)
CITY	varchar (255)
POSTALCODE	varchar (255)
TELEPHONE	varchar (255)
FAX	varchar (255)
EMAIL	varchar (255)

11.2.2.2.3 Tabelle FEEDBACKS

Ein Feedback besteht aus einer Schulnote, einem Kommentar, dem Erstelldatum und der UserID des Benutzers, über den das Feedback erstellt wurde.

NAME	VALUE
VICTIMID	varchar (255)
GRADE	int
DATE	bigint
COMMENT	varchar (255)

11.2.2.2.4 Tabelle RIGHTS

Ein Benutzer kann mehrere Rechte haben. Deshalb werden die Rechte nicht in die Tabelle PORTALUSER eingetragen, sondern in diese spezielle Tabelle. Einer UserID ist ein Recht in Form des Schlüsselwortes zugeordnet.

NAME	VALUE
USERID	varchar (255)
KEYWORD	varchar (255)

11.2.2.2.5 Tabelle INTERESTS

Wie auch schon bei den Rechten kann ein Benutzer mehrere Interessen haben, weshalb die Interessen in diese Tabelle eingetragen werden. Auch hier ist einer UserID ein Interesse in Form des Schlüsselwortes zugeordnet.

NAME	VALUE
USERID	varchar (255)
KEYWORD	varchar (255)

11.2.2.2.6 Tabelle Communication_log

Diese Tabelle gehört zwar nicht zum Subsystem Administration, sondern zum Subsystem Communication, sei hier aber vollständigkeitshalber aufgeführt.

NAME	VALUE
date	int
medium	int
status	int
description	text
userid	varchar (255)

11.2.2.3 Anlegen der Tabellen

Danach erstellt man die einzelnen Tabellen der neuen Datenbank ipsi.

Aufruf:

```
mysql --user=root --password=admin ipsi < sql-tablesetup.txt
```

sql-tablesetup.txt:

```
create table PORTALUSER (USERID varchar(255),PASSWORD varchar(255),
SHOPID varchar(255),SHOPPASSWORD varchar(255),CMSID varchar(255),
CMSPASSWORD varchar(255),OFFICEID varchar(255),OFFICEPASSWORD varchar(255),
SALUTATION varchar(255),FIRSTNAME varchar(255),SURNAME varchar(255),
STREET varchar(255),CITY varchar(255),POSTALCODE varchar(255),
TELEPHONE varchar(255),FAX varchar(255),EMAIL varchar(255),
HANDY varchar(255),AGENCYID varchar(255));

create table AGENCIES (AGENCYID varchar(255),AGENCYLEADERID varchar(255),
NAME varchar(255),STREET varchar(255),CITY varchar(255),
POSTALCODE varchar(255),TELEPHONE varchar(255),FAX varchar(255),
EMAIL varchar(255));

create table FEEDBACKS (VICTIMID varchar(255),GRADE int,DATE bigint,
COMMENT varchar(255));

create table RIGHTS (USERID varchar (255),KEYWORD varchar (255));

create table INTERESTS (USERID varchar (255),KEYWORD varchar (255));

create table Communication_log (date int,medium int,status int,
description text,userid varchar (255));
```

Anschließend führt man noch „reload“ durch, um die Änderungen auch tatsächlich auszuführen.

Aufruf:

```
mysqladmin --user=root --password=admin reload
```

Nun sind alle Tabellen der Datenbank ipsi erzeugt.

11.2.2.4 Zugriff über JAVA

Der Zugriff auf die Datenbank über JAVA funktioniert mit folgenden Parametern. Man beachte, dass der tatsächlich benutzte User mit dem Passwort geheim bleibt, stattdessen wird wieder der schon oben eingeführte exemplarische User (ipsiuser) benutzt.

```
Driver = „org.gjt.mm.mysql.Driver“;
URL = „jdbc:mysql://homer:3306/ipsi“;
User = „ipsiuser“;
Password = „ipsipassword“;
```

11.2.3 Einfügen der Testdaten

Die Testdaten wurden mit Hilfe des Skripts ipsidatenskript.txt eingefügt. Es genügt, hier das Einfügen eines Benutzers und einer Agentur zu demonstrieren. Weitere Benutzer fügt man analog ein.

ipsidatenskript.txt:

```
insert into PORTALUSER
(USERID,PASSWORD,SHOPID,SHOPPASSWORD,CMSID,CMSPASSWORD,OFFICEID,
OFFICEPASSWORD,SALUTATION,FIRSTNAME,SURNAME,STREET,CITY,POSTALCODE,
TELEPHONE,FAX,EMAIL,HANDY,AGENCYID) values
('muster00','mustermann','muster00','mustermann','muster00','mustermann',
'muster00','mustermann','Herr','Max','Mustermann','Baroper Str. 301 ',
'Dortmund','44227','0231/7555440','null','team@ipsi.de','null','0001');

insert into RIGHTS(USERID,KEYWORD) values ('muster00','agenturleiter');

insert into INTERESTS(USERID,KEYWORD) values
('muster00','Software');
insert into INTERESTS(USERID,KEYWORD) values
('muster00','Hardware/Zubehör');
insert into INTERESTS(USERID,KEYWORD) values
('muster00','Telekommunikation');

insert into AGENCIES
(AGENCYID,AGENCYLEADERID,NAME,STREET,CITY,POSTALCODE,TELEPHONE,FAX,EMAIL)
values ('0001','muster00','Agentur Mustermann',
'Baroper Str. 301','Dortmund','44227','0231/7555440','null',
'team@ipsi.de');
```

Die Feedbacks wurden mit Hilfe einer JAVA-Klasse eingefügt:

```
package testumgebung;

import java.util.*;
import java.lang.*;
import java.io.*;
import de.ipsi.subsystem.administration.*;
import de.ipsi.core.*;
import de.ipsi.core.http.*;
import java.rmi.*;

public class setfeedbacks {

 public static void main(String args[]){

  AdminSubsystemFacade fassade = null;
  SubsystemManager ssmgr = null;

  // System-property „RMINameServer“ auslesen -->
  // gibt den Namen des Rechners
  // an, auf dem der SubsystemServer Prozess läuft.
  String rmiNameServer = System.getProperty(„RMINameServer“);
  if (rmiNameServer == null) {
```

```
    System.err.println(„IPSI-Dispatcher: „ +
                       „Property RMINameServer nicht angegeben.“ +
                       „Kann NameServer somit nicht finden.“);
    System.exit(1);
  }

  // nun das SubsystemManager Remote-Object vom Nameserver holen
  try {
   ssmgr = (SubsystemManager) Naming.lookup(„rmi://“+ rmiNameServer
                                            +“/SubsystemManager“);
  }
  catch ( Exception e ) {
   System.err.println(„Kann SubsystemManager nicht finden. Sterbe dahin.“);
   e.printStackTrace();
   System.exit(1);
  }

  try{
   fassade = (AdminSubsystemFacade)
   ssmgr.getSubsystemForName(„Administration“);
  }
  catch (Exception e){
   e.printStackTrace();
  }

  // Erst werden die Calendars erzeugt.
  Calendar date206 = new GregorianCalendar(2000,5,20,13,45,33);
  Calendar date236 = new GregorianCalendar(2000,5,23,16,22,54);
  Calendar date246 = new GregorianCalendar(2000,5,24,20,10,35);
  System.out.println(„Calendars wurden erzeugt.“);

  // Dann werden die Victims erzeugt.
  User muster00 = null;
  try{
   muster00 = fassade.getUser(„muster00“,“mustermann“);
   System.out.println(„User wurden erzeugt.“);
  }
  catch (Exception e){
   System.out.println(e.getMessage());
  }

  // Jetzt werden die Feedbacks eingetragen
  try{
   fassade.insertFeedback(muster00,1,date206.getTime(),
                          „Nirgendwo wars billiger“);
   fassade.insertFeedback(muster00,5,date236.getTime(),
                          „Inkompetenter Misthaufen“);
   fassade.insertFeedback(muster00,2,date246.getTime(),
                          „Freundlich, kompetent, pünktlich“);
   System.out.println(„Feedbacks wurden eingetragen.“);
  }
  catch (Exception e){
   System.out.println(e.getMessage());
  }
  // Ende
 }
}
```

