Abschlussbericht

für das Vorhaben

WAVES – Wissensaustausch bei der verteilten Entwicklung von Software

Vorhabensbezeichnung	WAVES - Wissensaustausch bei der verteilten Entwicklung von Software
Ergebnis Id	REPORT
Förderkennzeichen	01ISF04A-01ISF04I
Kontakt	Peter Szulman Haid-und-Neu-Str. 10-14 76131 Karlsruhe Tel.: (0721) 9654-636 Fax: (0721) 9654-637 E-Mail: szulman@fzi.de
Laufzeit des Vorhabens	1. März 2006 bis 31. Oktober 2008
Autoren	Christoph Andriessens, Marcus Briesen, Markus Gebhard, Hans-Jörg Happel, Wassilios Kazakos, Guido Lohaus, Robert Neher, Rainer Neumann, Laura Plonka, Ralf Reyeg, Tim Romberg, Alexa Schumacher, Thomas Schuster, Peter Szulman, Ralph Traphöner, Max Völkel
Letzte Änderung	28.04.2009

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01ISF04A-I gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

FZI-Publication 4-6-1/09, ISSN 0944-3037

Verbundpartner

	Zuwendungsempfänger	Förderkennzeichen
	FZI Forschungszentrum Informatik, Karlsruhe (Verbundkoordinator)	01ISF04A
	Freie Universität Berlin	01ISF04B
	empolis GmbH, Gütersloh	01ISF04C
	Polarion Software GmbH, Stuttgart	01ISF04D
	Object International Software GmbH, Stuttgart	01ISF04E
	disy Informationssysteme GmbH, Karlsruhe	01ISF04F
	PTV Planung Transport Verkehr AG, Karlsruhe	01ISF04H
	CAS Software AG, Karlsruhe	01ISF04I

Inhaltsverzeichnis

1. Einführung

Eine Produktivitätssteigerung im Software Engineering ist durch die Verbesserung von mehreren Faktoren zu erreichen. Die Einführung von mächtigen Basistechnologien, wie zum Beispiel die neueren Ansätze aus dem Bereich Service-orientierte Architekturen oder Aspekt-orientiertes Programmieren sind nur zwei Beispiele hierfür. Moderne Automatisierungstechniken wie Modell-basierte Generatortechniken bzw. gut durchdachte Prozesse und Rollen in einem Softwareentwicklungsteam tragen ebenso zur Verbesserung der Produktivität im Software Engineering bei. Als Ergänzung zu diesen typischen Software Engineering Themen, fokussierte WAVES auf zwei weitere Aspekte, auf *qualifizierte Entwickler* und auf den Aufbau von *effizienten Teams*.

Entwickler kämpfen häufig auf der einen Seite mit einer großen Informationsflut (ständige Einarbeitung in neue Technologien und komplexe Domänen) und auf der anderen Seite mit Informationsmangel (Projektwissen ist häufig auf einige Know-How-Träger konzentriert, Informationen liegen verstreut vor, usw.). Das Wissen und Können der Entwickler muss daher sowohl mit dem Projektwissen als auch mit den Fortschritten im Software Engineering Schritt halten, um deren Potential optimal nutzen zu können. Desweiteren kann man heutzutage den Trend zur verteilten, team-orientierten Entwicklung beobachten. Einerseits, weil sich klassische Firmenstrukturen auflösen (Arbeit beim Kunden, Offshoring-Projekte), andererseits bedingt die Entwicklung komplexer Systeme eine entsprechende Arbeits- und Wissensteilung. Daraus folgt, dass nicht nur das Können der einzelnen Entwickler, sondern auch das Wissen zwischen den Projektbeteiligten ausgetauscht werden muss, um effizient funktionierende Teams aufzubauen.

Bei der eingehenden Analyse der Anforderungen und Ist-Analyse bei den Anwendungspartnern wurden in WAVES zwei Beobachtungen gemacht. Erstens, dass im Unternehmen in der Regel bereits ein großer Schatz an Lösungswissen vorhanden ist. Man denke beispielsweise an die Vielzahl explizit vorhandener Artefakte (Dokumente, Quelltext-Dateien, Informationen über vergangene Projekte), die in verschiedenen Repositories (Subversion, SourceSafe), Mitarbeiter-Rechner, oder Softwarewerkzeugen (z.B. CRM-Systeme) vorliegen. Es besteht ein großes Interesse daran, diesen großen Schatz an Lösungswissen in der Entwicklungskette miteinander zu vernetzen bzw. zu integrieren. Darüber hinaus ist in den Köpfen der Mitarbeiter viel Wissen implizit vorhanden. Während die Vernetzung der explizit vorhandenen Wissensartefakte vorwiegend eine technische Herausforderung darstellt, wird das Erfassen des Mitarbeiter-Know-Hows durch zahlreiche weitere Faktoren, wie zum Beispiel die häufig fehlende Motivation der Mitarbeiter ihr Wissen explizit zu erfassen, erschwert. Eine weitere Feststellung war, dass zwar Wissen bei den Unternehmen vorhanden ist, dieses Wissen erst mit der Zeit entsteht und dieser Wissensreifungsprozess selten ausreichend gut unterstützt wird (Happel, H.-J. und Schmidt, A. (2007)). Ausgehend von diesen Beobachtungen wurde das Ziel von WAVES abgeleitet:

Primäres Ziel von WAVES besteht darin, in verteilten Softwareprojekten den Aufbau und den Austausch von informellem Wissen zu fördern und seine schrittweise Strukturierung und Vernetzung zu unterstützen.

Um das Ziel zu erreichen, verfolgte WAVES einen Lösungsansatz auf zwei Ebenen: der methodischen Ebene und der technischen Ebene. Auf *methodischer Ebene* wurde ein sehr empirischer und fallstudienorientierter Ansatz gewählt, welcher die Ergebnisse der technischen Entwicklungsarbeit, der Anforderungsanalyse und der sozioökonomischen Untersuchungen verbindet. Basis für diese Fallstudien und damit ein zentraler Teil von WAVES ist eine im Projekt entwickelte Wissensmanagement-Plattform. Diese Plattform wurde während der Projektlaufzeit kontinuierlich - in vier Iterationen - bei den WAVES-Praxispartnern evaluiert. Dank der heterogenen Zusammenstellung der Praxispartner aus vier unterschiedlichen Anwendungsdomänen (disy/Datenerfassung und –auswertung, Konferenzsysteme; Object International/Individualentwicklung; PTV/Geomanagement, Logistik und Mobilty; CAS/CRM-Systeme) konnte während der erfolgreichen, Fallstudien-basierten Evaluation der WAVES-Wissensmanagement-Lösung, deren breites Einsatzspektrum aufgezeigt werden.

Neben der Integration bestehender Open-Source-Komponenten zum Wissnesmanagement wie z.B. Wikis spielen in der technologischen Lösungsarchitektur Erweiterungen und Innovationen im Bereich der effektiven Wissensnutzung durch kontextsensitive Retrieval-Methoden sowie die Wissensartikulation eine wichtige Rolle.

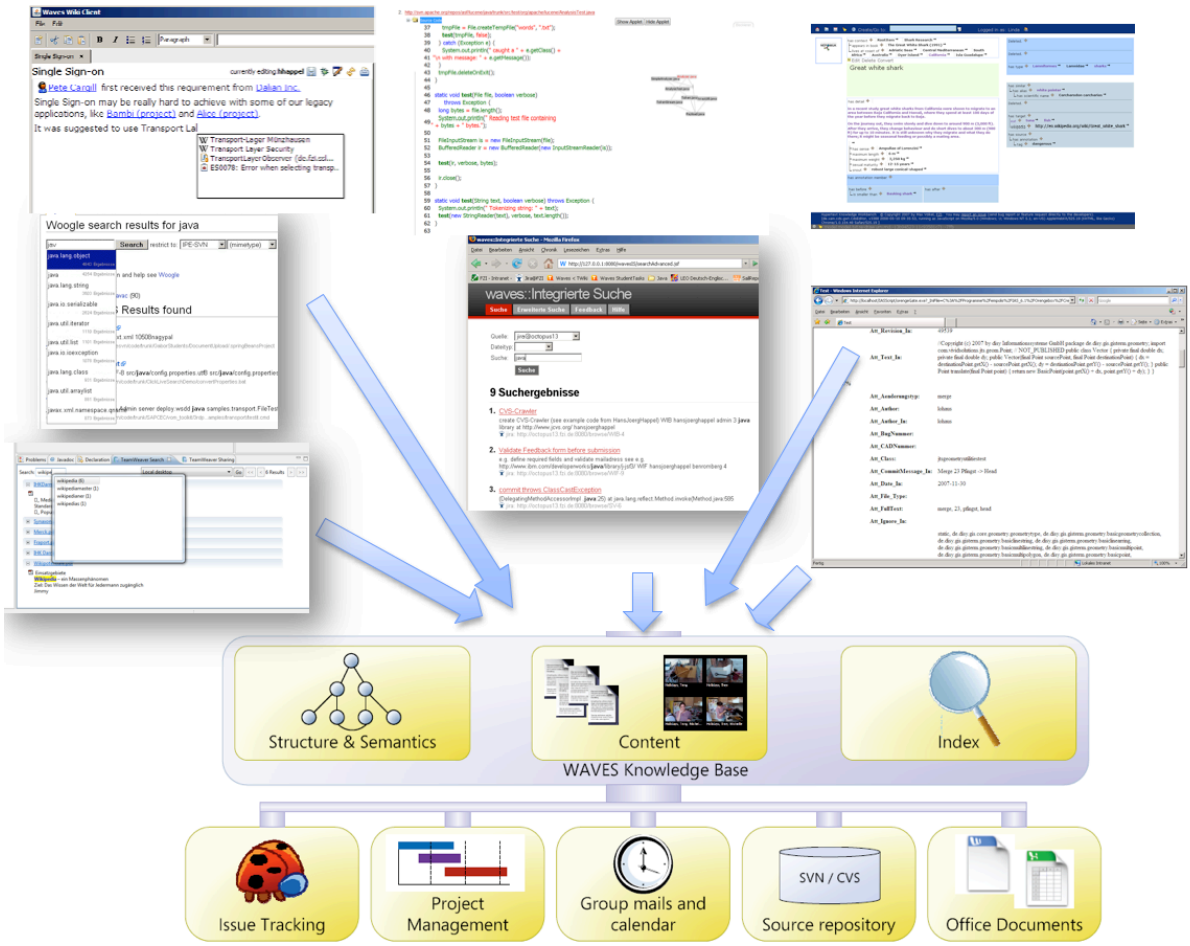


Abbildung 1: Die WAVES Wissensmanagement-Lösung

Das Backend der WAVES Wissensmanagement-Lösung bindet das explizit vorhandene Unternehmenswissen aus einer Vielzahl von Datenquellen ein. Dazu zählen unter anderem Issue-Tracking-Systeme (JIRA, Bugzilla, OTRS), Projektmanagement-Tools (XPlanner), Emails, das CRM-System genesisWorld, Dokumente (Word, PDF-s), Intranets (HTML-Seiten, JSP-Wiki, MediaWiki), Repositories (SVN, CVS, VSS) und nicht zuletzt der Quelltext von Softwareprojekten. Auf die durch das WAVES-Backend referenzierten Wissensartefakte bauen insgesamt sieben, in WAVES entwickelte Werkzeuge aus den folgenden vier Bereichen der Wissensnutzung bzw. Wissensartikulation auf:

- Informationszugriff: Die *WAVES integrierte Suche* (*WavesIS*) bietet eine einheitliche, web-basierte Oberfläche zur Suche nach den von dem WAVES-Backend indizierten Wissensartefekte. Das Werkzeug *KISSy* fokussiert speziell auf die Quelltext-Suche in Java-Systemen um wieder verwendbare Software-Komponenten oder Beispielcode für Software-Probleme aufzufinden.
- Kontext-Management: Das auf empolis IAS basierende Kontext Management Verfahren ermöglicht ein kontextabhängiges Angebot von Inhalten an die Benutzer.
- Bedarfsgetriebener Informationsastausch: bezeichnet die Erstellung und den Austausch von Wissen und Informationen der sich nach konkreten Informations-Bedürfnissen von Informationssuchenden richtet. Die *inverse Suche* ist eine Methode und ein Werkzeug zur Diffusion von Informationen aus privaten in öffentliche Informationsräume. Das Werkzeug

Woogle ermöglicht die Verzahnung von Wissenserstellung in Wikis mit unternehmensweiter Suche.

- Leichtgewichtige „Wiki-style“ Wissensartikulation: das Werkzeug *Wiquila* ermöglicht auf das bereits existierende, vom WAVES-Backend erfasste Unternehmenswissen weiter aufzubauen. Das Werkzeug *HKW* ist eine Erweiterung dieses Ansatzes auf die Ebene des persönlichen Wissensmanagements.

Durch die WAVES Werkzeug-Kette konnte eine in der Praxis erfolgreich erprobte Wissensmanagement-Lösung angeboten werden, die den Stand der Technik im Bereich Wissensaufbau und Wissensaustausch in verteilten Software-Entwicklungsprojekten verbessern konnte.

Der Rest dieses Berichts gliedert sich wie folgt: Zunächst wird in Kapitel 2 ein Überblick über den Stand der Technik gegeben, wie er zu Beginn des Projektes existierten, weiterhin es werden die wesentlichen Defizite beschrieben. In den nächsten drei Kapiteln werden die Ergebnisse des Projektes im Überblick vorgestellt: die am Projektbeginn erhobenen Use-Cases der WAVES-Anwendungspartner in Kapitel 3 untermauern die Praxisrelevanz der in Kapitel 4 vorgestellten und in WAVES entwickelten Konzepte und Verfahren. Kapitel 5 zeigt, wie die Konzepte und Verfahren in die Praxis übertragen wurden und stellt somit eine Evaluation der Konzepte und Verfahren bei den Anwendungspartnern dar. Der Bericht schließt mit einer kurzen Zusammenfassung und einem Ausblick auf weiterführende Fragestellungen.

2. Stand der Technik

Vorraussetzung für eine erfolgreiche Bearbeitung der Forschungsthemen aus WAVES war eine ausführliche Erarbeitung, Analyse und Darstellung des wissenschaftlichen bzw. in der Praxis etablierten Standes der Technik in den Themengebieten des Projektes, deren Ergebnisse in den Meilensteindokumenten (Happel, H-J., Kuttruff, V., Romberg, T., Szulman, P., Völkel, M. (2006) ; Happel, H-J., Schuster, T., Szulman, P., Traphöner, R., (2007); Romberg, T. (2007)) im Detail zu finden sind. Im Folgenden wird daher nur auf die Ergebnisse dieser Analyse in Form identifizierter Defizite des Standes der Technik zu Projektbeginn von WAVES eingegangen. Die Erhebung erfolgte anhand WAVES zu Grunde liegenden Themenbereiche und gliedert sich grob wie folgt:

- Wissensnutzung
- Kontext-Management
- „Wiki-style“ Wissensartikulation, Wikis in heutigen Softwareprojekten
- Bedarfsgetriebener Informationsaustausch
- Ontologien im Software Engineering

2.1 Wissensnutzung

2.1.1 Wissenssuche

Die Entwicklung komplexer Produkte und Dienstleistungen ist angewiesen auf hochgradig spezialisierte, arbeitsteilig organisierte Entwicklungsschritte. Einzelne Entwickler können den detaillierten Gesamtaufbau eines solchen Systems nur noch rudimentär nachvollziehen. Um ihre eigene Arbeit effizient zu erledigen, benötigen sie Kenntnisse der angrenzenden Schnittstellen. Diese Beschreibung trifft insbesondere auf die Entwicklung von Softwaresystemen zu, da durch die relativ leichte Wiederverwendung der Schnittstellen niedrigerer Systemschichten, Funktionsbibliotheken und nicht zuletzt beliebig verfügbarer Dienste eine extrem hohe Systemkomplexität entsteht. Sie erfordert von Entwicklern eine kontinuierliche Wissensaufnahme.

