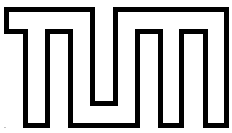


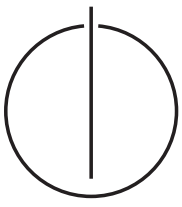
Technische Universität München

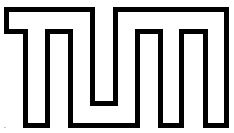
Fakultät für Informatik

Bachelor's Thesis in Informatik

Intelligent Assistent Lösung für die teilautomatisierte Erfassung von Incidents und Service Requests am Leibniz-Rechenzentrum

Benjamin Schnoy





Technische Universität München

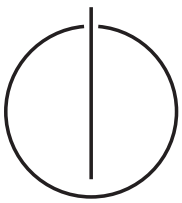
Fakultät für Informatik

Bachelor's Thesis in Informatik

Intelligent Assistent Lösung für die teilautomatisierte Erfassung von Incidents und Service Requests am Leibniz-Rechenzentrum

Intelligent Assistent solution for semi-automated recording and classification of incidents and service requests at the Leibniz Supercomputing Centre

Bearbeiter:	Benjamin Schnoy
Aufgabensteller:	Prof. Dr. Arndt Bode
Betreuer:	Dr. Michael Brenner Bastian Kemmler
Abgabedatum:	14. November 2014



Ich versichere, dass ich diese Bachelor's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 14.11.2014

.....
(Unterschrift des Kandidaten)

Abstract

Das Leibniz-Rechenzentrum (LRZ) ist zentraler IT-Dienstleiter für die Münchner Hochschulen und betreibt für diese mit dem Münchner Wissenschaftsnetz (MWN) einen Großteil der Netzwerkinfrastruktur und diverse IT-Dienste sowie einen der nationalen Höchstleistungsrechner. Nutzer der Dienste des Leibniz-Rechenzentrums können diesem Störungsmeldungen und Service-Anfragen entweder telefonisch oder elektronisch über ein Internet-Portal übermitteln. Diese werden dann in einem IT-Service-Management-Tool (iET ITSM) als Tickets erfasst und bearbeitet.
Aktuell existiert keine Möglichkeit zu definieren, welche Informationen zu einem Service Request oder bei einer Störungsmeldung vom Nutzer erfasst werden. Stattdessen gibt der Nutzer seine Anfrage in ein Freitext-Eingabefeld ein. Da der Nutzerkreis des LRZ sehr heterogen ist und nicht nur aus IT-Experten besteht, führt dies häufig dazu, dass vom Nutzer benötigte Informationen nach Erstellung des Tickets erfragt werden müssen. Dies erhöht die Bearbeitungszeit.
Im Rahmen dieser Arbeit wird das Internet-Portal für die Ticketerfassung so erweitert, dass Informationen strukturiert und kontextabhängig vom Nutzer erfragt werden. Bekannte Standardlösungen, die zu den Eingaben des Nutzers passen, werden vor Ticketerstellung automatisch ermittelt und dem Nutzer vorgeschlagen.

Inhaltsverzeichnis

1 Einleitung

Im Folgenden wird die im Rahmen dieser Arbeit entwickelte Lösung zur teilautomatisierten Erfassung von Nutzeranfragen am Leibniz-Rechenzentrum zuerst motiviert und dann auf die aus der Motivation abgeleiteten, konkreten Implementierungsziele eingegangen. Daran anschließend folgt eine kurze Darstellung der Struktur der vorliegenden Arbeit.

1.1 Motivation

Das Leibniz-Rechenzentrum ist, wie in Kapitel 2 beschrieben, zentraler IT-Dienstleiter für die Münchner Hochschulen und betreibt für diese mit dem Münchner Wissenschaftsnetz (MWN) einen Großteil der Netzwerkinfrastruktur und diverse IT-Dienste.
Im Rahmen des Betriebs dieser Dienste kommt es auch immer wieder zu Nutzeranfragen, die sich grob in die Kategorien Incidents, also Störungen bei der Nutzung eines Dienstes, und Service Requests, welche die Anfrage nach konkreten Leistungen oder Informationen zu einem Dienst darstellen, einteilen lassen. Diese werden vom LRZ im Rahmen eines Gesamtkonzeptes zum IT-Service-Management als Tickets in einer Software aufgenommen und bearbeitet. Ein zentrales Problem stellt hierbei die Datenqualität der erfassten Incidents und Service Requests dar, was sich an der hohen Rate an initialen Rückfragen nach Ticket-Erstellung, die weiter unten in diesem Absatz beschrieben wird, zeigt.
Nachdem der Nutzerkreis des LRZ bei weitem nicht nur aus IT-Experten besteht, sondern unter anderem mit Studierenden, Wissenschaftlern diverser Fachgebiete und weiteren Hochschulmitarbeitern um einiges diverser ist, kann nicht davon ausgegangen werden, dass ein Nutzer weiß, welche Informationen zur Bearbeitung seiner Anfrage benötigt werden.
Wendet sich ein Nutzer mit seinem Anliegen telefonisch an das LRZ, kann dies bei der Aufnahme eines Tickets durch gezieltes Nachfragen des Servicedesk-Mitarbeiters in einigen Fällen kompensiert werden. Da die Servicedesk-Mitarbeiter jedoch aufgrund des, in Abschnitt 2.1 dargestellten, breiten Dienst-Angebots am LRZ nicht bei jedem Dienst im Detail wissen können, welche Informationen in welchem Fall benötigt werden, können auch während der Bearbeitung durch einen Betreuer des jeweiligen Dienstes noch weitere Nachfragen notwendig werden.
Neben der telefonischen Erfassung wird dem Nutzer über ein Web-Portal auch die Möglichkeit geboten, seine Anfrage selbst elektronisch zu erfassen. Hierbei erhält der Nutzer keine direkte Hilfestellung, welche Informationen von ihm benötigt werden.

Von den 8645 im Jahr 2013 am LRZ zu Incidents und Service Requests erfassten Tickets wurde, vermutlich häufig aus den genannten Gründen, bei mindestens 963 dieser Tickets, also ca. 11 Prozent, nach der Erfassung eine Rückfrage an den Nutzer nötig.[1] Durch diese

[1] Hier ist ein Minimalwert angegeben, da zur Ermittlung Tickets gezählt wurden, die im IT-Service-Management-Tool des LRZ als ersten Eintrag im Kommunikationslog eine Rückfrage aufwiesen. Z.B. telefonische Rückfragen oder andere Nachrichten an den Nutzer blieben hierdurch unberücksichtigt.

Rückfragen steigt jedoch die Bearbeitungszeit des Tickets, was weder für den Nutzer noch das LRZ erstrebenswert ist.

Für bekannte Probleme bei der Nutzung von Diensten des LRZ existierten zudem in einigen Fällen bekannte Lösungen, die durch einen Betreuer des jeweiligen Dienstes und in manchen Fällen auch Servicedesk-Mitarbeiter durch einfache Diagnose-Schritte leicht erkannt werden können. Gibt der Nutzer seine Anfrage elektronisch ein und liegt ein solches Standardproblem vor, muss eine entsprechende Diagnose trotzdem erst manuell durch einen Mitarbeiter des Servicedesks oder schlimmstenfalls sogar einen Betreuer des Dienstes durchgeführt werden, selbst wenn der Nutzer alle nötigen Informationen direkt bei der Ersterfassung eingeben hat. Hier ließen sich die Mitarbeiter des Servicedesks entlasten und die Nutzererfahrung verbessern, wenn das Online-Portal dem Nutzer Lösungen zu Standardproblemen direkt auf Basis der eingegebenen Daten vorschlagen könnte.

1.2 Ziel der Arbeit

Ziel der Arbeit ist es, das sogenannte Servicedesk-Portal unter *servicedesk.lrz.de*, über das Nutzer elektronisch Tickets beim LRZ aufgeben können, um die Möglichkeit der kontextabhängigen, strukturierten Datenerfassung zu erweitern.
Meldet der Nutzer einen Incident, sollen ihm nach der Auswahl des entsprechenden Dienstes Fragen zur Diagnose gestellt werden. Die gestellten Fragen können hierbei von den vorherigen Antworten abhängen. Kann das System eine zu diesen Antworten passende Standardlösung ermitteln, soll diese dem Nutzer direkt vorschlagen werden. Dies entspricht, bis auf eine fehlende Komponente zur Erklärung der Entscheidungsfindung, weitgehend der Funktionalität eines Expertensystems, wie z.B. von Boersch et al. [BHSA07, S. 6-7] beschrieben.
Findet das System keine passende Standardantwort, sollen eventuell nötige Zusatzinformationen, die auch von den vorher gegebenen Antworten abhängen können, erfasst werden und hieraus dann ein Incident-Ticket generiert werden.
Das für die Entscheidungsfindung nötige Fachwissen soll hierbei nicht fest ins System einprogrammiert werden, sondern ohne Änderung des Programmcodes modifizierbar sein.

Für die Erfassung von Service Requests wiederum soll eine Möglichkeit zur Definition von speziellen Service Request-Templates gefunden und implementiert werden, bei der angegeben wird, welche Informationen vom Nutzer für den jeweiligen Service Request benötigt werden. Die so definierten Templates sollen dem Nutzer geeignet zur Verfügung gestellt werden.

Die zu entwickelnde Lösung wird in den folgenden Kapiteln in Anlehnung an ein früheres Unterstützungssystem zur teilautomatischen Incident Erfassung am LRZ, dessen Implementierung von Weishaupt [Wei96] vorgenommen und von Mayer [May99] beschrieben wurde, welches sich aber nicht mehr im Einsatz befindet, als Intelligent Assistent bezeichnet.

1.3 Struktur der Arbeit

Um den inhaltlichen und technischen Rahmen für die Umsetzung der Arbeitsziele darzustellen, wird im folgenden Kapitel auf die Rahmensituation am LRZ, also erbrachte Dienste,

konzeptionelle und technische Implementierung des IT-Service-Managements und vorangegangene Arbeiten zu diesem Thema eingegangen. Hieraus und aus der Motivation werden dann in Kapitel 3 konkrete Anforderungen an eine Lösung, deren Struktur an das Schema von Volere angelehnt ist, das von Robertson [RR13] beschrieben wird, abgeleitet. Danach wird in Kapitel 4 eine Lösung zur Umsetzung der Anforderungen gesucht und diese schließlich beschrieben. Auf Basis des entwickelten Lösungskonzepts und der Anforderungen wird in Kapitel 5 dann die durchgeführte Implementierung der Lösung vorgestellt und auf Details dieser eingegangen. Abgeschlossen wird die Arbeit dann von einer Zusammenfassung des Geleisteten und einem Ausblick auf mögliche zukünftige Erweiterungen und Verbesserungen.

2 Rahmensituation am LRZ

Das LRZ stellt den Münchner Hochschulen neben der Netzanbindung auch diverse IT-Dienste wie E-Mail, VPN und WLAN zur Verfügung. Außerdem ist am LRZ auch einer der nationalen Höchstleistungsrechner untergebracht, den neben den Münchner Universitäten auch andere deutsche Forschungseinrichtigungen nutzen können. [LRZ13b]
Mit seinen IT-Dienstleistungen versorgt das LRZ so über 100.000 Nutzer, vom Studierenden bis zum Hochschul-Professor. [LRZ13b] Um eine hohe Qualität der erbrachte Dienste sicherzustellen, beschäftigt sich das LRZ damit, ein IT-Service-Management-System nach ISO/IEC 20000 [ISO11] zu etablieren. [LRZ14]
Im Folgenden wird auf die Dienstleistungspalette des LRZ, sowie die etablierten Prozesse zu Incident- und Service Request-Management eingegangen, um das Einsatzfeld der zu entwickelnden Intelligent Assistent Lösung zu erarbeiten.

2.1 Vom LRZ erbrachte Dienste

Wie bereits aus der Beschreibung des LRZ ersichtlich, bietet das LRZ seinen Nutzern eine sehr vielfältige Palette an IT-Dienstleistungen. Diese werden im Dokument *Das Dienstleistungsangebot des LRZ* [LRZ13a] beschrieben und im Rahmen des IT-Service-Managements in einem sogenannten Service-Baum dargestellt (siehe Abbildung 2.1). Jeder Knoten, bis auf die Wurzel, stellt hierbei einen Service dar. Dieser Baum umfasst derzeit 69 öffentliche Dienste wie *Exchange* und zusätzlich 10 interne Dienste, die jedoch nur an Mitarbeiter des LRZ gerichtet sind.
Jeder dieser Dienste wird durch eine Gruppe an Personen betreut. Die hohe Anzahl an Diensten bedingt jedoch, dass es keine Personen gibt, die Betreuer für alle Dienste sind. Vielmehr sind die meisten Mitarbeiter des LRZ auf den Betrieb und die Betreuung einzelner Dienste spezialisiert.

2.2 Incident- und Service Request-Management am LRZ

Aktuell befindet sich ein nach ISO/IEC 20000 ausgerichtetes IT-Service-Management-System am LRZ im Aufbau. Zwar sind noch nicht alle hierfür notwendigen Prozesse zum Zeitpunkt dieser Arbeit fertig definiert und ausgerollt, jedoch ist der Prozess *Incident and Service Request Management*, der für diese Arbeit besonders relevant ist, bereits seit dem Jahr 2011 vollständig am LRZ eingeführt [LRZ14]. Die Prozessdokumentation dieses Prozesses [BRH14] formalisiert die in Abschnitt 2.1 erwähnten Betreuer eines Dienstes als *Support-Gruppe*. Innerhalb der eingesetzten IT-Service-Management-Software können derartige *Support-Gruppen* angegeben, und als Standard-Abgabegruppe für die Zuweisung der Rolle *Incident-Bearbeiter* definiert, werden. Die einem Service so zugewiesene Support-Gruppe wird daher im weiteren Verlauf dieser Arbeit als die Menge von Personen betrachtet,

Abbildung 2.1: Teil des Service-Baums am LRZ (Quelle: Eigene Darstellung)

die das Fachwissen für die Diagnose und Behebung von Incidents bzw. die Erfüllung von Service Requests des jeweiligen Dienstes besitzt.

2.2.1 Eingesetzte Software

Zur technischen Unterstützung der ITSM-Prozesse wird am LRZ die Software *iET ITSM*, die von der Firma iET Solutions hergestellt und vertrieben wird, eingesetzt. Diese implementiert bereits Funktionen um Prozesse nach ISO/IEC 20000 zu unterstützen. Hierzu bietet sie neben einem Webinterface eine *Workcenter* genannte Windows-Client-Anwendung (siehe Abbildung 2.2), die am LRZ hauptsächlich verwendet wird. In dieser werden die unterschiedlichen Prozesse als Applikationen abgebildet, die wiederum Elemente wie Formulare zum Anzeigen und Bearbeiten von einzelnen Datensätzen sowie Anzeigestrukturen für nach bestimmten Kriterien ausgewählten Datensatzmengen (als Queries bezeichnet) enthalten. [iet11b] Über diese wird dem Nutzer die eigentliche Funktionalität bereitgestellt.

Service Requests, die nach der ISO/IEC 20000 Norm auch in einem von Incidents unterschiedlichen Verfahren behandelt werden können [ISO11, S. 21], werden von iET ITSM als spezielle Typen von Incidents modelliert. Dies ermöglicht die Verwendung der gleichen Formulare und Queries für die Bearbeitung von Incidents und Service Requests. Die Unter-

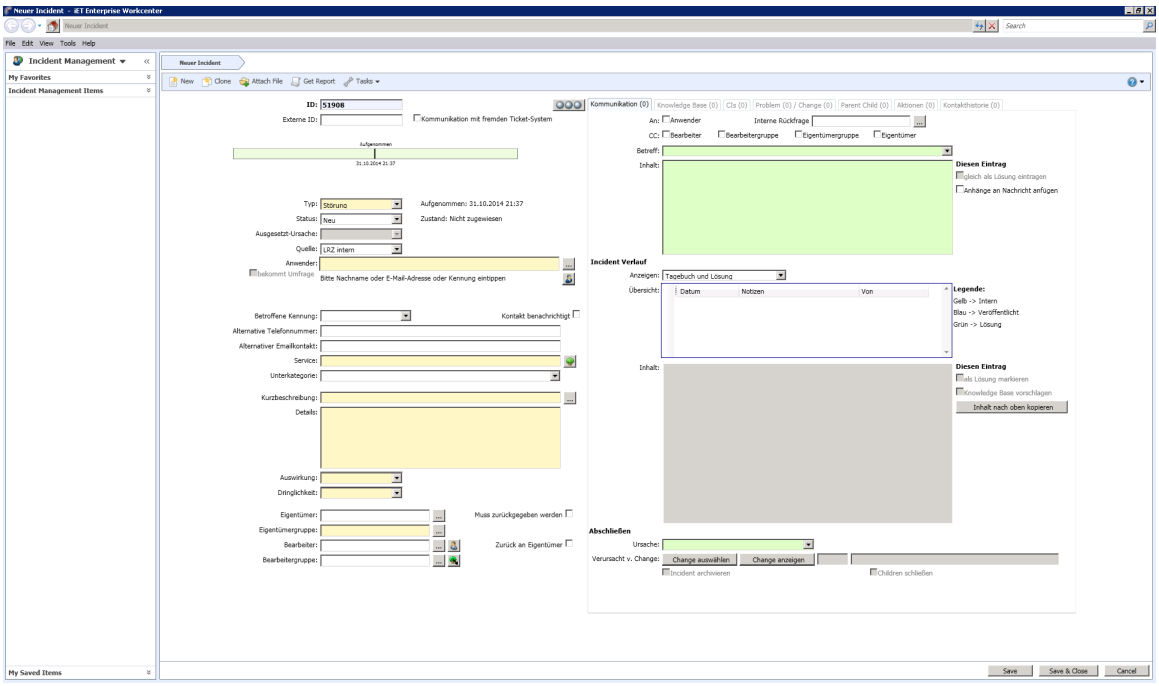


Abbildung 2.2: Screenshot des Formulars zur Incident-Datensatz-Erfassung in der Workcenter-Anwendung (Quelle: Eigene Darstellung)

scheidung erfolgt über das Attribut “Typ” eines Incidents. Handelt es sich um einen Incident der bisher verwendeten Definition nach ISO/IEC 20000, ist der Typ des Incidents *Störung*. Wird ein Serivce Request dargestellt, ist der Typ hingegen *Service Request*.
In den nachfolgenden Teilen der Arbeit wird für den Begriff Incident weiterhin die Definition nach ISO/IEC 20000 als “*unplanned interruption to a service, a reduction in the quality of a service or an event that has not yet impacted the service to the customer*” [ISO11, S. 4] verwendet. Soll hingegen die von iET ITSM verwendete Definition angesprochen werden, wird der Begriff *Incident-Datensatz* verwendet.

Erweiterbarkeit und Anpassbarkeit

Die Software iET ITSM lässt sich grob in zwei Teile gliedern. Zum einen ist dies die *iET Enterprise* genannte Backend-Technologie, die für das Ausführen und Anzeigen der Formulare, Versenden von Benachrichtigungen und die technische Realisierung der meisten anderen Prozesse zuständig ist [iet]. Diese beinhaltet auch einen Editor zur Erstellung und Bearbeitung von eigenen Formularen und Applikationen, die dann in der Workcenter Anwendung genutzt werden können, sowie auch die Möglichkeiten um z.B. Benachrichtigungen, das Datenbanklayout und weitere Objekte anzusehen und zu verändern [iet11a]. Auch gibt es Funktionen um die dynamische Manipulation von Datensätzen und Bedienelementen auf Formularen zu realisieren, wie eine einfache regelbasierte Skriptsprache, Workflow-Regeln genannt [iet11c], sowie APIs für verschiedenen Programmiersprachen und Einsatzwecke [iet11b].
Der zweite Teil der Software besteht aus vom Hersteller mitgelieferten Applikationen, Formularen und weiteren Objekten für die Benutzung mit der *iET Enterprise* Technologie, die

die verschiedenen ITSM-Prozesse modellieren. Diese sind wiederrum durch den Editor den eigenen Befürfnissen anpassbar. [iet]
Die Möglichkeit zur Anpassung der mitgelieferten Kompenenten und Eigenentwicklung neuer Komponenten wird am LRZ aktiv genutzt, um die Software möglichst weit an die spezifischen Ausarbeitungen der ITSM-Prozesse anzupassen.

Die Verwendung der unterschiedlichen Teilsysteme von iET ITSM ist noch einmal zusammenfassend in Abbildung 2.3 dargestellt.