11.2.4 Starten der Subsystem-Fassade

Die Startparameter sind der Datei rcAdministration.txt zu entnehmen:

```
# Realname des Subsystems z.B. 'Legacy'
export SUBSYS=„Administration“
# Pfad zur virtuellen Maschine
export JAVAVM=„/usr/local/JAVA/bin/JAVA“
# Startparameter für die virtuelle Maschine
export JAVAPARM“=-Xms512k -Xmx4M“
# Der nötige Classpath für das Subsystem
export CLASSPATH=„/home2/runtime/class-
override:/home2/runtime/lib/ipsi.jar:/home2/runtime/lib/mysql_comp.jar“
# Die Startklasse mit der Main-Methode
export STARTCLASS=
```

```
„de.ipsi.subsystem.administration.AdminSubsystemFacadeImpl“
# Die Startparameter für diese Klasse
export STARTPARM=
„-Djava.rmi.server.codebase=file:///home2/runtime/lib/ipsi.jar
 -DRMINameServer=localhost -DDebugMode=yes“
```

Der Classpath muss sowohl auf das ipsi.jar, in dem alle Klassen des IPSI-Portals enthalten sind, als auch auf das mysql_comp.jar zeigen, das nötig ist, um mit der Datenbank zu arbeiten. Die Fassadenklasse ist erreichbar über den Pfad de.ipsi.subsystem.administration.AdminSubsystemFacadeImpl. Die Startparameter für diese Klasse sind:

- java.rmi.codebase: hiert wird festgelegt, wo die Stub- und Skeleton-Klassen zu finden sind, die RMI benötigt (normalerweise das ipsi.jar)
- RMINameServer (normalerweise localhost)
- DebugMode: Sollen Debug-Meldungen ausgegeben werden (yes/no)

11.3 Subsystem Kommunikation

11.3.1 Abhängigkeiten vom Server

Das Subsystem Kommunikation benötigt für den Versand von Emails Zugriff auf einen SMTP-Server. Die Konfigurationsparameter, wie etwa die IP-Adresse, werden dem Subsystem über eine Konfigurationsdatei mitgeteilt. Der Aufbau dieser Konfigurationsdatei wird in Abschnitt 11.3.4 beschrieben. Die Einrichtung des SMTP-Servers an sich wird in Abschnitt 0 beschrieben.

11.3.2 Abhängigkeiten von Libraries

Das Subsystem Kommunikation in der aktuellen Version basiert auf den folgenden JAVA-Libraries:

- *JAVAEmail* (vgl. [Su00a]) Version 1.1.3
  Über diese Library ist die Funktionalität zum Versenden von Emails aus einem JAVA-Programm verfügbar.

- *JAVABeans Activation Framework* (vgl. [Su00d]) Version 1.0.1
  Diese Library wird von der Library *JAVAEmail* verwendet.

- *mysql_comp* (vgl. 11.2)
  Über diese Library werden Funktionen zum Zugriff auf eine SQL-Datenbank bereitgestellt.

11.3.3 Startparameter

Zum Starten des Subsystems Kommunikation müssen folgende Startparameter übergeben werden :

- java.rmi.server.codebase
  Ein Verweis auf die ipsi.jar-Datei. Diesen Verweis benötigt die RMI-Middleware.
  Beispiel: file:///home2/runtime/lib/ipsi.jar

- RMINameServer
  Die IP-Adresse des RMI-Nameservers.
  Beispiel: 192.168.10.1

- SubsystemManagerNameserverAddress
  Die IP-Adresse des Nameservers des SubsystemManagers
  Beispiel: 192.168.10.1

- CommunicationConfigFile
  Gibt die Position und den Namen der Konfigurationsdatei (siehe Abschnitt 11.3.4) an, die vom Subsystem verwendet werden soll.
  Beispiel: /home2/runtime/source/de/ipsi/subsystem/Communication/CommunicationConfig.txt

- DebugMode
  Optionaler Parameter. Wird für diesen Parameter der Wert „yes“ übergeben, so wird der Debug-Modus aktiviert und es werden Statusmeldungen während der Laufzeit ausgegeben.

11.3.4 Konfigurationsdatei

In der Konfigurationsdatei sind die Parameter für alle verwendeten Hardwareressourcen vermerkt. Diese Datei wird vom Subsystem gelesen und die Hardware entsprechend diesen Parametern angesprochen. Für jeden logischen Versandkanal muss ein Key-Value-Paar vorhanden sein, dass die Zuordnung dieses logischen Kanals zu der verwendeten physikalischen Hardwareressource (Modems, SMTP-Server) angibt. Die logischen Kanäle werden für jedes Versandmedium bei Null beginnend durchnummeriert. Zusätzlich werden für jede physikalische Hardwareressource spezielle Parameter erwartet. Ein SMTP-Server beispielsweise benötigt die Parameter IP, USER und PASSWD. Ein Modem erwartet vollkommen andere Parameter, die jedoch noch nicht spezifiziert sind, da sowohl der Fax- als auch der SMS-Versand nicht implementiert sind. Die Namen der speziellen Hardwareparameter werden gebildet, indem die ID der Hardwareressource (also SMTP oder Modem plus fortlaufende Nummer) mit einem Unterstrich und dem Namen des speziellen Parameters kombiniert wird. Zur Veranschaulichung hier ein Beispiel mit drei physikalischen und vier logischen Kanälen:

```
# Mapping zwischen verfügbaren logischen und physikalischen Versandkanälen:
EmailChannel0 = SMTP0
FaxChannel0 = Modem0
FaxChannel1 = Modem1
SMSChannel0 = Modem0

# Konfigurationsdaten für den Email-Server SMTP0:
SMTP0_IP = 192.1268.10.1
SMTP0_USER = username
SMTP0_PASSWD = passwort

# Konfigurationsdaten für das Modem Modem0:
# Die Parameter müssen noch spezifiziert werden.

# Konfigurationsdaten für das Modem Modem1:


# Die Parameter müssen noch spezifiziert werden.
```

11.3.5 SQL-Tabelle

Zum Speichern der Log-Einträge von erfolgreichen und fehlerhaften Versandaktionen des Subsystems Kommunikation wird eine SQL-Tabelle verwendet. Wie die SQL-Datenbank aufgesetzt wird, wurde im Abschnitt 11.2 beschrieben. Der Aufbau der SQL-Tabelle ist wie folgt:

NAME	VALUE
date	int
medium	int
status	int
description	text
userid	varchar (255)

11.4 Legacy Subsystem

Für das Legacy Subsystem wird die „JAVA API for XML Parsing“, die Sun auf seiner Webseite zur Verfügung stellt [Su00c], benötigt. Die Version 1.01 wird von uns verwendet und wird auch in unserer Library eingesetzt. Eingebunden sind die beiden Dateien parser.jar und jaxp.jar, die die Klassen der API beinhalten.

11.4.1 Wie startet man das Legacy Subsystem?

Das Legacy Subsystem wird auf dem Server gestartet und verlangt einen Startparameter, da dem System mitgeteilt werden muss, wo es die XML-Dateien auf der Festplatte finden kann, z.B.

-DXMLFilePath=/home2/runtime/source/de/ipsi/subsystem/legacy/xml-datenbank

Die XML-Dateien müssen im folgenden Format vorliegen

- Für die Vertragsdaten: „XKU“ + Versicherungsnummer + „.xml“
- Für die Geschichtsbücher: „XGB“ + Vertragsnummer + „.xml“

11.5 Subsystem Procurement

Wie bereits dargelegt, wurde SmartStore 2.0.2. für das Subsystem Procurement verwendet.

11.5.1 Vor der Installation

Vor der Installation von SmartStore 2.0.2 sollten folgenden Softwarekomponenten auf dem Shopserver installiert sein:

- Microsoft Internet Explorer 5.0
- NT Option Pack 4.0
- Windows NT Service Pack 4 oder höher
- MDAC 2.0

Auf den Installationsseiten, die nach dem Einlegen der SmartStore-CD erscheinen, können diese Pakete unter *smartstore installieren* > *Einzelplatzinstallation* installiert werden. Beim NT Option Pack 4.0 soll die Standardinstallation gewählt werden, wobei der Webserver *Microsoft Internet Information Server (IIS)* installiert wird [StCh99].

Die Microsoft Datenzugriffskomponente (MDAC) enthalten die Schlüsseltechnologien für den universellen Datenzugriff und -austausch. Diese Komponenten beinhalten ADO, OLE DB, ODBC und RDS. MDAC 2.0 soll vollständig installiert werden [Ms00].

11.5.2 Installation von SmartStore 2.0.2

Die Installation von SmartStore kann von den Installationsseiten unter *SmartStore installieren* > *Einzelplatzinstallation* > *SmartStore Standard Edition 2.0.2 installieren* gestartet werden. Dabei werden sowohl die Client-Anwendung als auch die Server-Dateien auf dem Shopserver installiert.