Die Unternehmenssuche bezeichnet ein Teilgebiet von Information Retrieval und bezieht sich auf den Vorgang der inhaltsorientierten Suche mit Hilfe einer unternehmensinternen Suchmaschine, welche Inhalt mittels so genannter Crawler indiziert. Die Suche wird jedoch in der Regel nicht live auf den ursprünglichen Datenquellen durchgeführt, sondern auf dem Suchindex. Dieser Index beinhaltet primär interne Datenquellen wie Dokumente von verschiedenen Datenbanken und Einträge von Dateisystemen. Treffer oder gefundene Dokumente werden im Kontext der Suchanfrage als Textauszug („Snippet“) angezeigt. Durch diese Vorschau lässt sich schnell die Relevanz der Ergebnisse beurteilen. Durch die fortlaufende Indexierung der einzelnen Datenquellen wird die Aktualität der Resultate (Result Set) gewährleistet. Aus Sicht von Unternehmen ist der Nutzen von Unternehmenssuche die Unterstützung der Mitarbeiter bei der Suche nach arbeitsrelevanten Informationen. Am Projektanfang ist die Suche nach einer integrierten Lösung zur Suche nach den verschiedensten Arten von Wissensartefakten im Unternehmen fehlgeschlagen.

2.1.2 Quelltextsuche und Wiederverwendung

Die Suche nach Quelltext stellt einen speziellen Aspekt der Wissensnutzung dar. Im Bereich des Software Engineerings tritt im Verlauf der verschiedenen Entwicklungsphasen einer Anwendung oftmals die Frage auf, ob ein auftretendes Problem bereits andernorts gelöst ist. In der Praxis hat sich gezeigt, dass die Entwickler auf diese Fragen oftmals durch entsprechende Suchanfragen an Suchmaschinen verschiedenster Ausprägung reagieren (Singer, J., Lethbridge, T., Vinson, N., Anquetil, N.). Aus diesem Grund hat sich die Suche nach Information zu einer zentralen Unterstützungsaktivität entwickelt, Studien zu Folge verbringen Beschäftigte im IT-Umfeld stellenweise bis zu 40% Ihrer täglichen Arbeit mit der Suche und dem Zugriff auf Informationen verschiedenster Arten (Henninger, S. (1997), Murphy, G.C., Kersten, M., Findlater, L. (2006)).

Aus Entwicklersicht sind daher besonders Suchmaschinen von Relevanz, die die Suche nach bereits existierenden Code-Fragmenten oder Komponenten unterstützen. Bekannter Vertreter solcher Suchmaschinen sind Google CodeSearch, Merobase, Koders oder Krugle. Wenn sich diese Suchmaschinen im Detail doch voneinander abgrenzen, so weisen sie als entscheidende Gemeinsamkeit die Suchmöglichkeit nach Quelltextfragmenten in quelltextoffenen Projekten auf. Diese Suchdienste arbeiten dabei üblicherweise rein textbasiert, vernachlässigen allerdings die Struktur der Software selbst. In Konsequenz ist die Suche dadurch zwar äußerst performant birgt aber zahlreiche andere Nachteile in sich. Besonders auffällig sind dabei das Auftreten stark verrauschter (und daher wenig aussagekräftiger) Ergebnisse, schwieriges bzw. ungenaues Ranking der Treffer und eine fehlende interaktive Navigation der in der Ergebnisliste (Henninger, S. (1997)).

Die bislang vorhandenen Einschränkungen der am Markt befindlichen Suchmaschinen führen daher dazu, dass die Suche nach Softwareartefakten durch ein erneutes überdenken der gesuchten Information und eine mehrfache Reformulierung der Suchanfragen geprägt wird, die oftmals nicht die gewünschte Information als Teilmenge der Ergebnisse zurückgeliefert wird. Letztendlich führt dies über einen entsprechenden Zeitverlust bis hin zur Nichterreichung der Entwicklerziele.

2.2 Kontext-sensitives Angebot von Wissensinhalten

Kontextbewusstsein und Personalisierung softwaregestützter Anwendungen umfasst inzwischen fast alle Anwendungsbereiche, angefangen von der Unterstützung mobiler Endgeräte bis hin zu kommerziellen Internetauftritten von Firmen. Die Personalisierung wird hierbei üblicherweise eingesetzt, um den Benutzer gegenüber der Flut von möglichen Informationen abzuschirmen und ihm so die Möglichkeit zu geben, den Überblick über den aktuell relevanten Kontext zu geben.

Kontextbewusstsein wird erstmals von Weiser propagiert, dieser hatte darin die Vision, dass der moderne Rechner bzw. die Anwendung dem richtigen Nutzer die richtige Information zur richtigen Zeit präsentieren kann. Diese Vorstellung wurde bislang besonders im Rahmen mobiler Anwendungen verfolgt; hierbei wird besonders der aktuelle geographische Kontext ausgewertet und entsprechende Empfehlungen für den Benutzer erzeugt.

Abgesehen von den sogenannten ubiquitären Anwendungen ist, bezogen auf das Software-Engineering bislang wenig Fortschritt im Bereich der Nutzung des Kontexts erfolgt, sodass eine Vielzahl wissenschaftlicher Fragestellung in diesem Themenfeld bislang unbeantwortet bleibt. Der Informationsflut in komplexen Entwicklungsprojekten ruft nach intelligenten Tools, die die Entwickler unterstützen. Während das Softwareingenieurwesen viele nutzbare Kontextinformationen anbietet, sind kontext-basierte Anwendungen noch selten. Dennoch zeigen die ersten Prototypen, die eine beschränkte Anzahl von Kontextfaktoren analysieren, zahlreiche versprechende Ergebnisse. Ein solches Ergebnis ist das mylar Projekt, das als ein Eclipse Projekt läuft. Während anwendungsorientierte Werkzeuge sich auf einen bestimmten Bereich von Kontextfaktoren beschränken, bieten die Rahmenwerke die Möglichkeit, aus einer Vielzahl von Informationquellen Kontextinformationen abzuleiten. Jedoch wurde festgestellt, dass Entwickler bezüglich Privatsphäre, der zu stark automatisierten Unterstützung und fehlender Transparenz eher empfindlich reagieren. Dementsprechend sind Transparenz und Kontrollierbarkeit der Assistenzfunktionalität wichtige Voraussetzungen für ein erfolgreiches Kontext-Managament Tool.

Die meisten Tools haben eine modularisierte Architektur, bieten APIs an und sind durch Plugins erweiterbar. Folglich können Rahmenwerke wie Hackystat sich leicht an einige Szenarien und neue Kontextfaktoren anpassen. Andererseits stützen sich die beschriebenen Tools auf die Client-Server Architektur, indem das Client Daten sammelt und sie an einen zentralen Server schickt. Diese statische Architektur könnte eine Beschränkung bedeuten, wenn man in einer offenen heterogenen Umwelt arbeitet. Während mylar mit dem Austausch von Task-Kontext in einem verteilten Entwicklungsprojekt adressiert, beschäftigt sich keiner der untersuchten Ansätze mit Kontext- und Wissensaustausch. Die Idee kontextualisierter Informationsdienste im Softwareentwicklungsprozess ist noch nicht weit verbreitet, verspricht aber durch die starke Strukturierung und reichhaltige formalisierte Indizien die Möglichkeit, Kontext in guter Detaillierung und Breite zu ermitteln und somit mächtige kontextbewußte Anwendungen zu gestalten.

2.3 „Wiki-style“ Wissensartikulation, Wikis in heutigen Softwareprojekten

Die Mitglieder eines Softwareprojekts – also sowohl Entwickler als auch Tester, Designer, Fachliche Analysten und Vertreter des Endkunden – brauchen heute Zugriff auf sehr unterschiedliche Arten von Wissen und Informationen, so wie:

1. Die Konzepte und Methoden, die ihre jeweilige Qualifikation definieren (so z.B. Konzepte der OO-Programmierung oder des Lambda-Kalküls für Programmierer, oder Regeln der Perspektive für einen Designer);
2. die Fertigkeit im Umgang mit den spezifischen Werkzeugen und Materialien in ihrem Beruf (Programmiersprachen, Bibliotheken und Laufzeitumgebungen, Grafiksoftware, etc.)
3. Hintergrundwissen über die Problemdomäne, die ihre Arbeit behandelt (z.B. steuerliche Gesetzgebung bei einer Finanzsoftware oder Wissen um die Arbeitsgewohnheiten der Benutzer;
4. spezifische Infrastruktur und Vorgehensweisen in ihrer Arbeitsumgebung (wie man sich auf einen Demoserver aufschaltet, wer bei bestimmten Änderungen benachrichtigt werden muss);
5. und schließlich Wissen über das eigentliche Projekt – wofür seine einzelnen Module verantwortlich sind, warum bestimmte Designentscheidungen getroffen wurden, welche Beschränkungen und Workarounds existieren, etc.

Viel ist über die Unterscheidung zwischen *Informationen* und *Wissen* geschrieben worden. Ohne dieses Thema zu vertiefen, können wir einerseits feststellen, dass für Kernprozesse im Softwareengineering heute standardisierte Vorgehensweisen und damit auch vorstrukturierte Informationsrepositorien angeboten werden, wie z.B. für das Change Management, Bugtracking, Release Management oder Testcoverage-Analyse – Application-Lifecycle-Management-Suiten wie Polarion integrieren diese Werkzeuge miteinander. Dabei handelt es sich also vor allem um Projektinformationen (Punkt 5 in der obigen Liste). Daneben haben sich Wikis als populäres Allround-Werkzeug in Softwareteams etabliert, um einerseits operative Informationen zu verwalten, die (noch) in keines der vorstrukturierten Schemata passen, oder für die das Hypertext-Paradigma besonders geeignet ist (z.B. Entwurfsmuster), als auch zur Unterstützung des Wissensaustauschs bei den Punkten 1-4. Zu den Vorteilen von Wikis zählt:

- Sie erlauben Benutzern, sich allmählich von Konsumenten (Lesern) zu Produzenten (Autoren) zu entwickeln, und motivieren Benutzer dazu, weil diese selbst für kleine Beiträge in existierenden Artikeln eine große Leserschaft erwarten können – im Gegensatz zu anderen Techniken, bei denen ein Benutzer erst einen ganzen Artikel schreiben muss, und sich dann nicht sicher sein kann, dass er überhaupt gelesen wird.
- Sie fördern kollektive Autorenschaft und ständige Verbesserung an bestehenden Inhalten statt einer Vielzahl von sich inhaltlich überschneidenden Dokumente unterschiedlicher Autoren.
- Das flexible Hypertext-Strukturmodell erlaubt recht große Flexibilität und Mächtigkeit bei der Ad-hoc-Definition von problemspezifischen Strukturen, während bei vielen anderen problemspezifischen Anwendungen nur ihre Entwickler in der Lage sind, ihre Strukturen anzupassen.

Allerdings förderten Gespräche mit Projektpartnern und anderen Firmen, die Wikis zur Projektunterstützung einsetzen, folgende Mängel zutage:

1. *Benutzbarkeit*. Beim Bearbeiten präsentiert sich der Wikiinhalt völlig anders als beim Lesen. Viele Features sind nur durch Lernen einer bestimmten Wikisyntax nutzbar. Es wird kein direktes Feedback z.B. beim Einfügen von Links gegeben, so dass man häufig Fehler im Nachhinein korrigieren muss.

2. *Produktivität*. Selbst sehr erfahrene Nutzer brauchen aufgrund der komplizierten Bedienung mehr Zeit als nötig für viele Operationen (z.B. Suchen & Ersetzen), und werden von technischen Aspekten vom eigentlichen Inhalt abgelenkt.
3. *Wucherung und andere Qualitätsmängel.* Große Wikis tendieren oft zum Wildwuchs, wenn keine „Wikigärtner“ da sind, die einen Überblick behalten, ab und zu aufräumen und dazu auch mit den anderen Autoren in direkten Kontakt treten.
4. *Integration.* Klassische Wikis stellen isolierte Silos dar; die einzige Verbindungsmöglichkeit mit anderen Systemen besteht darin, externe Hyperlinks einzufügen; aber dieser Prozess ist sehr fehlerträchtig – Links können z.B. User- oder Session-IDs enthalten, die bei einem späteren Aufruf nicht mehr gültig sind.
5. *Offline*-Zugriff. Vertrieb- und Support-Mitarbeiter sind viel unterwegs und brauchen daher Offline-Zugriff (mindestens Lesezugriff, am Besten auch Schreibzugriff). Weil heutige Wikis dies nicht bieten, bevorzugen diese Mitarbeiter oft weiterhin E-Mail, mit allen Problemen, die dies für die ganze Organisation bringt.

Bisherige Ansätze für eine bessere Integration von Wikis in Softwareengineering-Prozesse gingen im Wesentlichen in zwei Richtungen:

Integration von Wikis und Bug-Trackern. Zum Beispiel Confluence und Jira der Firma Atlassian; oder Trac von Edgewall. Allerdings beschränkt sich die Integration im Wesentlichen auf eine gemeinsame Benutzerverwaltung. Wenig Unterstützung wird für eine wirkliche Bezugnahme der Inhalte aufeinander geliefert.

Modellierung von Anforderungen mit semantischen Wikis. Dieser Ansatz wird seit längerer Zeit von Fraunhofer IESE (Projekte RISE, SOP) und der Universität Leipzig (OntoWiki und SoftWiki) verfolgt. Hier besteht das Ziel im Ersetzen der bisher verwendeten Werkzeuge zur Anforderungsmodellierung, es findet daher auch keine Integration statt.

2.4 Bedarfsgetriebener Informationsaustausch

Während die im Abschnitt 2.1.1 beschriebenen Ansätze den Informationszugriff erleichtern, fokussieren die Verfahren im Abschnitt 2.2 auf eine benutzbare und effiziente Wissensartikulation. Allerdings adressieren diese Werkzeuge nicht, *welches* Wissen artikuliert werden sollte und *wie* es mit Anderen ausgetauscht werden kann.

Gerade bei Softwareentwicklern sind Tätigkeiten wie Dokumentation und Wissenserstellung relativ unbeliebt, da sie nicht direkt zu "produktiven" Erstellung von Systemfunktionalität beitragen und ihr Nutzen meist nur schwer greifbar ist. Weiterhin ist es ökonomisch wenig effizient bzw. unmöglich, sämtliches entwicklungsrelevantes Wissen explizit zu dokumentieren. Softwareentwickler haben in der Regel aber keine systematischen Informationen darüber, welches Wissen für ihr Projekt/ihre Organisation den größten Nutzen bringt. Anders ausgedrückt - ein *effizienterer Wissensaustausch sollte sich am Bedarf einer Organisation orientieren und dabei möglichst zielgerichtet erfolgen*.