Abbildung 2.3: Use Case Diagramm iET ITSM (Quelle: Eigene Darstellung)

Servicedesk-Portal Webschnittstelle

Seit April 2014 nutzt das LRZ ein selbst implementiertes Web-Portal, das die vorher eingesetzte Web-Portal-Lösung des Herstellers abgelöst hat, um den Nutzern das Anlegen von neuen Incidents und die Kommunikation bezüglich bestehender Incidents mit dem LRZ zu ermöglichen (Screenshot in Abbildung 2.4). Dieses besitzt keine eigene Datenbank sondern ist über eine vom Hersteller der *iET ITSM* Software zur Verfügung gestellte SOAP-Schnittstelle direkt an *iET ITSM* angebunden und kann somit Datensätze aus *iET ITSM* anlegen, anzeigen und manipulieren. Das Backend des Web-Portals ist in PHP mit dem Symfony2-Framework implementiert und nutzt für das Frontend HTML5 sowie verschiedene Javascript Bibliotheken. [Sch14b]

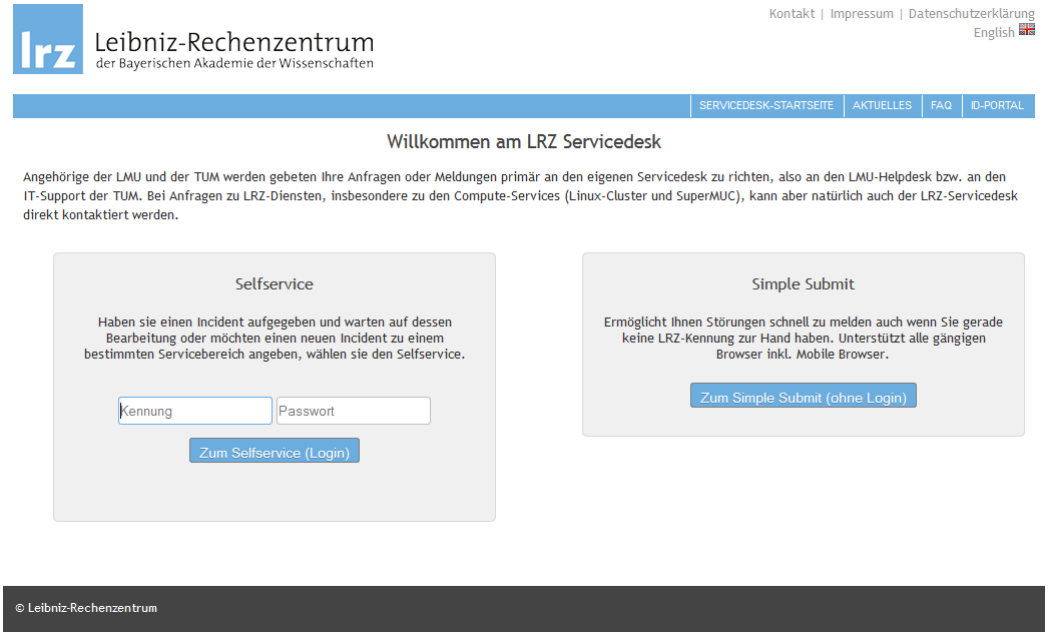


Abbildung 2.4: Screenshot der Startseite des Web-Portals (Quelle: Eigene Darstellung)

2.2.2 Incident- und Service Request-Erfassung

Die Prozessdokumentation [BRH14] zum *Incident and Service Request Management* definiert drei Arten, auf die ein Incident-Datensatz erfasst werden kann. Dies sind

- Erfassung durch einen Mitarbeiter (z.B. bei Meldung über Telefon)
- Webinterface / Self-Service (Erfassung über das im letzten Absatz beschriebene Servicedesk-Web-Portal)
- Integration mit externen Ticket-Systemen (z.B. Tickets aus dem OTRS-Ticketsystem des TUM-IT-Supports)

Als Output der Erfassung legt die Prozessdokumentation fest, dass die Felder *Kurzbeschreibung*, *Details* und *Kontakt* des Incident-Datensatzes erfasst sind. Die eigentlichen Informationen zum Incident-Datensatz werden hierbei im Feld *Details* unstrukturiert, also als Freitext, gespeichert.
Als Erfassungshilfe können im Workcenter Templates für das *Details*-Feld hinterlegt werden, um für das Anlegen von Incidents Ausfüllhilfen oder benötigte Informationen zu definieren. Da diese Templates nur über ihren Namen und die Zuordnung zu einem Dienst auswählbar sind, können sie vor allem bei der telefonischen Erfassung durch einen Mitarbeiter, der nicht selbst der Support-Gruppe des betroffenen Dienstes angehört, eine Hilfe sein. Für einen Nutzer kann es aber aufgrund mangelnden Hintergrundwissens so durchaus schwierig werden, ein passendes Template für die Erfassung seines Incidents selbst auszuwählen.
Nachdem die Möglichkeit der Template-Auswahl in der Web-Portal-Lösung des Herstellers implementiert war, wurde diese beim Wechsel auf die selbst implementierte Web-Portal-Lösung daher vorerst nicht erneut umgesetzt.

Auch der Erfassung durch einen Mitarbeiter vorbehalten ist die Möglichkeit, dass der Nutzer bei bekannten Problembildern direkt einen Lösungsvorschlag erhält. Gibt er seine Frage elektronisch ein, muss er also auf jeden Fall warten, bis seine Anfrage bearbeitet wird. Dies ist auch der Fall, wenn das von ihm gemeldete Problem eine bekannte Lösung besitzt. Dies ist neben dem Vorliegen einer bereits diagnostizierten Dienststörung vor allem der Fall, wenn die Probleme nicht durch eine Störung bei dem Betrieb des Dienstes, sondern durch einen Konfigurations– oder Bedienungsfehler des Nutzers verursacht werden.

Bei Service Requests wiederum gibt es aktuell bei der Erfassung über das Webinterface (im Folgenden *Servicedesk-Portal* genannt) keine Möglichkeit sicherzustellen, dass der Benutzer alle für die Umsetzung des Service Requests benötigten Informationen angibt. Dieser Fall könnte sicherlich teilweise durch die erwähnten Templates für das *Details*-Feld abgedeckt werden, jedoch hat diese Möglichkeit einen Nachteil: Aufgrund der unstrukturierten Informationen können einzelne Felder nicht als Pflichtfelder definiert werden und auch Angaben, bei denen eine aus mehreren Optionen gewählt werden soll, sind so nicht realisierbar.

Insgesamt müssen aufgrund der aktuellen Art der Incident-Datensatz-Erfassung häufiger als eigentlich nötig Rückfragen an den Nutzer gestellt werden, was eine Erhöhung des Bearbeitungsaufwandes und der Bearbeitungszeit von Incidents und Service Requests bedeutet. Die These, dass ein gezieltes Erfragen von Informationen vom Nutzer weniger initiale Rückfragen nötig macht, lässt sich durch Betrachtung der Anzahl an Incident-Datensätzen, die im Jahr 2013 aufgegeben wurden und als erste Kommunikationsinteraktion mit dem Nutzer eine Rückfrage im Incident-Tagebuch aufweisen, stützen. Von 1857 durch einen Mitarbeiter erfassten Incident-Datensätze weisen 191 als ersten Eintrag im Incident-Tagebuch eine Rückfrage auf (9,2%), von den elektronisch von Nutzern erzeugten 5647 Incident-Datensätzen 1019 (18,0%). Hieraus lässt sich die Vermutung ableiten, dass sich durch gezielte Abfrage von Informationen vom Nutzer bei der Erfassung die Zahl der initialen Rückfragen reduzieren lässt.

2.2.3 Knowledge Base und CMDB

Im vergangenen Abschnitt wird von *bekannten Problemen* gesprochen. In diesem Kontext stellt sich die Frage, ob es im Rahmen des IT-Service-Managements am LRZ bzw. der eingesetzten ITSM-Software eine Möglichkeit gibt, ein bekanntes Problem zu modellieren. Offensichtlicherweise ist es nicht sinnvoll, dass häufig auftretende Probleme bei der Nutzung eines Dienstes, die durch Bedien- und oder Konfigurationsfehler des Nutzers ausgelöst werden, nur den Betreuern des Dienstes bekannt sind. Viel mehr sollte derartiges Wissen auch den Mitarbeitern im First-Level-Support, also dem Servicedesk, zur Verfügung gestellt werden, damit dieser etwaige Anfragen direkt beantworten kann. Hierfür bietet die eingesetzte ITSM-Software eine sogenannte *Knowledge Base* an, in der neben Templates für häufig genutzte Kommunikationstexte auch Standardlösungen abgelegt werden können. Einträge aus dieser Knowledge Base können dann direkt als Lösung einem Incident-Datensatz zugewiesen werden. Zusätzlich findet die Dokumentation von Standardlösungen für den Servicedesk teilweise auch außerhalb der ITSM-Software auf einer Microsoft Sharepoint Installation statt. Hierbei wäre es im Bedarfsfall mit etwas Aufwand jedoch möglich, diese in die Knowledge Base des ITSM-Tools zu importieren.
Die Knowledge Base bietet sich somit als Quelle für Lösungsvorschläge einer Intelligent As-

sistent Lösung an.

Eine zweite Art der Standardlösung, nämlich dass ein Dienst gerade gestört ist, kann jedoch nicht durch die Knowledge Base modelliert werden. Zur Zurverfügungstellung solcher Informationen bietet sich eine Configuration Management Database (CMDB) an, in der alle wichtigen Komponenten eines Dienstes, egal ob Software oder Hardware, als sogenannte Configuration Items (CIs) modelliert werden. In dieser lassen sich auch, entweder manuell oder automatisch durch Monitoring Systeme, Ausfälle von CIs eintragen. Aus eingetragenen Ausfällen wiederum ließe sich dann auch auf Beeinträchtigungen von Diensten schließen.
Zwar unterstützt die iET ITSM-Software die Nutzung einer CMDB, jedoch befindet sich diese am LRZ erst im Aufbau und steht daher leider noch nicht als Quelle für derartige Informationen zur Verfügung

2.2.4 Vorangegangene Arbeiten zur Verbesserung der Erfassung

Zum Schluss dieses Kapitels sollen noch vorangegangene Arbeiten zum Thema der teilautomatisierten Erfassung bzw. Klassifikation von Nutzeranfragen an das LRZ als möglicher Ausgangspunkt für diese Arbeit betrachtet werden. Mayer [May99] beschreibt ein *Intelligent Assistent* genanntes System, das früher am LRZ zur teilautomatisierten Erfassung von Incidents im Einsatz war und auf Entscheidungsbäumen basierte, die mit Fachwissen entsprechend qualifizierter Mitarbeiter aufgebaut wurden. Im Gegensatz zur aktuellen Aufgabenstellung wurde dieses System jedoch nur für bestimmte Teilbereiche, nämlich “Verbindung”, “Durchsatz” und “Mail” genannt, realisiert. Mayer [May99] untersucht hierbei, inwieweit die Diagnosephase durch den Einsatz von Bayeschen Netzen optimiert werden kann, kommt jedoch im Fazit zu dem Schluss, dass diese aufgrund der hohen Komplexität eher ungeeignet erscheinen.
Des Weiteren ist fraglich, inwieweit sich Erkenntnisse aus der von Mayer beschriebenen Implementierung auf die aktuelle Fragestellung übertragen lassen. Aus der Arbeit von Mayer lässt sich der Eindruck gewinnen, dass dieses System hauptsächlich dazu gedacht war festzustellen, ob das vom Benutzer beobachtete Verhalten wirklich einer Störung entspricht und der Support-Gruppe dann relevante Informationen zur Verfügung zu stellen. Das Ziel dieser Arbeit geht mit dem Ansatz, Nutzern nach Möglichkeit auch direkt einen Lösungsvorschlag zu geben und der Möglichkeit, je nach Antworten auf die Fragen noch zusätzliche Daten vom Nutzer zu erfassen, falls kein direkter Lösungsvorschlag möglich ist, einen Schritt weiter. Außerdem haben sich seit der Anfertigung der beschriebenen Arbeit sowohl die technische Umgebung zum Incident-Management als auch die hierfür definierten Prozesse am LRZ geändert.

2.3 Zusammenfassung

In diesem Kapitel wurden zuerst die vom LRZ angebotenen Dienste beschrieben und es wurde darauf eingegangen, dass diese aufgrund der hohen Anzahl von unterschiedlichen Personengruppen, den Support-Gruppen, betreut werden. Danach wurde der am LRZ nach ISO/IEC 20000 implementierte Prozess zum Incident- und Service Request-Management beschrieben und die hierfür eingesetzte Software iET ITSM eingeführt, sowie auf Möglichkeiten zur Anpassung dieser Software eingegangen. Anschließend wurden die Knowledge Base als Quelle für bekannte Lösungen und die CMDB als Quelle für Komponenteninformationen in iET

ITSM vorgestellt. Geschlossen wurde das Kapitel mit einer Betrachtung der vorangegangenen Arbeiten zur Umsetzung einer Intelligent Assistent Lösung am LRZ.

3 Anforderungen an die Intelligent Assistent Lösung

In diesem Kapitel werden, ausgehend von der in Kapitel 2 beschriebenen Rahmensituation, zuerst Use Cases und dann Anforderungen an den Intelligent Assistent am LRZ abgeleitet. Diese dienen dann als Grundlage für die in Kapitel 4 beschriebene Lösungsentwicklung und schließlich für die Implementierung. Der in den folgenden Abschnitten vorgenommenen Klassifikation der Anforderungen in Kategorien und den zu jeder Anforderung angegebenen Informationen liegt das von Robertson [RR13] beschriebene *Volere* Template zugrunde, wobei nur ein sehr kleiner und unvollständiger Teil des Templates leicht verändert verwendet wird.

3.1 Use Cases

Im Folgenden findet sich eine Auflistung der abgeleiteten Use Cases entsprechend der von Robertson [RR13, S.82 ff.] beschriebenen Definition eines Product Use Cases. Ein Business Event ist nach dieser Definition ein Ereignis, was Auslöser für den Ablauf eines Use Case ist. Zusätzlich sind die Use Cases in Abbildung 3.1 in Form eines UML Use Case-Diagramms dargestellt.

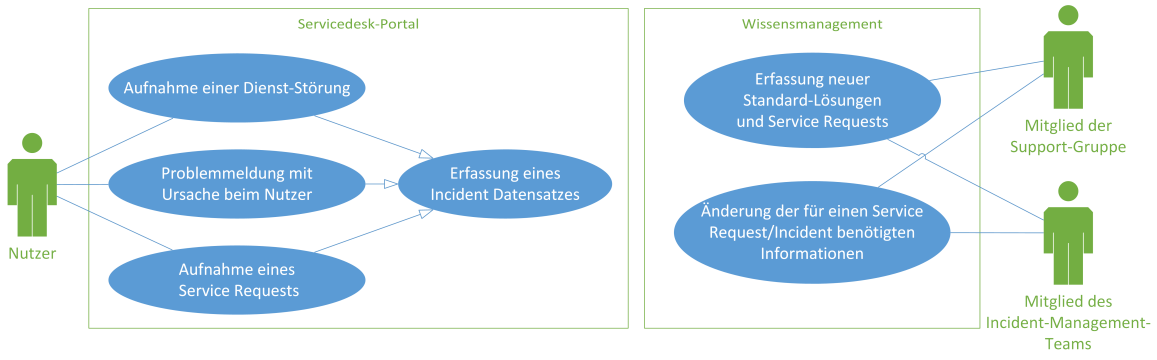


Abbildung 3.1: Use Cases der zu entwickelnden Lösung. (Quelle: Eigene Darstellung)

- **Nummer:** 1
  Name: Aufnahme einer Dienst-Störung
  Akteur: Nutzer eines Dienstes
  Business Event: Dienststörung
  Beschreibung Der Nutzer stellt Probleme bei der Dienstnutzung fest und meldet das Problem über das Servicedesk-Portal als Incident. Der Intelligent Assistent erfragt die

relevanten Informationen um dem Servicedesk ohne weitere Rückfragen, die laut Abschnitt 2.2.2 bei elektronisch erfassten Incident-Datensätzen aktuell häufig nötig sind, die Zuweisung des Tickets an die Support-Gruppe zu ermöglichen. Die erfassten Informationen sind für die Support-Gruppe nun wiederum ausreichend, um ohne weitere Rückfragen schnell festzustellen zu können, ob und wo genau eine Störung vorliegt und diese zu beheben.

- **Nummer:** 2
  Name: Problemmeldung mit Ursache beim Nutzer
  Akteur: Nutzer eines Dienstes
  Business Event: Nutzer kann einen Dienst aufgrund eines Konfigurations- oder Bedienfehlers nicht nutzen
  Beschreibung Der Nutzer stellt Probleme bei der Dienstnutzung fest und meldet das Problem dem LRZ über das Servicedesk-Web-Portal. Der Intelligent Assistent erfragt die relevanten Informationen und stellt fest, dass es sich um ein Konfigurationsproblem / einen Bedienungsfehler des Nutzers handelt. Daraufhin wird dem Nutzer vom Intelligent Assistent mitgeteilt, was er an seiner Konfiguration ändern muss, um den Dienst problemfrei nutzen zu können. Der Nutzer erhält die Möglichkeit die Standardlösung zu akzeptieren, oder diese abzulehnen und einen Incident-Datensatz zu erzeugen.
  Im Gegensatz zu der in Abschnitt 2.2.2 dargestellten Situation erhält der Nutzer in diesem Use Case also direkt bei Eingabe des Incidents einen Lösungsvorschlag. Akzeptiert der Nutzer den Vorschlag, müssen die Mitarbeiter des Servicedesks und der Support-Gruppe des betroffenen Dienstes zudem im Gegensatz zur aktuellen Situation nicht mehr tätig werden und werden somit entlastet.

- **Nummer:** 3
  Name: Aufnahme eine Service Requests
  Akteur: Nutzer eines Dienstes
  Business Event: Der Nutzer hat eine konkrete Service Anfrage, z.B. die Erweiterung seines E-Mail-Postfachs
  Beschreibung Die für die Durchführung des Service Requests nötigen Informationen werden vom Nutzer durch den Intelligent Assistent erfasst und der Support-Gruppe des zugehörigen Dienstes als Teil des Incident-Datensatzes zur Umsetzung zur Verfügung gestellt.
  Auch dieser Use Case führt wie Use Case 1 zu einer Reduktion der in Abschnitt 2.2.2 genannten hohen Zahl an nötigen Rückfragen durch den Servicedesk und die Support-Gruppe.

- **Nummer:** 4
  Name: Änderung der für einen Service Request/Incident benötigten Informationen
  Akteur: Mitglied der Support-Gruppe eines Dienstes
  Business Event: Die für die Diagnose von Incidents oder zur Abarbeitung eines definierten Service Requests benötigten Informationen ändern sich.
  Beschreibung Ein Mitglied der Support-Gruppe hinterlegt im Intelligent Assistent, welche Informationen nun unter welchen Bedingungen benötigt werden. Der Intelligent Assistent passt daraufhin die Abfrage von Nutzerinformationen für die zukünftige Erfassung von Incident-Datensätzen an, nachdem die Änderung durch ein Mitglied des Incident-Management-Teams bestätigt wurde. Hierdurch wird der Support-Gruppe

ermöglicht, das Wissen im Intelligent Assistent aktuell zu halten und somit die korrekte Umsetzung der Use Cases 1 und 2 zu unterstützen.

- **Nummer:** 5
  Name: Erfassung neuer Standardlösungen und Service Requests
  Akteur: Mitglied der Support-Gruppe eines Dienstes
  Business Event: Eine neue Standardlösung oder ein neuer Service Request wird für einen Dienst definiert
  Beschreibung Die Standardlösung/der Service Request wird von einem Mitglied der Support-Gruppe des Dienstes in iET ITSM hinterlegt. Danach wird im Intelligent Assistent angegeben, unter welchen Bedingungen sie anwendbar ist. Der Intelligent Assistent nimmt diese daraufhin nach Freigabe durch ein Mitglied des Incident-Management-Teams mit in die Menge der möglichen Standardlösungen oder Service Requests auf, die dem Nutzer angeboten werden können. Hierdurch kann der Intelligent Assistent Use Case 3 möglichst effektiv umsetzen.

3.2 Mandated Constraints

In Abbildung 3.1 lässt sich gut erkennen, dass sich für die Softwarearchitektur des zu entwickelnden Systems eine Einteilung in zwei Komponenten anbietet. Deshalb werden die Constraints und Requirements in diesem und den folgenden Abschnitten direkt einer der beiden möglichen Komponenten zugeordnet. Dies sind die *Wissensverwaltung* und das *Nutzer-Frontend*.
Für mögliche Softwarearchitekturen, die nur eine Komponente in der Implementierung vorsehen, gelten dann natürlich alle Constraints und Requirements entsprechend.
Aus den aufgelisteten Rahmenbedingungen und Use Cases lassen sich die folgenden Mandated Constraints ableiten:

MC1. **Name:** Implementierung innerhalb des bestehenden Servicedesk-Portals
Komponente: Nutzer-Frontend
Beschreibung: Die Incident-Datensatzerfassung soll in das in Abschnitt 2.2.1 beschriebene Servicedesk-Portal integriert werden. Hierbei soll die dort bestehende Incident-Erfassungsmöglichkeit als Ausgangspunkt für die Erweiterung dienen.
Grund der Anforderung: Die Anforderung, dass das Nutzer-Frontend im bestehenden Servicedesk-Portal integriert werden soll, ergibt sich direkt aus den in Abschnitt 3.1 beschriebenen Use Cases der Incident-Datensatzerfassung. Eine Aufteilung der Erfassung der Incident-Informationen durch den Intelligent Assistent und der Anzeige von bestehenden Incidents in zwei separate Nutzer-Frontends würde aufgrund des hohen inhaltlichen Zusammenhangs der beiden Funktionen vermutlich nicht gut durch die Nutzer angenommen werden.

MC2. **Name:** Anbindung an iET ITSM
Komponente: Wissensverwaltung
Beschreibung: Es soll mindestens lesend eine direkte Anbindung (durch Integration der zu entwickelnden Komponente in die ITSM-Lösung oder über eine Schnittstelle) oder indirekte Anbindung (z.B. über ein Zwischensystem) an die iET ITSM-Datenbank vorgenommen werden.
Grund der Anforderung: Die in Abschnitt 2.2.3 beschriebene Knowledge Base

wird in iET ITSM verwaltet. Daher sollte die zu entwickelnde Wissensverwaltungs-Komponente die entsprechenden Daten aus iET ITSM abrufen können, um eine doppelte Eingabe der bekannten Lösungen für die Verwendung im Intelligent Assistent zu vermeiden.