Zusätzlich sollten die *SmartStore Server Extensions* installiert werden, die unter anderem das Email-Modul für den Shop bereitstellen. Das Packet befindet sich unter dem Verzeichnis WWS auf der SmartStore-Installations-CD.

Wenn die SmartStore-Clientsoftware zum ersten mal gestartet wird, müssen die Lizenzierungsdaten eingegeben werden. Danach kann man sich als Administrator ohne Passwort anmelden. Nach der Anmeldung kann dann im Verwaltungsbereich ein Administrationspasswort eingegeben werden [Sm00].

11.5.3 Erstellung von Shops

Nach der Anmeldung erscheint ein Fenster zur Auswahl von Shops, die in SmartStore auch Mandanten genannt werden. Unter *Verwalten* > *Neuen Mandant anlegen* kann ein neuer Shop erstellt werden. Wenn der Portal-Shop neuinstalliert werden muss, muss der Mandant *ipsishop* heißen. Nach diesem Schritt kann man sich bei dem neuen Mandanten anmelden. Als erstes wird in der Vorlagenauswahl nach einer Vorlage gefragt. Für den IPSI-Shop sollte die benutzerdefinierte Vorlage gewählt werden. Nach der Übernahme der gewählten Vorlage werden die entsprechenden HTML- und ASP-Seiten kopiert und ein virtuelles Verzeichnis innerhalb des IIS erstellt. Beim Auftreten von Problemen kann das virtuelle Verzeichnis auch manuell eingerichtet werden. Das geschieht im IIS unter erweiterte Optionen, wobei das virtuelle Verzeichnis *ipsishop* heißen muss und standardmäßig auf das Verzeichnis C:\Programme\SmartStore Standard Edition\Stores\<ipsishop_stage> gerichtet ist. Um festzustellen, ob die Installationen erfolgreich waren, sollte der Browser durch Eingabe von http://localhost/ipsishop den Benutzer zur Startseite des Shops führen.

11.5.4 Wiederherstellung des IPSI-Shop

Nach der Erstellung des Mandanten *ipsishop* wird einerseits die Datenbank *ipsishop.mdb* angelegt, die unter dem Verzeichnis *C:\Programme\SmartStore Standard Edition* zu finden ist, und andererseits wird das Verzeichnis *ipsihop-stage* erzeugt, das alle HTML-, ASP- und Konfigurationsdateien beinhaltet. Dieses Verzeichnis befindet sich unter C:\Programme\SmartStore Standard Edition\Stores. Zur Wiederherstellung des Shops müssen die o.g. Datenbank und die Verzeichnisse durch die Dateien, die sich im CVS befinden, ersetzt werden. Darüber hinaus muss die Datei check.asp im Root-Verzeichnis von IIS (C:\inetpub\wwwroot) kopiert werden [HoSu00]. Diese Datei hat die Funktion, die erhaltene *UserID* in ein Cookie zu schreiben und die entsprechenden Seiten im Shop aufzurufen, wo das Cookie zur Identifikation des Benutzers gelesen wird. Somit gilt diese Datei als Verbindung zwischen Portal und Shopsystem [Ai97]. Wenn lediglich die UserID übergeben wird, ruft diese Datei die Eingangsseite des Shops auf. Bei zusätzlicher Übergabe einer Artikelnummer wird die Artikeldetailseite des Shops aufgerufen.

11.5.5 Start des Subsystems Procurement

Aus den schon genannten Gründen, die im Kapitel 8.3 beschrieben wurden, sollte die Subsystem-Fassadenklasse ShopSubsystemFacadeImpl.class auf dem Shopserver gestartet werden. Dazu wird eine Netzlaufwerkverbindung mit dem Verzeichnis *runtime* auf dem IPSI-Server benötigt. Wenn beispielsweise diesem Netzlaufwerk der Buchstabe R zugeordnet ist, sieht der Startbefehl für die Subsystemfasade wie folgt aus:

```
JAVA -Djava.rmi.server.codebase=file:///r:/lib/ipsi.jar -
DSubsystemManagerNameserverAddress=192.168.10.1 -DRMINameServer=192.168.10.1
-cp r:\lib\ipsi.jar de.ipsi.subsystem.shop.ShopSubsystemFacadeImpl
```

Dabei ist ipsi.jar die komprimierte Fassung aller Portalklassen und 192.168.10.1 die interne IP-Adresse des IPSI-Servers. Während der Implementierung des Subsystems wurde festgestellt, dass das JAVA Development Kit (JDK) in der Version 1.2 Probleme mit dem Zugriff auf den ODBC-Treiber von MS Access hatte. Diese Probleme traten aber bei einem Update auf die Version 1.3 des JDK nicht mehr auf, so dass für einen einwandfreien Verlauf des Subsystems die Nutzung des JDK 1.3 notwendig wurde.

11.5.6 Anmerkungen

Für die Realisierung von Anforderungen an das Subsystem Procurement wurden viele Änderungen an der ursprünglichen Shop-Datenbank und an den ASP-Skripten vorgenommen [HoSu00] [Ai97]. Alle geänderte Daten befinden sich in dem internen Versionierungssystem CVS. Von daher sollte die Verwaltung von Shop-Benutzern im Portal-Administrationsbereich erfolgen. Bei der Neuaufnahme oder Änderung von Benutzerdaten werden Datensätze sowohl in der Portal-Datenbank als auch in der Shop-Datenbank aktualisiert. Dabei werden Attribute wie beispielsweise Budgetzeitraum in der Shop-Datenbank eingetragen, die ursprünglich nicht in SmartStore vorhanden waren. Die Produkte müssen aber im Gegensatz zum Benutzer in dem Client-Programm von SmartStore verwaltet werden. Das Clientprogramm ist selbsterklärend und beinhaltet spezielle Hilfethemen zur Anwendung.

11.6 Der Server

11.6.1 Allgemeines

Die beschriebenen Verfahren und Anleitungen in diesem Abschnitt beziehen sich auf ein lauffähig installiertes Linux-System. Insbesondere gehören bestimmte Tools und Programme zur Standardinstallation eines Unix-Systems, wie z.B. find, make, cat, usw. Deren Vorhandensein und Funktionsfähigkeit werden in diesem Kapitel und zur Funktion der vom IPSI-Entwicklerteam erstellten Skripten und Programme stillschweigend vorausgesetzt. Weitere Nicht-Standard-Tools und –Programme, die zur Ausführung oder Erstellung von IPSI notwendig sind, werden hier explizit aufgeführt.

11.6.2 Netztopologie

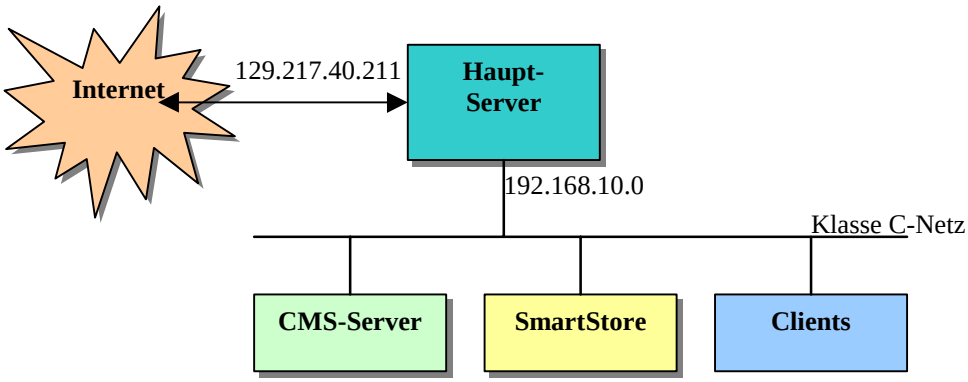


Abbildung 62 - Architektur des Netzwerks

Das Netzwerk besteht aus einem Haupt-Server, der typische Serverdienste wie Emailserver, Fileserver, Webserver bereitstellt, aber auch als Domänencontroller für die Windows-NT-Clients arbeitet (s. Abbildung 62). Er verwaltet außerdem das gesamte Sourcecode-Repository. Weiterhin laufen die meisten der Serverprozesse des IPSI-Portals auf diesem Rechner. Als weitere Server befinden sich ein Windows-NT4-Server mit dem Contentmanagementsystem (CMS) Pirobase [Pi00] und eine Windows-NT4-Workstation mit dem Shopsystem SmartStore [Sm00] im Netz.

Als Betriebssystem des Servers wurde Linux [Li00] (SuSE-Linux 6.3 [Su00b]) gewählt.