Eine große Anzahl von Wissensmanagement-Systmen wurde entwickelt, um verschiedene Aspekte des Wissensmanagements, inklusive Wissensaustausch zu unterstützen. Beispiele hierfür sind kollaborative Systeme, Dokumenten-Ablagen, Suchsysteme für Unternehmen, Expertensysteme oder Wikis (Alavi, M., Leidner, D.E. (2001), Maier, R. (2003)). Der Hauptunterschied des *bedarfsgetrieben Informationsaustauschs* im Vergleich zu diesen Systemen besteht darin, dass beim letzteren diejenigen, die ihr Wissen für die Öffentlichkeit anbieten möchten, selbständig entscheiden müssen, welche Informationen überhaupt für die Öffentlichkeit relevant sind. In unserem Ansatz (inverse Suche bzw. Woogle, Abschnitt 4.3) signalisieren die Informationsnutzer (mit der Hilfe von Anfragestatistiken) ihren Informationsbedarf. Anschließend kann der Informationsanbieter entscheiden, ob er seine Informationen angesichts des bereits bekannten Informationsbedarfs tatsächlich freigeben möchte. Das System *Answer Garden System* (Ackerman, M.S., Malone, T.W. (1990)) ist eine Ausnahme, bei dem die potentiellen Informationsanbieter basierend auf dem

Informationsbedarf der Nutzer getriggert werden. Allerdings bündelt Answer Garden weder die Informationen über die Informationsbedarfe noch macht es für den Nutzer, der die benötigten Informationen besitzt, Vorschläge, welche er freigeben soll.

Zahlreiche Arbeiten aus dem Bereich *Information retrieval* beschäftigen sich mit der Notifikation von Nutzern, wenn Informationen über bereits durchgeführte Suchanfragen zur Verfügung stehen. Ein bekanntes Beispiel hierfür ist das System Google Alerts. In der Literatur ist dieser Ansatz als *prospective search* (Irmak, U., Mihaylov, S., Suel, T., Ganguly, S., Izmailov, R. (2006)), *continous querying* (Kukulenz, D., Ntoulas, A. (2007)) *retroactive query answering* (Yang, B., Jeh, G. (2006)) bezeichnet.

Peer2Peer Ansätze zum Information retrieval adressieren die Probleme von traditionellen, zentralisierten Suchsystemen, die schlecht skalieren und keine Möglichkeit bieten das gesamte Web zu indizieren. Systeme wie Minerva oder P-Grid4 setzen voraus, dass die Nutzer die Indizierung selbstständig durchführen, die Indizes eigenständig verwalten, freigeben und Suchdienste anbieten. Zusätzlich stellt ein Informationsanbieter anderen Nutzern im P2P-Netzwerk Statistiken und Metadaten über die bei ihm vorliegenden Informationen zur Verfügung. In unserem Ansatz kann der Informationsanbieter basierend auf den Anfragen der Informationsnutzer entscheiden, welche Informationen er preisgibt.

2.5 Ontologien für Software Engineering

Ontologien sind formale Konzeptualisierungen, die in einer bestimmten Domäne gemeinsam genutzt werden. Ontologien beschreiben die Domäne üblicherweise anhand von Konzepten und ihren Beziehungen untereinander. Typische Anwendungsfelder sind Informationsintegration, Medizininformatik und in zunehmenden Maße auch Software Engineering.

Im Software Engineering bieten sich wiederum eine Vielzahl konkreter Anwendungszwecke für Ontologien an. Beispielsweise wird unter dem Stichwort "Ontology-driven Architecture" der Einsatz von Ontologien als Artefakte im Software-Design diskutiert, aus dem entsprechend dem Paradigma der modellgetriebenen Entwicklung weitere Artefakte wie z.B. Quellcode abgeleitet werden.

Explizite, formale Ontologien im Sinne des Wissensmanagements finden sich im Software Engineering noch eher selten. Während es im Bereich der Software-Strukturen einige Ontologien gibt, die von konkreten Programmiersprachen abstrahieren, und somit breit einsetzbar sind, gibt es kein allgemein verwendbares Modell für den Software-Lebenszyklus. Die vorhandenen Modelle sind entweder zu abstrakt (z.B. Falbo), decken nur ein spezifisches Anwendungsszenario ab (Dhruv) oder liegen nicht formal in einer Wissensrepräsentation vor. So benutzen z.B. (Falbo et al., 2004) Ontologien zum Beschreiben von SE-spezifischen Organisationsgedächtnis-Inhalten, auch im oben erwähnten IE-SE Ansatz spielen Ontologien eine zentrale Rolle zur Beschreibung von Lessons Learned (vgl. z.B. (Tautz & Wangenheim, 1998; Althoff et al., 2001). (Wille et al., 2003) diskutieren ontologische Modellierungsprobleme, die im Kontext der Standardisierungsbestrebungen für den Guide to the Software Engineering Body of Knowledge (ISO/IEC TR 19759) aufgetreten sind. Von verschiedenen Herstellern und Projekten wurden zur Zeit des Projektanfangs von WAVES Ontologien zum Software Lifecycle Management erstellt, welche in erster Linie dazu dienen, die verschiedenen am Software-Erstellungsprozess beteiligten Werkzeuge zu integrieren.

Neu in WAVES war die Zielsetzung, eine SE-spezifische Referenzontologie zu schaffen, so wie die Idee, über Wikis die Ontologienutzung und –änderung deutlich zu vereinfachen. Hochinnovativ für den WM-Bereich ist die Idee – aufbauend auf Ergebnissen aus dem bmb+f Projekt Compobench und QBench – z.B. Quellcode-Strukturen zu extrahieren, um sie auf der Ebene des Wissensmanagements (Kontextualisierung etc.) als ontologische Struktur nutzbar zu machen.

3. Die WAVES Use-Cases (Arbeitspakete AP1/AP3)

Die Anforderungserhebung und die daraus resultierenden Ergebnisse spielen eine zentrale Rolle für den Projekterfolg, da sie die Basis für die im Rahmen des Projektes zu entwickelnden Funktionalitäten bilden. Im folgenden Abschnitt wird die Art der Anforderungsbeschreibung und die Methodik zur Anforderungsanalyse erläutert, anschließend wird die Ausgangsituation und die Use Cases der einzelnen Anwendungspartnern vorgestellt und abschließend die in WAVES erarbeiteten Use Cases zusammengefasst.

3.1 Narrative Use Cases

Aufgrund der verschiedenen Ausgangssituation der Anwendungspartnern war bereits zu Projektbeginn zu erkennen, dass die einzelnen Partner unterschiedliche Use Cases formulieren werden, die anschließend gemeinsam diskutiert und priorisiert werden sollten. Daher wurden im Rahmen des Projekts WAVES narrative Use Cases verwendet, da diese eine gute Grundlage für die Kommunikation und Diskussion der Anforderungen bietet. Die Use Cases wurden in strukturierter Textform festgehalten.

Ein Use Case (Anwendungsfall) ist hierbei eine narrative Beschreibung des Verhaltens eines Systems. Dabei reagiert das System auf die Aufträge eines Hauptakteurs, der damit ein bestimmtes Ziel verfolgt. Der Use Case beschreibt die Interaktion zwischen System und Akteur im Normalfall (Haupt-Erfolgsszenario) sowie evtl. in abweichenden Fällen wie Varianten, Erweiterungen oder Fehlersituationen mit jeweils nur wenigen (3-9) Schritten. Er benennt seine Voraussetzungen und Auswirkungen. Szenario nennt man eine konkrete Ausprägung eines Use Case; ein Szenario hat also nur einen Ablauf.

Die Use Cases in WAVES sollten speziell die Möglichkeit der Kommunikations,- und Diskussionsgrundlage bieten, wie auch die Möglichkeit die Anforderungen zu validieren. Um dies zu gewährleisten wurden für die Use Cases eine bestimmte Struktur und verschiedene Detail-Ebenen festgelegt. Dies ermöglichte allen Anwendungspartner an dem Prozess der Formulierung wie auch Validierung der Use Cases beteiligt zu sein.

Die Use Cases in dem Projekt WAVES hatten die folgende Struktur:

Bisheriges Problem und Ziele:

- Warum ist dieser Use Case formuliert worden?
- Was sind die bisherigen Probleme?

Beteiligte und Interessen:

- Wer ist der Akteur und welche Interessen verfolgt er?

Auslöser:

- Was ist der Auslöser für den Use Case?

Ziel:

- Welches Ziel verfolgt der Akteur?

Zusicherung im Erfolgsfall:

- Was sollte das Ergebnis des Use Cases im Erfolgsfall sein?

Haupt-Erfolgsszenario

- Beschreibung des Haupt-Szenarios

Varianten und Erweiterungen:

- Welche Varianten und Abweichungen kann es vom Hauptszenario geben?

Fehlerfälle:

- Welche Probleme und Fehler können auftreten?

Offene Fragen und Anmerkungen:

Diese Struktur konnte für die wie folgt definierten Detail-Ebenen verwendet werden:

Überblick:

Abläufe, die nicht in einem Schritt durchlaufen werden, sondern z.B. mehrere menschliche Akteure betreffen.

Benutzerziel:

Abläufe, die von einem Benutzer in einer Sitzung durchlaufen werden und ein für die Arbeit des Benutzers relevantes Ziel erreichen.

Teilziel:

Abläufe, die aus Komplexitätsgründen aus Benutzerziel-Use-Cases herausgebrochen werden.

Detail:

Abläufe, die aus Komplexitätsgründen aus Benutzerziel- oder Teilziel-Use-Cases herausgebrochen werden und von mehreren verschiedenen Use-Cases verwendet werden. Detail-Use-Cases stehen also auf einer Ebene, auf der es um wiederkehrende technische Abläufe geht.

3.2 Vorgehensweise bei der Anforderungsermittlung

Die Anforderungsermittlung umfasst neben den Anforderungen der einzelnen Partner an eine Wissensmanagement-Lösung zur verteilten Entwicklung von Software ebenfalls die Erhebung der Werkzeug-Landschaft, Informationen zum Entwicklungsprozess bei den einzelnen Partnern, sowie die Abhängigkeiten der Werkzeuge. Diese Informationen gewährleisten eine spätere Integration von WAVES in die Firmen Umgebung.

Der Datenerhebungs-Prozess lässt sich in die folgenden drei Schritte unterteilen:

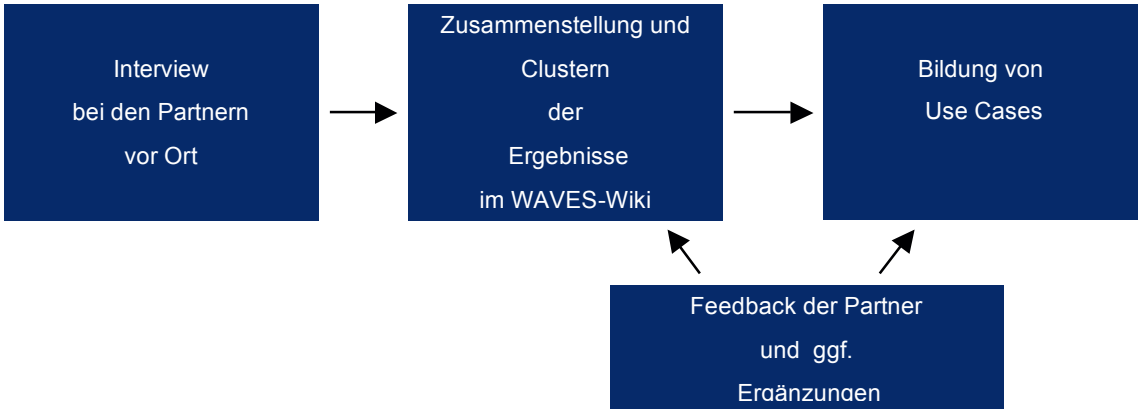
1. Initiale Datenerhebung bei den einzelnen Partnern
2. Konsensbildung der Anforderungen aller Partner
3. Fortschreibung der Anforderungen

3.2.1 Initiale Datenerhebung bei den einzelnen Partnern

Ziele des ersten Schrittes:

- Erhebung der Anforderungen bei den einzelnen Partnern vor Ort
- Clustern der Anforderungen jedes einzelnen Partners
- Umwandlung der Anforderungen in Use Cases

Für diesen Schritt wurden Interviews mit den einzelnen Anwendungspartnern bei den Partnern vor Ort geführt. Die erhobenen Daten wurden in Form von Rohdaten zusammengestellt und nach Themenbereichen geclustert. Das Ergebnis wurde den Endanwendern zum Review übergeben, so dass bereits vor der Formulierung der Use Cases mögliche Ergänzungen oder Verbesserungen vorgenommen werden konnten.



Vorgehen bei der initialen Anforderungserhebung

Anschließend wurden die Anforderungen in Use Cases umgewandelt. Die Uses Cases durchliefen anschließend ein erneutes Review durch die jeweiligen Anwendungspartnern.

Nach Abschluss des Review-Prozessese erfolgte der Schritt 2 der Anforderungsanalyse.

3.2.2 Konsensbildung der Anforderungen aller Partner

Ziele des zweiten Schrittes:

- Clustern der Use Cases von allen Partnern
- Priorisierung der Use Cases von allen Anwendungspartnern
- Konsens aller Anwendungspartner

Im 2.Schritt wurden die Use Cases der einzelnen Anwendungspartner zusammengeführt und nach Themengebieten geclustert. Das Ziel des 2. Schrittes war die Priorisierung der Use Cases bzw. Themengebiete sowie die Erreichung eines Konsenses aller Anwendungspartner. Hierfür wurden zuerst die Use Cases aller Partner nach Themengebieten zusammengefasst. Anschließend erfolgt ein Review aller Use Cases durch alle Anwendungspartnern. Durch das gemeinsame Review konnte ein Konsens über die Themengebiete wie auch die Priorisierung der initialen Anforderungen gefunden werden.

3.2.3 Fortschreibung der Anforderungen

Im Laufe des Projektes wurden die Anforderungen iterativ fortgeschrieben. Dabei wurden einerseits bereits bestehende Anforderungen angepasst, als auch neue Anforderungen aufgedeckt. Das iterative Fortschreiben wurde durch eine kontinuierliche Evaluation (siehe Kapitel Evaluation) der einzelnen WAVES Releases durch die Partner gewährleistet.

3.3 CAS Software AG

3.1.1 Kurzvorstellung CAS Software AG

Die CAS Software AG ist führender deutscher CRM-Spezialist für den Mittelstand und Komplettanbieter für das Kunden- und Informationsmanagement mit Standardsoftware und Individuallösungen. Die CAS Software AG entwickelt und vermarktet Software-Lösungen speziell für mittelständische Unternehmen, um diese nachhaltig erfolgreicher zu machen. Als führender deutscher Spezialist für das Customer Relationship Management (CRM) im Mittelstand bietet CAS Software eine bereichsübergreifende CRM-Lösung, die den Aufbau, die Pflege und den Ausbau von dauerhaften und profitablen Kundenbeziehungen ermöglicht.