3.3 Functional Requirements

Nach der Auflistung der Mandated Constraints im letzten Abschnitt werden nun die *Functional Requirements* angegeben:

FR1. **Name:** Strukturierte Erfassung aller für die Bearbeitung eines Incident-Datensatzes nötigen Daten
Komponente: Nutzer-Frontend
Beschreibung: Es sollen vom Nutzer alle Informationen abgefragt werden, die die Support-Gruppe als für die Bearbeitung nötig definiert. Welche Informationen nötig sind, um die Anfrage des Nutzers zu bearbeiten, kann hierbei bei Incidents von den vorher vom Nutzer eingegebenen Informationen abhängen. Bei Service Requests hängen diese nur von der Art des Service Requests, aber nicht von weiteren Angaben des Nutzers, ab.
Grund der Anforderung: Die Anforderung ergibt sich aus den Use Cases 1,2 und 3. Sie stellt eine der Kern-Funktionalitäten des Intelligent Assistent dar

FR2. **Name:** Erstlösungsvorschlag bei bekanntem Konfigurations-/Bedienungsproblem
Komponente: Nutzer-Frontend
Beschreibung: Wenn die Angaben des Nutzers auf eine in der Knowledge Base hinterlegte Standardlösung passen, soll diese dem Nutzer angezeigt werden. Welche Informationen vom Nutzer benötigt werden kann hierbei wiederum von den vorher gegebenen Informationen abhängen. Der Nutzer bekommt nach der Anzeige der Standardlösung die Möglichkeit diese zu akzeptieren, womit seine Anfrage als gelöst betrachtet wird. Alternativ kann er die Standardlösung auch ablehnen, was zur Generierung eines Incident-Datensatzes führt.
Grund der Anforderung: Die Anforderung lässt sich aus Use Case 2 ableiten und stellt eine weitere Kern-Funktionalität dar.

FR3. **Name:** Angabemöglichkeit der vom Nutzer benötigten Informationen
Komponente: Wissensverwaltung
Beschreibung: Es soll eine Eingabemöglichkeit für Mitarbeiter der Support-Gruppe eines Dienstes existieren, welche Angaben eines Nutzers bei Incidents zu einer Standardlösung führen und welche Informationen bei bestimmten Nutzer-Angaben zur weiteren Bearbeitung benötigt werden, wenn keine Standardlösung für diesen Fall existiert. Auch bei Service Requests müssen die von einem Nutzer zu erfassenden Angaben definiert werden können. Diese von der Support-Gruppe einzugebenden Definitionen werden im Weiteren entsprechend der Definition von Beierle und Kern-Isberner [BKI08, S. 17] als *Wissensbasis* bezeichnet.
Grund der Anforderung: Die bisher genannten Functional Requirements und Use Cases 4 und 5 begründen diese Anforderung. Dass die Eingabemöglichkeit für die Support-Gruppe des jeweiligen Diensts bestehen muss und nicht für eine zentrale Ver-

walterrolle, begründet sich mit der Erkenntnis aus Absatz 2.1, dass es keine Person gibt, die über genügend Fachwissen für alle Dienste des LRZ verfügt.

FR4. **Name:** Generierung und Visualisierung einer Entscheidungs-Datenstruktur für die Datenerfassung im Nutzer-Frontend
Komponente: Wissensverwaltung
Beschreibung: Die Wissensbasis soll durch den Intelligent Assistent nach zu definierenden Regeln in eine Datenstruktur konvertiert werden, die im Nutzer-Frontend für eine Dialogführung zur Erfassung der Nutzerangaben genutzt werden kann. Sie dient somit auch der Entscheidung, in welchen Fällen Standardlösungen angezeigt, bzw. wann welche Zusatzinformationen erfragt werden. Hierbei soll sich auch die Funktionalität der in Abschnitt 2.2.2 beschriebenen Templates, die im Details-Feld des Incident-Datensatzes eingefügt wurden, auf Bedarf der Support-Gruppe modellieren lassen.
Die generierte Struktur soll zudem geeignet visualisiert werden, damit die Support-Gruppe prüfen kann, ob die Wissensbasis korrekt erstellt und umgesetzt wurde.
Grund der Anforderung: Das von der Support-Gruppe eingegebene Wissen muss für die Incident-Datensatzerfassung im Nutzer-Frontend nutzbar gemacht werden.

3.4 Nonfunctional Requirements

Auf die Functional Requirements folgend werden nun die Nonfunctional Requirements aufgelistet, die sich in mehrere Unterkategorien gliedern.

3.4.1 Ease of Use Requirements

NR1. **Name:** Nachvollziehbarkeit der vom Nutzer erfragten Informationen
Komponente: Wissensverwaltung
Beschreibung: Die in *FR4* aus Abschnitt 3.3 geforderte Visualisierung der generierten Entscheidungsstruktur muss ausreichend übersichtlich sein.
Grund der Anforderung: Die Support-Gruppe eines Dienstes soll die Möglichkeit haben, schnell zu prüfen, dass die dem Nutzer dargestellte Dialogführung ihren Erwartungen entspricht. Ist für die Support-Gruppe hingegen nicht nachvollziehbar, ob die eingegebene Wissensbasis entsprechend ihren Erwartungen in die Dialogführung umgesetzt wurde, sind Eingabefehler der Support-Gruppe schlechter für diese zu erkennen. Außerdem ist davon auszugehen, dass hierdurch die Akzeptanz des Systems durch die Support-Gruppe sinkt.

NR2. **Name:** Einfache Administrierbarkeit der Wissensbasis
Komponente: Wissensverwaltung
Beschreibung: Die Verwaltung der Wissensbasis soll einfach und komfortabel genug sein, dass Mitglieder der Support-Gruppen bereit sind, Änderungen an der Wissensbasis zu ihrem Dienst regelmäßig und zeitnah einzupflegen.
Grund der Anforderung: Ist die Administration der Wissensbasis zu aufwendig, ist zu erwarten, dass diese nicht ausreichend durch Mitglieder der Support-Gruppe gepflegt wird. Ohne eine entsprechende Pflege der Wissensbasis werden eventuell nicht alle benötigten Informationen vom Nutzer ermittelt oder der Nutzer erhält im schlimmsten Fall sogar nicht mehr aktuelle bzw. falsche Lösungsvorschläge.

NR3. **Name:** Gute Bedienbarkeit für unterschiedliche Nutzergruppen
Komponente: Nutzer-Frontend
Beschreibung: Technisch nicht versierte Nutzer sollen durch den Intelligent Assistent nicht überfordert werden, wobei gleichzeitig Nutzer mit großen Fachkenntnissen durch die Erfassungsart in ihrer Eingabegeschwindigkeit nicht zu stark ausgebremst werden sollen.
Grund der Anforderung: Das LRZ bedient, wie in der Einleitung von Kapitel 2 beschrieben, einen sehr heterogenen Nutzerkreis mit stark variierenden Fachkenntnissen.

3.4.2 Integrity Requirements

NR4. **Name:** Autorisierung von Änderungen nur durch Incident-Management-Team
Komponente: Wissensverwaltung
Beschreibung: Die Freischaltung von generierten Entscheidungs-Datenstrukturen für den Einsatz im Nutzer-Frontend soll nur für Mitglieder des Incident-Management-Teams möglich sein.
Grund der Anforderung: Die Anforderung ergibt sich aus Use Case 4 und 5. Inhaltlich lässt sie sich zudem damit begründen, dass die Rolle *Incident Manager* laut Prozessdokumentation [BRH14] für die Prozessschritte *Aufnahme von Incidents* und *Diagnose und Lösung* des Prozesses *Incident and Service Request Management* verantwortlich ist.

3.4.3 Personalization and Internationalization Requirements

NR5. **Name:** Zweisprachiger Erfassungsdialog
Komponente: Nutzer-Frontend
Beschreibung: Die durch den Intelligent Assistent durchgeführte Incident-Erfassung sowie die Service Request-Erfassung sollen auf Deutsch und auf Englisch möglich sein.
Grund der Anforderung: Zum bereits erwähnten heterogenen Nutzerkreis des LRZ gehören auch Nutzer ohne Kenntnisse der deutschen Sprache. Nachdem die bisherigen Funktionen des Servicedesk-Portals komplett auf Deutsch und Englisch verfügbar sind, würde eine fehlende englische Benutzerführung des Intelligent Assistent den Nutzen für nicht deutschsprachige Nutzer stark mindern.

3.4.4 Speed and Latency Requirements

NR6. **Name:** Maximale Ladezeit von einer Sekunde
Komponente: Nutzer-Frontend
Beschreibung: Die Ladezeit nach jeder Nutzereingabe, z.B. für die nächste Frage an den Nutzer oder einen anzuzeigenden Lösungsvorschlag, soll maximal eine Sekunde bei 3 *Concurrent Usern* betragen.
Grund der Anforderung: Laut dem Whitepaper *Why Web Performance Matters* [gom] erwartet der durchschnittliche Nutzer eines Online-Shopping-Portals maximale Ladezeiten einer Webseite von zwei Sekunden oder weniger. Außerdem zeigt dieses Whitepaer auch einen Trend der Erwartung zu kürzeren Ladezeiten auf, weshalb eine Sekunde als zeitgemäßes Performance-Ziel für ein Online-Portal erscheint. Die Zahl der Concurrent User ist hierbei eine Schätzung, die anhand der durchschnittlichen Zahl

neuer Incident-Datensätze und Tagebucheinträge, die pro Tag über das Servicedesk-Portal generiert werden, vorgenommen wurde. Dies waren im Zeitraum vom 01.04.2014 bis 01.10.2014 durchschnittlich 14 Incident-Datensätze und Tagebucheinträge pro Tag. Nimmt man dazu noch an, dass diese von unterschiedlichen Nutzern aufgegeben wurden und legt eine durchschnittliche Dauer von 10 Minuten für das Anlegen eines Incident-Datensatzes oder Tagebucheintrags zugrunde und nimmt dazu noch eine Gleichverteilung der Anfragen in der Zeitspanne von 8-18 Uhr an (in dieser wurden im betrachteten Zeitraum 92% der Incident-Datensätze angelegt), lässt sich die Zahl der Concurrent User nach Barber [Bar07] wie folgt berechnen:

$$\frac{\frac{\text{Nutzer pro Tag}}{\text{Nutzungs-Zeitspanne (in Stunden)}}}{\frac{60}{\text{Nutzungsdauer (in Minuten)}}} = \frac{\frac{14}{10}}{\frac{60}{10}} = 0,23$$

Nachdem die Annahme einer Gleichverteilung natürlich nicht der Realität entspricht, stellt die Annahme von 3 Concurrent Usern eine gute obere Schranke für die berechneten 0,23 Concurrent User dar, die sehr sicher eingehalten werden sollte.

3.5 Zusammenfassung

Zu Anfang des Kapitels wurden die Use Cases der Intelligent Assistent Lösung definiert. Hieraus wurden dann zwei Mandated Constraints, vier Functional Requirements und schließlich sechs Nonfunctional Requirements abgeleitet. Die Klassifizierung lehnt sich hierbei an das *Volere*-Template, welches von Robertson [RR13] beschriebenen wird, an.

4 Lösungsentwurf

In diesem Kapitel wird nun ausgehend von den im vorigen Kapitel abgeleiteten Anforderungen eine Lösung für die Entwicklung der Intelligent Assistent Lösung entworfen. Hierbei wird zuerst auf die grundlegende Softwarearchitektur eingegangen. Danach werden Lösungen für die einzelnen Komponenten und Funktionalitäten entwickelt und vorgestellt. Hiermit wird die Grundlage für die in Kapitel 5 beschriebene Implementierung geschaffen.

4.1 Softwarearchitektur

Nachdem durch die Mandated Constraints im vorherigen Kapitel bereits feststeht, dass das Nutzer-Frontend des Intelligent Assistent in das bestehende Servicedesk-Portal integriert werden soll, stellt sich nun die Frage, wo die Wissensverwaltungskomponente technisch anzusiedeln ist. Im Use Case Diagramm der bestehenden Komponenten von iET ITSM (Abbildung 2.3) aus Kapitel 2 ist gut zu erkennen, dass die Mitarbeiter der Support-Gruppe bisher ausschließlich mit der Workcenter-Anwendung arbeiten. Nachdem auch die Verwaltung der Knowledge Base über diese Anwendung vorgenommen wird, ist es nicht sinnvoll, die Wissensverwaltung als einen eigenen Bereich des Servicedesk-Portals oder eine eigene Anwendung zu realisieren, da sonst die Verwaltung der im Intelligent Assistent verwendeten Daten auf zwei Anwendungen aufgeteilt würde. Die vor den genannten Aspekten sinnvollere Alternative scheint die Integration in die Workcenter-Anwendung zu sein, die deshalb für die Softwarearchitektur gewählt wird. Der Datentransfer aus der Wissensverwaltung an das Servicedesk-Portal kann hierbei über die vom Servicedesk-Portal genutzte SOAP-Schnittstelle erfolgen, da in dieser individuell konfiguriert werden kann, welche Daten übertragen werden und auch die Ausgabe der Daten von Query-Objekten möglich ist [Sch14a].
Die resultierende Softwarearchitektur ist in Abbildung 4.1 dargestellt.

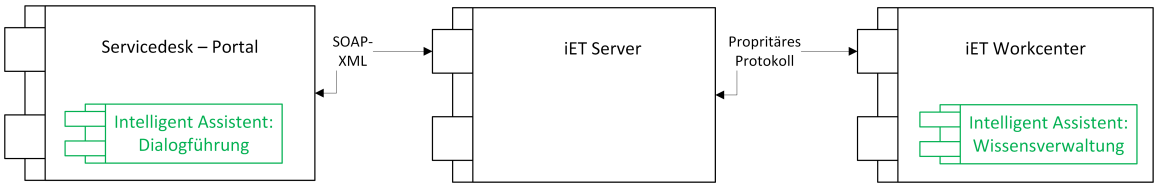


Abbildung 4.1: Softwarearchitektur der Lösung. Die grün dargestellten Komponenten sind die in dieser Arbeit neu Entwickelten. (Quelle: Eigene Darstellung)

Im Folgenden wird nun zuerst der Lösungsentwurf für die Wissensverwaltungs-Komponente (Abschnitt 4.2) und danach der für die Nutzer-Frontend-Komponente (Abschnitt 4.3) beschrieben.

4.2 Wissensverwaltungs-Komponente

Nachdem der Intelligent Assistent, wie in Abschnitt 1.2 dargestellt, einem Expertensystem oder allgemeiner einem Wissensbasierten System nach Beierle und Kern-Isberner [BKI08] ähnelt, stellt sich zuerst die Frage der Wissensrepräsentation. Hierbei muss zwischen der Wissensbasis, wie z.B. Ableitungsregeln, und der Wissensverarbeitung, z.B. der Interpretation dieser Ableitungsregeln, unterschieden werden [BKI08, S.17]. Als erste Designentscheidung ist hierbei zu nennen, für die unterschiedlichen Dienste des LRZ eine getrennte Wissensbasis zu verwalten. Dies hat zwar zur Konsequenz, dass ein Nutzer zuerst den betroffenen Dienst angeben muss, bevor er Unterstützung durch den Intelligent Assistent erhält, spiegelt jedoch die Zuordnung von unterschiedlichen Support-Gruppen zu den einzelnen Diensten wieder.
Wie bereits in der Anforderung *FR3: Angabemöglichkeit der vom Nutzer benötigten Informationen* aus Abschnitt 3.3 dargestellt, besitzen einzelne Personen auch gar nicht das Fachwissen, eine globale Wissensbasis für alle Dienste zu verwalten. In den folgenden Abschnitten ist mit der Wissensbasis also immer die Wissensbasis eines Dienstes gemeint, wobei sich mit der implementierten Lösung dann natürlich parallel Wissensbasen für die einzelnen Dienste verwalten lassen.

4.2.1 Repräsentation der Wissensbasis

Für die Lösungsentwicklung muss nun also eine geeignete Repräsentation der Wissensbasis gefunden werden.
Nachdem die Intelligent Assistent Lösung entsprechend der Anforderungsliste das Erfassen von Service Requests und Incidents unterstützen soll, muss dies für beide Incident-Datensatz-Arten erfolgen.

Service Requests

Da laut Use Case 3 für Service Requests keine Dialogführung zur Ermittlung des für den Nutzerwunsch zutreffenden Service Request erfolgt, können die Service Requests als Liste gespeichert werden, in der diese mit einem von der Support-Gruppe definierten Namen abgespeichert werden. Jedem der so definierten Service Requests können vom Nutzer zu erfassende Informationen in Form von Fragen zugeordnet werden. Ein solches Element aus Name des Service Requests und zugeordneten Fragen wird im Folgenden als *Service Request Template* bezeichnet. Zusätzlich enthält dieses Template auch noch die Information, welche Fragen optional sind und welche für die Erfassung unbedingt ausgefüllt werden müssen.

Incidents

Bei Incidents ist die benötigte Wissensrepräsentation komplexer, da hier eine Dialogführung durch den Intelligent Assistent erfolgen soll. Ein naheliegender Lösungsansatz wäre hierbei die Eingabe des Wissens als Entscheidungsbaum, bei dem Fragen innere Knoten darstellen und Standardlösungen (also Knowledge Base-Einträge) sowie *Incident Templates*, die analog zu den *Service Request Templates* definiert werden, die Blätter. Die Kanten sind hierbei die Antworten auf die jeweiligen Fragen.
Ein einfaches Beispiel für einen solchen Baum ist in Abbildung 4.2 zu sehen.
Würde das Wissen als ein solcher Baum eingegeben, könnte dieser direkt für eine Dialogführung im Nutzer-Frontend genutzt werden, da sich aus einer Antwort des Nutzers un-

mittelbar die passende Kante und somit nächste Frage, bzw. Lösung, ermitteln ließe. Es müsste also keine weitere Wissensverarbeitung stattfinden.

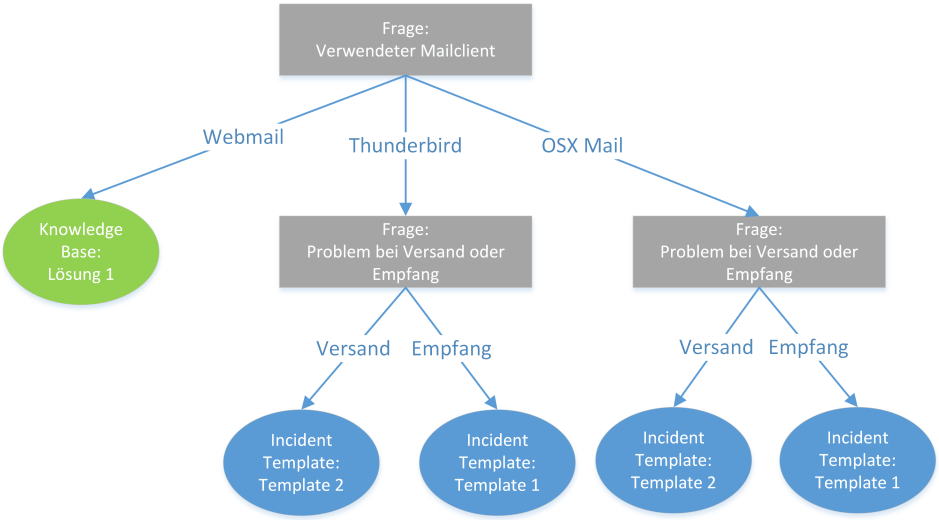


Abbildung 4.2: Beispiel für die Repräsentation einer einfachen Wissensbasis zum Dienst E-Mail als Baum. Fragen sind grau, Lösungen aus der Knowledge Base grün und Incident Templates blau gefärbt. (Quelle: Eigene Darstellung)

Obwohl das Workcenter diese Art der Wissensrepräsentation mit den sogenannten *Graphical Explorer* Objekten, mit denen sich eine Menge verknüpfter Knoten erstellen und anzeigen lassen [iet11a, Folie 15-2 ff.] (also auch Bäume), unterstützen würde, hat diese Form der Wissensrepräsentation auch entscheidende Nachteile.
Vor allem ist hier der hohe manuelle Wartungsaufwand bei größeren Bäumen zu nennen, falls Fragen und oder Antworten hinzugefügt oder entfernt werden sollen. Je nach Verwendungshäufigkeit der Frage oder Antwort erfordert dies mitunter manuelle Veränderungen an vielen Knoten und Kanten. Dies lässt sich schon am sehr kleinen Beispielbaum aus Abbildung 4.2 gut nachvollziehen: Möchte man hier zwischen den beiden Fragen eine neue Frage mit drei Antwortmöglichkeiten einfügen, wobei für mindestens zwei der Antwortmöglichkeiten Teile des restlichen Baums wiederverwendet werden, muss zuerst für jede Antwort der ersten Frage ein Knoten für die neue Frage eingefügt werden und die Kanten von der ersten Frage mit den neuen Knoten verbunden werden. Nun müssen die wiederzuverwenden Teile des restlichen Baums manuell dupliziert und über Kanten mit den Knoten der neu eingefügten Frage verbunden werden.

Auch entsprechend aufwendig ist aus gleichem Grund die Umsortierung der Fragereihenfolge. Hier müsste für das Beispiel aus Abbildung 4.2 zuerst die Verbindung von allen Kanten zu Knoten gelöst werden um danach die Frage nach dem verwendeten Mailclient zu duplizieren. Danach würde einer der beiden *Problem bei Versand oder Empfang*-Knoten gelöscht, damit der andere zur neuen Wurzel werden kann. Nun müssten die Antwortkanten manuell neu mit den so erstellten Knoten und Blättern verbunden werden.