In den weiteren Abschnitten dieses Kapitels werden die Dienste des Hauptservers – im folgenden einfach mit Server bezeichnet – beschrieben.

11.6.3 CVS

Der erstellte Sourcecode wird in einem Source-Repository gespeichert und verwaltet. IPSI verwendet dazu das CVS-System [Cv00]. Das Repository ist unter /home/cvsipsi abgelegt.

Hinweis: Um mit dem CVS-Client sinnvoll zu arbeiten, sollte die Umgebungsvariable *CVSROOT* auf /home/cvsipsi gesetzt sein. Das ist insbesondere für die Erstellung der Runtime-Umgebung nötig.

Beim Auschecken der Sourcen werden Verzeichnisse angelegt, die der Package-Struktur der Sourcen entsprechen. Die Verzeichnisse sind in der zweiten Spalte angegeben. Es existieren folgende Module im Source-Repository:

Modulname	ausgecheckt nach	Inhalt
Administration	de/ipsi/subsystem/administration	Administrations-Subsystem
CMS	de/ipsi/subsystem/cms	CMS-Subsystem
Communication	de/ipsi/subsystem/Communication	Communication-Subsystem
Controller	de/ipsi/controller	Controller
Core	de/ipsi/core	Core-System
Database	./database	Skripten zur Initialisierung der IPSI-Benutzer-Datenbank
Formatter	de/ipsi/formatter	Formatter
Guibase	de/ipsi/guibase	GUIbase-Library
Guilib	de/ipsi/guilib	GUIlib-Library
Legacy	de/ipsi/subsystem/legacy	Legacy-Subsystem
Office	de/ipsi/subsystem/office	Office-Subsystem inkl. der C++-Sourcen der Office.dll
Runtime	./	Das Makefile zur Erzeugung der kompletten IPSI-Runtimeumgebung und die für IPSI nötigen fremden jar-Libraries
Search	de/ipsi/subsystem/search	Search-Subsystem

Shop	de/ipsi/subsystem/shop	Shop-Subsystem
SmartStore	./smartstore	Modifizierte ASP-Skripte und die modifizierte SmartStore-Datenbank mit den Anpassungen zur Integration mit IPSI[1]
Testdaten	./testdaten	Testdaten des IPSI-Portals[2]

11.6.4 Samba

Samba [Sa00] ist ein Softwarepaket unter Unix, das einen Windows-Server emuliert. Wie unter Windows 9x/NT/2000 können Verzeichnisse eines Filesystems und Drucker als Freigaben in die Windows-Welt exportiert werden. Weiterhin kann ein Samba-Server als Domänencontroller in einem Windows-NT-Netzwerk fungieren.

User-, Entwicklungs- und Runtime-Umgebungsverzeichisse werden vom Samba-Server in das Windows-Netzwerk exportiert. Es ist ein Samba-Server der Version 2.0.5a installiert. Weiterhin dient der Samba-Server im lokalen IPSI-Netzwerk den angeschlossenen Windows-NT-Clients als Domänencontroller. Alle Benutzeraccounts und Profile für das Windows-NT-Netzwerk werden vom Samba-Server gespeichert und verwaltet. Die Konfiguration ist in /etc/smb.conf gespeichert.

Es existieren folgende Freigaben für das Windows-NT-Netzwerk:

Name	Inhalt
profiles	Die Windows-NT-Benutzerprofile und Desktops
netlogon	Nötig für Benutzeranmeldungen
IPSI	Benutzerübergreifendes Projektverzeichnis für alle IPSI-Entwickler
homes	Für jedes IPSI-Teammitglied stellt Samba ein Homeverzeichnis bereit
Webmaster	Zugriff auf Dokumente des Webauftritts für den Webmaster
Runtime	Enthält die IPSI-Runtime-Umgebung
ljet4ks	Die Freigabe des Netzdruckers

11.6.5 Sendmail

Sendmail [Se00] wird für IPSI nur als SMTP-Server verwendet und dient dem Communication-Subsystem als Grundlage zum Versenden der Emails. Es wird eine Standardinstallation durchgeführt, wie sie von der Linux-Distribution vorgesehen ist. Die Konfiguration wird an das jeweilige Netzwerk angepasst. Die Konfigurationsfiles sind /etc/sendmail.cf und /etc/mail/*.

11.6.6 JAVA-Installation

Die IPSI-Subsysteme verwenden das Sun JAVA2-JDK [Su00] in der Version 1.2.2. Das JDK ist auf dem Server unter /usr/local/JAVA1.2.2 installiert. Ein Link /usr/local/JAVA zeigt auf das Installationsverzeichnis. Der systemweite PATH verwendet den Link auf das JDK, um die JAVA-Tools aufzunehmen. Durch Änderung des Links kann das gesamte System auf eine neue JDK-Version umgestellt werden.

Die Windows-Clients benötigen ebenfalls eine JAVA-Umgebung. Hier genügt es, das JAVA2 JRE in der Version 1.2.2 zu installieren. Lediglich der SmartStore-Rechner benötigt eine JDK1.3-Installation, weil die virtuelle Maschine des JDK1.2.2 nicht mit den verwendeten ODBC-Treibern der Datenbankanbindung zusammen läuft.

[1] Die Datenbank beinhaltet auch die Shop-Testdaten.
[2] Die Testdaten des Shopsystems sind in der Shop-Datenbank enthalten, die im CVS-Modul *smartstore* enthalten ist.

11.6.7 Firewall und Routing

Der Linux-Server hat als einziger Rechner im Netz eine offizielle IP-Adresse und kann damit aus dem Internet erreicht werden. Alle weiteren Rechner im lokalen Netz besitzen lokale Klasse-C-Adressen. Über IP-Masquerading erhalten sie vollen Zugriff auf das Internet.

Da der Server im Verbund des Universitätsrechenzentrums liegt und bereits durch mehrstufige Firewalls geschützt ist, wird keine eigene weitere Firewall eingesetzt.

Durch die Notwendigkeit, den Pirobase-Server vom Internet erreichen zu können, wurde ein Port-Forwarding implementiert: Verbindungen, die auf dem Port 1024 des Servers ankommen, werden auf eine interne (lokale) Adresse umgeleitet.

Die Skripten zur Konfiguration von Routing und Firewall sind unter /sbin/init.d/firewall und /sbin/init.d/route zu finden.

11.6.8 Webserver und Servlet-Engine

Als Webserver kommt der Apache-Server [Ap00] in der Version 1.3.6 zum Einsatz. Die Konfiguration liegt unter /etc/httpd, die Webdokumente unter /usr/local/httpd.

Als Servlet-Engine ist *JServ* [Ja00] in der Version 1.1b1 installiert. Die Engine wird als Apache-Modul in das Verzeichnis /usr/lib/apache installiert, Dokumentation findet sich unter /usr/local/jserv. Der Classpath, aus dem die Servlets geladen werden, ist so definiert, dass die Klassen aus der IPSI-Runtime-Library (ipsi.jar) geladen werden. Wird die referenzierte Klasse dort nicht gefunden, sucht die virtuelle Maschine im Verzeichnis /usr/local/httpd/servlets danach. Der Classpath wird in der Servlet-Konfiguration definiert, die sich unter /etc/httpd/jserv findet.

Beim Start einer JAVA-Applikation werden der Main-Methode die Startparameter im *args*-Array übergeben. Da dies bei Servlets nicht möglich ist, werden die Startparameter in einer eigenen Datei gespeichert, die von der Servlet-Engine bei der Initialisierung des Servlets ausgelesen und dem Servlet in einem Objekt zur Verfügung gestellt wird. Die Datei mit den Startparametern hat den gleichen Namen und Pfad wie die class-Datei des Servlets, erhält jedoch die Endung *.initArgs*. IPSI verwendet für sein HTTP-Servlet eine Startparameter-Datei, die sich zusammen mit der class-Datei in der ipsi.jar-Library befindet.

11.6.9 Runtime-Umgebung

Alle getesteten und freigegebenen Quellen werden in einer Runtime-Umgebung zusammengefasst. Diese Umgebung soll allen Entwicklern und den Produktivsystemen als Quelle für Klassen und Libraries dienen. So ist sichergestellt, dass die Entwickler nicht mit alten Libraries und Klassen arbeiten und daraus schwer zu findende Fehler und Inkompatibilitäten entstehen.

Diejenigen Systeme, die auf dem Server laufen, greifen direkt über das Filesystem auf die Klassen und Libraries zu, Windows-Rechner im lokalen Netzwerk bekommen über den Samba-Server eine NT-Freigabe, die Lesezugriff auf alle Inhalte der Runtime-Umgebung gestattet.