Mit den Software-Lösungen von CAS können mittelständische Firmen eine effiziente Prozessorganisation einrichten, Risiken minimieren, das Unternehmenswissen besser nutzen und das Potenzial des Unternehmens optimal verwerten. Über 100.000 Anwender arbeiten täglich mit Produkten von CAS Software. Damit zählt das Karlsruher Unternehmen zu den führenden Komplettanbietern für das Kunden- und Informationsmanagement im Mittelstand.

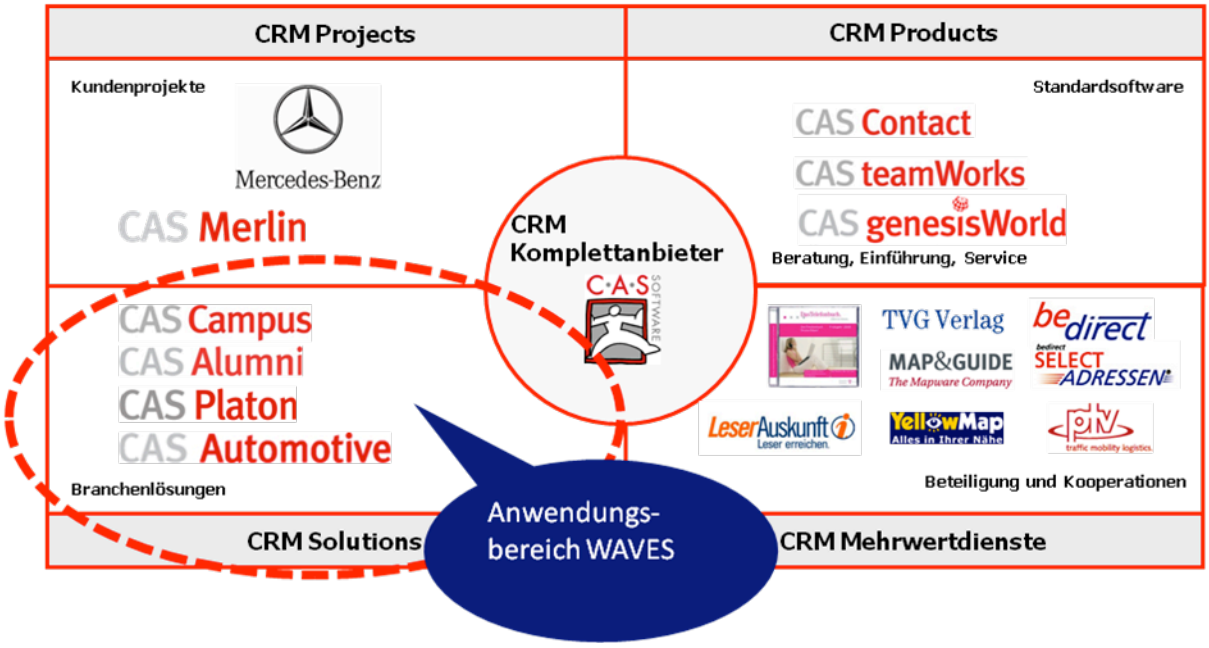
Seit 1986 steht die CAS Software für Innovation, solides Wachstum und Software-Lösungen, die unsere Kunden erfolgreicher machen. Das Unternehmen wurde von dem heutigen Vorstandsvorsitzenden Martin Hubschneider und dem Vorstand Ludwig Neer gegründet und ist komplett in Besitz von Vorständen und Mitarbeitern. Heute arbeiten bei CAS über 100 Spezialisten an aufregenden Projekten und marktführenden Lösungen für mittelständische Firmen und Großunternehmen.

Unser erster Kunde für Individuallösungen war 1986 die Mercedes-Benz AG. Seitdem entwickeln und erweitern wir für die Nutzfahrzeugsparte der DaimlerChrysler AG das Mercedes-Benz Kundenberatungssystem mit heute weltweit über 3.000 Anwendern. Unser Verständnis von nachhaltigem Kundenbeziehungsmanagement beweist sich darin, dass unser langjähriger Projektkunde DaimlerChrysler weiterhin unsere aktuelle Referenzliste anführt.

Im Bereich Standard-Software bieten wir praxisorientierte Lösungen für das operative Kundenmanagement und eine effektive, unternehmensweite Zusammenarbeit. Unsere Produkte zählen zu den innovativsten und erfolgreichsten Lösungen in Europa. Mit dem Kontaktmanager CAS Contact, der CRM-Groupware CAS genesisWorld und der Intranet-Groupware CAS teamWorks können mittelständische Unternehmen neue Kunden gewinnen, langfristige Kundenbeziehungen aufbauen und die Effizienz ihrer Mitarbeiter verbessern. Wir fokussieren uns auf die Bedürfnisse des Mittelstands und setzen daher auf praktikable Software-Einführungen, kurze Projektlaufzeiten sowie preiswerte Lizenzen und Wartungsgebühren.

Für unsere innovative Produktpalette und herausragende Erfolge im Mittelstand erhielten wir zahlreiche Auszeichnungen und Preise, darunter den 'IT Prize der Europäischen Union', den 'CRM Award Mittelstand', den 'TeleTalk Best of CeBIT Award', die Auszeichnung 'Member of TOP 100 des deutschen Mittelstands' sowie den Sonderpreis 'Bestes Innovationsklima' in der Vergleichsstudie TOP 100 jeweils in 2003, 2004 und 2005. In 2006 erhält CAS Software bereits den Preis des innovativsten Unternehmens im deutschen Mittelstand der TOP 100 Studie.

Die CAS Software ist in vier Geschäftsbereiche unterteilt, CRM Projects, CRM Products, CRM Solutions, CRM Mehrwertdienste, welche sich jeweils auf Standardprodukte, Mehrwertdienste, Individuallösungen oder Branchenlösungen spezialisiert haben.



Relevante abgeschlossene F&E-Projekte:

Ziel des Projektes WAVES für CAS ist es, die Produktivität und Qualität bei der Softwareerstellung erheblich zu steigern. Die CAS hat an folgenden relevanten F&E-Projekten teilgenommen:

- COMPOBENCH (BMBF): Methoden und Werkzeuge zur Konstruktion komponentenorientierter Softwaresysteme
- QBench (BMBF): Methoden und Werkzeuge zur Sicherung der inneren Qualität bei der Evolution objektorientierter Softwaresysteme
- UNCODE (Europäische Kommission – ESSI): Unified Modelling Language for Component Design
- ProSoft (Wirtschaftsministerium Baden-Württemberg): Professionalisierung der Softwareentwicklungsprozesse (KMU-adaptiertes Reifegrademodel)

3.1.2 Ausgangssituation

Die CAS Software AG ist ein typisches, mittelständisches Softwareunternehmen und kennt deswegen sehr gut die typischen Herausforderungen, denen sich derartige Unternehmen gegenübersehen: Das Personal ist oft für einzelne Funktionen eher knapp, viel Wissen ist ausschließlich in den Köpfen von Schlüsselmitarbeitern, es wird mit einer großen Flexibilität auf Kundenwünsche eingegangen.

Der innerhalb der CAS an WAVES teilnehmende Geschäftsbereich CRM Solutions wurde 2005 neu mit 8 Mitarbeitern und einem Produkt gegründet und wuchs in den Folgejahren schnell auf (am Ende der Projektlaufzeit) 24 Mitarbeiter, 8 Studenten und 5 Produkte. In diesem Einsatzszenario waren ein anfangs geringer Organisationsgrad, der Aufbau von Know-How, Strukturen, Prozessen und Dokumenten wichtige Themen. Dies zeigt sich im Wissensmanagement im Entwicklungsprozess vor der Teilnahme an WAVES: Entsprechende Infrastruktur war erst im Aufbau: Viel Wissen war (vermutlich) vorhanden, aber Ablageorte und zu Wissensträger waren unklar. Anderes Wissen musste aufgebaut und verfügbar gehalten werden. Es kam zu konkurrierenden Systemen und Insellösungen. Ein CRM-System mit Kunden- und Projektdaten existierte, wurde aber je nach Team (Vertrieb, Projektmanagement, Produktmanagement, Vertrieb) unterschiedlich stark genutzt. Für diese Situation waren folgende Probleme charakteristisch: Unterschiedliche Nutzung von Systemen je nach Team, uneinheitliche Dateiablagen, Informationen lagen vor wurden aber entscheidenden Personen nicht oder erst spät bekannt.

Mit dieser Grundlage passte die CAS sehr gut als Anwendungsunternehmen zu WAVES und konnte wichtige Anforderungen zu WAVES beitragen.

3.1.3 Informationsquellen: Übersicht Systeme, Tools und Informationsartefakte

Die CAS verwendet folgende Werkzeuge:

- *Wiki:* MediaWiki, Wiki aus Codebeamer
- *Versionssystem:* SVN, Microsoft Source Safe
- *Bugtracking:* Track+, Codebeamer (hier werden auch zur Kollaboration Pflichtenhefte und Tasklisten hinterlegt)
- *Office:* Excel, Word (Lastenheft, Pflichtenheft)
- *Projektplanung:* Excel, zeitweise auch mit MS Project, Aufgabenplanung erfolgt auch im Wiki
- *Entwicklungsumgebungen:* Eclipse, Borland

- *Sonstiges:* CRM-Groupware CAS genesisWorld unter anderem zum Dokumentenmanagement / Ablage, Terminplanung, Verwaltung und Durchführung der Kundenkommunikation.
- *Als Programmiersprachen kommen dabei zum Einsatz:* Java, XML, VBA, Delphi

Es gibt kein festgeschriebenes Softwareentwicklungsmodell, es werden agile Methoden eingesetzt.

3.1.4 Anforderung: Integrierte Suche

Verschiedene Artefakte aus dem Entwicklungsprozess werden an verschiedenen Orten aufbewahrt. Beispiele: Wiki, Bugtracker, Konfigurationsmanagementsystem (CM).

Dazu behindern Kulturengrenzen wie etwa zwischen Vertrieb und Entwicklung den Fluss von Information: Unterschiedliche Beteiligte pflegen eigene Aufbewahrungsorte für Informationen. Das ist oft sinnvoll und für die Beteiligten jeweils auch effizient. Beispiel: Vertrieb nutzt hauptsächlich das CRM-System, die Entwickler hauptsächlich Bugtracker, CM, …

Die Anforderung integrierte Suche bedeutet: Mit einer integrierten Suche jedem Artefakt, jedem Beteiligten maximal gerecht zu werden und trotzdem gleichzeitig alle Informationen für alle zur Verfügung stellen zu können.

3.1.5 Anforderungen: Anforderungsverwaltung /- verfolgung in SW-Projekten, Indizierung von Meetingprotokollen

Die Erhebung und Modifikation von Anforderungen geschieht durch verschiedene Personen im Laufe eines Softwareentwicklungsprojekts: Im Consulting / in der Projektleitung, im Produktmanagement, im Vertrieb oder auch durch Softwareentwickler selbst. In auch nicht immer formellen Besprechungen finden zudem manchmal informellere Anforderungsänderungen oder -aufnahmen statt. Alle Anforderungen müssen an verschiedene Beteiligte innerhalb des Softwareentwicklungsprozesses kommuniziert werden.

Die Verwaltung und Verfolgung von Anforderungen ist eine zentrale Aktivität im Softwareentwicklungsprozess. Damit stehen Anforderungen im Mittelpunkt beim Wissensmanagement im Entwicklungsprozess und müssen entsprechend berücksichtigt werden.

3.1.6 Anforderung: Cockpit, freie Projektsichten

Jedes Projekt kann durch spezifische Kunden und Ziele eigene Kennzahlen besitzen. Diese Kennzahlen können sich außerdem je nach Projektsituation und Projektalter ändern. Sichten auf Informationen im Projekt müssen sich deswegen anpassen lassen: An das Projekt, dessen Situation und Alter.

3.1.7 Anforderung: Integration des CRM-Systems

Die CAS Software AG ist Hersteller eines erfolgreichen CRM-Systems und setzt dieses auch selber ein. Dieses CRM-System enthält viel Wissen aus Vertriebs- und Anforderungsphase: Darunter Workshop-Mitschriften, Notizen zu Vertriebsterminen, Notizen zu Kundentelefonaten, Aufgaben und Termine. Diese Informationen sind auch für den Softwareentwicklungsprozess wichtig und müssen entsprechend einfach verfügbar sein.

3.4 disy Informationssysteme GmbH

3.4.1 Kurvorstellung disy Informationssysteme GmbH

Die disy Informationssysteme GmbH (kurz disy) wurde 1997 als Spin-off des Forschungszentrums Informatik (FZI) gegründet und hat seitdem ihren Firmensitz in Karlsruhe. Heute beschäftigt disy über 60 Mitarbeiter. disy bietet maßgeschneiderte Lösungen für Umweltinformationssysteme (UIS), Berichts- und Auswertesysteme mit Raumbezug (Spatial Reporting), Geodateninfrastrukturen (GDI),

Geoinformationssysteme (GIS), Metadateninformationssysteme auf der Basis eigener Software-Produkte und in Kooperation mit Partnern an.

Die Kernkompetenz von disy liegt im Aufbau von Geo- und Sachdateninfrastrukturen auf der Basis eigener Technologien. Hierzu hat sich das Unternehmen auf die Entwicklung, Integration und Optimierung von Datenbanken und (Geo-)Data Warehouses, den Aufbau von Geodateninfrastrukturen (GDI) sowie auf die Erstellung von Web-basierten Anwendungen auf der Basis von Java- und XML-Technologien spezialisiert.

Das Unternehmen besitzt tief gehendes Wissen und umfangreiche Erfahrung insbesondere bei der Integration, dem übergreifenden Zugriff und der Auswertung von heterogenen und verteilten Daten und beim Aufbau von Internet-basierten Lösungen. disy setzt bei der Lösungserstellung agile und testgetriebene Entwicklungsmethoden ein. Als Spin-Off des Forschungszentrums Informatik und der Universität Karlsruhe haben die disy Mitarbeiter auch langjährige Erfahrung in der Durchführung von Forschungsprojekten.

Ein besonderer Schwerpunkt von disy ist die Konzeption und Umsetzung von Geodateninfrastrukturen (GDI), geographischen Informationssystemen (GIS) und von Auswertungen mit geographischem Bezug (Spatial Reporting). Mit disy-Produkten und -Lösungen werden in Deutschland mehr als 10000 Arbeitsplätze bedient.