Diese Überlegungen rechtfertigen die Annahme, dass mit dieser Art der Wissensrepräsen-

tation die Anforderung *NR2: Einfache Administrierbarkeit der Wissensbasis* aus Abschnitt 3.4 nicht erfüllt werden würde.

Als alternativer Weg der Wissensrepräsentation wird daher der Folgende gewählt: Die Support-Gruppe definiert *Entscheidungsfragen* für die Dialogführung mit einer Menge aus fest vorgegebenen Antwortmöglichkeiten. Außerdem definiert sie *Zusatzfragen*, die entweder mehrere vorgegebene Antwortmöglichkeiten haben oder eine Freitext-Antwortmöglichkeit. Zusätzlich wird für beide Fragemengen eine Reihenfolge der Fragen angegeben. Den Knowledge Base-Einträgen und Incident Templates können nun jeweils mehrere Mengen aus Antworten zu *Entscheidungsfragen* zugeordnet werden, wobei eine solche Menge im Folgenden als *Antwortsatz* bezeichnet wird. Jeder *Antwortsatz* gibt hierbei eine Menge von Bedingungen an, unter denen der Knowledge Base-Eintrag oder das Incident Template zutreffend ist. Zusätzlich werden Incident Templates noch eine Menge der Zusatzfragen zugeordnet. Ein *Incident Template* modelliert hierbei dem Fall, dass für das vom Nutzer geschilderte Problem nicht direkt eine Lösung vorgeschlagen werden kann, allerdings von der Support-Gruppe zur Bearbeitung des Incidents noch Zusatzinformationen benötigt werden, die über die Zusatzfragen definiert sind.
Die beschriebene Wissensmodellierung ist in Abbildung 4.3 dargestellt.

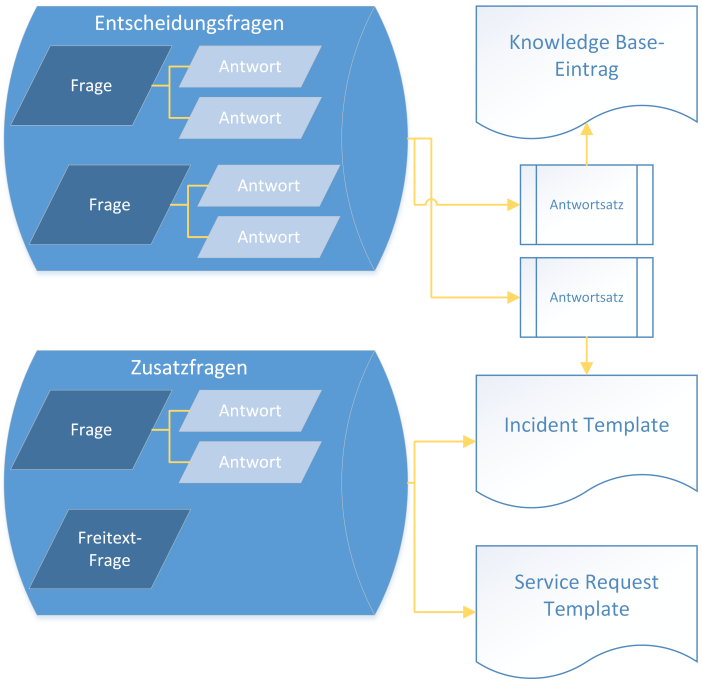


Abbildung 4.3: Modellierung der Wissensbasis. (Quelle: Eigene Darstellung)

Um das Beispiel aus Abbildung 4.2 noch einmal aufzugreifen wird in Tabelle 4.1 und Tabelle 4.2 dargestellt, wie man eine Beispielwissensbasis für den Dienst E-Mail mit dieser Methode der Wissensrepräsentation angeben könnte (wobei die Zuordnung der Zusatzfragen im Beispiel ausgelassen wird).

Reihenfolge	Frage	Antworten
1	Verwendeter Mailclient	[Webmail, Thunderbird, OSX Mail]
2	Problem bei Versand oder Empfang	[Versand, Empfang]

Tabelle 4.1: Fragen und Antworten der Beispiel-Wissensbasis zum E-Mail Dienst

Typ	Name	Antwortsätze
Knowledge Base-Eintrag	Lösung 1	[Verwendeter Mailclient = Webmail]
Incident Template	Template 1	[Problem bei Versand oder Empfang = Empfang]
Incident Template	Template 2	[Problem bei Versand oder Empfang = Versand]

Tabelle 4.2: Knowledge Base-Einträge und Incident Templates mit zugeordneten Antwortsätzen der Beispiel-Wissensbasis zum E-Mail Dienst

Mit diesem Ansatz lässt sich das Hinzufügen, Ändern und Umsortieren von Fragen mit der Modifikation an einer einzigen Stelle, der Frageliste, realisieren. Jedoch macht dies eine Wissensverarbeitung erforderlich, um das Wissen in eine Datenstruktur, die für die Dialogführung genutzt werden kann, umzuwandeln. Diese Datenstruktur muss zur Erfüllung der Anforderung *NR1: Nachvollziehbarkeit der vom Nutzer erfragten Informationen* aus Abschnitt 3.4 auch geeignet visualisierbar sein, da bei mehreren Knowledge Base-Einträgen und Incident Templates mit zugeordneten Antwortsätzen sonst nicht mehr direkt ersichtlich ist, welche Antworten des Nutzers zu welcher Entscheidung des Intelligent Assistent führen.

4.2.2 Wissensverarbeitung

Entsprechend der im vorigen Abschnitt gewählten Wissensrepräsentation muss das Wissen im Zuge der Wissensverarbeitung in eine für die Dialogführung und Visualisierung geeignete Form konvertiert werden.
Nachdem der Entscheidungsbaum im vorigen Abschnitt bereits als eine für die Dialogführung und Visualisierung gut geeignete Form herausgearbeitet wurde, bietet sich die Generierung eines solchen Baumes als Ziel für die Wissensverarbeitung an. Mit dieser Herangehensweise wird auch der genannte Negativpunkt, die Wartung und Änderung des Baumes, durch einen Automatismus übernommen. Trotz der umfangreichen Modellierung des Wissens in der Wissensbasis ist der zu generierende Baum durch diese noch nicht eindeutig definiert. Dies zeigt sich z.B. daran, dass nicht festgelegt ist was geschehen soll, falls ein Antwortsatz auf die Antworten des Nutzers passt aber ein Antwortsatz eines anderen Blattes existiert, für dessen Prüfung noch Informationen fehlen.
Im Folgenden werden zwei mögliche Arten einen Entscheidungsbaum aus der Wissensbasis zu generieren untersucht.

Generierung mit C4.5 Algorithmus

Der C4.5 Algorithmus ist ein auf dem Prinzip der *Top Down Induction of Decision Trees* basierendes Verfahren zur Generierung eines Entscheidungsbaums aus Beispieldatensätzen, die in diesem Anwendungsfall den im vorigen Abschnitten definierten Antwortsätzen eines Inci-

dent Templates oder Knowledge Base-Eintrags entsprechen. Hierbei wird vom Algorithmus die Auswahlreihenfolge der Fragen selbst vorgenommen. Die Sortierung der Fragen geschieht grob gesagt nach ihrem *Informationsgehalt*, also wie gut sie die Menge der noch zur Verfügung stehenden Blätter unterteilen. Hiermit soll eine möglichst geringe Tiefe des Baums, die über die Anzahl von Kanten von der Baumwurzel bis zu den Blättern definiert ist, erreicht werden. Außerdem werden einige weitere Methoden angewendet, um einen möglichst simplen Entscheidungsbaum zu erhalten. [BKI08, S.116-119]
Die in Abschnitt 4.2.1 beschriebene Vorsortierung der Fragen durch die Support-Gruppe wäre hiermit unnötig. Jedoch ist die Zielsetzung dieses Algorithmus, aus Beispieldaten Sachverhalte zu lernen und diese in einem möglichst einfachen Entscheidungsbaum darzustellen eine etwas andere, als die dieser Arbeit. Hier geht es vor allem darum, der Support-Gruppe die Möglichkeit zu geben Sachverhalte zu definieren, unter denen Nutzer bestimmte Lösungsvorschläge erhalten oder weitere Informationen angeben müssen. Es soll nicht versucht werden, von diesem Wissen auf nicht direkt eingegebene Sachverhalte zu schließen und für diese auch eine Nutzerführung anzubieten. Auch ist bei den durch diesen Algorithmus generierten Bäumen bei den Beispieldaten eine geringe Rate an Fehlklassifikationen möglich [Qui93, S.37-S.42], was im Widerspruch zu der Anforderung *NR1: Nachvollziehbarkeit der vom Nutzer erfragten Informationen* aus Abschnitt 3.4 steht. Da der C4.5 Algorithmus zudem eine mögliche, vorgegebene Ordnung von Ergebnis-Klassen (die in diesem Fall den Knowledge Base-Einträgen und Incident Templates entsprechen) nicht verarbeiten kann [Qui93, S.103], ist es auch nicht einfach möglich vorzugeben, ob Incident Templates oder Knowledge-Base-Einträge bei der Auswahl priorisiert werden sollen.
Insgesamt ist der C4.5 Algorithmus zwar ein sehr mächtiges Verfahren, was für die Anwendung im Kontext des Intelligent Assistent aber mitunter zu komplex und schwierig zu steuern ist, da diese nicht wirklich eine Lernaufgabe im eigentlichen Sinn darstellt.

Generierung mit selbst entwickeltem Algorithmus
Nachdem der C4.5-Algorithmus für die Wissensverarbeitung im beschriebenen Fall als nicht ideal erscheint, wird im Folgenden ein Algorithmus entwickelt, der besser zu den Anforderungen passen soll:
Wie schon zuvor sind hierbei Incident Templates und Knowledge Base-Einträge die möglichen Blätter, die Fragen die inneren Knoten des Entscheidungsbaums und die Antworten auf Fragen die Kanten. Nachdem in der Wissensbasis bereits eine Reihenfolge der Fragen angegeben ist, muss diese nicht weiter ermittelt werden.
Der Algorithmus ist in Abbildung 4.4 als Aktivitätsdiagramm dargestellt und wird im Folgenden genauer beschrieben. Die Verbindung zwischen den Elementen des Aktivitätsdiagramms und der Beschreibung wird durch die in Klammern angegebene Nummerierung hergestellt.
Der Algorithmus startet mit der Menge der definierten Incident Templates sowie den Knowledge Base-Einträgen, denen Antwortsätze zugeordnet wurden, als mögliche Blätter (1).

Für jede Entscheidungsfrage aus der Wissensbasis wird entsprechend der angegebenen Reihenfolge geprüft, ob diese die Menge der möglichen Blätter weiter unterteilt (2). Falls nein wird diese verworfen, da sie keinen Informationsgewinn herbeiführt (3). Falls ja wird die Frage als Frageknoten in den Baum eingefügt (4) und es werden für jede Antwort auf die Frage die Teilmengen der möglichen Blätter gebildet (7), die Antwortsätze besitzen, die mit der Antwort zutreffen oder noch zutreffen könnten. Falls sich aus dieser Menge noch nicht nach den unten genannten Abbruchregeln eindeutig ein Blatt bestimmen lässt (9), wird der

Vorgang mit der nächsten Frage wiederholt. Hierbei wird die Antwort eine Kante von der beantworteten Frage zum nächsten zu ermittelnden Frageknoten (13,14). Falls dieser Knoten nicht existiert, da alle Antwortsätze durch diese Antwort unzutreffend geworden sind, wird die Kante hingegen nicht eingefügt (8).
Als unzutreffend wird ein Antwortsatz angesehen, wenn er eine andere Antwort auf eine Frage im Pfad enthält, als zutreffend hingegen, wenn der Pfad alle Antworten enthält, die im Antwortsatz enthalten sind.

Zusätzlich existieren noch zwei Priorisierungsregeln, falls für einen Pfad im Baum mehrere Blätter in Frage kommen. Diese werden im Folgenden in absteigender Anwendungsreihenfolge angegeben:

- **Priorisierung nach Blatt-Typ**: Passen Antwortsätze von mehreren Blättern auf den Pfad, werden Knowledge Base-Einträge bevorzugt.
  Begründung der Regel: Tritt der genannte Fall ein, treffen also ein Lösungsvorschlag und ein Template zur weiteren Erfassung von Daten für die manuelle Bearbeitung durch die Support-Gruppe auf die Antworten des Nutzers zu. Dem Nutzer den Lösungsvorschlag anzubieten verspricht hier die größere Arbeitsreduktion, da bei Annahme des Vorschlags durch den Nutzer die Support-Gruppe nicht aktiv werden muss.

- **Priorisierung nach Spezialität des Antwortsatzes**: Trifft ein Antwortsatz auf den Pfad zu, aber existieren noch weitere, speziellere Antwortsätze (also solche die noch auf weitere Fragen Antworten benötigen), wird der Algorithmus noch nicht abgebrochen. Speziellere Antwortsätze werden dabei für die Auswahl des Blattes bevorzugt.
  Begründung der Regel: Aufgrund dieser Regel kann die Support-Gruppe einen Antwortsatz für einen Spezialfall anlegen (der z.B. auf eine von vier möglichen Antworten einer Frage zutrifft) und braucht dann nicht für alle weiteren Einzelfälle einen eigenen Antwortsatz anzulegen (was in diesem Beispiel die drei anderen Antworten der Frage wären), sondern nur einen Weiteren für einen allgemeineren Fall (hier also einen Antwortsatz, der die Frage nicht enthält).

Beendet wird der Algorithmus für einen Pfad, wenn das zu verwendende Blatt entsprechend der obigen Regeln eindeutig bestimmt ist (9). Dieses wird dann an der entsprechenden Stelle im Baum statt einer nächsten Frage eingefügt (10,11), wobei die letzte Antwort wieder der Kante zugeordnet wird (12).
Eine weitere Endbedingung für einen Pfad ist der Fall, dass es bereits Knowledge Base-Einträge mit passenden Antwortsätzen gibt und in der Menge von Blättern mit noch zu prüfenden Antwortsätzen nur noch Incident Templates vorhanden sind (9). Dies hat den Grund, dass nach der ersten Priorisierungsregel hier in jedem Fall der Knowledge Base-Eintrag als Blatt gewählt wird.

Außerdem wird für jeden inneren Antwortknoten noch eine leere Antwort als Kante eingefügt, falls an dieser Stelle im Pfad schon zutreffende Antwortsätze vorhanden sind. Wenn es hierbei mehrere mögliche Blätter mit passenden Antwortsätzen gibt, wird das entsprechende Blatt für die Kante nach den obigen Priorisierungsregeln ausgewählt (15,16,17). Somit kann der Intelligent Assistent eventuell ein Incident Template oder sogar einen Lösungsvorschlag

anbieten, obwohl der Nutzer eine Frage nicht beantworten kann.

Fordert man nun noch von der Eingabe an den hier definierten Algorithmus, dass keine Incident Templates und keine Knowledge Base-Einträge einen identischen Antwortsatz zugeordnet haben dürfen, ist der durch den Algorithmus erzeugte Baum eindeutig definiert. Der hier dargestellte Algorithmus wird für die im folgenden Kapitel beschriebene Implementierung verwendet.

4.3 Nutzer-Frontend

In diesem Abschnitt wird nun die Lösung für das Nutzer-Frontend innerhalb des Servicedesk-Portals entworfen. Der Entwurf erfolgt aufgrund der unterschiedlichen Anforderungen wieder separat für Service Requests und Incidents. Zur Auswahl ob ein Service Request oder ein Incident aufgegeben wird, werden dem Benutzer zwei separate Buttons angeboten, über die die in den folgenden Abschnitten beschriebene Funktionalität aufgerufen werden kann. Der Nutzer muss sich also zuerst bewusst entscheiden, ob er einen Incident oder einen Service Request aufgeben möchte. Dies sollte aber aufgrund der unterschiedlichen zugrunde-liegenden Ursachen für die beiden Meldungsarten, in Kapitel 3 als *Business Events* bezeichnet, auch für nicht technisch versierte Nutzer in den meisten Fällen kein großes Problem darstellen.

4.3.1 Service Requests

Die beschriebene Nutzerführung ist in Abbildung 4.5 als Aktivitätsdiagramm dargestellt und wird wieder durch die Zahlen in Klammern mit der folgenden Beschreibung verbunden.
Nachdem der Nutzer ausgewählt hat, dass er einen Service Request aufgeben möchte (1), bekommt er zuerst eine Übersichtsseite mit den für alle Dienste definierten Service Requests in alphabetischer Sortierung angezeigt (2). Er hat nun die Möglichkeit, nach einem zu seiner Anfrage passenden Service Request zu suchen und kann die Liste hierzu nach Dienst oder nach dem Namen des Service Requests filtern (3).
Findet der Nutzer den gesuchten Service Request in der Liste, kann er diesen auswählen und kommt zu einem Formular, dass alle dem Service Request zugeordneten Zusatzfragen untereinander anzeigt (4). Zusätzlich bekommt er das bisher vorhandene *Details*-Freitextfeld angezeigt, um noch eigene Angaben zu machen, die für den Service Request nicht explizit erfragt werden. Nach Beantworten der Fragen (5) kann der Nutzer durch einen Bestätigungsbutton die Generierung eines Incident-Datensatzes auslösen, wobei überprüft wird, dass auch alle Pflichtangaben vom Nutzer ausgefüllt wurden.
Der generierte Incident-Datensatz enthält im *Details*-Feld dann zuerst alle durch den Intelligent Assistent erfassten Angaben und danach optisch getrennt die Zusatzangaben des Nutzers aus dem *Details*-Freitextfeld bei der Erfassung.
Die durch den Intelligent Assistent strukturiert erfassten Antworten werden nach der Erfassung also wieder unstrukturiert gespeichert. Da aktuell noch kein sichtbarer Bedarf besteht, diese strukturiert zu speichern, ist dies eine Designentscheidung um die Lösung möglichst simpel zu halten. Sollte ein entsprechender Bedarf in Zukunft entstehen, ließe sich die Speicherung jedoch leicht anpassen.

Findet der Nutzer hingegen keinen passenden Service Request in der Liste, hat er die Möglichkeit, seinen Wunsch wie gehabt über das *Details*-Freitextfeld komplett als Freitext zu formulieren (6,7). Diese Funktion ist vor allem für die Übergangsphase wichtig, in der

noch nicht alle Standard-Service Requests von den Support-Gruppen im Intelligent Assistent hinterlegt sind.

Die hier erarbeitete Lösung erfüllt die Anforderung *NR3: Gute Bedienbarkeit für unterschiedliche Nutzergruppen* aus Absatz 3.4, da nicht technisch versierte Nutzer durch die Anzeige der Liste eine Übersicht über alle Service Requests erhalten und erfahrene Nutzer durch den Freitext-Namensfilter einen bekannten Service Request sehr schnell auswählen können.

4.3.2 Incidents

In Abbildung 4.6 ist die im Folgenden beschriebene Dialogführung wiederum als Aktivitätsdiagramm dargestellt.
Wählt der Nutzer aus, dass er einen Incident erfassen möchte (1), erhält er, wie bereits im Servicedesk-Portal implementiert, zuerst einen Auswahldialog für den Dienst (2). Nachdem er einen Dienst ausgewählt hat (3), prüft der Intelligent Assistent, ob ein Entscheidungsbaum für den ausgewählten Dienst hinterlegt ist. Falls nein, wird ein Erfassungsdialog mit einem Freitext-Eingabefeld für die Beschreibung und einem für die Details des Incidents angezeigt (was auch dem vor dieser Arbeit implementierten Verhalten entspricht), im Folgenden *Default-Erfassungsdialog* genannt (4,5). Falls ein Entscheidungsbaum hinterlegt ist, beginnt der Intelligent Assistent vom Wurzelknoten aus, dem Nutzer Fragen zu stellen (6). Mit der Antwort des Nutzers (7) kann im Baum jeweils die nächste Frage für die Anzeige ermittelt werden (8). Der Nutzer erhält hierbei in jedem Schritt die Möglichkeit, die Dialogführung explizit abzubrechen (Eine “Weiß ich nicht”-Antwort auf eine Frage wird hierbei nicht als expliziter Abbruchwunsch, sondern als eine leere Antwort gewertet).
Die Dialogführung wird durch das Eintreten eines der folgenden Fälle beendet:

- **Fall 1**: Der Nutzer bricht die Dialogführung ab.
  Reaktion des Intelligent Assistent: Der Nutzer erhält den Default-Erfassungsdialog angezeigt (4). Im Details-Feld des Incident-Datensatzes werden alle bis zum Abbruch der Dialogführung gegebenen Antworten und die Eingabe aus dem Default-Erfassungsdialog erfasst.

- **Fall 2**: Für eine Antwort des Nutzers existiert keine Kante im Baum.
  Reaktion des Intelligent Assistent: Wie in Fall 1.

- **Fall 3**: Der Intelligent Assistent erreicht ein Blatt (9).
  Reaktion des Intelligent Assistent: Falls das Blatt ein Knowledge Base-Eintrag ist, wird dem Nutzer der Lösungsvorschlag angezeigt (10). Er bekommt die Auswahl, die Lösung als zutreffend zu akzeptieren und damit die Erfassung abzubrechen, oder die Lösung abzulehnen. Lehnt er die Lösung ab, erhält er wieder den Default-Erfassungsdialog (4). Im Details-Feld des Incident-Datensatzes werden dann die im Dialog gegebenen Antworten, die Ablehnung der Knowledge Base-Lösung und die Daten aus dem Default-Erfassungsdialog vermerkt.
  Ist das Blatt hingegen ein Incident Template, wird analog zur Erfassung eines ausgewählten Service Requests verfahren (11,12). Im Details-Feld des Incident-Datensatz werden dann natürlich wieder zusätzlich die Antworten aus der Dialogführung eingetragen.