Die Verzeichnisse der Umgebung gehören der Gruppe *ipsi* und sind i.A. gruppenschreibbar, d.h. alle Mitglieder der Benutzergruppe ipsi dürfen in diesen Verzeichnissen lesen und schreiben, um ein neues Build aus den aktuellen Sourcen zu erstellen[3], etc. Die Umgebung befindet sich z.Zt. unter /home2/runtime (im Folgenden mit „<runtime>„ bezeichnet) und enthält folgende Unterverzeichnisse/Inhalte:

[3] In der NT-Freigabe wird die Runtime-Umgebung aber read-only exportiert um „Unfälle“ und unbeabsichtigtes Überschreiben der Umgebung zu verhindern.

- Makefile
  Mit Hilfe des Makefiles wird die Runtime-Umgebung erstellt
- bin/
  Enthält Skripten zum starten und stoppen der Subsysteme und Skripten zum Erzeugen der Runtime-Umgebung
- classes/
  Enthält die class-Dateien und die daraus erstellten Stubs und Skeletons der Remote-Objekte
- class-override/
  Der Classpath der Subsysteme ist so gesetzt, dass er alle benötigten Klassen aus Libraries (jar-Files) lädt. Klassen, die in diesem Verzeichnis liegen, werden bevorzugt geladen. Gleichnamige Klassen der Libraries, werden also „überladen“. Das Verzeichnis kann verwendet werden, um das System zu patchen, ohne ein neues Build zu erstellen
- distribute/
  In diesem Verzeichnis werden die (automatisch) erstellten Archive (zip oder tgz) der Runtime-Umgebung abgelegt. Mit Hilfe der Archive kann die Runtime-Umgebung leicht an einen anderen Ort transferiert werden, z.B. für die Arbeit an einem anderen Ort
- doc/
  Hier liegt die komplette JAVAdoc-Dokumentation aller JAVA-Klassen des IPSI-Portals
- lib/
  Hier findet sich die IPSI-Library (ipsi.jar) und alle Libraries, von denen die IPSI-Klassen abhängen.
- logs/
  Werden die Subsysteme gestartet, wird deren Standardausgabe in Logfiles geleitet, die sich in diesem Verzeichnis befinden
- makelogs/
  Logfiles, die bei der Erstellung der Runtime-Umgebung entstehen (Compilerausgaben, etc.), sind hier abgelegt
- pids/
  Die Startskripten der Subsysteme merken sich in diesem Verzeichnis die PID (Prozess-ID) der gestarteten Prozesse
- source/
  In diesem Verzeichnis werden die Sourcen aus dem CVS ausgecheckt
- tmp/
  Kann beliebig verwendet werden

Um eine IPSI-Runtime-Umgebung in einem leeren Verzeichnis (oder auf einem neuen Server) zu erstellen, muss zuerst das Modul *runtime* aus dem CVS ausgecheckt werden. Anschließend wird in diesem Verzeichnis das Kommando *make init* aufgerufen, wodurch die nötigen Verzeichnisse erstellt und deren Dateirechte gesetzt werden. Zuletzt wird mit *make all* die Erstellung der IPSI-Library durchgeführt. Das Ergebnis der Prozedur ist die ipsi.jar-Library, die alle JAVA-Klassen des IPSI-Portals enthält[4].

11.6.9.1 Sourcen, Release-Tags und Builds

Zu bestimmten strategischen Zeitpunkten werden alle Sourcen, die sich im Source-Repository (CVS) befinden mit Release-Tags versehen. Dadurch entsteht ein reproduzierbarer Zustand (Schnappschuß) des Gesamtsystems. Als Tag wird jeweils „build<Nr.>„ verwendet, wobei <Nr.> eine fortlaufende natürliche Zahl beginnend mit 1 ist. Diese Aktion wird als *Codefreeze* bezeichnet. Nach jedem Codefreeze wird auch die Runtime-Umgebung neu erstellt. Das Erzeugen der Umgebung kann unter dem Benutzeraccount jedes Mitglieds der Gruppe *ipsi* durchgeführt werden.

11.6.9.2 Makefile

Das Makefile (in Verbindung mit den Unix Make-Tool) automatisiert die Erstellung der Runtime-Umgebung. Das Makefile kann durch verschiedene Variablen im Kopf der Datei an andere Umgebungen angepasst werden. Die Variablen sind selbsterklärend. Die beiden wichtigsten Variablen sind MODULES und RELEASE: MODULES definiert eine Liste von CVS-Modulen, aus denen die Aktionen (checkout, compile) die Information bezie-

[4] Zusätzlich zu dieser Library sind die jar-Libraries aus dem lib-Verzeichnis zum Betrieb von IPSI nötig. Für das Office-Subsystem ist weiterhin die Office.dll-Windows-Library zum Betrieb nötig.

hen, welche Module bearbeitet werden sollen. RELEASE gibt an, welche Version der Sourcen aus dem Repository ausgecheckt werden soll.

Make wird im Verzeichnis <runtime> aufgerufen und kann folgende Kommandos erhalten:

- init
  Im aktuellen Verzeichnis werden die für die Runtime-Umgebung nötige Verzeichnisstruktur angelegt und die erforderlichen Dateirechte gesetzt. Dieses Kommando wird i.A. nur bei einer Neuerstellung der Runtime-Umgebung aufgerufen.[5] Es kann aber auch verwendet werden, um die Dateirechte der Verzeichnisstruktur zu korrigieren.
- all
  Führt der Reihe nach die Aktionen *clean, checkout, compile, stubs, ipsijar, link, JAVAdoc, tgz* aus
- clean
  Löscht alle Sourcen, Classfiles, Dokumentation und das Verzeichnis *class-override*
- checkout
  Checkt die Sourcen aus dem CVS aus. Es werden alle Module ausgecheckt, die in der Variablen MODULES definiert sind und das Release-Tag haben, das durch die Variable RELEASE bestimmt wird[6]. Alle Sourcen werden in das durch die Variable SRCDIR definierte Verzeichnis ausgecheckt, i.A. <runtime>/source. Es wird ein Logfile erstellt.
- compile
  Kompiliert alle Sourcen, die im Baum <runtime>/source liegen. Es wird ein Logfile erstellt.
- stubs
  Zu allen RMI Remote-Klassen werden die Stubs und Skeletons erstellt. Es wird ein Logfile angelegt.
- ipsijar
  Alle Klassen im Verzeichnis <runtime>/classes werden zu einem jar-Archiv ipsi-<Release-Tag> im Verzeichnis <runtime>/lib zusammengefasst.
- link
  Erstellt einen Softlink <runtime>/lib/ipsi.jar, der auf das jar-File <runtime>/lib/ipsi-<Release-Tag>.jar zeigt. Durch diesen Link ist es einfacher, das Gesamtsystem zu einem anderen Build umzustellen, wenn alle Benutzer der IPSI-Library den Link auf ipsi.jar verwenden. Durch Änderung des Links verwenden alle Systeme gemeinsam ein anderes Build.
- JAVAdoc
  Erstellt im Verzeichnis <runtime>/doc die gesamte JAVAdoc-API der im Source-Baum enthaltenen Sourcen.
- tgz
  Erstellt ein tgz-File (gezipptes tar-File) mit den wichtigsten Komponenten der Runtime-Umgebung: Classes, Sources, JAVAdoc, ipsi.jar. Fremde Libraries werden nicht eingepackt, weil sie sich i.A. nicht zwischen zwei Builds ändern.
- zip
  Wie tgz, erstellt aber ein ZIP-File.

11.6.9.3 bin-Verzeichnis

- rc*-Skripte
  Die rc*-Skripte starten ein Subsystem. Die Skripten verstehen die Parameter *start* und *stop*. Damit wird ein Subsystem gestartet oder gestoppt. Diverse Parameter, wie der Classpath, Steuerparameter für die virtuelle Maschine, Startparameter für das Subsystem, etc. werden im Kopf des jeweiligen rc-Skripts eingestellt. Die Ausgaben der Prozesse werden in das jeweilige Logfile im Verzeichnis <runtime>/logs protokolliert.
  Das Starten der Prozesse muss von keinem bestimmten User durchgeführt werden. Die Prozesse können also unter jedem Benutzeraccount mit normalen Benutzerrechten laufen. Damit in einer Entwicklergruppe die Prozesse auch von anderen Entwicklern (neu) gestartet und gestoppt werden können, wurde der Benutzer *ipsirun* eingeführt. Alle IPSI-relevanten Prozesse sollten unter diesem Account gestartet werden.
- .bat-Dateien
  Diese Dateien sind Batchfiles für Windows-Computer. Sie erfüllen für die unter Windows-NT laufenden Subsysteme den gleichen Zweck wie die rc-Skripte: Sie starten ein Subsystem. Die Ausgaben werden hier aber nicht in einem Logfile protokolliert[7].