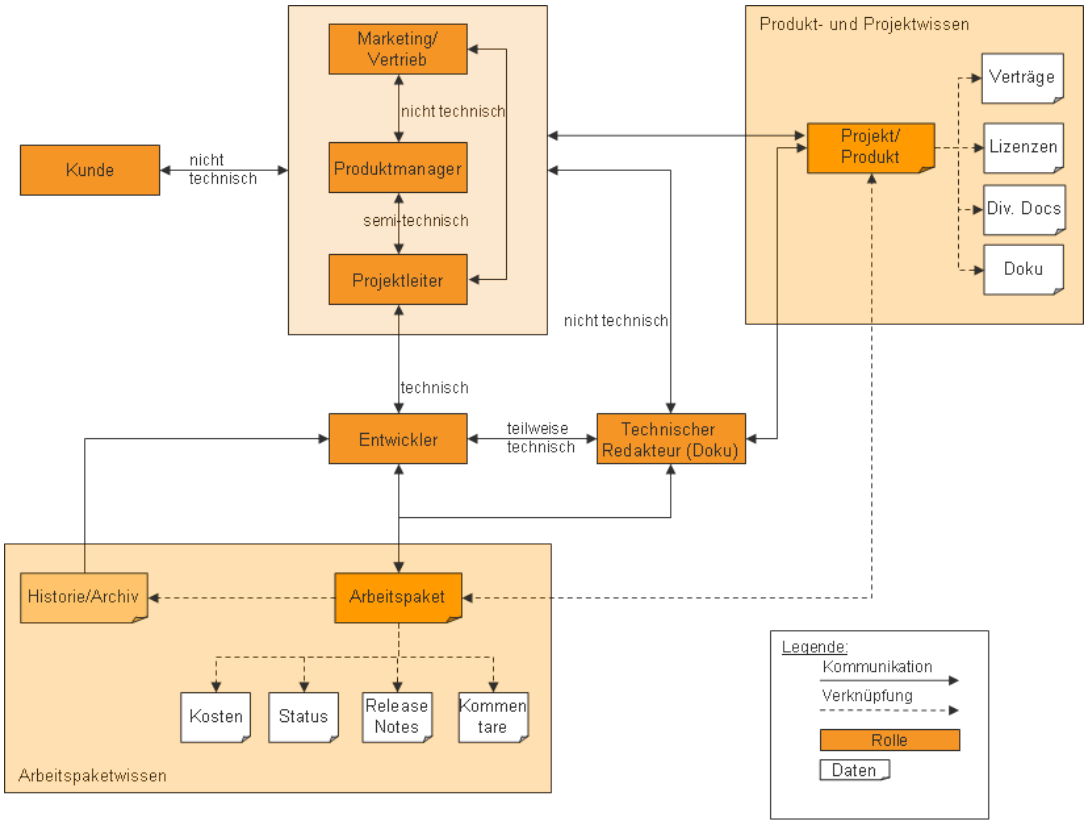
Über 50 hoch qualifizierte Mitarbeiter, überwiegend Informatiker, Mathematiker und Software-Entwickler aus einer der renommiertesten Hochschulen Deutschlands, der Universität Karlsruhe, bilden das Rückgrat des Erfolgs. Die über 10-jährige erfolgreiche Tätigkeit am Markt sowie die vollständige Finanzierung durch private Investoren mit enger Firmenbindung unterstreichen das solide Geschäftskonzept.

3.4.2 Ausgangssituation Wissensmanagement vor WAVES

Ziel der Beteiligung der disy Informationssysteme GmbH war es, den Wissensaustausch zwischen Entwicklerteams, Projektleitung, Produktleitung, Dokumentation, Vertrieb und Kunden zu verbessern. disy hatte bereits damit begonnen, Wissensmanagement-Werkzeuge wie Wikis für den Informationssaustausch einzusetzen, hat jedoch enormes Potential bei der Weiterentwicklung solcher Plattformen hin zu kollaborativen Wissensaustauschplattformen gesehen.

Die Kommunikations- und Wissensaustauschprozesse sowie das Wissen, das die Mitarbeiter von disy bei der Projektabwicklung und Produktentwicklung benötigen, hängen stark von der Rolle, die die Mitarbeiter in dem Prozess einnehmen ab. Während die Kommunikation auf der Ebene des Vertriebs, der Projekt- und Produktleitung eher informell und weniger technisch ist, wird die Kommunikation mit und zwischen den Entwicklern wesentlich technischer. Eine zentrale Rolle übernimmt auch der technische Redakteur, der aus technischen Eigenschaften häufig eine nicht technische Beschreibung formulieren muss.

Weder ein zentrales Feature/Bug-Repository ist vorhanden (wo landen also die Probleme, wer ist zuständig), noch ist dies mit Release Notes verknüpft. Es fehlten auch Automatismen, die einen unterstützenden Charakter haben. Eine wesentliche Eigenschaft von Informationen, die beim Prozess anfallen ist, dass sie sehr dynamisch sind. Dokumente ändern sich teilweise häufig oder werden neu erzeugt (oder sind veraltet). Als wesentliche Herausforderung hat disy daher das Problem der Verwaltung von Verknüpfungen zwischen Dokumenten, Arbeitspaketen und Aufgaben unter dem Aspekt einer zeitlichen Veränderung gesehen.



disy folgt weitgehend einem eXtreme-Programming-Prozess und setzt dementsprechende Werkzeuge (z.B. XPlanner) ein. Dabei hat disy eine eigene auf die Bedürfnisse des Unternehmens angepasste Vorgehensweise auf Basis von XP und SCRUM entwickelt. Das Vorgehen ist nur an dedizierten Stellen formal, um einen möglichst großen Spielraum für innovation zu lassen. So wird auch zwischen externen Dokumenten (die gemeinsam mit den Kunden entwickelt werden) und internen Arbeitspaketen unterschieden, die für die internen Belange konzipiert und angeordnet werden.

Bei der Umsetzung wird ein recht intensiver Einsatz von Offline Medien (z.B. Karteikarten, Whiteboards, Flip-Charts genutzt. Auf CMMI wird momentan verzichtet.

Die wesentlichen Rollen, ihr Wissen und Kommunikationswege sind in obiger Abbildung dargestellt.

Neben dem informellen Austausch von Wissen, kommen unterschiedliche Werkzeuge zu Einsatz:

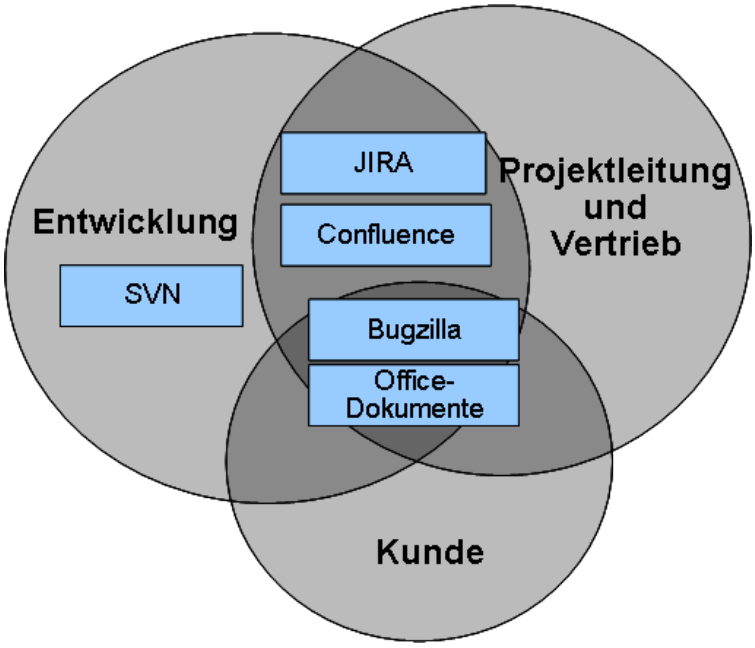
- Das Wiki(JSPWiki) dient für Dokumente der internen Arbeitspakete und wird von allen Mitarbeitern genutzt. Es steht ausschließlich im Intranet zur Verfügung. Allerdings ist eine Umstellung auf Confluence angedacht, so dass auch Teile des Wissen abhängig von Nutzungsrechten Kunden direkt bereitgestellt werden kann
- Als Bugtracking-System wird Bugzilla eingesetzt, dass jedoch nicht mehr neuesten Standard genügt. Eine Umstellung auf Jira ist daher geplant.
- Für die Dokumente der externen Arbeitspakete wird das Datei-System genutzt
- Zur Versionsverwaltung und Verwaltung des Quellcodes wird CVS eingesetzt. Eine Umstellung auf SVN ist geplant.
- Die Programmierung erfolgt in Java. Eine Umstellung auf eine andere Programmiersprache ist nicht angedacht.
- Office: MS-Word/MS-Excel im Dateisystem
- Projektplanung: XPlanner
- für die Ablage von Aufwänden: Stundenplänen, Zuständigkeiten, Planung der Iterationen, Status

- Programmcode-Dokumentation: Javadoc
- Sonstige:
  - Eclipse mit diversen Plugins
  - Ant
  - CruiseControl
  - ReleaseNotes
  - Wiki: FitNesse zur Durchführung von Akzeptanztests (bislang nicht breit im Einsatz)

Zu Projektbeginn waren bei disy bereit einige Werkzeuge im Einsatz, die jedoch für einzelne Aufgaben genutzt wurden ohne eine integrative Betrachtungsweise. Ein Werkzeug, das existierende Wissensquellen und Werkzeuge verknüpft fehlte komplett.

3.4.3 Anforderungen und Problemstellungen

Aus der Beschreibung der Ausgangssituation werden bereits die wesentlichen Anforderungen sichtbar, die die Grundlage für die Entwicklung darstellen. Sie lassen sich in funktionale und technische Anforderungen unterteilen.



Die **technischen Anforderungen** können im Wesentlichen auf die Erhaltung bestimmter Systeme reduziert werden, ohne die eine Wissensaustausch innerhalb des Unternehmens nicht möglich ist. Hierzu zählt, dass die E-Mail-Kommunikation bestehen bleiben muss. Jede Lösungs muss also berücksichtigen, dass weiterhin ein großer Teil des Wissensaustauschs über E-Mail erfolgt. Das gleiche gilt für die Anwendung von Office-Anwendungen, die auch weiterhin ein zentrales Austauschformat darstellen. Insbesondere bei der Kommunikation nach Außen sind weder E-Mail noch Office-Anwendungen wegzudenken. Ansonsten stellt disy keine expliziten technischen Anforderungen und ist flexible, was den Einsatz von neuen Werkzeugen bzw. den Ersatz von vorhandenen Werkzeugen betrifft.

Die funktionalen Anforderungen werden im Folgenden getrennt nach Wissensnutzung und Wissensartikulation betrachtet.

Die Grundlage für jegliche **Wissensnutzung** ist die Suche. Hierbei besteht das größte Problem, dass in allen verfügbaren Informationsquellen einzeln gesucht werden muss. Bei der Vielzahl der Werkzeuge und Technologien, die bei disy im Einsatz sind, bedeutet dies, dass der Nutzer entweder schon grob wissen muss, in welchem Werkzeug er die gesuchte Information finden kann oder er muss

einen mühsamen Prozess der Wissensrecherche starten. Da dies selten zielführend ist werden übergreifende Recherchen selten gemacht und eher versucht implizites Wissen heranzuziehen, durch Diskussion mit anderen Mitarbeitern, die vielleicht wissen, wo was zu finden ist.

Da dieser Ansatz zum Einen nicht skaliert und gleich mehrere Mitarbeiter involviert sind, was zusätzlich Kosten für die Wissensnutzung bedeutet, ist eine zentrale Anforderung, einen zentralen Einstiegspunkt für sämtliche Informationsquellen zu schaffen. Die Minimale Anforderung ist, dass alle Informationsquellen zumindest berücksichtigt werden, wobei eine intelligentere Suche angestrebt wird, in der beispielsweise bereits in den Werkzeugen automatisch angezeigt wird, in welche anderen Dokumenten, Dateien oder Bugs ein Begriff vorkommt. Durch die Interpertation des aktuellen Benutzerkontexts, wäre eine für den Benutzer zugeschnittene priorisierte Ergebnisliste sicherlich von Vorteil, so dass der Recall auf das Wesentliche reduziert wird und das Ergebnis möglichst präzise wird.

Dies kann auch schon als Übergang zu einer eher proaktiven Informationsnutzung gewertet werden, die auch eine wichtige Anforderung an die Möglichkeiten der Wissensnutzung darstellt. Eine wesentliche Hilfe wäre so beispielsweise die automatische Generierung von Projektseiten oder gar Arbeispaketseiten mit allen relevanten Informationen vorab ausgewählt und gruppiert. Minimalvoraussetzung hierfür ist sicherlich die technische Verknüpfung der einzelnen Informationsquellen. Allerdings sollte der Ansatz weiter gehen, so dass beispielsweise Anforderungen aus einem Projekt mit einem Testfall verknüpft werden, die wiederum mit dem Ergebnis des Testfalls verknüpft sind. Darüber hinaus wäre es wichtig, dass in der Projektsicht nur bestimmte Teile der Dokumente sichtbar sind. Da Weboberflächen häufig für die Eingabe großer Datenmengen nicht geeignet sind (Office ist einfacher) oder der Kunde Office Dokumente verwendet, müssen die Informationen aus Office oder PDF-Dokumenten integriert werden können. Gegebenenfalls auch mit festen Formatvorlagen. Wichtig dabei ist, dass alle Informationen auch in Druckformaten vorliegen müssen um beispielsweise dem Kunden die Informationen zu einem bestimmten Bereich in einem Projekt vorlegen zu können

Das zweite große Thema bei den funktionalen Anforderungen ist die Frage nach den Möglichkeiten der **Wissensartikulation**. So ist ein wesentlicher Bestandteil der Wissensartikulation die E-Mail, die in den unterschiedlichen Phasen eines Projekts, vom Vertrieb bis zum Abschluss, maßgeblich genutzt wird. Momentan sind viele Informationen in E-Mails vorhanden, die jedoch in keinerlei Wissensplattform nachgenutzt werden können, da sie mit den restlichen Informationen nicht verknüpft sind und somit nicht allen zugänglich sind.

Um dem zu entgegnen sollten Möglichkeiten geschaffen werden, E-Mails explizit, z.B über ein cc oder ein bestimmten Betreff, oder implizit durch Filterung des Inhalts bestimmten Projekten zugeordnet werden können. Hier wäre beispielsweise ein separater Bereich „Diskussion“ sinnvoll, in dem die einzelnen E-Mail verwaltet, gruppiert und angeordnet werden können.

3.5 Object International Software GmbH

3.5.1 Kurzvorstellung Object International Software GmbH

Die *Object International* gehört seit ihrer Gründung im Jahr 1991 zu den technologisch führenden Unternehmen im Bereich Objekttechnologie. Basierend auf unseren Frameworks und Softwarekomponenten und mit dem Engagement und dem Wissen unserer Mitarbeiter realisieren wir qualitativ hochwertige, wartungs- und bedienerfreundliche Software für die Zukunft unserer Kunden.

Die Kompetenz des Unternehmens resultiert aus den Erfahrungen vieler erfolgreicher Projekte in Industrie und Wirtschaft sowie aus der Entwicklung technologisch führender Spitzenprodukte. Mit Produkten wie dem vielfach prämierten UML-Modellierungswerkzeug *Together*, dem *NOWAIT Application Framework (*Application Framework zur schnellen und stabilen Anwendungsentwicklung in C++) dem *NOWAIT EAI Server (*Ressourcenschonender Hochleistungsapplikationsserver mit Schwerpunkt EAI (Enterprise Application Integration) und Dienstbereitstellung (Webservices,

Unternehmensdienste, Software-as-a-service, usw.)) sowie dem Business Framework *ecomex (*Softwaresystem für Warenwirtschaft und Buchhaltung) hat *Object International* wiederholt den enormen Technologievorsprung bewiesen. Die Erfahrungen aus diesen Softwareprodukten und den zahlreichen erfolgreichen Softwareprojekten zusammen mit dem offenen Dialog mit den Kunden tragen zum Erfolg der Produkte, Projekte und Dienstleistungen maßgeblich bei.