Die Abbruchmöglichkeit für den Nutzer ist hierbei als Feature explizit vorgesehen, um erfahrenen Nutzern bei der Erkenntnis, dass die Entscheidungsführung ihr Problemgebiet nicht abdeckt, direkt das Überspringen der weiteren Fragen zu ermöglichen. Dieses Feature dient daher dem Einhalten der Anforderung *NR3: Gute Bedienbarkeit für unterschiedliche Nutzergruppen* aus Absatz 3.4.
Auch kann aufgrund der bisherigen Darstellungen geschlossen werden, dass die für Incidents und Service Requests angegebenen Erfassungsarten alle in Absatz 3.3 aufgelisteten *Fuctional Requirements* für die Komponente *Nutzer-Frontend* erfüllen.

4.4 Zusammenfassung

In diesem Kapitel wurde zuerst die Softwarearchitektur, die aus der Wissensverwaltungs-Komponente und der Nutzer-Frontend-Komponente besteht, erarbeitet. Danach erfolgte eine Lösungsfindung für die Wissensverwaltungs-Komponente, für die zuerst eine Art der Wissensrepräsentation und dann ein Algorithmus für die Wissensverarbeitung entwickelt wurde, mit dem ein Entscheidungsbaum generiert wird. Auf Basis dieses Entscheidungsbaums wurde dann die Dialogführung in der Nutzer-Frontend-Komponente für Incidents und Service Requests entwickelt und auf die Einhaltung der Anforderungen aus Kapitel 3 geprüft.

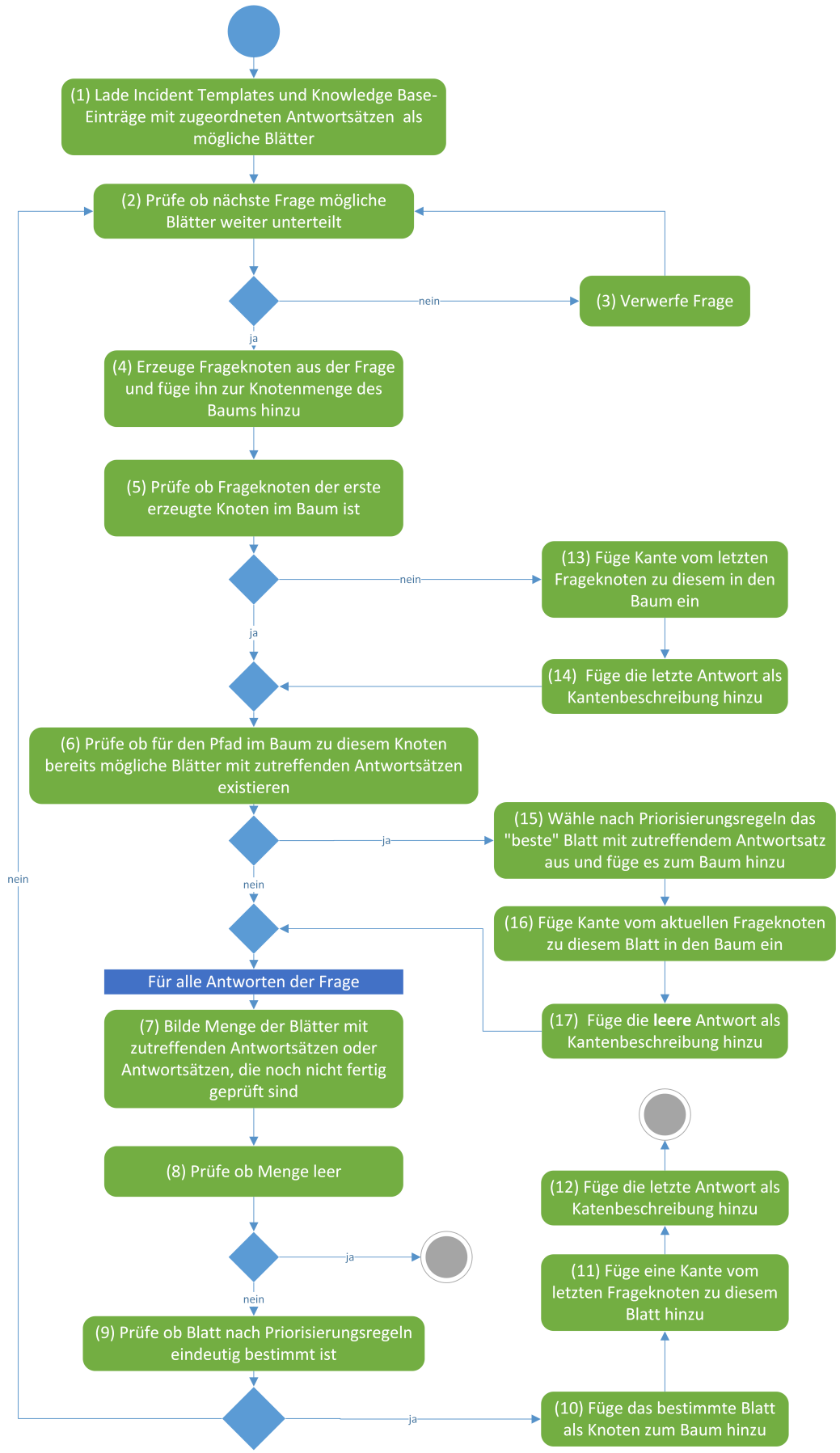


Abbildung 4.4: Selbst entwickelter Algorithmus zur Generierung eines Entscheidungsbaums aus der Wissensbasis. (Quelle: Eigene Darstellung)

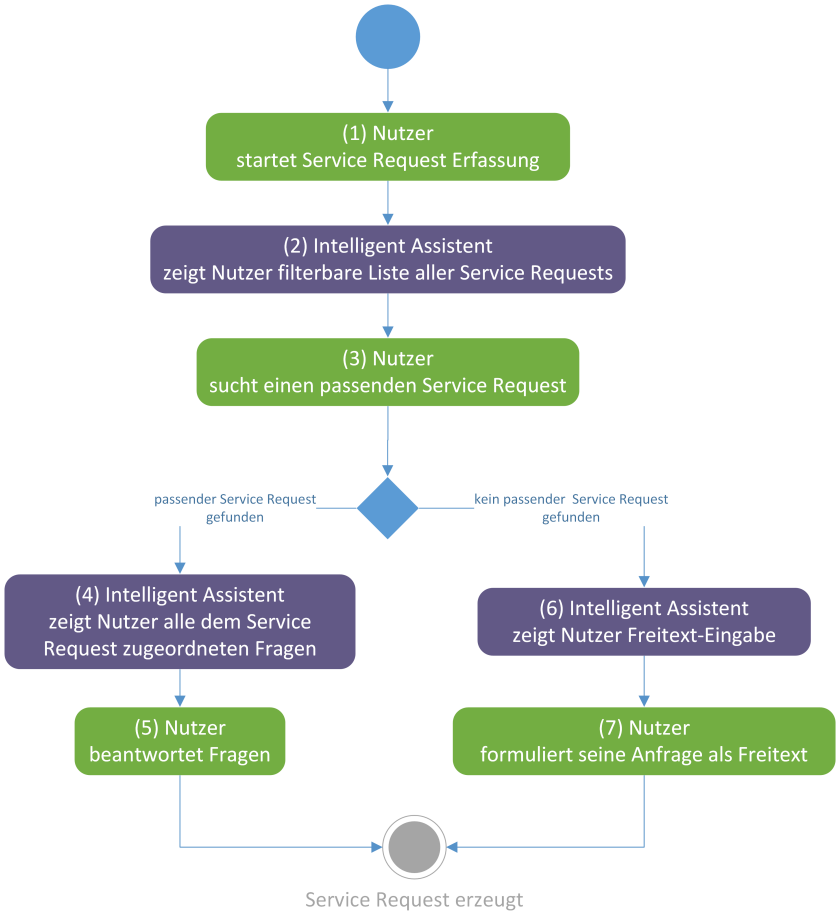


Abbildung 4.5: Service Request-Erfassung als Aktivitätsdiagramm. (Quelle: Eigene Darstellung)

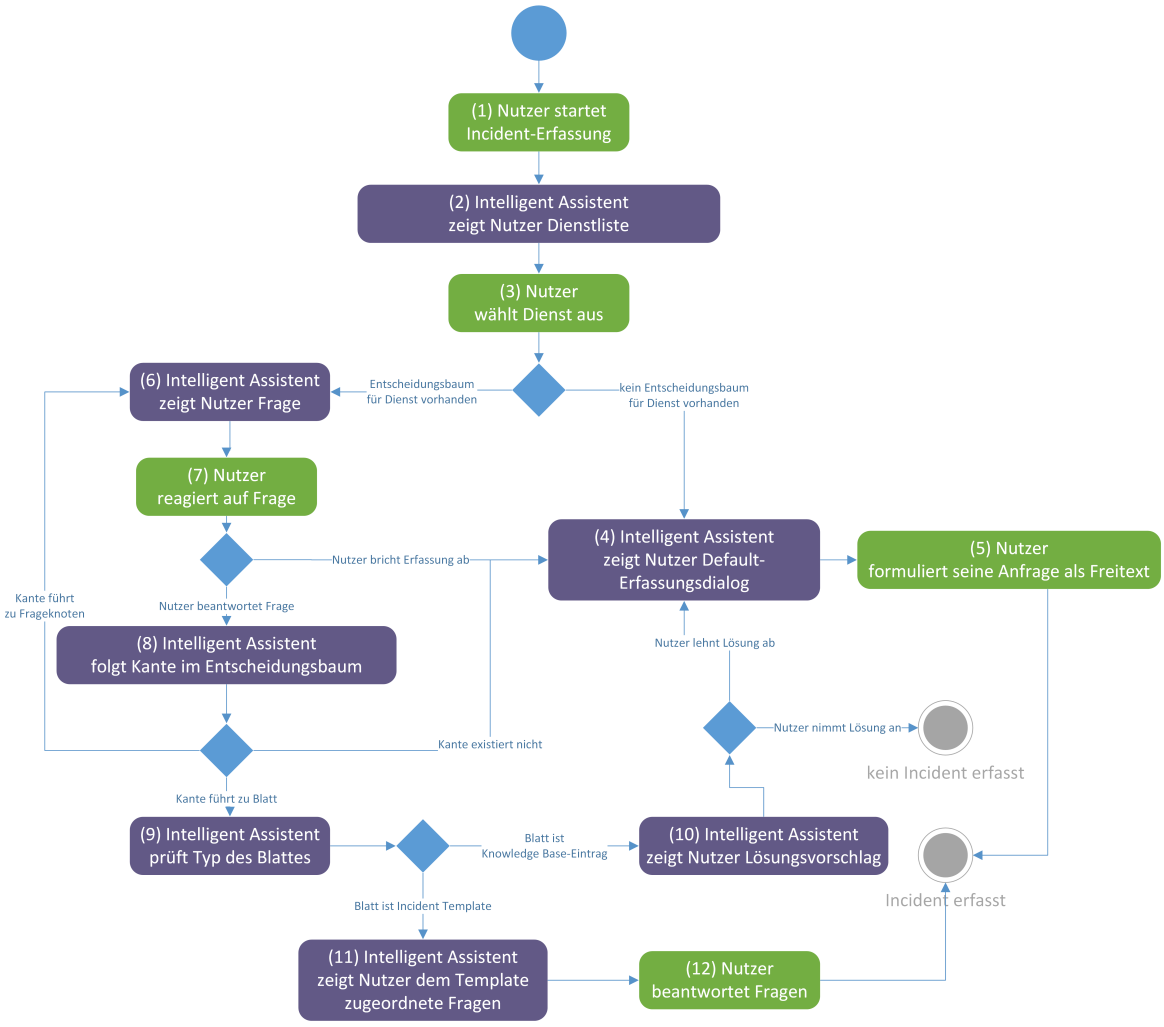


Abbildung 4.6: Incident-Erfassung als Aktivitätsdiagramm. (Quelle: Eigene Darstellung)

5 Implementierung

Im Folgenden wird die Implementierung der in Kapitel 4 entwickelten Lösung für den Intelligent Assistent beschrieben. Da eine Lösung gewählt wurde, die aus zwei Komponenten besteht, wird auch die Beschreibung in diesem Kapitel entsprechend aufgeteilt. Zuerst kommt hierbei die Implementierung der Wissensverwaltungs-Komponente, danach wird kurz auf die konkrete Implementierung der Datenübertragung zwischen den beiden Komponenten mittels der in Abschnitt 4.1 hierfür ausgewählten Webservices-Schnittstelle eingegangen. Zuletzt erfolgt dann eine Beschreibung der Implementierung der Frontend-Komponente.

5.1 Wissensverwaltung im Workcenter

Die Wissensverwaltungs-Komponente wird entsprechend der Designentscheidung in Abschnitt 4.1 innerhalb der Workcenter-Anwendung realisiert. Da der Intelligent Assistent hier ein komplett neues Feature darstellt, wird er im Workcenter als Applikation mit dem Namen *Intelligent Assistent* umgesetzt. Dies bedeutet, dass er als eigene Option in der im Workcenter angezeigten Applikationsliste aufgeführt wird und somit leicht gefunden werden kann. Im Folgenden wird nun zuerst die Menüführung beschrieben und danach auf das Rechtemanagement innerhalb der Applikation eingegangen. Hierauf folgt ein Blick auf das zugrunde liegende Datenbanklayout für die verwaltete Wissensbasis und den daraus generierten Baum. Als letzter Punkt wird kurz die Realisierung der Entscheidungsbaum-Generierung beschrieben.

5.1.1 Formulare und Benutzerführung

Startpunkt der *Intelligent Assistent*-Applikation ist ein zentrales Formular mit dem Namen *Fragen- und Zuordnungsverwaltung*. Dieses Formular ist auch das einzige Element, dass in der *Intelligent Assistent*-Applikation über die seitliche Navigation des Workcenters direkt aufgerufen werden kann. Nachdem die Wissensverwaltung entsprechend Abschnitt 4.2 getrennt nach Dienst verwaltet wird, muss auf diesem Formular nun zuerst ein Dienst ausgewählt werden. Danach werden die anderen Elemente des Formulars sichtbar. Die nach Auswahl eines Dienstes dargestellte Ansicht ist in Abbildung 5.1 dargestellt. Das Formular wird durch 6 Reiter unterteilt, deren Funktionen nun beschrieben werden.

Der erste Reiter mit dem Namen *Fragen* dient der Verwaltung der beiden Fragearten, die in Abschnitt 4.2.1 definiert wurden, wobei die Entscheidungsfragen hier als *Klassifizierungsfragen* bezeichnet werden. Die Reihenfolge der Fragen lässt sich durch Benutzung der Buttons mit den Pfeilen nach oben und unten durch Mitglieder der Support-Gruppe verändern. Um Fragen anzulegen oder zu bearbeiten existiert ein weiteres Formular, die *Fragenverwaltung*. Diese lässt sich über die entsprechenden Buttons unter der Fragenliste aufrufen. Dort lassen sich Fragetext und Antwortmöglichkeiten, sowie ein Ausfüllhinweis,

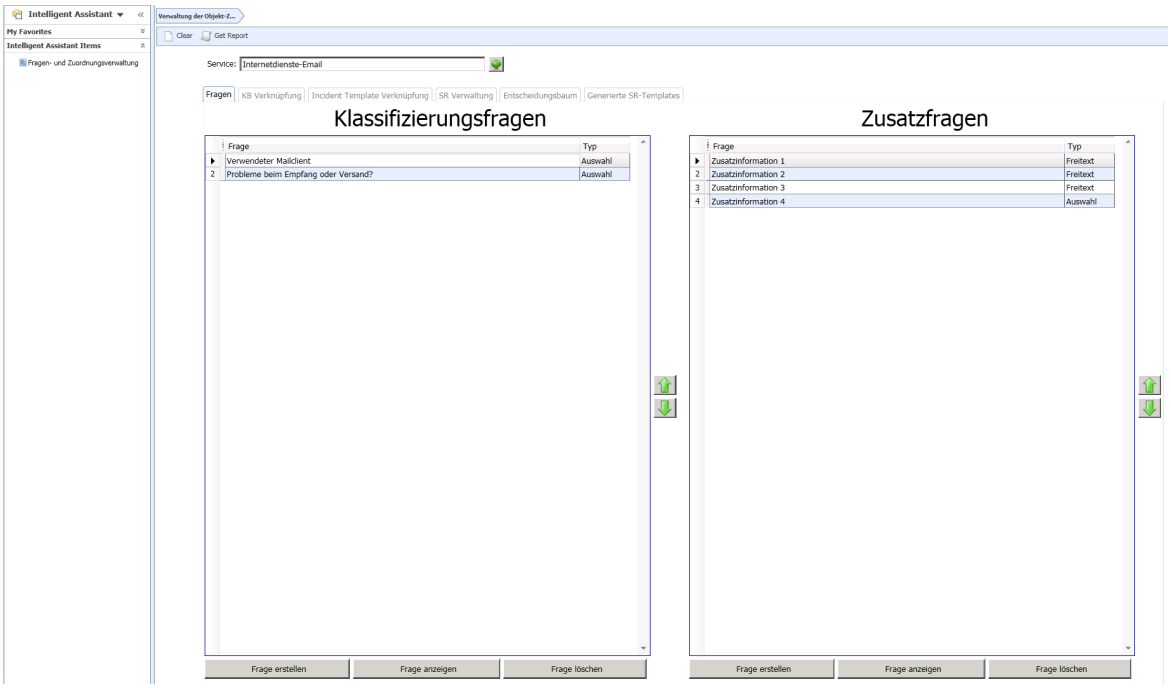


Abbildung 5.1: Screenshot des Formulars *Fragen- und Zuordnungsverwaltung* im Workcenter nach Auswahl eines Dienstes. (Quelle: Eigene Darstellung)

angeben (siehe Abbildung 5.2). Dies ist auf Deutsch und Englisch möglich, um der Anforderung *NR5: Zweisprachiger Erfassungsdialog* aus Abschnitt 3.4 gerecht zu werden.

Im Reiter *KB Verknüpfung* können die in Abschnitt 4.2.1 definierten Antwortsätze angelegt und Einträgen aus der Knowledge Base zugeordnet werden. Nachdem in Abschnitt 2.2.3 beschrieben wurde, dass in der Knowledge Base nicht nur Lösungen gespeichert werden, nimmt diese Ansicht gleich eine Filterung der Knowledge Base vor. Nach dieser Filterung werden nur aktuelle Lösungen für die Verknüpfung angeboten. Sollten Knowledge Base-Einträge, die durch den Filter aussortiert würden bereits zugeordnete Antwortsätze besitzen (da sie z.B. zur Zeit der Zuweisung noch nicht ausgefiltert wurden), werden diese weiterhin eingeblendet. Allerdings werden solche Knowledge Base-Einträge dann farblich entsprechend auffällig in der Liste markiert und auch nicht mehr für die Baumgenerierung verwendet.

Der Reiter *Incident Template* funktioniert ähnlich wie der *KB Verknüpfung*-Reiter, allerdings lassen sich hier Incident Templates auch anlegen, bearbeiten und löschen, da diese nicht wie die Knowledge Base-Einträge bereits vorher als Datenstrukturen im Workcenter implementiert waren, sondern durch diese Arbeit in Abschnitt 4.2.1 neu eingeführt wurden. Das Layout des Reiters ist in Abbildung 5.3 dargestellt.

Im Reiter *SR Verwaltung* lassen sich, analog zu Incident Templates, Service Request-Templates definieren, allerdings ohne die Zuordnung von Antwortsätzen, da diese in der in Abschnitt 4.2.1 beschriebenen Wissensrepräsentation für Service Requests nicht vorgesehen

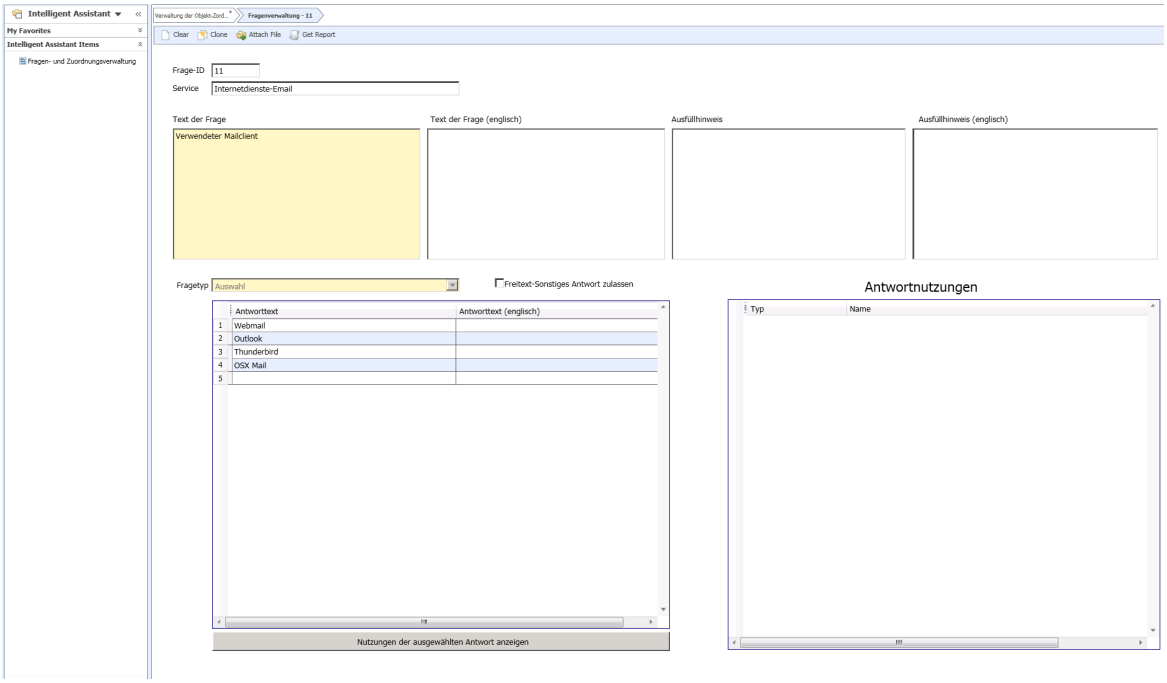


Abbildung 5.2: Screenshot der Fragenverwaltung in der Wissensverwaltungs-Komponente. (Quelle: Eigene Darstellung)

sind.
Die bisher beschriebenen Reiter erfüllen also zusammen die Anforderungen *FR3: Angabemöglichkeit der vom Nutzer benötigten Informationen* und *NR2: Einfache Administrierbarkeit der Wissensbasis* aus Kapitel 3.