---

[5] Um eine neue Umgebung zu erstellen, wird i.A. in einem leeren Verzeichnis das Modul *runtime* aus dem CVS ausgecheckt und dann in diesem Verzeichnis *make init* aufgerufen, um die Verzeichnisstruktur zu erstellen. Anschließend kann mit *make all* die IPSI-Library erzeugt werden.

[6] Wenn RELEASE mit dem Release-Tag *HEAD* belegt wird, werden die aktuellsten Sourcen aus dem Repository ausgecheckt.

[7] Da die Runtime-Umgebung und damit die Batchfiles über eine Freigabe von den Windows-Rechnern erreichbar ist, müssen die Batchfiles zum Start nicht in ein lokales Verzeichnis kopiert werden, sondern können direkt

- rcAdministration
  Startet und stoppt das Administrations-Subsystem.
- rcCommunication
  Startet und stoppt das Communications-Subsystem.
- rcDispatcher
  Startet und stoppt den serverseitigen Dispatcher[8].
- rcGeneric
  Dieses Hilfsskript beinhaltet die Funktionalität für die anderen rc-Skripte. Es wird nicht selbst aufgerufen.
- rcLegacy
  Startet und stoppt das Legacy-Subsystem.
- rcSubsystemManager
  Startet und stoppt den Subsystem-Manager.
- rcTemplate
  Dieses Skript kann als Vorlage für ein neues rc-Skript für ein Subsystem verwendet werden.
- runOfficeSubsystem.bat
  Dieses Batchfile startet die Subsystem-Fassade für ein Office-(Outlook-)Subsystem. Der Benutzer, unter dem sich IPSI beim lokalen Outlook anmeldet, ist im Batchfile enthalten und muss ggf. angepasst werden.
- runShopSubsystem.bat
  Dieses Batchfile startet die Subsystem-Fassade des Shop-(SmartStore-)Servers. Es muss vom Windows-Server aus aufgerufen werden, auf dem das Shopsystem installiert ist.
- setBuildTag
  Dieses Skript versieht eine Liste von CVS-Modulen mit einem Release-Tag, welches dem Skript als Parameter übergeben wird. Die Modulliste wird im Kopf des Skripts eingetragen.
- showProcesses
  Zeigt alle IPSI-relevanten Systemprozesse an. Es kann verwendet werden, um den Status eines Subsystems zu überprüfen. Weil die Ausgabe des Skriptes sehr lang ist, sollte die zum Programm *less* gepiped werden: showProcess | o -S
- startAll
  Startet alle Subsysteme.
- stopAll
  Stoppt alle Subsysteme.

11.6.9.4 lib-Verzeichnis

In diesem Verzeichnis befinden sich die Libraries, von denen die IPSI-Portale abhängen, sowie die IPSI-Library selbst. Die Subsysteme binden diese Libraries in den jeweiligen Classpath des Subsystems ein, auf Windows-Systemen kann über die Runtime-Freigabe auf die Libraries zugegriffen werden.

Es befinden sich folgende Libraries in diesem Verzeichnis:

Name	Version	Zweck
Activation.jar	1.0.1	Wird für die JAVA-Email-API benötigt
ipsi.jar	Build<nr>	Enthält allen JAVA-Code, der für den Betrieb von IPSI nötig ist.
jaxp.jar	1.01	JAVA API for XML Parsing. API zum Parsen von XML-Dokumenten
parser.jar	1.01	JAVA API for XML Parsing. API zum Parsen von XML-Dokumenten
mail.jar	1.1.3	JAVA Email-API. API zur Unterstützung von Email-Anwendungen in JAVA
mysql_comp.jar	1.2c	JDBC-Treiber für die mySQL-Datenbank
Office.dll	Build<nr>	JAVA-Native-Interface-Bibliothek für die Anbindung des IPSI-Portals an Microsoft Outlook 2000

aus der Freigabe gestartet werden. Bei einer Fehlfunktion sollte die PATH-Variable auf korrekte Werte überprüft werden.

[8] Es kann auch ein Dispatcher auf jedem Client-Rechner gestartet werden. Dieses Feature ist nur zu Entwicklungs- und Debugzwecken nötig.

12 Zusammenfassung und Ausblick

Dieser Abschnitt soll, nachdem die PG abgeschlossen wurde, eine kurze rückblickende Zusammenfassung der Projektgruppe sowie einen zukunftsgerichteten Ausblick geben. Im Gegensatz zu den restlichen Kapiteln muss sich dieser Abschnitt daher durch einen eher subjektiven Charakter kennzeichnen lassen.

Zusammenfassend lässt sich die Arbeit der PG als erfolgreich bezeichnen. Diese Aussage kann durch folgende Teilaspekte begründet werden:

- der wichtigste und größte Teil der Anforderungen, die am Anfang der PG definiert wurden, wurde auch in ein lauffähiges System umgesetzt (s. Kapitel 6 Anforderungsanalyse)
- die Vorgehensweise war (wenn auch nicht immer formal und strikt im Voraus definiert) in großen Bereichen systematisch durch die begleitende Definition des PG-Prozesses und weitergehende Planung des Vorgehens im Project Management Board. Dies zeigt sich nicht zuletzt darin, dass Entscheidungen durchweg auf eine solide Informationsbasis gestellt wurden und selten „aus dem Bauch heraus“ gefällt wurden (z.B. bei der Erstellung von verschiedenen Machbarkeitsprototypen).
- es bestand eine gute Mischung zwischen den Zielen einer Lehrveranstaltung und den Bedingungen, wie man sie bei der realen Softwareentwicklung vorfindet. So haben alle PG-Teilnehmer ausnahmslos in ihren Rollen neue Kenntnisse und Erfahrungen erworben, die von der Projektplanerstellung, dem Gestalten von Meetings über das Halten von Vorträgen bis hin zu so technischen Kenntnissen wie der Benutzung von CORBA und RMI oder des JNI-APIs reichen und sicherlich über das Ende der PG hinweg von Nutzen sein werden. Zudem mussten alle diese Kenntnisse schließlich in ein fertiges Produkt einfließen, so dass keineswegs von einem wissenschaftlichen „herumprobieren“ auf einem zu hohen Abstraktionsniveau gesprochen werden kann. Hierfür steht das lauffähige System des IPSI-Portals als Beweis.
- nicht zuletzt war die Zusammenarbeit aller PG-Teilnehmer sehr gut; es kam nur selten zu Differenzen, die dann auch nicht langfristig waren; die Argumente und Diskussionen waren durchweg sachlich geprägt. Auch außerhalb der PG bestand zwischen vielen PG-Teilnehmern ein freundschaftliches Verhältnis.

Trotz der insgesamt überaus positiven Beurteilung des Verlaufes der Projektgruppe sollen nicht die mehr oder weniger kleinen Unwägbarkeiten verschwiegen bleiben, die untrennbar mit einem solchen langfristigen Projekt verbunden sind – nicht zuletzt um nachfolgenden Projektgruppen hilfreiche Anregungen und Vorbereitungen geben zu können. So war die gesamte PG durch viele abwechselnde Phasen von Hochs und Tiefs gekennzeichnet, die sich in der Stimmung der PG-Teilnehmer widerspiegelten. So erschien das gesamte Vorhaben zu bestimmten Phasen (insbesondere bei nicht funktionierenden Prototypen und Detailproblemen wie bei der Benutzung von RMI oder scheinbar nichtdeterministischen Problemen bei der Outlook-Anbindung) schlicht nicht oder nur mit größten Einschränkungen machbar. Eine dieser Phasen war schon direkt nach der Anforderungsanalyse eingetreten, die nach Meinung der PG-Teilnehmern mit über drei Monaten zu lange ausfiel. Rückblickend haben sich jedoch die langen Planungszeiten (insbesondere im OO-Design) durchweg ausgezahlt und resultierten in einer sauberen Systemarchitektur. Eine Empfehlung an nachfolgende PGs besteht somit auch gerade darin, zunächst detaillierte Planungsphasen vorzusehen, in denen dann die getroffenen Entscheidungen z.B. durch Prototypen realisiert werden – trotz der Verlockung, schnell mit der Programmierung anfangen zu wollen.