Mit unserem praxisorientierten Seminarangebot transferieren wir aktuelles Wissen über Objekt- und Komponententechnologie sowie zu Vorgehensmodellen zu unseren Kunden.

Geschäftsbereiche der Object International

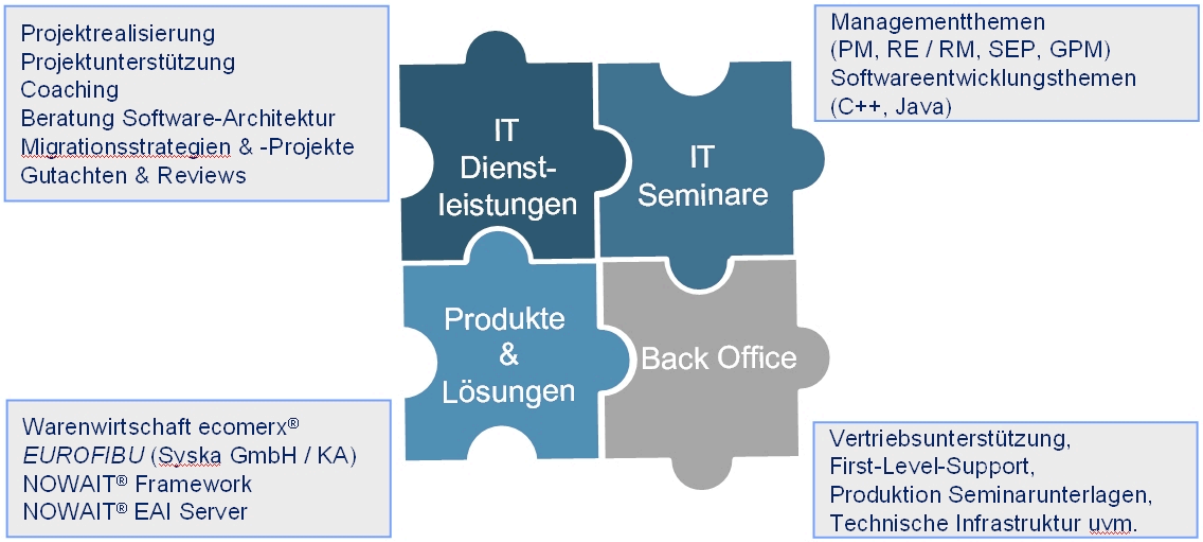


Abbildung 2: Die Geschäftsbereiche der *Object International* und deren Themenschwerpunkte

3.5.2 Ausgangssituation und Status Wissensmanagement vor *WAVES*

Ausgangssituation

KMU typische Rahmenbedingungen: Aufgrund der Unternehmensgröße herrschen bei *Object International* die für KMU typischen Rahmenbedingungen wie flache Hierarchien, eine partnerschaftliche Unternehmenskultur, geringe Zugriffsrestriktionen auf Informationen und Daten sowie eine hohe Flexibilität hinsichtlich des Leistungsspektrums vor. Diese hohe Flexibilität erfordert ein breites Einsatzspektrum der Mitarbeiter und eine rasche Einarbeitung in neue Themen und Trends.

Des Weiteren sind die finanziellen und insbesondere die personellen Ressourcen in KMUs zumeist knapper als in konzernartigen Unternehmensstrukturen - dies betrifft im besonderen personelle Ressourcen, die nicht direkt mit der Leistungserbringung im Zusammenhang stehen („Personalgemeinkosten“).

Die *Object International* arbeitet im Rahmen von IT-Projekten oder Seminaren intensiv mit einem Netzwerk, bestehend aus Partnerunternehmen und einzelnen Freelancern, zusammen. Die Leistungserbringung der Mitarbeiter erfolgt zu überwiegenden Anteilen direkt vor Ort beim Kunden - beispielsweise weil nur der Kunde über die benötigte IT-Infrastruktur verfügt oder die enge Kommunikation mit dem Kunden dies erforderlich macht.

Status Wissensmanagement

Folgendes Ausgangsszenario des Wissensmanagements vor *WAVES* ist bei *Object International* gegeben:

„Wissen“ über Prozesse und/oder Technologien ist in hohem Maß als implizites Wissen vorhanden und sehr stark an einzelne Personen gebunden.

Die Akquisition und das Management von Wissen sind bedarfsgetrieben, beispielsweise weil sich ein Mitarbeiter im Rahmen eines Projektes oder einer Seminarvorbereitung in eine neue Thematik einarbeitet. Ein kontinuierlicher Prozess zur Wissensakquisition oder zum Management von Wissen ist nicht etabliert - des Weiteren existiert kein allgemein verwendetes Werkzeug für das Management von Wissen.

Expliziertes Wissen auf eine Vielzahl von Quellen (IT-Systeme & Datenformate) verteilt - ein zentraler Einstiegspunkt bzw. eine Möglichkeit zur übergreifenden Recherche ist nicht gegeben. Dies erfordert ausgeprägte Kenntnisse der Ablage- und Verzeichnissystematik.

Der Zugriff auf „Wissen“ ist aufwändig – die Recherche erfolgt manuell – beispielsweise mittels Dateisuche mit dem Microsoft Explorer und ist mit einem Wechsel des Applikationskontextes verbunden.

Ein relativ hoher Anteil der IT-Dienstleistungen wird durch die Mitarbeiter direkt vor Ort beim Kunden erbracht - beispielsweise weil nur der Kunde über die benötigte IT-Infrastruktur verfügt oder die enge Kommunikation mit dem Kunden dies erfordert - daher ist der direkte Zugriff auf diese Mitarbeiter und deren Know-How nur eingeschränkt und mit den entsprechenden zeitlichen Verzögerungen möglich. Im Gegensatz dazu ist es für diejenigen Mitarbeiter, die direkt beim Kunden arbeiten, schwierig bis unmöglich nach Informationen oder Daten zu recherchieren, die sich innerhalb der IT-Infrastrukturgrenzen der *Object International* befinden.

Als weiterer Aspekt ist es für eine schnelle Einarbeitung in bereits laufende Kundenprojekte zwingend erforderlich, sehr schnell eine Übersicht über die für das jeweilige Projekt relevanten Daten und Informationen (von den Anforderungsdokumenten bis zum Sourcecode) zu erhalten. In der Regel sind diese Informationen in den unterschiedlichsten Formaten in einem weit verzweigten Dateisystem und/oder in mehreren IT-Systemen abgelegt - was die Recherche nach relevanten Informationen insbesondere beim Einstieg in ein neues Projekt sehr zeitaufwändig macht.

Der jeweilige Mitarbeiter baut sich in der Regel lokal auf seinem Laptop sein eigenes „Wissensmanagement-System“ (Unterschiedliche und nicht integrierte Tools vom Notepad über Outlook bis zu lokalen Wiki-Systemen) auf bzw. ergänzt sein bestehendes Wissensmanagement-System sukzessive um Wissen, das im Rahmen des Kundenprojektes erarbeitet wird. Zur Sicherung des Wissens wäre es wünschenswert dieses bereits erschlossene Wissen - oder zumindest Teile davon - ohne aufwändige manuelle Interaktion und Transformationsprozesse in eine zentrale Wissensbasis zu überführen.

Informationsquellen: Übersicht Systeme, Tools und Informationsartefakte

Nachfolgend eine Übersicht über die bei *Object International* eingesetzten IT-Systeme und Tools und die Verwendung durch die einzelnen Geschäftsbereiche.

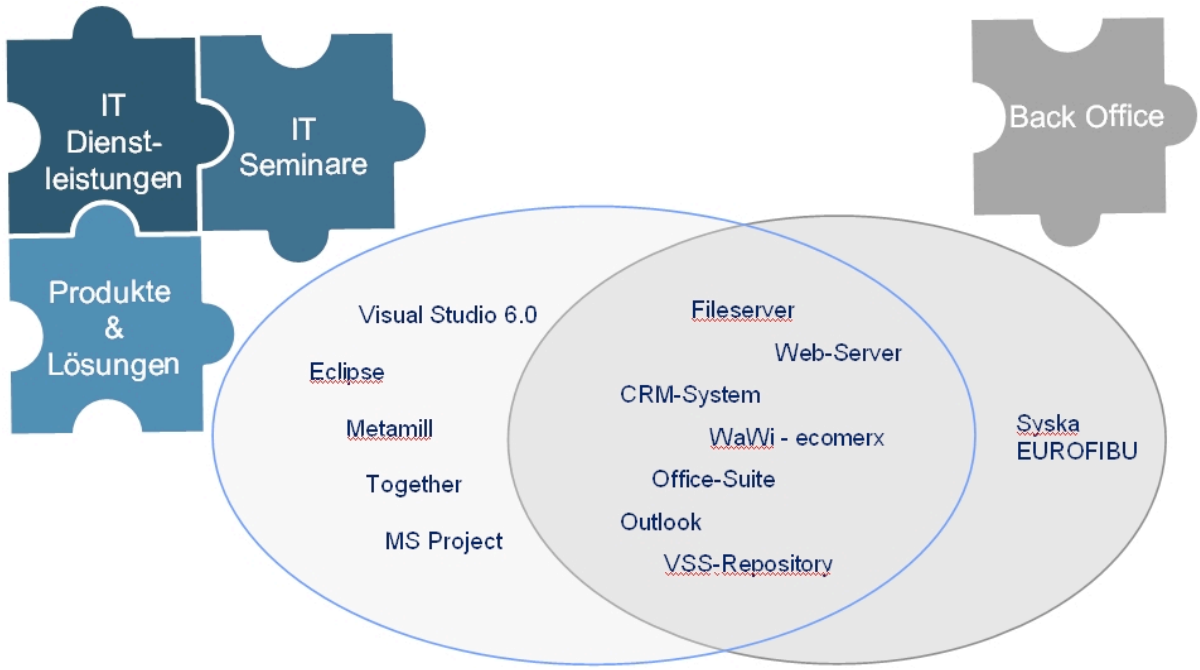


Abbildung 3: Übersicht über die bei Object International eingesetzten Systeme und Tools

Nachfolgend sind die internen und externen Informationsquellen aufgeführt und näher beschrieben.

Interne Systeme und Datenquellen: Zu den internen Systemen zählen der als zentrales Filesystem dienende Windows 2003 Server, das CRM-System cobra Adress Plus 10, das Warenwirtschaftssytem ecomerx, das globale Outlook-Postfach des sowie das zentrale Visual SourceSafe Repository.

Zu den *internen Datenquellen* und *-formaten* zählen Microsoft-Office-Dateien (.doc, .xls, .ppt), PDF- und Textdateien (.rtf, .txt, .asc, .xml), Adress- und Kontaktinformationen sowie Informationen über geschäftliche Transaktionen und die Mailkommunikation mit unseren WaWi-Kunden. Des Weiteren zählen die im Visual SourceSafe Repository gespeicherten Softwareentwicklungsartefakte, Projektdokumente, Schulungsunterlagen (Präsentationsfolien, Übungen & Lösungen, Sourcecode) und versionierte sonstige Dokumente (bspw. Vorlagen, Marketingunterlagen,…) ebenfalls zu den *internen Datenquellen*.

Zu den e*xternen Datenquellen* zählen Web-Portale (bspw. Partnerportale oder allgemeine Portale (Beispielsweise Portale wie Cetus-Links.org, www.cmcrossroads.com, www.itwissen.info, Technorati, www.tecchannel.de oder de.wikipedia.org.) mit Informationen über Produkte, Technologien) sowie Systeme und Datenquellen von Consulting- oder Projektkunden und die *persönlichen Datenquellen* der Mitarbeiter.

Zu den *persönlichen Datenquellen* zählen Outlook (1 – n Postfächer), das lokales Filesystem des Laptops, externe Speichermedien sowie ein lokales Repository (bspw. Subversion und/oder Visual SourceSafe), die lokalen Link- und Infosammlungen (bspw. Web-Browser, Outlook-Notizen, Outlook-Aufgaben, …) und ggf. ein lokales Wiki-System. Je nach Mitarbeiter sind unterschiedliche Ausprägungen der verwendeten Systeme und Formate möglich.

Informationsquellen: Übersicht über die Informationsartefakte

Nachfolgend eine Übersicht über die bei *Object International* vorhandenen Informationsartefakte und ihre Verwendung durch die einzelnen Geschäftsbereiche.

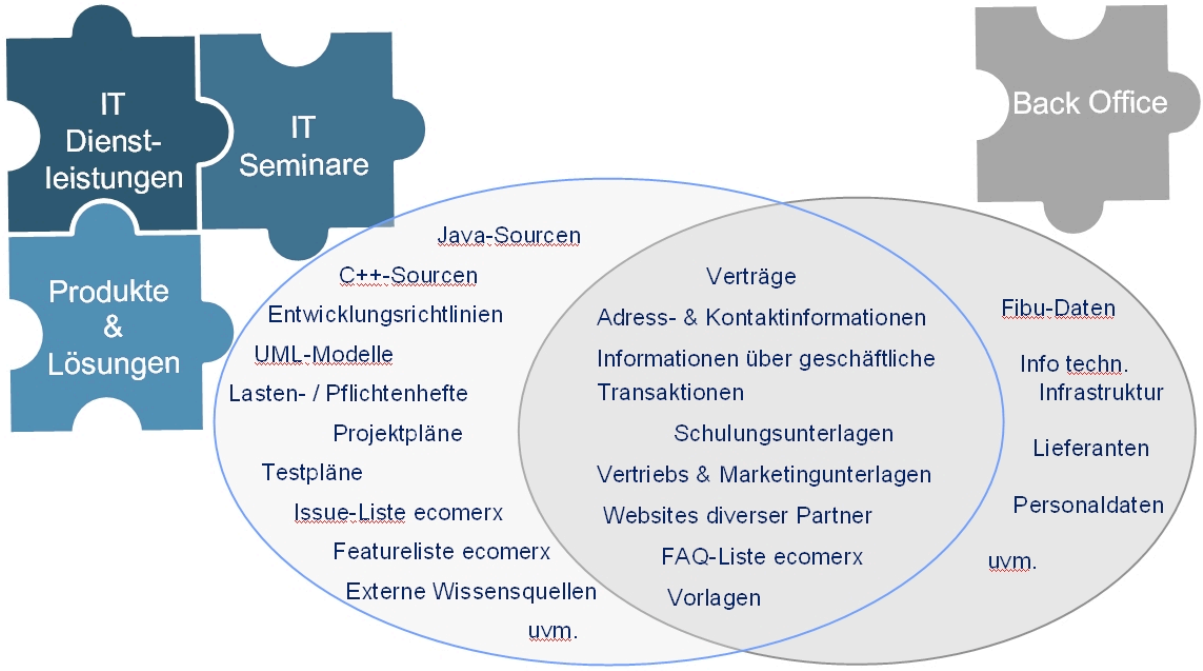


Abbildung 4: Übersicht über die Informationsartefakte der Object International

3.5.3 Anforderungen und Problemstellungen

Bedarfsanalyse - Erwartungen an ein integriertes Wissensmanagement

Bei der Bedarfsanalyse wurden die generellen Anforderungen an das Wissensmanagement sowie an die Wissensmanagement-Plattform identifiziert – nachfolgend die einzelnen Kernaspekte:

Anforderung Wissensakquisition- und transfer:

Ein wichtiger Aspekt ist der Ausbau und Sicherung der Wissensbasis des gesamten Unternehmens – dies bedeutet insbesondere der Schutz vor „Wissensverlust“ durch ausscheidende Mitarbeiter sowie die Reduktion der Abhängigkeit von einzelnen „Wissensträgern“ (u.a. auch wegen bereits beschriebenen „On Site“-Thematik).