Über die Reiter *Entscheidungsbaum* und *Generierte SR-Templates* wiederum lassen sich aus der Wissensbasis der Entscheidungsbaum und die Service Request-Liste für die Dialogführung in der Frontend-Komponente generieren. In beiden Fällen ist es möglich, erst ein *Preview* zu erstellen, damit die Support-Gruppe vor Veröffentlichung einer Veränderung prüfen kann, ob diese auch ihren Vorstellungen entspricht. Über diese beiden Reiter wird also die Realisierung der Anforderung *FR4: Generierung und Visualisierung einer Entscheidungs-Datenstruktur für die Datenerfassung im Nutzer-Frontend* aus Abschnitt 3.3 vorgenommen. Die generierten Entscheidungsbäume werden hierbei über die bereits in Abschnitt 4.2.1 angesprochenen *Graphical Explorer*-Objekte visualisiert (siehe Abbildung 5.4). Die generierten Service Request Templates wiederum werden als Liste dargestellt.
Diese Visualisierung dient auch der Erfüllung der Anforderung *NR1: Nachvollziehbarkeit der vom Nutzer erfragten Informationen.*

5.1.2 Berechtigungen

Nach dem Formularentwurf stellt sich natürlich die Frage, welchen Mitarbeitern am LRZ welche Berechtigungen in der implementierten Applikation zu gewähren sind. Aus den Anforderungen *FR3: Angabemöglichkeit der vom Nutzer benötigten Informationen* und *FR4: Ge-*

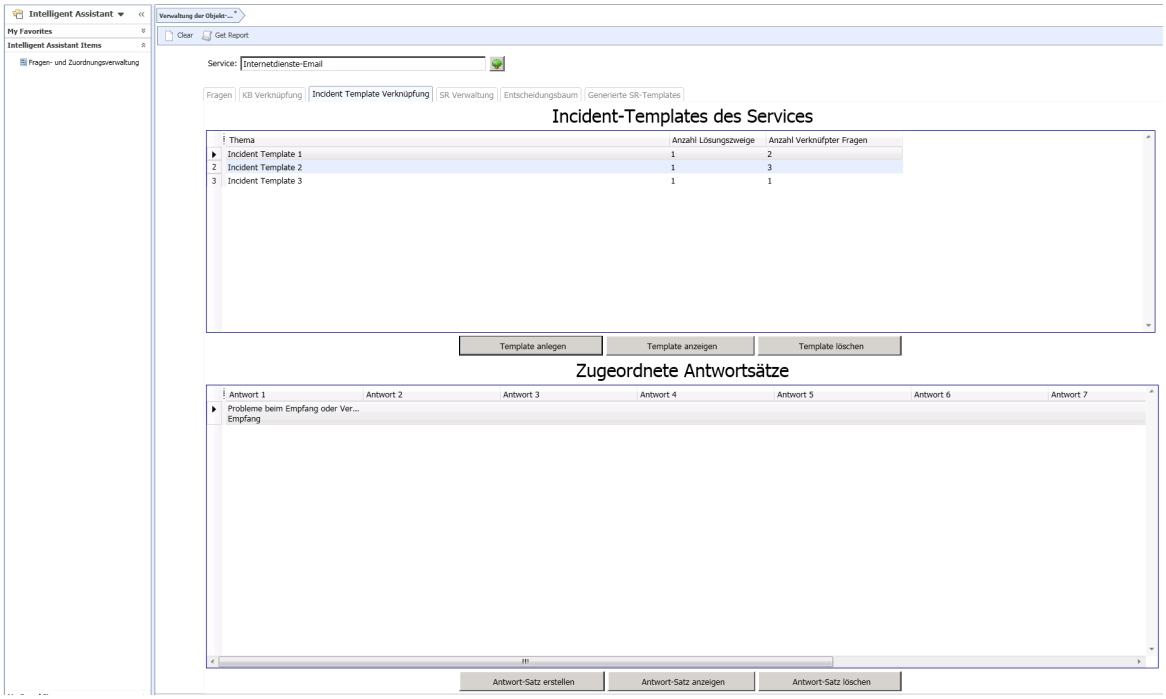


Abbildung 5.3: Screenshot der Incident Template-Verwaltung in der Wissensverwaltungs-Komponente. (Quelle: Eigene Darstellung)

nerierung und Visualisierung einer Entscheidungs-Datenstruktur für die Datenerfassung im Nutzer-Frontend lässt sich ableiten, dass die Mitglieder der Support-Gruppe eines Dienstes Wissen der Wissensbasis verändern können müssen. Außerdem sollen sie ein Preview der im Rahmen der Wissensverarbeitung zu generierenden Datenstrukturen erstellen können. Nachdem die Support-Gruppe aber auch der einzige Personenkreis ist, der für die Veränderung der Wissensbasis zuständig ist, ist es also nicht sinnvoll, weiteren Mitarbeitern des LRZ Zugriffsberechtigungen hierauf zu geben. Eine einzige Ausnahme bildet die Support-Gruppe zur Verwaltung der IT-Service-Management-Software, um gegebenenfalls technische Unterstützung bei der Verwendung des Intelligent Assistent leisten zu können. Andere Mitarbeiter des LRZ erhalten hingegen nur lesenden Zugriff auf die Listen und Zuordnungen, sowie die Möglichkeit, die generierten Entscheidungsbäume und Service Request-Listen anzusehen. Die Berechtigung, ein generiertes *Preview* eines Entscheidungsbaums bzw. einer Service Request-Liste als Produktivversion zu übernehmen, erhält entsprechend Anforderung *NR4: Autorisierung von Änderungen nur durch Incident-Management-Team* aus Abschnitt 3.3 nur das Incident-Management-Team.
Die genannten Zugriffsbeschränkungen werden hierbei durch Deaktivieren der entsprechenden Buttons auf dem *Fragen- und Zuordnungsverwaltung*-Formular realisiert.

5.1.3 Datenbanklayout

Im mit iET Enterprise mitgelieferten Editor lässt sich auch indirekt die von iET Enterprise verwendete Datenbank bearbeiten. Dies geschieht durch Anlegen oder Modifizieren soge-

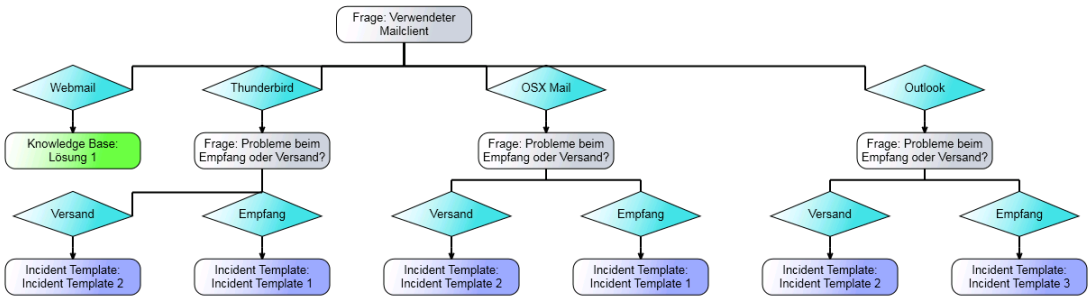


Abbildung 5.4: Ein Entscheidungsbaum, der mit einem Graphical Explorer dargestellt ist. Fragen sind grau, Lösungen aus der Knowledge Base grün, Incident Templates blau und Kantenbeschriftungen türkis eingefärbt. (Quelle: Eigene Darstellung)

nannter Data-Dictionary Elemente. Da hierdurch aber letztendlich die entsprechenden Elemente in der Datenbank angelegt bzw. verändert werden [iet11a, Folie 2-7 ff], wird in diesem Abschnitt der Einfachheit halber direkt das Datenbanklayout anstatt der Data-Dictionary Elemente betrachtet.
Dieses unterteilt sich wiederum in die für die Wissensverwaltung verwendeten Tabellen und Tabellen, die für die in der Wissensverarbeitung generierten Datenstrukturen verwendet werden. Diese komplette Unterteilung ist eine sehr wichtige Eigenschaft des Layouts, da sie verhindert, dass durch Modifikationen an der Wissensbasis die der Frontend-Komponente zur Verfügung gestellten Daten ohne explizit neu durchgeführte Wissensverarbeitung und Autorisierung durch das Incident-Management-Team verändert werden können. Hiermit unterstützt das Datenbanklayout also die Einhaltung der Anforderung *NR4: Autorisierung von Änderungen nur durch Incident-Management-Team*. Allerdings entsteht hierdurch auch eine gewisse Datenredundanz, da auch bei der Wissensverarbeitung nicht veränderte Strukturen wie Zusatzfragen nach Durchführung der Wissensverarbeitung mehrfach in der Datenbank vorhanden sind. Aufgrund der geringen Größe der einzelnen Datensätze ist dies aber nicht als Problem anzusehen. Im Folgenden wird eine kurze Übersicht über die angelegten Tabellen und ihre Verwendung gegeben. Das Präfix *lrz_ia* im Namen erleichtert hierbei die Zuordnung der Tabelle zur Intelligent Assistent-Applikation im Workcenter bei Betrachten der Datenbank.
Die für die Wissensverwaltung angelegten Datenbank-Tabellen sind in Tabelle 5.1 aufgelistet, die Datenbank-Tabellen für die Ergebnisse der Wissensverarbeitung in Tabelle 5.2.
Als Besonderheit bei der Speicherung der Service Request-Listen ist noch zu nennen, dass diese intern auch als Baum gespeichert werden, bei dem die Service Request Templates Blätter der Wurzel sind. Die Wurzel ist hierbei eine Dummy-Frage, die nicht dargestellt wird. Diese Repräsentation wurde gewählt, um nicht noch weitere Datenbanktabellen nötig zu machen.

Name	Beschreibung
lrz_ia_questions	Speicherung der Entscheidungsfragen und Zusatzfragen inklusive ihrer Sortierreihenfolge.
lrz_ia_question_answers	Speicherung der Antwortmöglichkeiten von Fragen mit Auswahlmöglichkeit.
lrz_ia_incident_template	Speicherung von Incident Templates.
lrz_ia_service_req_template	Speicherung von Service Request Templates.
lrz_ia_answerset_leaf_map	Speicherung von Antwortsätzen und Zuordnung der Antwortsätze zu Incident Templates oder Knowledge Base-Einträgen.
lrz_ia_question_leaf_map	Zuordnung von Zusatzfragen zu Incident Templates und Service Request Templates.

Tabelle 5.1: Datenbank-Tabellen der Wissensverwaltung

Name	Beschreibung
lrz_ia_generated_tree	Speicherung der IDs von Entscheidungsbäumen und Service Request-Listen.
lrz_ia_tree_status	Zuordnung von Entscheidungsbäumen und Service Request-Listen zu Diensten.
lrz_ia_generated_nodes	Speicherung der Knoten von Entscheidungsbäumen und von Service Request Templates.
lrz_ia_generated_edges	Speicherung der Kanten von Entscheidungsbäumen.
lrz_ia_generated_addi_questions	Speicherung von Zusatzfragen inklusive ihrer Sortierreihenfolge.
lrz_ia_generated_answers	Speicherung der Antwortmöglichkeiten von Zusatzfragen und Frageknoten.
lrz_ia_generated_question_map	Zuordnung von Zusatzfragen zu Blättern des Entscheidungsbaums.

Tabelle 5.2: Datenbank-Tabellen für die im Rahmen der Wissensverarbeitung generierten Datenstrukturen

5.1.4 Generierung des Entscheidungsbaums

Die Generierung des Entscheidungsbaums wird mit Hilfe einer in Java geschriebenen Implementierung des in Abschnitt 4.2.2 beschriebenen Algorithmus vorgenommen. Hierzu wurden Klassen in Java erstellt, die in Anlehnung an das Konzept des *objekt-relational-mapping*, kurz ORM, Datensätze aus den Tabellen für die Speicherung der generierten Datenstrukturen repräsentieren [Wik14b]. Der Unterschied zu einem klassischen ORM ist jedoch, dass die Objekte der Klassen mit den Daten aus den Tabellen der Wissensverwaltung erstellt werden und somit nur beim Speichern auf die Tabellen für das Ergebnis der Wissensverarbeitung zurückgreifen. Hierbei fällt der Klasse *lrz_ia_DecisionTreeManager* eine Sonderrolle zu. Diese ist nicht zur Repräsentation von Daten gedacht, sondern enthält die Operationen zum Erstellen und Löschen der Entscheidungsbäume und Service Request-Listen unter Verwendung der anderen Klassen.
Ein Klassen-Diagramm der erstellten Klassen und ihrer Beziehungen untereinander fin-

det sich in Abbildung 5.5. An diesem Klassen-Diagramm lässt sich auch gut das Konzept zur Speicherung der generierten Datenstrukturen erkennen: Jede Klasse besitzt eine *dump*-Methode. In dieser sind die SQL-Statements zur Speicherung der relevanten Informationen definiert. Außerdem ruft die *dump*-Methode auch die *dump*-Methoden der jeweils assoziierten Objekte auf. Somit genügt ein *dump*-Aufruf auf einem Objekt der *lrz_ia_DecisionTree* Klasse, um den Baum komplett in die Datenbank zu schreiben.
Auch ist am Diagramm auffällig, dass die Attribute der Klassen zwei unterschiedlichen Namenskonventionen folgen. Die Attribute mit Unterstrichen im Namen sind hierbei Attribute, die aus den Datenbank-Tabellen gelesen werden und der Übersichtlichkeit halber nach den entsprechenden Datenbank-Spalten benannt sind. Die nicht aus der Datenbank geladenen Attribute der Klassen hingegen sind entsprechend der CamelCase Notation [Wik14a] benannt.

Zur Einbindung der Funktionalität der Java-Klassen in das GUI des Workcenters wird die im *iET Enterprise Application Developer's Guide* beschriebene Möglichkeit verwendet, für Formulare eine Java-Klasse anzugeben, auf der dann für bestimmte Formular-Ereignisse Callback-Methoden aufgerufen werden [iet11a, Folie 2-34]. Hiermit wird die Generierung nach einem Klick auf den entsprechenden Formularbutton durch Aufrufen der *updatePreviewTree*- bzw. *updatePreviewSrList*-Methode einer Instanz der *lrz_ia_DecisionTreeManager*-Klasse gestartet.

5.2 Datentransfer zwischen den Komponenten

Wie in Abschnitt 4.1 beschrieben, soll die Datenübertragung zwischen der Wissensverwaltungs-Komponente und der Frontend-Komponente über die iET Webservices SOAP-Schnittstelle erfolgen. Hierzu wird in der Webservices-Schnittstelle, die mittels der Workcenter-Anwendung verwaltet werden kann [Sch14a, S. 3], für die Übertragung der Entscheidungsbäume und der Service Request-Listen jeweils ein sogenannter Webservice angelegt. Dieser lädt mit den bei einem Aufruf übertragenen Eingabedaten ein Formular und übermittelt als Antwort eine definierte Menge von auf dem Formular angezeigten Daten als XML [Sch14a, S. 3].
Für die beiden Webservices wird jeweils ein Formular in iET Enterprise angelegt, das die für die Ausgabedaten benötigten Formularelemente enthält.
Als Eingabe für den Entscheidungsbaum-Webservice wird die ID eines Dienstes, die sein Primärschlüssel in der Datenbank ist, verwendet. Somit werden die Entscheidungsbäume der einzelnen Dienste durch getrennte Anfragen an die Schnittstelle abgerufen. Aufgrund der in Abschnitt 4.3.2 beschriebenen Trennung der Dialogführung nach Dienst ist für die Dialogführung trotzdem jeweils nur maximal eine Anfrage an die Webservices-Schnittstelle notwendig.
Die Service Request-Liste hingegen wird für alle Dienste bei einer Anfrage übertragen, da sie trotz der Trennung der Wissensbasis dem Nutzer entsprechend Abschnitt 4.3.1 als eine dienstübergreifende Liste dargestellt werden soll. Daher ist für diesen Webservice auch kein Eingabe-Parameter notwendig.
Insgesamt wurden die bei einer Anfrage an einen der beiden Webservices zu übertragenden Daten also so gewählt, dass für jeden Schritt der Nutzerführung im Intelligent Assistent maximal eine Anfrage nötig ist, bei der zudem eine möglichst geringe Menge an Daten übertragen wird. Dies unterstützt die Einhaltung der in Abschnitt 3.4 definierten Anforderung

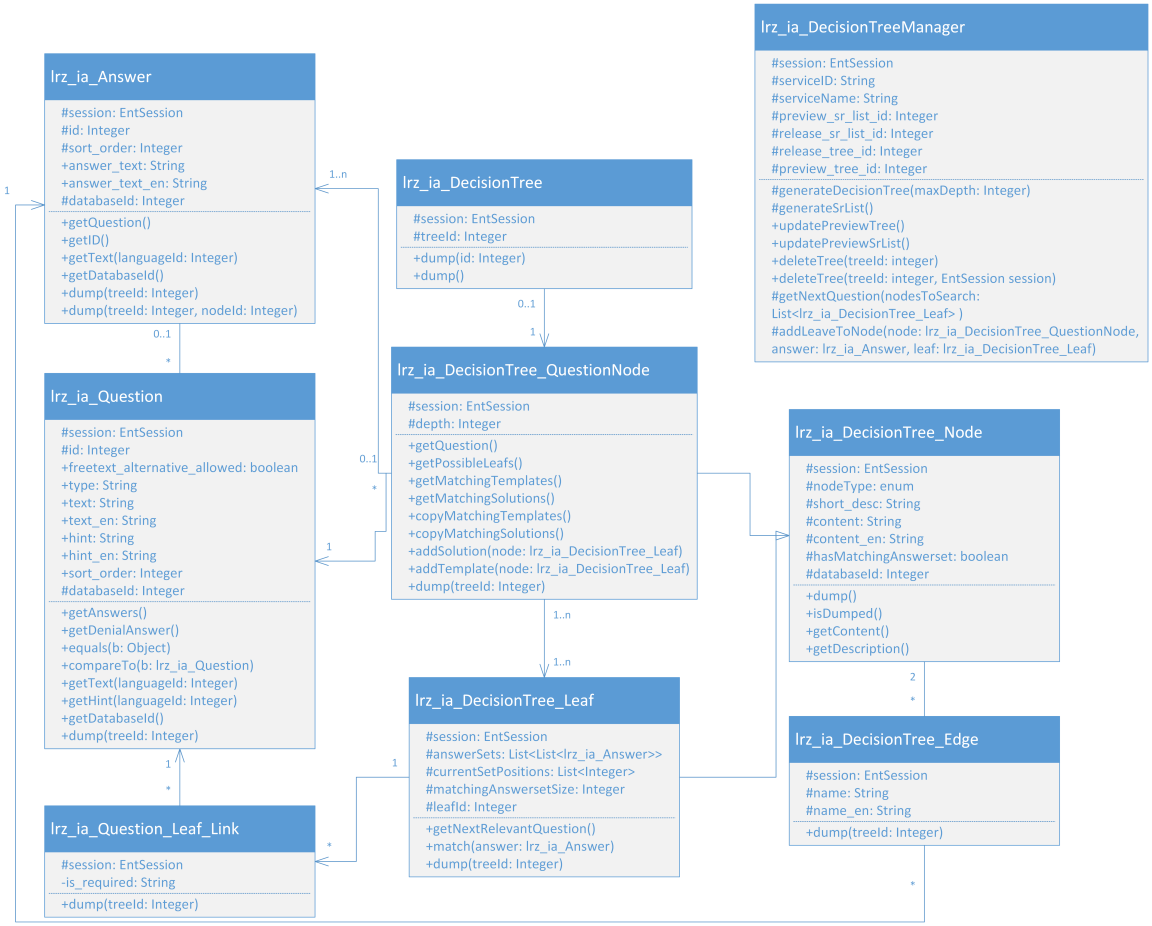


Abbildung 5.5: Klassendiagramm der für die Generierung von Entscheidungsbäumen und Service Request-Listen verwendeten Java-Klassen. (Quelle: Eigene Darstellung)

NR6: Maximale Ladezeit von einer Sekunde, da die Bearbeitungszeit von einer Anfrage an die Webservices Schnittstelle auf den Produktiv-Systemen des LRZ laut eigenen Messungen ohne andere Last bereits ca. 500ms benötigen kann.