Wichtig schien ebenfalls die Schaffung einer gemeinsamen von allen PG-Teilnehmern getragenen Vision – sowohl die Fachlichkeit als auch die technische Systemarchitektur betreffend. Hierzu waren insbesondere Initiativen und Beiträge von einzelnen Teilnehmern von Bedeutung, die durch Präsentationen in den PG-Sitzungen auf eine breite Zustimmungsbasis gestellt wurden. Insgesamt lebt eine PG allerdings von dem Engagement aller Teilnehmer.

Die vielen Erfolgsfaktoren (die sicherlich auch (noch) nicht durch die PG IPSI identifiziert werden konnten) können hier nicht in ihrer gesamten Breite dargelegt werden. Die zuvor genannten sollen als wichtige Faktoren stellvertretend dienen.

Mit Blick in die Zukunft bleibt zu hoffen, dass die Arbeit und das Ergebnis der PG noch weitere positive Anstöße geben kann. Sei es bei der Vergabe von Diplomarbeitsthemen oder den Veröffentlichungen über interessante Aspekte wie der Architektur oder des Prozesses. Themen hierzu bietet die PG IPSI sicherlich hinreichend genug, sowohl in quantitativer als auch in qualitativer Hinsicht. Die Grundlage hierfür konnte dadurch geschaffen werden, dass das gesamte IPSI-Portalsystem bei einem der Partnerunternehmen auch nach Beendigung der PG weiterhin gehostet und somit lauffähig gehalten wird. Der Zugriff auf das System bleibt somit dauerhaft gewährleistet.

Ebenso bleibt zu hoffen, dass die freundschaftlichen Beziehungen, die während der PG entstanden sind, über das Ende der PG hinaus bestehen. „Post-Mortem“-Treffen werden das Ihre hierzu beitragen.

Allen nachfolgenden PGs wünscht das gesamte IPSI-Team genau so viel Spaß und Erfolg bei der Projektgruppenarbeit!

13 Danksagungen

Die Durchführung und der erfolgreiche Abschluss des Projektes wäre nicht in dieser Form möglich gewesen ohne die Mithilfe von Partnerunternehmen, die die Projektgruppe in den verschiedenen Phasen unterstützt haben. Ihnen gilt unser besonderer Dank.

Im Einzelnen waren dies die folgenden Unternehmen (in alphabetischer Reihenfolge):

adesso AG, Dortmund
Agrippina Versicherung, Dortmund
Albingia Versicherung, Dortmund
Barmenia Versicherungen, Dortmund
Continentale Versicherung, Dortmund
LVM Versicherungen, Münster
Microsoft GmbH, Unterschleißheim
Netscape GmbH, München
Oracle Deutschland GmbH, München
Pironet AG, Köln
Signal Iduna, Dortmund
SmartStore AG, Dortmund
Volkswohl Bund Versicherungen, Dortmund

14 Literatur

[Ai97] P. Aitken: „JavaScript und VBScript“, MITP Verlag Bonn, 1997

[Al96] P. Alpar: „Kommerzielle Nutzung des Internets“, Springer, 1996

[Ap00] Apache Organisation, http://www.apache.org, (zitiert: August 2000)

[Ba82] H: Balzert: “Lehrbuch der Software-Technik: Software-Entwicklung“, Spektrum Akademischer Verlag, 1996

[Ba98] H. Balzert: „Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung“, Spektrum Akademischer Verlag, 1998

[BoUl99] M. Böhnlein, A. Ulbrich vom Ende: „XML – Extensible Markup Language“, In: Wirtschaftsinformatik, Hrsg.: Verlag Vieweg (GWV Verlagsgesellschaft mbH) 41/1999

[CaPi94] L. Catledge, J. Pitkow: „Characterizing Browsing Strategies in the World-Wide Web”, http://www.igd.fhg.de/www/www95/papers/80/userpatterns/UserPatterns.Paper4.formatted.html (zitiert: Februar 2000)

[Co00] Continentale: Die Continentale – Mehr als eine Versicherung. http://www.continentale.de (zitiert: August 2000)

[CuKe92] B. Curtis, M. I. Kellner: „Process Modeling“, In: Communications of the ACM, 35(9), September 1992

[Cv00] CVS Organisation, http://www.cvshome.org, (zitiert: August 2000)

[De00] Deja News Usenetsuche, http://www.deja.com/usenet, (zitiert: August 2000)

[De00a] developer.com: „The Leading Source for Technical Information by Earth Web“, http://www.developer.com, (zitiert: Februar 2000)

[De95] M. Deutsch: „Electronic-Commerce“, Vieweg Verlag, 1995

[Di88] DIN 66234, Teil 8: Bildschirmarbeitsplätze: Grundsätze der Dialoggestaltung. Auszüge verfügbar unter http://www.ergo.uni-bremen.de/normen/auszug.html (zitiert: Februar 2000)

[DoDi99] E. Doberkat, S. Dißmann: „Einführung in die objektorientierte Programmierung mit JAVA“, München, 1998

[EcOr98] Electronic-Commerce Info NRW – ECIN http://www.electronic-Commerce.org (zitiert: Februar 2000)

[Fl96] D. Flanagan: „JAVA in a nutshell“, O’Reilly, 1996

[GaHe96] E. Gamma, R. Helm, R. Johnson, J. Vlissides: „Entwurfsmuster – Elemente wiederverwendbarer

objektorientierte Software“, Addison Wesley, 1996
[GoJo97] J. Gosling, B. Joy, G. Steele: „JAVA - Die Sprachspezifikation“, Addison-Wesley, 1997
[Hi97] S. Hillier: „Active Server Pages Programmierung“, Microsoft Press, 1997
[HoSu00] A. Homer, D. Sussman: „ASP 3.0 professionell. Active Server Pages“, MITP Verlag Bonn, 2000
[Ib90] IBM Corp., Systems Application Architecture: „Common User Access: Advanced Interface Design Guide“, IBM Document SC26-4582-0, 1990
[Is93] ISO 9241, Teil 10. Auszüge verfügbar unter http://www.ergo.uni-bremen.de/normen/auszug.html (zitiert: Februar 2000)
[Ja00] The JAVA Apache Project, http://java.apache.org, (zitiert: August 2000)
[Ju99] F. Jung: „eXtensible Markup Language“, In: it-Fokus, Hrsg.: IT-Verlag, Höhenkirchen 02/2000 und 03/2000
[Kr97] D. J. Kruglinski: „Inside Visual C++ Version 5“, Microsoft Press, 1997
[La96] F. Lampe: „Business im Internet“, Vieweg 1996
[Li00] Linux Organisation, http://www.linux.org, (zitiert: August 2000)
[Mi00] Microsoft GmbH, http://www.microsoft.com/germany (zitiert: Februar 2000)
[Mi99] Microsoft GmbH, „Visual InterDev 6.0: Internet-Entwicklung mit ASP-Technologie“, http://www.microsoft.com/germany/msdn/vinterdev/asp.htm (zitiert: Dezember 1999)
[Ms00] MSDN Online, http://msdn.microsoft.com, 2000
[Ni83] J. Nievergelt: „Die Gestaltung der Mensch-Maschine-Schnittstelle“, In: Informatik-Fachbericht 72: Sprachen für Datenbanken, Hrsg.: J.W. Schmidt, Springer Verlag 1983
[Ni96] J. Nielsen: „Why Frames Suck (Most of the Time)“, Alertbox for December 1996. http://www.useit.com/alertbox/9612.html (zitiert: August 1999)
[Ni97] J. Nielsen: „How People Read on the Web“, Alertbox for October 1, 1997. http://www.useit.com/alertbox/9710a.html (zitiert: August 1999)
[Oe97] B. Oestereich: „Objektorientierte Softwareentwicklung mit der Unified modeling language“, Oldenbourg, 1997 (3. Auflage)
[OeHr99] B. Oestereich, P. Hruschka, N. Josuttis, H. Kocher, H. Krasemann, M. Reinhold: „Erfolgreich mit Objektorientierung – Vorgehensmodelle und Managementpraktiken für die objektorientierte Softwareentwicklung“, Oldenbourg, 1999
[oV00] Ohne Verfasser, http://ls4-www.informatik.uni-dortmund.de/PGB/info.html, (zitiert: Februar 2000)
[Pi00] Pironet AG, http://www.Pironet.de, (zitiert: August 2000)
[RoMe95] H. Rombach, M. Verlage „Directions in Software Process Research”, in: M. Zelkowitz (eds.) Advances in Computers Vol. 41, Academic Press, 1995
[Sa00] Samba Organisation, http://www.samba.org, (zitiert: August 2000)
[Sc91] H.-J. Schneider (Hrsg.): „Lexikon der Informatik und Datenverarbeitung“, Oldenbourg Verlag, 1991
[Se00] Sendmail Organisation, http://www.sendmail.org, (zitiert: August 2000)
[Sm00] SmartStore AG, http://www.smartstore.de, (zitiert: Februar 2000)
[St98] B. Stearns: „JAVA Native Interface in Suns JAVA Tutorial“, http://java.sun.com/docs/books/tutorial/native1.1/index.html, 1998
[StCh98] J. M. Stewart, R. Chandak: „Crash Test, IIS4“, MITP Verlag Bonn, 1998
[Su00] Sun Microsystems, „The Source for JAVA™ Technology by Sun“, http://java.sun..com (zitiert: Februar 2000)
[Su00a] Sun Microsystems, JAVAEmail™ API. http://java.sun.com/products/JAVAmail/index.html (zitiert: August 2000)
[Su00b] SuSE GmbH, http://www.suse.de, (zitiert: Februar 2000)
[Su00c] Sun Microsystems, „JAVA™ Technology & XML“, http://java.sun.com/xml (zitiert: August 2000)
[Su00d] Sun Microsystems, „JAVABeans™ Activation Framework (JAF)“. http://java.sun.com/products/JAVAbeans/glasgow/jaf.html (zitiert: August 2000)
[Su97] Sun Microsystems, „JAVA Native Interface Specification“, http://java.sun.com/products/jdk/1.2/docs/guide/jni/spec/jniTOC.doc.html, 1997
[Su98] Sun Microsystems, „JAVAEmail™ API Design Specification. Version 1.1, Revision 01“, August 1998. http://java.sun.com/products/JAVAmail/JAVAEmail-1.1.pdf (zitiert: Februar 2000)
[Th97] G. Thaller: „Der Individuelle Spftware-Prozess“, BHV Verlag, 1997
[Th98] G. Thaller: „SPICE – ISO 9001 und Software in der Zukunft“, BHV Verlag, 1998
[W300] W3C Architecture domain, „Extensible Markup Language (XML)“, http://www.w3.org/XML (zitiert: August 2000)