Des Weiteren soll die WM-Plattform die effiziente Einarbeitung von Mitarbeitern in für sie neue Themen sowie die Einarbeitung von neuen (oder externen) Mitarbeitern in die Geschäftsprozesse des Unternehmens unterstützen.

Anforderung Steigerung der Produktivität und Useability:

Ein weiterer Aspekt ist die signifikante Reduktion des Zeitaufwandes bei der Recherche nach Informationen und Daten und damit eine direkte Verbesserung der Produktivität der Mitarbeiter.

Das System soll eine ad-hoc-Informationsversorgung unterstützen, damit der Nutzer kurzfristig – beispielsweise für kurzfristig anberaumte Meetings, Präsentationen oder Telefonkonferenzen – in die Lage versetzt wird, Informationen zu recherchieren und diese ggf. anschließend aufzubereiten.

Von einer Wissensmanagementplattform erwarten wir uns auch eine höhere Kunden- und Mitarbeiterzufriedenheit aufgrund der schnelleren Bereitstellung qualitativ hochwertiger Informationen – hier wäre insbesondere die Reduktion der Durchlauf-/Bearbeitungszeit bei Supportanfragen (Bereich Produkte & Lösungen) wünschenswert. Durch eine produktivere

Rechercheumgebung könnte eine größere Anzahl von Supportanfragen sofort online (d.h. während des Telefonats mit dem Kunden) beantwortet werden.

Alle Mitarbeiter des Unternehmens – und nicht nur die Experten – sollen mit der Softwarelösung (Wissensplattform) arbeiten - dies erfordert eine einfache und intuitive Bedienung der Software.

Anforderung Bereitstellung und Verteilung von Wissen:

Die WM-Plattform soll einen Grossteil der relevanten Datenquellen und Datenformate erschließen und ein kollaboratives Erarbeiten von Wissen unterstützen.

Das „persönliche" Wissensmanagement sollte durch geeignete Mechanismen ebenfalls unterstützt werden (alleine auch wegen der bereits beschriebenen „On Site"-Thematik) – des Weiteren sollte die WM-Plattform in der Lage sein „Wissenslücken" – d.h. Recherchen ohne Treffermenge – aufzudecken.

Ökonomische- und technische Aspekte:

Ein geringer „TCO" (Total Cost Ownership) der WM-Plattform ist zwingend erforderlich – daraus leiten sich folgende Forderungen ab:

Die Wissensplattform darf keine hohen Anforderungen an die Zielhardware (Prozessor, Hauptspeicher, Festplattenausstattung) stellen und soll sich einfach - d.h. mit ein bis max. zwei AT Aufwand - in Betrieb nehmen lassen. Idealerweise genügt ein handelsüblicher PC mittlerer Ausstattung. Des Weiteren sollte sich die Lösung gut in die vorhandene IT-Infrastruktur und Toolkette integrieren lassen und möglichst auf Open Source Komponenten basieren. Damit werden einerseits die sonst anfallenden Lizenzkosten eingespart – eine Weitergabe bzw. Inbetriebnahme der Plattform bei Dritten (Kunden oder Partnerunternehmen) wäre dann bei Bedarf/Interesse ebenfalls möglich - und andererseits können Teile der Wissensplattform bei Bedarf selbst angepasst.

Die Anbindung von Dritten - beispielsweise von Partnerunternehmen und/oder Kunden - an die gesamte „Wissensbasis" des Unternehmens (oder dedizierte Teile der „Wissensbasis") sollte möglich sein.

3.5.4 Die WAVES Use Cases bei Object International

Übersicht über die für Object International wesentlichen Anwendungsfälle

- Use Case 1: Räumlich verteilte Wissensnutzung
- Use Case 2: Integrierte Suche über Dokumente und Software spezifische Quelldaten
- Use Case 3: Einführung von Wikis im Unternehmen
- Use Case 4: Informationsbedarf im Unternehmen identifizieren
- Use Case 5: Sharing von Wissen und Motivation zur Wissensartikulation

Nachfolgend werden das jeweilige Anforderungsszenario, die Nutzergruppen und die Anforderungen der einzelnen Use Cases im Detail beschrieben.

Use Case 1: Räumlich verteilte Wissensnutzung

Kurzbeschreibung Szenario:

Räumlich verteiltes Arbeiten (Suche & Artikulation von Wissen) im Team ggf. unter Einbeziehung von externen Mitarbeitern und Mitarbeitern von Projekt- oder Consulting-Kunden.

Die „Wissensbasis" des Unternehmens soll auch für Mitarbeiter, die räumlich verteilt arbeiten – beispielsweise weil sie sich beim Kunden vor Ort, im Home-Office oder auf Geschäftsreise befinden – zugänglich sein.

Nutzergruppen: Primär: Alle Mitarbeiter des Unternehmens / Sekundär: Partnerunternehmen, Kunden (=>erfordert Rechtesystem)

Anforderungen:

- Effiziente Nutzung im Intra- / Internet unter Berücksichtigung der technischen Barrieren (eingeschränkte Bandbreite, Kundenfirewall) sowie der Anforderungen an die Sicherheit des Zugriffs auf Daten und Informationen.
- Generell gilt: Falls fremde Dritte Zugang zur Wissensplattform erhalten, ist eine selektive Offenlegung der „Wissensbasis" bei allen Szenarien / Use Cases zwingend erforderlich – dies setzt allerdings ein aufwändiges Rollen- / Rechtesystem voraus.

Beispielszenario (Bereich IT-Dienstleistungen):

Ein Mitarbeiter ist bei einem Kunden vor Ort mit dem Thema Anforderungsmanagement betraut und benötigt sehr kurzfristig für eine Präsentation eine Checkliste sowie ein Template für das Management und zur Verfolgung von Changes.

Er hat die Vermutung, dass auf dem Firmenserver derartige Artefakte sowie ergänzende Informationen zum Change-Prozess existieren – kennt allerdings den genauen Ablageort (Verzeichnispfad) nicht.

Istzustand: Der Mitarbeiter greift zum Telefon und bittet eine Kollegin aus dem Back Office um Hilfestellung – diese durchsucht die diversen Verzeichnisse des Servers nach Checkliste und Template – hat selbst aber keine oder nur geringe Kenntnisse des fachlichen Kontextes „Change-Management". Die Suche gestaltet sich daher relativ aufwändig – des Weiteren besteht die Gefahr, dass die Kollegin aus dem Back-Office falsche Vorlagen versendet was wiederum zu einer mehrfachen und zeitaufwändigen Interaktion zwischen den beteiligten Personen führt.

Zielzustand: Der Mitarbeiter wird in die Lage versetzt selbst in den diversen Informationsquellen seines Unternehmens nach den gewünschten Informationen zu recherchieren.

Die Mitarbeiterin aus dem Back-Office kann in einer zentralen Datenbasis nach den relevanten Begriffen („Change" / „Change-Management" / oder synonymen Begriffen) suchen während sie mit dem Kollegen telefoniert.

Use Case 2: Integrierte Suche über Dokumente und Software spezifische Quelldaten

Kurzbeschreibung Szenario:

Integrierte Suche über alle relevanten Informationsquellen und Informationsartefakte des Unternehmens. Je nach Problemstellung soll die Suche selektiv in einer oder mehreren Informationsquellen (bspw. Filesystem und/oder Adressdatenbank) erfolgen.

Nutzergruppen: Alle Mitarbeiter des Unternehmens

Anforderungen:

- Indizierung von Office-Dokumenten, Postfächern, Webseiten / Wikis, sowie SW spezifische Informationsartefakte (Java- bzw. C++ - Quellcode, XML-Dateien)
- Einbindung SW spezifischer Systeme (Subversion, Visual Source Safe, Issue-Tracker,…) und Datenbank basierter Systeme (CRM, Warenwirtschaftssystem,...)
- Einheitliche und überschaubare Oberfläche, leichte Bedienbarkeit
- Einbindung in die Toolkette eines Entwicklers (beispielsweise als Plug-In für die Entwicklungsumgebung)
- Hohe Aktualität der Metadaten (erfordert effiziente Indizierungstechnologie / inkrementelle Indizierung)

Use Case 3: Einführung von Wikis im Unternehmen (u. a. Benutzbarkeit)

Kurzbeschreibung Szenario:

Zum Aufbau einer gemeinsamen Wissensbasis soll ein Wiki verwendet werden. Die Inhalte des Wikis sollen von allen Geschäftsbereichen und vom Back Office erstellt und gepflegt werden.

Ist bereits „Wissen" zu einem Thema im Unternehmen verfügbar, soll während der Bearbeitung einer Wiki-Seite eine Möglichkeit zur Referenzierung und Erschließung dieses „Wissens" angeboten werden (=> Vermeidung von Redundanzen).

Nutzergruppen: Alle Mitarbeiter des Unternehmens

Anforderungen:

- Gute Benutzbarkeit und Produktivität für alle Mitarbeiter des Unternehmens - d.h. die Softwarelösung sollte Textverarbeitungsfunktionalität und keine ausschließliche Wiki-Syntax bieten.
- Redundante Informationen (Wucherungen) sollen möglichst vermieden werden => das „Wissen" des Unternehmens muss erschlossen sein.
- Der Offline-Zugriff auf Informationen sollte aufgrund der Vor-Ort-Tätigkeit der Mitarbeiter bei Kundenprojekten („On Site"-Thematik) möglich sein.
- Eine Versionierung von Inhalten sollte möglich sein.

Use Case 4: Informationsbedarf im Unternehmen identifizieren

Kurzbeschreibung Szenario:

Ein Nutzer sucht nach Informationen („Wissensartefakten") in der globalen Wissensbasis – die Suche verläuft erfolglos (beispielsweise ist Treffermenge leer oder irrelevante Fundstellen) – mögliche Lösungsvarianten wären:

- Ein Mitarbeiter mit entsprechenden Spezial-/Detailkenntnissen erfasst das angefragte Wissensartefakt.
- Das angefragte Wissensartefakt befindet sich im privaten Bereich eines Mitarbeiters – dieser stellt das Artefakt zur Verfügung.

Nutzergruppen: Alle Mitarbeiter des Unternehmens

Anforderungen:

Die Wissensplattform muss in der Lage sein nicht hinreichend befriedigten Informationsbedarf zu erkennen, diesen statistisch auszuwerten und den Nutzern der Plattform bei Bedarf mitzuteilen sobald die angefragte Information verfügbar ist - mögliche Benachrichtigungsvarianten wären:

- *Aktiv asynchron*: Beispielsweise Adressierung der recherchierten Information per Mail sobald diese bereitgestellt wurde.
- *Aktiv synchron*: Beispielsweise mittels „Info-Ticker" oder eines dedizierten Fensterbereiches zur Darstellung des Informationsinhaltes.
- *Passiv*: Der Nutzer holt sich die Information selbst bei Bedarf ab – er erhält vom WM-System die Information, dass zwischenzeitlich Ergebnisse („Wissensartefakte") zu seiner früheren Recherche vorliegen (d.h. Treffermenge > 0).

Use Case 5: Sharing von Wissen und Motivation zur Wissensartikulation

Kurzbeschreibung Szenario:

Implizites Wissen wird durch Mitarbeiter expliziert und anderen Personen / Mitarbeitern proaktiv oder bei Bedarf zur Verfügung gestellt - denkbare Varianten wären beispielsweise:

a) Direkte Artikulation des Wissens in der zentralen Wissensbasis (beispielsweise mittels eines oder mehreren Wikis) – dies erfordert dann allerdings permanenten Zugang zur WM-Plattform.

b) Übertragung von „privatem" Wissen (d.h. von Wissensartefakten) aus dem lokalen WM-System in die zentrale Wissensbasis.

Nutzergruppen: Alle Mitarbeiter des Unternehmens

Anforderungen:

Die technische Barriere zur Artikulation von Wissen muss niedrig sein:

- Dies erfordert eine ergonomische und effiziente Artikulation des Wissens
- Falls das nachgefragte Wissensartefakt bereits im lokalen WM-System des Mitarbeiters artikuliert wurde, soll der einfache Transfer des Wissensartefaktes in das globale WM-System oder ein „Sharing" des lokalen Artefaktes möglich sein.

Ergänzende Anforderungen nichttechnischer Art:

- Die Unternehmenskultur sollte das Sharing von Wissen unterstützen und belohnen – d.h. es darf keineswegs ein Klima des Misstrauens oder ein ungesunder Wettbewerb zwischen den Mitarbeitern eines Unternehmens herrschen. Dieser „menschliche" Faktor ist für das Teilen von „Wissen" erfahrungsgemäß weitaus kritischer als die zuvor genannten technischen Barrieren.

3.6 PTV AG

Die PTV AG ist seit 30 Jahren ein führender Anbieter von Dienstleitungen und Produkten rund um den Kontext Verkehrsoptimierung. Mit seinen ca. 500 Mitarbeitern innerhalb der PTV AG und weiteren rund 250 Mitarbeitern in Tochterfirmen ist die PTV ein weltweit agierendes und entsprechend vernetztes Unternehmen.

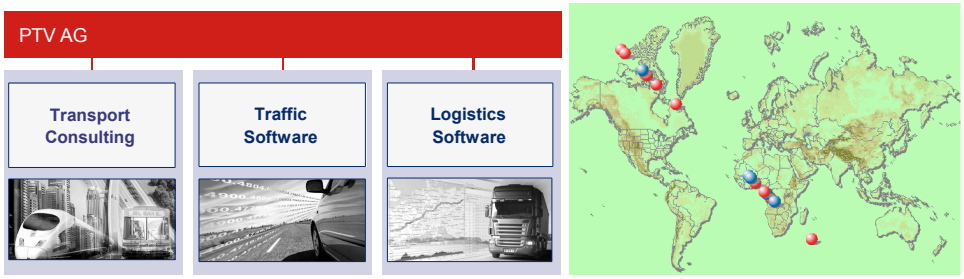


Abbildung 5: PTV AG – Geschäftsfelder und geographische Struktur

Die wichtigsten Kernkompetenzen der PTV AG sind dabei Wissen und Software im Kontext Verkehrsplanung und –steuerung, sowie Wissen und Produkte im Bereich der logistischen Planung. Grundlage hierfür in allen Bereichen ist eine qualitativ hochwertige digitale Geographie und deren Nutzung in Form von Kartenmaterialien und darauf basierenden Softwarekomponenten.

3.6.1 Anforderungen an Wissensverwaltung aus Sicht der PTV

Die Anforderungen der PTV AG an WAVES lassen sich aus drei verschiedenen Sichten definieren:

- Anforderungen aus der technischen Infrastruktur und Systemlandschaft heraus

- Anforderungen aus der Unternehmensstruktur heraus
- Anforderungen aus der Prozessorganisation heraus

Diese Anforderungen werden nachfolgend getrennt nach den Sichten betrachtet.