5.3 Frontend-Komponente im Servicedesk-Portal

Nachdem nun bereits die Implementierung der Wissensverwaltungs-Komponente und die Einrichtung der Webservices-Schnittstelle in diesem Kapitel beschrieben wurden, bleibt noch die Implementierung der Frontend-Komponente im Servicedesk-Portal, mit der die aus dem Wissen generierten Datenstrukturen genutzt werden können.

Das Servicedesk-Portal ist in zwei unterschiedliche Nutzerinterfaces unterteilt, den Selfservice und das Simple Submit-Formular. Der Selfservice-Bereich erfordert hierbei eine Au-

thentifizierung des Nutzers mit LRZ-Kennung und Passwort und erlaubt neben dem Anlegen von Incident-Datensätzen auch das Ansehen bereits durch den Nutzer angelegter Incident-Datensätze. Außerdem sind die Aktivierung von Javascript und die Unterstützung des HTML5 File-Elements Voraussetzungen für die Nutzung des Selfservice-Bereichs.
Das Simple Submit-Formular hingegen kann auch ohne Authentifizierung, Javascript und HTML5 genutzt werden. Dafür lassen sich hierüber natürlich keine bereits angelegten Incident-Datensätze anzeigen, sondern nur Neue anlegen. [Sch14b, S. 3-5]

Im Rahmen dieser Arbeit wird der Intelligent Assistent für den Selfservice-Bereich implementiert. Da der Selfservice-Bereich und das Simple Submit Formular jedoch technisch im gleichen PHP-Backend mit dem Symfony2-Framework realisiert sind, ist eine nachträgliche Implementierung für das Simple Submit Formular leicht möglich.

Im Folgenden wird nun zuerst beschrieben, wie die Daten aus der Wissensverwaltungs-Komponente über die Webservices-Schnittstelle mit PHP abgerufen und verarbeitet werden. Danach wird auf die implementierte Dialogführung für die Erfassung von Service Requests und Incidents eingegangen.

5.3.1 Abruf der Daten für die Dialogführung

Zum Abruf der Daten über die Webservices-Schnittstelle können die bereits für den Selfservice implementierten ORM-Klassen genutzt werden. Dies erfolgt über Vererbung von der bereits vorhandenen *AbstractIetEnterpriseObject* PHP-Klasse. In der jeweiligen Kind-Klasse können dann Parameter wie der zu nutzende Webservice gesetzt werden. [Sch14b, S. 6-10]

Eine erste wichtige Beobachtung für die Umsetzung des Datenabrufs ist, dass sich die Daten aus der Wissensverwaltungs-Komponente im Gegensatz zu Incident-Datensätzen gut zwischenspeichern lassen, da sie sich zum einen nicht häufig ändern und es zum anderen in den meisten Fällen vertretbar ist, wenn entsprechende Änderungen erst mit einer gewissen Latenz durch das Servicedesk-Portal übernommen werden. Daher bietet es sich an, die von der Webservices-Schnittstelle empfangenen Daten für Entscheidungsbäume und die Service Request-Liste als XML-Datei zwischenzuspeichern.
Nachdem eine entsprechende Implementierung bereits für eine Klasse des Servicedesk-Portals vorhanden ist [Sch14b, S. 13], wurde sie im Rahmen dieser Arbeit als konfigurierbare Option in die *AbstractIetEnterpriseObject*-Klasse übernommen, sodass sie nun für alle über die Webservices-Schnittstelle abgerufenen Daten zur Verfügung steht.
Die Lebenszeit des Caches wird dabei mit 24 Stunden so eingestellt, dass sie einen guten Kompromiss aus Aktualität und Aufwand zur Aktualisierung des Caches bietet. Löst man nun noch, z.B. durch automatisches Laden der entsprechenden URLs, gegen 3 Uhr nachts die Aktualisierung des Caches aus, lässt sich die Wahrscheinlichkeit eines für den Nutzer wahrnehmbaren Effekts bei den Ladezeiten minimieren, da z.B. im Zeitraum vom 01.04.2014 bis 01.10.2014 nur 5 von 4270 Incidents, also 0,1%, nachts zwischen 02:00 Uhr und 04:00 Uhr über das Servicedesk-Portal erfasst wurden.

Für die Modellierung des Baumes wird in der Frontend-Komponente eine Implementierung verwendet, die von der Aufteilung der Klassen der in Abschnitt 5.1.4 beschriebenen Modellierung für die Wissensverarbeitung ähnelt. Das zugehörige Klassen-Diagramm ist in Abbildung

5.6 dargestellt. Zu beachten ist hierbei, dass die über die Webservices-Schnittstelle geladenen Attribute der Klassen im Diagramm nicht auftauchen, da sie nicht im Quelltext der Klassen definiert werden, sondern über die *AbstractIetEnterpriseObject*-Klasse dynamisch angelegt werden [Sch14b, S. 6-7]. Auch gut zu erkennen ist, dass die Klassennamen hier kein Präfix tragen, was durch die Nutzung des Namespace-Features von PHP begründet ist.
Die *DecisionTree*- und *ServiceRequestList*-Klasse übernehmen hierbei die Initialisierung der Objekte der anderen Klassen aus den über die Webservices-Schnittstelle empfangenen Daten. Außerdem werden diese beiden Klassen auch genutzt, um die Entscheidungsbäume und Service Request-Liste für die Nutzung im Symfony2-Framework zur Verfügung zu stellen.

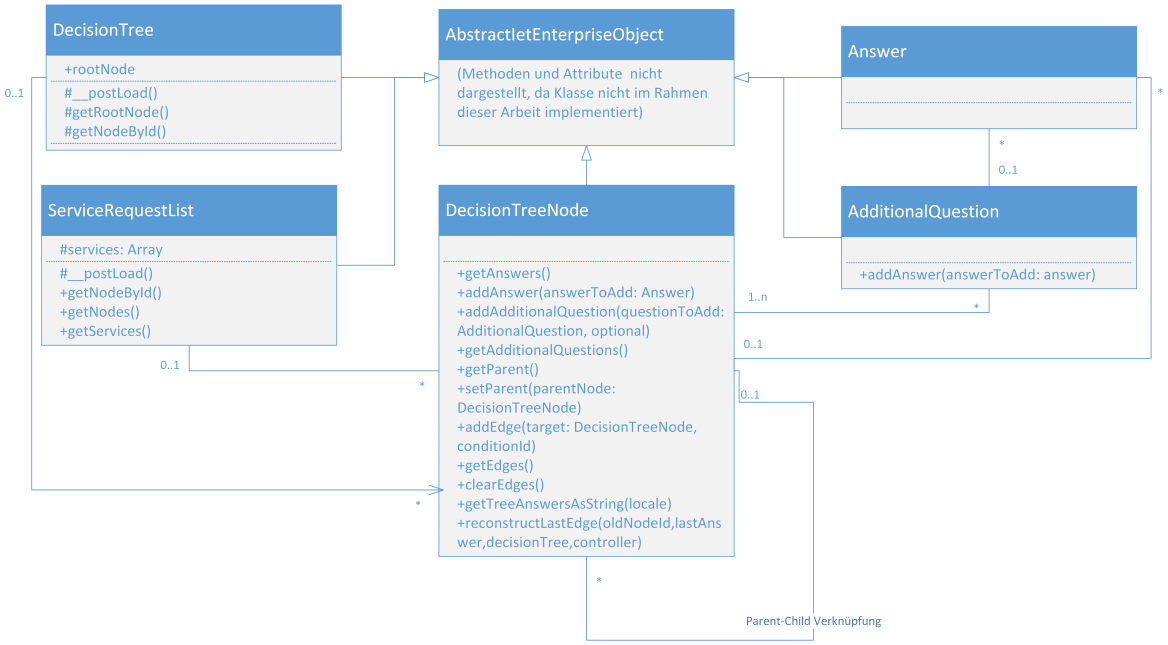


Abbildung 5.6: Klassendiagramm der in PHP verwendeten Klassen zur Repräsentation der Datenstrukturen aus der Wissensverarbeitung. (Quelle: Eigene Darstellung)

5.3.2 Service Request-Erfassung

Nachdem die aus dem Wissen generierten Datenstrukturen nun im Symfony2-Framework zur Verfügung stehen, können diese genutzt werden, um die Nutzerführung im Selfservice-Bereich des Servicedesk-Portals zu implementieren. Da es bisher nur einen Button zur Erfassung eines Incident-Datensatzes gab, ist hier der erste Schritt, den in Abschnitt 4.3 beschriebenen zweiten Button einzubauen. Dieser wird direkt auf der Startseite des Selfservice-Bereichs, unter dem Button zur Incident-Erfassung, platziert und erhält den Namen *Neuer Service Request*. Somit ist kein neues Untermenü zur Unterscheidung zwischen Incident und Service Request nötig. Klickt der Nutzer nun auf den *Neuer Service Request*-Button, erscheint ein Auswahlformular mit den in 4.3.1 beschriebenen Funktionalitäten. Der Nutzer kann hier aus einer nach Dienst und Namen filterbaren Liste ein Service Request Template auswählen (siehe Abbildung 5.7). Alternativ kann er hier für einen beliebigen Dienst einen Service

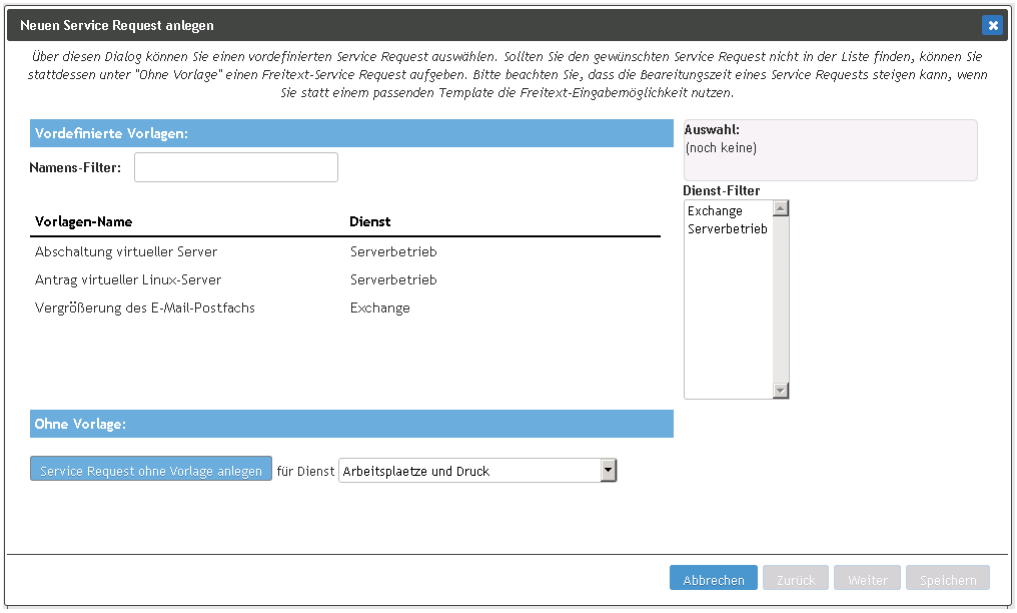


Abbildung 5.7: Screenshot des Dialogs zur Service Request-Erfassung. (Quelle: Eigene Darstellung)

Request ohne Template generieren.
Wählt der Nutzer ein Template und bestätigt durch Klick auf den *Weiter*-Button, erhält er ein Formular, das neben den normalen Eingabemöglichkeiten auch Eingabeelemente für die im Template definierten Zusatzfragen enthält (Abbildung 5.8). Dieses wird per Ajax unter Verwendung der ID des entsprechenden Template-Knotens vom Server abgerufen. Sollte das abzurufende Template nicht existieren, wird stattdessen das Standard-Formular mit Freitext-Eingabefeld vom Server ausgeliefert. Dieser Fall kann aber nur eintreten, wenn zwischen dem Anzeigen der Template Liste und der Auswahl eines Templates durch den Nutzer eine Auffrischung des in Abschnitt 5.3.1 beschriebenen Caches stattfindet und die Service Request-Liste seit der letzten Cache-Aktualisierung in der Wissensverwaltungs-Komponente neu erstellt wurde. Da aber auch in Abschnitt 5.3.1 beschrieben wurde, dass im Zeitraum für die Cache-Aktualisierungen nur extrem wenige Incident-Datensätze erfasst werden, rechtfertigt dies die Annahme, dass das Nutzererlebnis hierdurch nicht merklich beeinflusst wird.
Die Übertragung der jeweiligen Templates erfolgt hierbei bereits als formatiertes HTML, das auf Client-Seite per JQuery-Javascript-Bibliothek direkt in ein *div*-HTML-Element eingefügt wird.

Benutzt der Nutzer nun ein Service Request Template, ist das weiterhin angezeigte Freitext-Feld *Details* im Gegensatz zu einer Eingabe ohne Template kein Pflichtfeld mehr, da davon ausgegangen wird, dass alle vom Nutzer benötigten Informationen bereits über die Eingabeelemente für die Zusatzfragen erfasst werden. Trotzdem wird das Feld weiterhin angeboten, um dem Nutzer die Möglichkeit zu geben, weitere Informationen oder Mitteilungen hinzuzufügen. Aus den Antworten auf die Zusatzfragen und den im *Details*-Feld erfassten Informationen wird dann bei Generierung des Service Requests der Details-Text für den Incident-Datensatz generiert.

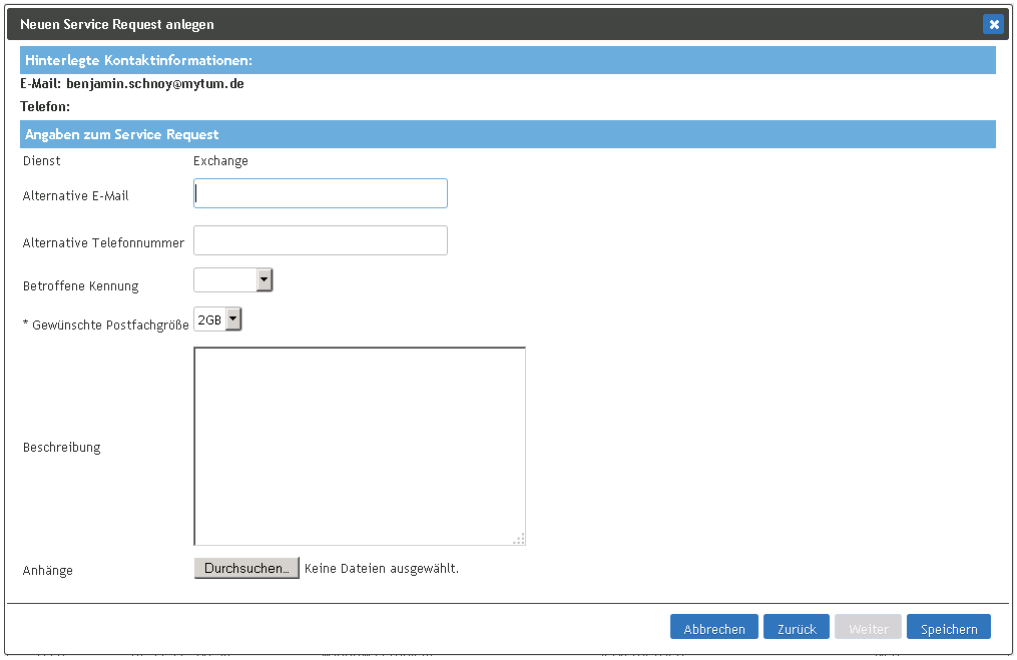


Abbildung 5.8: Screenshot eines vom Intelligent Assistent dargestellten Service Request Templates. Die Frage *Gewünschte Postfachgröße* stammt hierbei aus der Template-Definition. (Quelle: Eigene Darstellung)

Die hier dargestellte Implementierung entspricht damit dem Lösungsentwurf aus Abschnitt 4.3.1 und erfüllt somit auch die durch diesen abgedeckten Anforderungen.

5.3.3 Incident-Erfassung

Nach der Implementierung der Service Request-Erfassung wird nun die Implementierung der Incident-Erfassung beschrieben. Diese setzt im bereits bestehenden Erfassungsdialog für Incidents nach der Auswahl eines Dienstes aus dem Service-Baum an. Klickt der Nutzer hier den *Weiter*-Button, wird per Ajax der erste Knoten des Entscheidungsbaumes, also die Wurzel, abgerufen. Existiert hingegen kein Entscheidungsbaum für den gewählten Dienst, antwortet der Server mit dem in Abschnitt 4.3.2 erwähnten Default-Erfassungsdialog, der dem bisher eingesetzten Dialog zur Incident-Erfassung entspricht.
Die per Ajax abgerufenen Daten werden hierbei wieder als bereits formatiertes HTML übertragen und auf Seite des Clients in einen *div*-Container im Erfassungsfenster eingefügt. Der Client muss für den Abruf also nicht wissen, um was für einen Typ Knoten es sich handelt. Ist der Knoten eine Frage, wird dem Nutzer der Fragetext, ein in der Wissensverwaltungs-Komponente definierbarer Ausfüllhinweis und ein Dropdown-Menü mit den Antwortoptionen angezeigt (siehe Abbildung 5.9 für ein Beispiel). Dieses Dropdown-Menü ist über das *select*-HTML-Element realisiert. Als *value*-Attribut enthalten die entsprechenden *option*-Kindelemente des select-Elements die ID des Knotens, zu dem die durch die Antwort repräsentierte Kante im Entscheidungsbaum führt.
Nachdem es im Baum für Antwortsätze, die keinem Incident Template oder Knowledge

Base-Eintrag zugeordnet sind, auch keine Kanten gibt, erhalten entsprechende Antworten, die keine Kante im Baum besitzen, den Wert *0* als *value*-Attribut und verweisen so auf den Default-Erfassungsdialog. Zusätzlich enthalten die *option*-Kindelemente des Antwort-Dropdowns per HTML5-Data-Attribut [W3C14] noch die ID der Antwort. Diese wird bei einer Antwort, die zum Default-Erfassungsdialog führt, für die weiter unten beschriebene Rekonstruktion der Nutzerantworten in der Dialogführung benötigt.

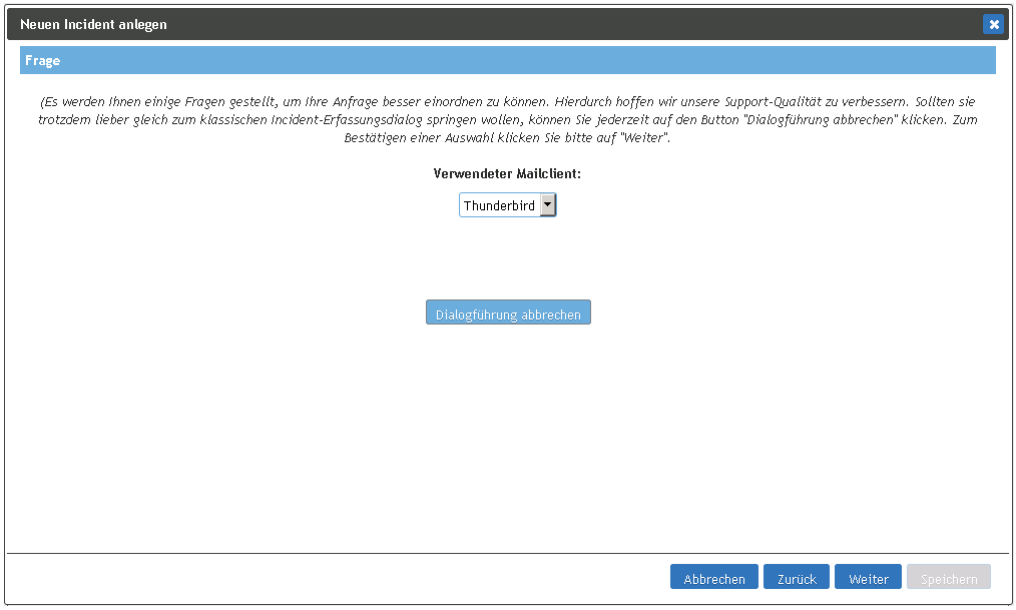


Abbildung 5.9: Screenshot einer Frage aus der Dialogführung des Intelligent Assistent. (Quelle: Eigene Darstellung)

Klickt der Nutzer nach Auswahl einer Antwort auf den *Weiter*-Button des Popup-Fensters, wird per HTTP GET-Request der nächste Knoten vom Server abgerufen. Hierzu werden im GET-Request die ID des Dienstes, die ID des momentan angezeigten Knotens, sowie die eben bereits erwähnte ID des zu ladenden Knotens und die ID der gegebenen Antwort als Parameter mitgesendet.
Auch ist, wie in Abbildung 5.9 zu sehen, der in Abschnitt 4.3.2 entworfene Button zum Abbruch der Dialogführung enthalten. Dieser erzeugt eine Lade-Anfrage an den Server, die 0 als ID für den nächsten Knoten und die ID der gegebenen Antwort enthält, also zur Anzeige des Default-Erfassungsdialogs führt.
Beantwortet der Nutzer die Frage hingegen und existiert eine Kante für die Antwort im Entscheidungsbaum des Dienstes, sendet der Server entweder den Inhalt eines weiteren Frageknotens, ein Incident Template, oder aber einen Lösungsvorschlag aus der Knowledge Base.