15 Anhang A – Leseanleitung zu den Prozessmodellen

Die Dokumentation der Prozesse bei der PG IPSI ist mit Hilfe des Ablauf-Modellierungstools LeuSmart entstanden. LeuSmart bietet die Möglichkeit, Ablaufmodelle zu erstellen, zu verändern und zu simulieren. Die Modellierung basiert auf FUNSOFT-Netzen, die aus folgenden Bestandteilen zusammengesetzt sind:

- Tätigkeiten (dargestellt durch Rechtecke)
- Informationsspeichern (dargestellt durch Kreise)
- Verbindungen (dargestellt durch Pfeile zwischen Tätigkeiten und Informationsspeichern)

Ein Ablaufmodell enthält die fachliche Reihenfolge des zu modellierenden Geschäftsprozesses. Dabei stellen Tätigkeiten die verschiedenen Aktivitäten innerhalb des Geschäftsprozesses dar. Die Informationsspeicher können die Informationen zwischen den Bearbeitungen durch Tätigkeiten speichern. Tätigkeiten und Informationsspeicher werden durch Verbindungen miteinander verknüpft, wodurch der fachliche Ablauf des Geschäftsprozesses festgelegt wird. Verfeinerungen sind Tätigkeiten, die es ermöglichen, einen zusammenhängenden Teil des Ablaufmodells auf einer eigenen Seite darzustellen, und tragen somit zur Übersichtlichkeit bei.

Abbildung 63 zeigt die benutzen Symbole.

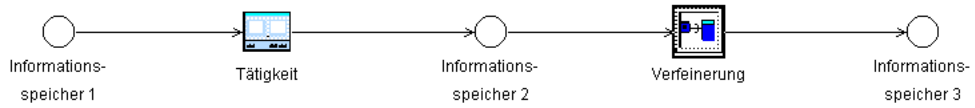


Abbildung 63 - Benutzte Symbole in einem beispielhaften Prozess

Das Teilmodell, das sich hinter der Verfeinerung verbirgt, wird separat dargestellt (s. Abbildung 64). Die Einbettung des Teilmodells in das Gesamtbild ergibt sich durch die vor- und nachgeschalteten Informationsspeicher.

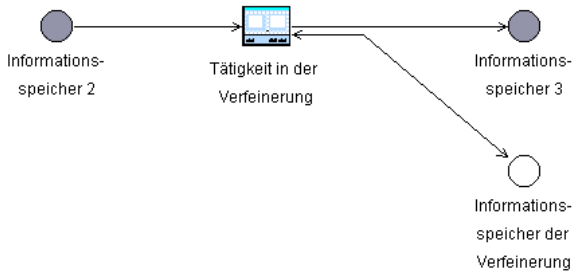


Abbildung 64 - Das Teilmodell zu der Verfeinerung aus Abb. 65

Sie sind mit identischen Namen (hier „Informationsspeicher 2“ und „Informationsspeicher 3“) sowohl auf der übergeordneten Ebene wie auch auf dem Bild der Verfeinerung zu sehen. Das Bild der Verfeinerung zeigt diese sogenannten „Konnektions-Kanäle“ grau unterlegt, um sie von anderen Speichern der Verfeinerung zu unterscheiden.

Neben den einfachen Informationsspeichern gibt es noch sogenannte Systeminformationsspeicher, die Schnittstellen zwischen verschiedenen Prozessen oder direkt zum Kunden darstellen. Das bedeutet, dass an diesen Stellen im Prozess Daten mit anderen Prozessen bzw. mit dem Kunden ausgetauscht werden. Die grafische Repräsentation dieser Systeminformationsspeicher erfolgt durch einen Kreis mit einer vertikalen Linie (s. Abbildung 65).

Abbildung 65 - Systeminformationsspeicher zur Darstellung von Schnittstellen

16 Anhang B – die Teilnehmer

Nachfolgend sei die Organisationsstruktur der Projektgruppe dargestellt. Diese soll späteren Lesern des Endberichts einen schnellen Überblick darüber geben, welche Person in welcher Rolle für bestimmte Bereiche verantwortlich zeichnete. Diese Darstellung soll die gezielte Kontaktaufnahme mit den PG-Teilnehmern zwecks Erfahrungsaustausch erleichtern.

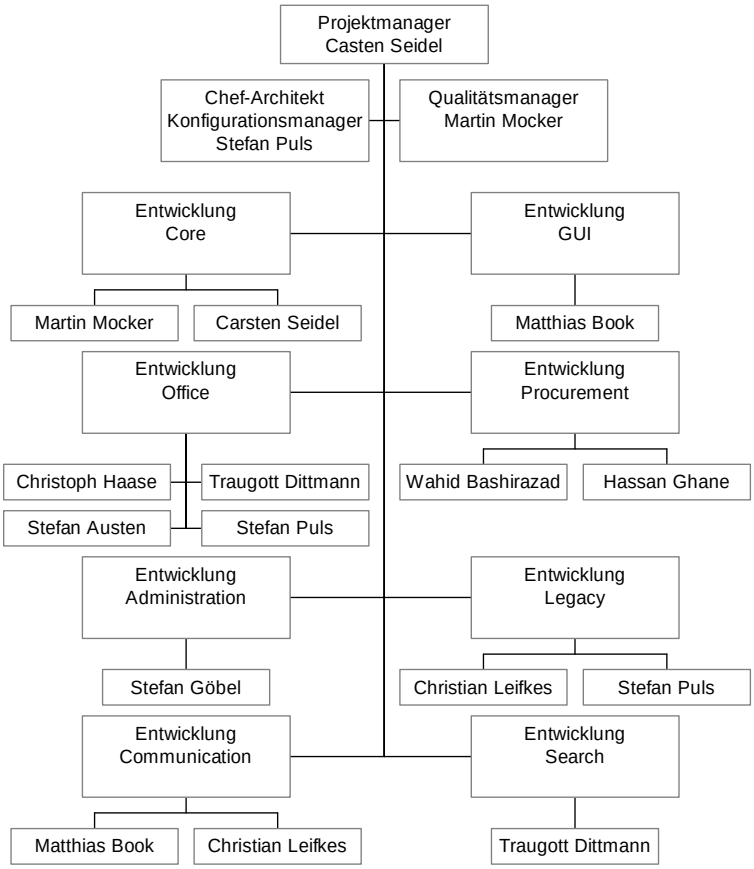


Abbildung 66 - Projektstruktur PG IPSI