Erhält der Nutzer eine Lösung aus der Knowledge Base, wie in Abbildung 5.10 dargestellt, wird ihm diese mit einem Hinweis zum weiteren Vorgehen und den in Abschnitt 4.3.2 vorgesehenen Möglichkeiten zur Ablehnung oder Annahme dargestellt. Eine Annahme der Lösung löst eine Schließung des Dialogs ohne weitere Interaktion mit dem Server aus. Lehnt der Nut-

zer die Lösung hingegen ab, wird eine Anfrage zum Abruf des Default-Erfassungsdialogs mit dem Wert *0* für die ID der gegebenen Antwort an den Server gesendet.
Erreicht der Nutzer in der Dialogführung schließlich ein Incident Template oder den Default-

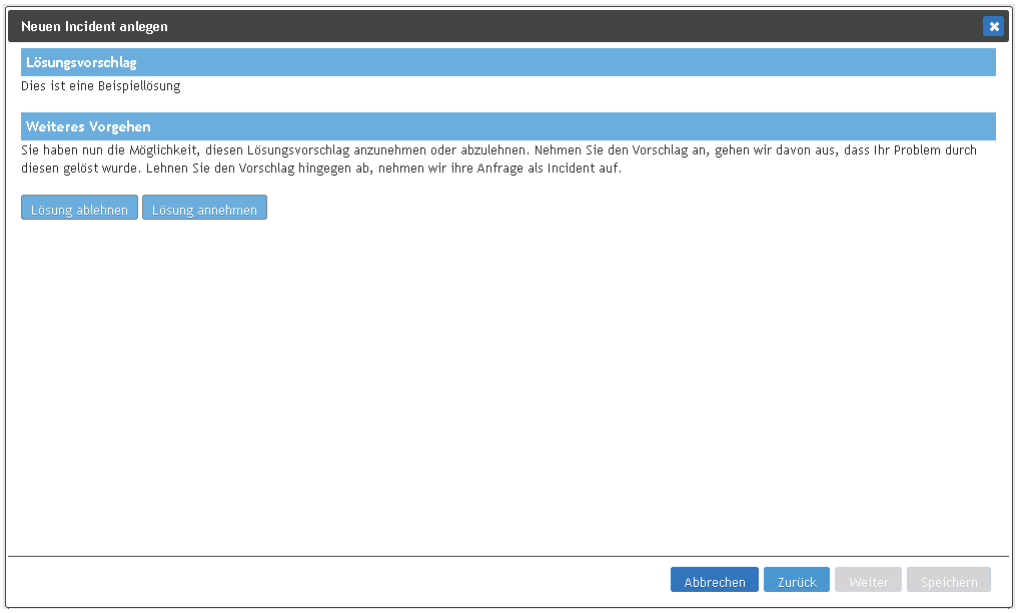


Abbildung 5.10: Screenshot der Lösungsanzeige aus der Dialogführung des Intelligent Assistent. (Quelle: Eigene Darstellung)

Erfassungsdialog, rekonstruiert der Intelligent Assistent anhand dem zum Blatt gehörigen Pfad im Baum die Antworten des Nutzers in der Dialogführung und stellt ihm diese dar (Abbildung 5.11). Sollte der Nutzer den Default-Erfassungsdialog erreicht haben, ist dies jedoch nicht direkt möglich, da der Default-Erfassungsdialog im Baum nicht vorhanden ist. Hier kann dann über die beim Abrufen jedes Knoten mitgesendete ID des letzten Knotens und der dort gegebenen Antwort jedoch rekonstruiert werden, von welchem Knoten im Entscheidungsbaum der Default-Erfassungsdialog erreicht wurde, um so die gegebenen Antworten des Nutzers zu erhalten.
Auch lässt sich so ermitteln und protokollieren, ob der Nutzer den Default-Erfassungsdialog durch Ablehnung einer Lösung oder durch Auswahl einer Antwort ohne zugeordnete Kante erreicht hat.

Nachdem der Nutzer alle benötigten Informationen des Incident Templates oder Default-Erfassungsdialogs eingegeben hat, kann er nun einen Incident-Datensatz generieren, der im Details-Feld die in Abschnitt 4.3.2 genannten Informationen enthält.

Die hier genannte Implementierung der Erfassung von Service Requests und Incidents erfüllt damit alle in Abschnitt 3.3 genannten Functional Requirements für die Nutzer-Frontend-Komponente sowie das in Abschnitt 3.4 genannte Nonfunctional Requirement *NR3: Gute Bedienbarkeit für unterschiedliche Nutzergruppen*

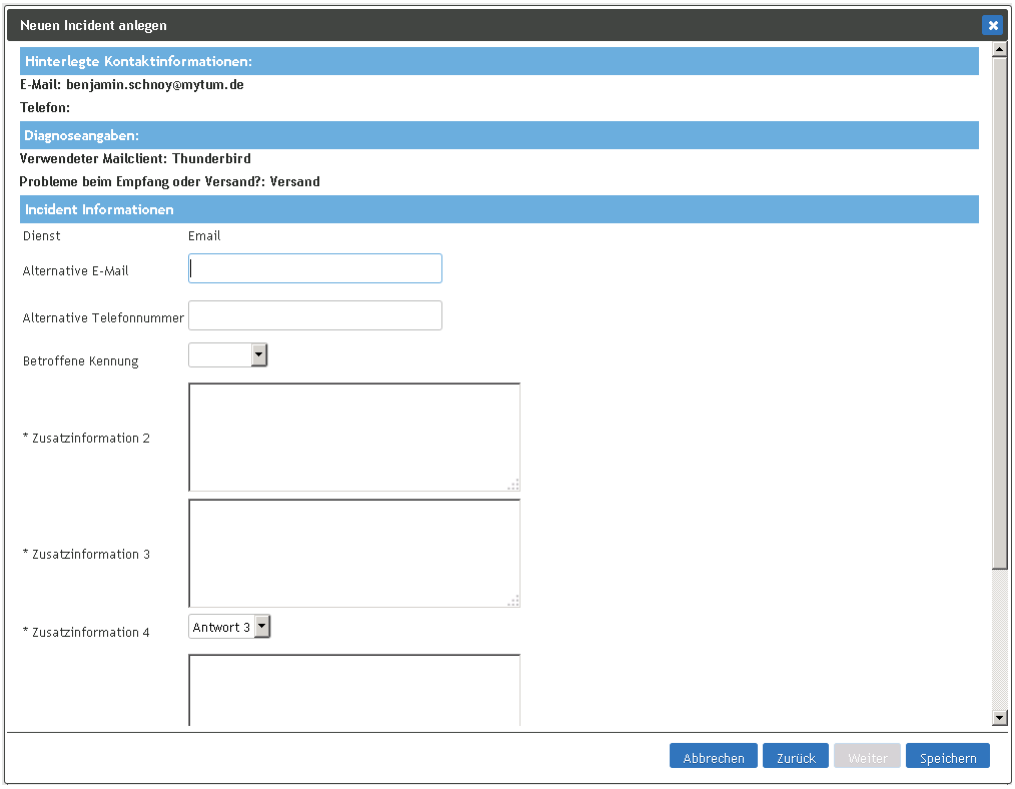


Abbildung 5.11: Screenshot eines vom Intelligent Assistent angezeigten Incident Templates. (Quelle: Eigene Darstellung)

5.3.4 Mehrsprachigkeit

Für das Servicedesk-Portal ist die Umschaltbarkeit zwischen Deutsch und Englisch, die im Nonfunctional Requirement *NR5: Zweisprachiger Erfassungsdialog* gefordert wird, bereits mit Hilfe der Übersetzungsfunktion des Symfony2-Frameworks realisiert. Mit dieser lassen sich Übersetzungen für verschiedene Sprachen in Übersetzungsdateien speichern. Die Übersetzungen werden dann über die *Translator*-Komponente des Symfony2-Frameworks in den Controller-Klassen und den für die Generierung des HTML-Codes genutzten Views zur Verfügung gestellt [Sen]. Diese Methode wurde für die Übersetzung sämtlicher Zeichenketten des Intelligent Assistent im Nutzer-Frontend verwendet, die nicht über die Webservices-Schnittstelle geladen werden.
Für die über die Webservices-Schnittstelle geladenen Daten lässt sich im Klassen-Diagramm der Wissensverwaltungs-Komponente (Abbildung 5.5) bereits erkennen, dass einige Member noch einmal mit dem Suffix *_en* vorhanden sind. Diese englischen Übersetzungen werden auch über die Webservices-Schnittstelle übertragen und stehen daher im Symfony2-Framework zur Verfügung. In den entsprechenden View-Dateien wird dann über ein *if*-Konstrukt entsprechend der gewählten Sprache entweder das englische oder deutsche Attribut ausgegeben. Hierbei wird bereits in der Wissensverwaltungs-Komponente dafür gesorgt, dass für nicht eingegebene englische Übersetzungen stattdessen die deutsche Übersetzung verwendet wird, da dies auch für einen Nutzer ohne deutsche Sprachkenntnisse einen höheren Informations-

gehalt bietet als die entsprechenden Elemente gar nicht darzustellen.
Somit erfüllt die dargestellte Implementierung auch die Anforderung *NR5: Zweisprachiger Erfassungsdialog.*

5.3.5 Ladezeiten

Als Anforderung an die Ladezeiten der Intelligent Assistent Lösung wurde in Abschnitt 3.4 das Nonfunctional Requirement *NR6: Maximale Ladezeit von einer Sekunde* definiert.
Um diese Anforderung zu erfüllen wurde in Abschnitt 5.2 eine Zwischenspeicherung der über die Webservices-Schnittstelle abgerufenen Daten implementiert, da die Bearbeitungszeit einer Anfrage an die Webservices-Schnittstelle, wie auch in Abschnitt 5.2 beschrieben, auf dem Produktivsystem ca. 500ms betragen kann. Durch diese Maßnahme ist serverseitig für die Ladezeit einer Anfrage an den Intelligent Assistent nur die Laufzeit des PHP-Codes im Symfony2-Framework entscheidend. Da das für die Implementierung verwendete Server-System bei der Anzahl der eingesetzten Server und der Ausstattung nicht dem Produktivsystem am LRZ entspricht, ist jedoch eine aussagekräftige Messung der Ladezeiten im Rahmen dieser Arbeit nicht möglich.
Stattdessen wird über Stichproben die Beobachtung gemacht, dass Anfragen an das Servicedesk-Portal, die keine Daten über die Webservices-Schnittstelle abrufen müssen, auf dem Produktivsystem Ladezeiten von weit unter einer Sekunde aufweisen.
Nachdem in den Funktionen der Nutzer-Frontend-Komponente des Intelligent Assistent keine aufwendigen Berechnungen durchgeführt werden, rechtfertigt dies die Annahme, dass sich diese Beobachtung auf den Intelligent Assistent übertragen lässt.
Daher kann mit hoher Wahrscheinlichkeit davon ausgegangen werden, dass die Anforderung *NR6: Maximale Ladezeit von einer Sekunde* erfüllt wird.

5.4 Zusammenfassung

In diesem Kapitel wurde die vorgenommene Implementierung der in Kapitel 4 entwickelten Lösung für den Intelligent Assistent zuerst für die Wissensverwaltungs-Komponente, dann für die Datenübertragung zwischen den beiden Komponenten und schließlich für die Frontend-Komponente beschrieben. Die Erfüllung der in Kapitel 3 beschriebenen Anforderungen wurde hierbei bis auf die Anforderung *NR6: Maximale Ladezeit von einer Sekunde* überprüft und festgestellt. Für die Anforderung *NR6: Maximale Ladezeit von einer Sekunde* wurde zumindest die begründete Einschätzung getroffen, dass auch diese erfüllt wird.
Eine Übersicht über die Erfüllung der Anforderungen ist in Tabelle 5.3 dargestellt.

Name der Anforderung	**Anforderung erfüllt?**	**Referenz-Abschnitte**
FR1: Strukturierte Erfassung aller für die Bearbeitung eines Incident-Datensatzes nötigen Daten	✓	5.3.3
FR2: Erstlösungsvorschlag bei bekanntem Konfigurations-/Bedienungsproblem	✓	5.3.3
FR3: Angabemöglichkeit der vom Nutzer benötigten Informationen	✓	5.1.1, 5.1.2
FR4: Generierung und Visualisierung einer Entscheidungs-Datenstruktur für die Datenerfassung im Nutzer-Frontend	✓	5.1.1
NR1: Nachvollziehbarkeit der vom Nutzer erfragten Informationen	✓	5.1.1
NR2: Einfache Administrierbarkeit der Wissensbasis	✓	5.1.1
NR3: Gute Bedienbarkeit für unterschiedliche Nutzergruppen	✓	5.3.2, 5.3.3
NR4: Autorisierung von Änderungen nur durch Incident-Management-Team	✓	5.1.2, 5.1.3
NR5: Zweisprachiger Erfassungsdialog	✓	5.3.4
NR6: Maximale Ladezeit von einer Sekunde	Erfüllung abgeschätzt, aber nicht überprüft	5.3.5

Tabelle 5.3: Übersicht über die Erfüllung der Anforderungen aus Kapitel 3.

6 Zusammenfassung und Ausblick

Im ersten Kapitel wurde die Motivation dieser Arbeit erarbeitet: Durch die Abfrage von Diagnose-Informationen bei der Incident-Datensatz-Erfassung über das Servicedesk-Portal des LRZ sollen direkt alle für die Bearbeitung von Incident-Datensätzen nötigen Informationen strukturiert erfasst werden oder dem Nutzer für bekannte Probleme gleich ein Lösungsvorschlag angeboten werden.
Um diese Motivation weiter zu begründen und die nötigen Rahmenbedingungen für eine Lösungsfindung zu erarbeiten, wurden dann im zweiten Kapitel das LRZ und die von diesem angebotene Dienstpalette sowie der Nutzerkreis vorgestellt. In diesem Zuge wurde auch darauf eingegangen, wie aktuell Incidents und Service Requests aufgenommen werden und welche Software das LRZ zur Unterstützung seiner IT-Service-Management Prozesse einsetzt.
Auf dieser Grundlage wurden dann in Kapitel 3 konkrete Use Cases und Anforderungen an eine Intelligent Assistent Lösung am LRZ erarbeitet.
Hieraus wurde in Kapitel 4 eine Lösung für die Realisierung des Intelligent Assistent entwickelt, die zwei Komponenten vorsieht. Dies sind die Wissensverwaltungs-Komponente und die Nutzer-Frontend-Komponente. Die Komponente zur Wissensverwaltung wurde hierbei in die iET Workcenter-Anwendung integriert, die Nutzer-Frontend-Komponente im Servicedesk-Portal.
Das Wissen für den Intelligent Assistent wird von den Mitgliedern der Support-Gruppe eines Dienstes eingegeben und getrennt nach Dienst in einer selbst entworfenen Form der Wissensrepräsentation gespeichert. Diese wird dann durch einen im Rahmen dieser Arbeit entworfenen Algorithmus in einen Entscheidungsbaum und eine Liste von Service Requests konvertiert und so für die Nutzer-Frontend-Komponente nutzbar gemacht.
Die Implementierung dieser Lösung wurde dann in Kapitel 5 beschrieben und die Einhaltung der in Kapitel 3 definierten Anforderungen bis auf eine Ausnahme überprüft und festgestellt.

Die einzige Anforderung, deren Einhaltung nicht überprüft, sondern nur abgeschätzt, wurde ist das Nonfunctional Requirement *NR6: Maximale Ladezeit von einer Sekunde*. Dies begründet sich damit, dass das System auf dem die Implementierung durchgeführt wurde von der Ausstattung nicht dem Produktivsystem am LRZ entspricht.
Um die Einschätzung zur Einhaltung dieser Anforderung zu bestätigen, müsste daher ein Performance-Test entwickelt werden, der das Nutzerverhalten im Servicedesk-Portal realistisch modelliert. Dieser müsste dann auf einem System durchgeführt werden, das von der Hardwareausstattung identisch zum Produktivsystem des LRZ ist. Die Entwicklung und Durchführung eines solchen Tests liegt nicht im Rahmen dieser Arbeit und stellt somit eine erste mögliche Folgearbeit dar.
Des Weiteren sind als Ausblick auch noch einige mögliche Erweiterungen des Funktionsumfangs des Intelligent Assistent zu nennen:
Aktuell wird bei der Generierung der Knoten des Entscheidungsbaums nicht gespeichert, aus welchem Knowledge Base-Eintrag oder Incident Template ein entsprechender Baumknoten

erzeugt wurde. Dies macht z.B. bei der Korrektur von Tippfehlern oder einer reinen Aktualisierung des Beschreibungstextes eines dieser Elemente in der Wissensbasis eine komplette Neugenerierung des Entscheidungsbaums nötig. Würde mit der Baumstruktur gespeichert, aus welcher Quelle ein Knoten generiert wurde, ließe sich eine Aktualisierung der Knoteninhalte ohne Neugenerierung der Baumstruktur implementieren.
Des Weiteren ist eine Nutzung von Informationen aus der CMDB, wenn diese am LRZ fertig implementiert und ausgerollt ist, wünschenswert. Hierdurch könnte der Intelligent Assistent dann so erweitert werden, dass der Nutzer bei bestimmten Diagnoseangaben auf einen aktuellen Dienstausfall hingewiesen wird, wenn bestimmte, für einen Dienst relevante, CIs in der CMDB als ausgefallen markiert sind.
Als weitere mögliche Funktionserweiterung ist zu nennen, die Möglichkeit von Deep-Links von der Incident-Erstellung auf die Auswahl einzelner Service Request Templates zu erweitern. Hiermit könnten Links zu bestimmten Service Request Templates direkt auf den Informationsseiten zum jeweiligen Dienst auf der Webseite des LRZ angeboten werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Bar07] Barber, Scott: *Get performance requirements right - think like a user. Whitepaper.* `http://www.perftestplus.com/resources/requirements_with_compuware.pdf`, 2007. Abgerufen am: 11.11.2014.

[BHSA07] Boersch, Ingo, Jochen Heinsohn und Rolf Socher-Ambrosius: *Wissensverarbeitung: Eine Einführung in die Künstliche Intelligenz für Informatiker und Ingenieure.* Elsevier, Spektrum Akademischer Verl., Heidelberg, 2. Auflage, 2007.

[BKI08] Beierle, Christoph und Gabriele Kern-Isberner: *Methoden wissensbasierter Systeme: Grundlagen, Algorithmen, Anwendungen.* Studium. Vieweg + Teubner, Braunschweig [u.a.], 4., verb. Auflage, 2008.

[BRH14] Brenner, Michael, Christian Richter und Norbert Hartmannsgruber: *Incident and Service Request Management: Prozessbeschreibung*, 25.02.2014.

[gom] gomez: *Why Web Performance Matters: Is Your Site Driving Customers Away?* `http://www.mcrinc.com/Documents/Newsletters/201110_why_web_performance_matters.pdf`. Abgerufen am: 31.10.2014.

[iet] iet Solutions: *iET Enterprise: Plattform für Service Management, xRM und individuelle Applikationen: Datenblatt.* `http://www.iet-solutions.com/files/9313/7717/9427/ger_iET_Enterprise_DB.pdf`. Abgerufen am: 31.10.2014.

[iet11a] iet Solutions: *iET Enterprise Application Developer's Guide*, 08.2011.

[iet11b] iet Solutions: *iET Enterprise Application Programming Interface*, 08.2011.

[iet11c] iet Solutions: *Workflow*, 08.2011.

[ISO11] ISO/IEC: *20000-1:2011. Information technology - Service management - Part 1: Service management system requirements*, 15.04.2011.

[LRZ13a] LRZ: *Das Dienstleistungsangebot des LRZ.* `https://www.lrz.de/wir/regelwerk/dienstleistungskatalog.pdf`, 20.12.2013.

[LRZ13b] LRZ: *Einführung in das LRZ.* `http://www.lrz.de/wir/Einfuehrung-LRZ.pdf`, 2013.

[LRZ14] LRZ: *Jahresbericht 2013 des Leibniz-Rechenzentrum.* `http://www.lrz.de/wir/berichte/JB/JBer2013.pdf`, Mai 2014.

[May99] Mayer, Birgitt: *Einsatz von Bayesschen Netzen für den Intelligent Assistant des LRZ.* Diplomarbeit, Technische Universität München, 1999.

[Qui93] Quinlan, J. R.: *C4.5: Programs for machine learning.* The Morgan Kaufmann series in machine learning. Morgan Kaufmann Publishers, San Mateo and Calif, 1993.

[RR13] Robertson, Suzanne und James Robertson: *Mastering the requirements process: Getting requirements right.* Addison-Wesley, Upper Saddle River and NJ, 3. Auflage, 2013.

[Sch14a] Schnoy, Benjamin: *Technische Dokumentation Webservice-Schnittstelle*, 04.2014.

[Sch14b] Schnoy, Benjamin: *Technische Dokumentation Servicedesk-Portal*, 06.2014.

[Sen] SensioLabs: *Symfony: Translations.* `http://symfony.com/doc/2.3/book/translation.html`. Abgerufen am: 10.11.2014.

[W3C14] W3C: *HTML5: A vocabulary and associated APIs for HTML and XHTML. W3C Recommendation 28 October 2014.* `http://www.w3.org/TR/2014/REC-html5-20141028/dom.html#embedding-custom-non-visible-data-with-the-data-*-attributes`, 2014. Abgerufen am: 10.11.2014.

[Wei96] Weishaupt, O.: *Implementierung des Intelligent Assistant für das LRZ: Fortgeschrittenenpraktikum*, 1996. Technische Universität München.

[Wik14a] Wikipedia: *CamelCase — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/w/index.php?title=CamelCase&oldid=632758820`, 2014. Abgerufen am: 09.11.2014.

[Wik14b] Wikipedia: *Objektrelationale Abbildung — Wikipedia, The Free Encyclopedia.* `http://de.wikipedia.org/w/index.php?title=Objektrelationale_Abbildung&oldid=131861296`, 2014. Abgerufen am: 09.11.2014.