Anforderungen aus der technischen Infrastruktur und Systemlandschaft heraus

Bedingt durch verschiedene Technologien, unterschiedliche Themenschwerpunkte und heterogene und gewachsene Prozesse findet sich bei der PTV ein relativ weites Spektrum an Systemen, in denen Wissen und Informationen vorliegen. Die nachfolgende Abbildung zeigt einen Ausschnitt aus der Systemlandschaft, der im Kontext WAVES betrachtet werden soll.

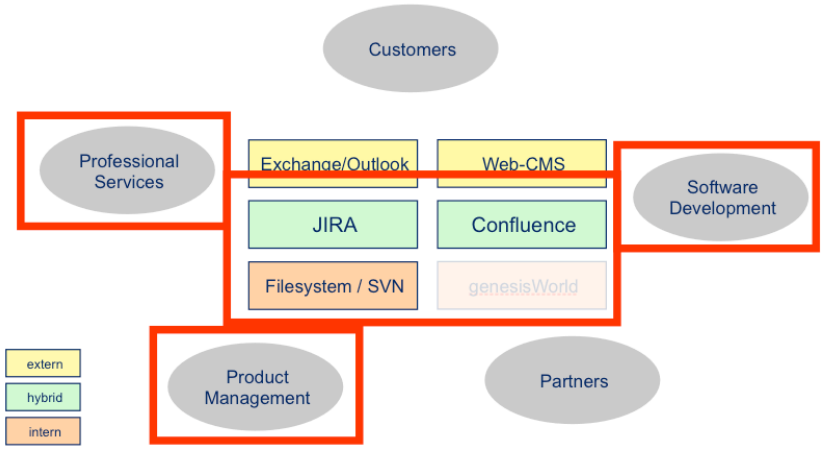


Abbildung 6: Systemlandschaft der PTV (Auszug)

Kommunikation mit Kunden und Partnern erfolgt zumeist mittels e-Mail auf Basis von Outlook und Exchange. Benachrichtigungen und Veröffentlichungen werden via Web-CMS zur Verfügung gestellt. Kommunikation mit Partnern und ausgewählten Key-Kunden erfolgt zudem über JIRA zur Fehlerverfolgung und Confluence (Wiki) für Informationen und Kommentare. Ein Ticketing-System und eine Kundenverwaltung auf Basis des genesisWorld Systems der CAS wird als CRM-System und zur Bearbeitung von Anfragen und Fehlermeldungen genutzt. Schließlich liegen wichtige Spezifikationen und Dokumente, sowie Quellcode im Filesystem und zumeist unter SVN Versionsverwaltung.

Aus dieser Systemlandschaft heraus ergeben sich folgende Anforderungen an eine Wissensverwaltung:

- Wissen liegt verteilt in den verschiedenen Systemen vor – eine Suche muss die verschiedenen Quellen geeignet integrieren.
- Sowohl Fehlermeldungen, als auch Aufgabenlisten, Statusinformationen und Kundenhistorien liegen in verschiedenen Systemen zu unterschiedlichen Zwecken vor. Eine integrierte Sicht – beispielsweise in Form eines Cockpits oder einer personalisierten bzw. thematischen Web-Seite – ist sehr wünschenswert.
- In jedem Fall müssen Zugriffsbeschränkungen und Datenschutzbestimmungen berücksichtigt werden können.

Anforderungen aus der Unternehmensstruktur heraus

Die weltweite Verteilung des Unternehmens in Bezug auf Kunden, Partner und Niederlassungen erfordert auch im Zeitalter elektronischer Kommunikation Mobilität und Reisen. Dadurch ergeben sich zusätzliche Anforderungen:

- Der Zugriff auf Informationen muss nahezu von überall erfolgen können. Dies bezieht sich sowohl auf die einzelnen Systeme und Dokumente als auch auf die integrierten Suchen und Sichten.
- Während längerer Reisen ist es üblich, Dokumente zu sichten, oder zu erstellen, auch wenn keine Online-Verbindung besteht, sei es aus Kostengründen, oder weil prinzipiell keine Verbindungsmöglichkeit besteht. Erforderlich ist damit die Offline-Fähigkeit der Wissensbearbeitung.

Anforderungen aus der Prozessorganisation heraus

Entwicklungsseitig setzt PTV in verstärktem Maß auf agile Methoden wie SCRUM. Diese Methoden erfordern ein hohes Maß an Transparenz für alle Beteiligten. Dies wird bei PTV durch die Ablage der sogenannten Backlogs, d.h. der ansetehenden oder aktuell laufenden Tätgkeiten, im JIRA-System, sowie die Ablage von Konzepten, Review-Informationen, Präsentationen und Produkten im Dateisystem und im Wiki.

Hieraus ergeben sich die folgenden Anforderungen:

- Integration der Systeme mit Fokus auf Priorisierung, übersichtliche Listen und Requirements-Management
- Integration zu Übersichtsseiten, die den einzelnen Rollen gerecht werden.
- Integrierte Suche in den verschiedenen Datenbeständen um schnell einen Überblick über die Aufgaben oder Konzepte zu einem Thema zu bekommen, oder auch um die Wichtigkeit eines Themas durch Abgleich mit Kundenanfragen und -wünschen zu belegen oder zu überprüfen.

Datenmengen

Bei den oben genannten Systemen handelt es sich um lebende Systeme, d.h. sie werden aktiv genutzt und sind ständig im Fluss, sei es durch Erweiterungen, Ergänzungen, oder Restrukturierungen. In der folgenden Tabelle sind einige Kennzahlen zu den einzelnen Systemen Stand Oktober 2008 dargestellt

System	Anzahl Dokumente	Neue Dokumente pro Monat
Confluence (WIKI)	12925	340
JIRA (Requirements und Bugs)	68269	1500
genesisWorld (CRM + Ticketing)	29416	600

Die Beherrschung dieser Datenmengen ist für den Systemnutzer nur mit Hilfe einer geeigneten integrierten Suche möglich. Diese Suche muss Stichworte wie Themenbezüge auffinden lassen und stellt aus Sicht der PTV den zentralen Anwendungsfall dar.

3.7 Zusammenfassung

Trotz der wie oben beschriebenen verschiedenen Ausgangssituation und damit einhergehenden unterschiedlichen Anforderungen der einzelnen Partner an ein Wissensmanagement System, wurde im Rahmen des Projektes ein Konsens über die umzusetzenden Anforderungen erreicht. Die Ausgangsituation der Partner kann durch folgende gemeinsame Herausforderungen zusammenfassend charakterisiert werden:

- Heterogene Systemlandschaft

  Das Wissen innerhalb der Unternehmen liegt in unterschiedlichen Systemen, die zum Großteil nicht mit einander vernetzt sind. Dies erschwert speziell die Nutzung von bereits vorhandenem Wissen (Reuse) und kann somit zur kostenintensiven redundanten Wissensartikulation führen.

- Unterschiedliche Verwendungskulturen/Insellösungen

Innerhalb der Unternehmen werden zum Teil je nach Abteilung, zum Beispiel Entwicklung und Vertrieb, unterschiedliche Systeme zur Wissensartikulation verwendet, so dass die Kommunikation und die systematische Wissensnutzung erschwert werden. Es können somit Insellösungen für einzelne Abteilungen entstehen.

- Systematische Wissenserfassung/ Persönliches Wissensmanagement/Wucherung

Bislang sind keine Systeme zur systematischen Erfassung von Wissen im Einsatz. Das entstandene Wissen wird teilweise in privaten, also nicht firmenweit zugänglichen, Bereichen abgelegt. Bislang ist es nicht möglich dieses persönliche Wissen später in eine organisatorische Wissensbasis zu überführen. Ein weiteres Problem ist das Wachstum der Wissensbasis; ohne systematische Erfassung wird neues Wissen zwar generiert, jedoch kann dies zur Wucherung von einzelnen Systemen wie zum Beispiel Wikis führen und kann somit die Wissensnutzung trotz vorhandenem Wissen nicht unterstützen.

- Wissen bei Schlüsselmitarbeitern/Wissensartikulation

Durch die fehlende systematische Wissenserfassung sammelt sich das Wissen bei einzelnen Mitarbeitern; somit liegt dieses Wissen im Unternehmen nur implizit vor. Die Bereitschaft das vorhandene Wissen zu teilen, setzt einerseits die entsprechende Firmenkultur, aber auch eine effiziente Form der Wissensartikulation, wie zum Beispiel die Überführung von persönlichem Wissen oder eine benutzerfreundliche, effiziente Form des Wissensartikulation voraus.

- Räumliche Verteilung

Die Mitarbeiter in den Unternehmen arbeiten räumlich verteilt an verschiedenen Standorten wie auch beim Kunden vor Ort. Häufig erschwert dies die Kommunikation bzw. der Zugriff (ggf. offline) auf die notwendige Wissensbasis kann nicht gewährleistet werden.

- Starkes Wachstum der Anzahl der Mitarbeiter

Die Anwendungspartner des WAVES Projektes unterliegen einem ständigen Wachstum und somit der Herausforderung neue Mitarbeiter einzuarbeiten. Die Einarbeitung kann nur effizient gewährleistet werden, wenn das benötigte Wissen den neuen Mitarbeitern möglichst einfach und schnell zur Verfügung gestellt werden kann. Das bedeutet, dass die Mitarbeiter sich schnell einen Überblick über die für sie relevanten, bereits vorhandenen Informationen verschaffen können sollten.

Überblick: Herausforderungen und Anforderungen

Aus den oben beschriebenen Herausforderungen ergeben sich für die einzelnen Anwendungspartner spezielle Anforderungen (siehe die vorigen vier Abschnitte). Im Folgenden werden die Anforderungen an WAVES Partner unabhängig kurz zusammengefasst:

- **Vernetzung von Wissen**

  Bisheriges Problem:

  Bislang kann das Wissen, welches in der heterogenen Systemlandschaft liegt nicht Toolübergreifend miteinander verknüpft werden. Dies ist speziell bei der Artikulation von neuem Wissen ein Problem, da einzelne Informationen redundant in verschiedene Systeme eingepflegt werden müssen, zum Beispiel bei der Erfassung eines Meetingprotokolls. Des Weiteren stehen bei der Wissensnutzung keine Verknüpfungen der gefundenen Artefakte zur Verfügung

  Anforderung:

  Bei der Wissensartikulation eine Referenz auf andere bereits bestehende Objekte setzen zu können, so wie eine halbautomatische Vorschlag eines einzufügenden Objekts während der Eingabe. Bei der Wissensnutzung können explizite Abfragen formuliert werden, welche alle Textstücke, oder Ausschnitte daraus anzeigen, welche auf ein bestimmtes Objekt verweisen.

- **Integrierte Suche**

  Bisheriges Problem:

  Es muss bislang in allen Informationsquellen einzeln gesucht werden, es gibt keine Möglichkeit zentral in allen relevanten Tools zu suchen. Wird in einer Informationsquelle etwas gefunden, so besteht bislang keine Verknüpfung zu den anderen Quellen.

  Anforderungen:

  Um eine effiziente Wissensnutzung zu erreichen, soll eine Integrierte Suche die Möglichkeit bieten über alle relevante Informationsquellen und Informationsartefakte des jeweiligen Unternehmens zu suchen. Die Integrierte Suche soll hierfür eine zentrale Such-Startseite zur Verfügung stellen, so dass der Benutzer nicht wissen muss, in welcher Informationsquelle die gesuchte Information zu finden ist. Durchsucht werden sollen je nach Unternehmen unterschiedliche Quellsysteme, generell sollen aber sowohl Office-Dokumente, E-Mail, Wikis, wie auch Software spezifische Systeme (Bugtracker, Quellcode) indiziert und durchsucht werden

können. Die Indizierung der Daten sollte inkrementell einmal täglich durchgeführt werden, um die Aktualität der durchsuchten Daten zu gewährleisten. Die Bedienbarkeit der Suchmaske soll einfach und leicht überschaubar sein, dennoch soll sie die Möglichkeit bieten, den Suchraum auf bestimmte Informationsquellen einzuschränken. Des Weiteren soll die Integrierte Suche adäquat in die Technologie-Landschaft und somit die Prozesse der Benutzer durch eine Integration in die Entwicklungsumgebung eingebetet werden bzw. über ein Webinterface zur Verfügung gestellt werden.

- **Projektsicht/Cockpit**

Bisherige Probleme:

Es gibt bislang keinen zentralen Einstiegspunkt in ein Projekt von dem aus alle aktuellen Informationen über ein Projekt angezeigt werden und von dort aus weitere Informationen zu dem Projekt angesteuert werden können.

Anforderungen:

Jeder Benutzer soll eine individuellen Einstiegsseiten zu einem Projekt generieren und speichern können (Projektsicht). Dabei darf ein Benutzer mehrere Sichten (eventuell für jede Rolle, die er im Softwareentwicklungsprozess annehmen kann) generieren. Der Benutzer stellt sich die für ihn relevanten Informationen für ein Projekt zusammen, dabei darf der Benutzer nur aus Informationsquellen bzw. Informationen aus den einzelnen Quellen wählen, für die entsprechenden Rechte besitzt. Wenn der Benutzer sich mit seiner Rolle einloggt und zu dem Projekt navigiert, wird immer seine Projektsicht angezeigt.

- **Offline Zugriff/ Räumlich verteilte Wissensnutzung**

Bisherige Probleme:

Die Mitarbeiter der Anwendungspartner arbeiten zum Teil räumlich verteilt bzw. beim Kunden vor Ort. Die Wissensbasis ist durch das räumlich verteilte Arbeiten häufig weder für die Wissensnutzung noch –artikulation verfügbar. Oft hat der entsprechende Mitarbeiter auch keinen Online Zugriff und benötigt eine Offline Unterstützung zum Beispiel für die Verwendung des Wikis.

Anforderungen:

Offline - Der Mitarbeiter kann festgelegte Inhalte des Wikis online lesen, verändern. Die offline Änderungen werden später korrekt in das Wiki übertragen.

Raümliche Verteilung - Effiziente Nutzung im Intra- / Internet unter Berücksichtigung der technischen Barrieren muss soll gewährleistet werden. Ein entsprechendes Rollen- und Rechtesystem muss entwickelt werden.

- **Wissensartikulation**

Bisherige Probleme:

Im Rahmen des Projektes wurde gemeinsam mit allen Partnern das Wiki als geeignete Form zur kooperativen Wissensartikulation festgelegt. Allerdings bieten bisherigen Wikis, keine Möglichkeit einfach und schnell unterschiedliche Informationsquellen zu verlinken. Des Weiteren ist eine wichtige Vorraussetzung für die Artikulation von Wissen, dass Wissen effizient und einfach erfasst werden kann. Dies setzt eine leichte und den Benutzer unterstützende Bedienbarkeit voraus.

Anforderung:

Das Wiki muss eine gute Usability (Textverarbeitungsfunktionalität und nicht ausschließlich Wiki-Syntax) bieten, um die Produktivität für alle Mitarbeiter des Unternehmens bei der Artikulation von Wissen zu gewährleisten. Des Weiteren sollen redundante Einträge durch Vernetzung mit den

anderen bestehenden Informationsquellen und Link Vervollständigung über diese Quellen vermieden werden. Besteht bereits Wissen zu einem bestimmten Thema, so soll der Benutzer die Information darüber bereits während der Artikulation erhalten und die Möglichkeit zur Referenzierung soll angeboten werden. Zudem soll das Wiki ebenfalls Offline-Funktionalität besitzen.

- **Identifikation von Wissensbedarf**

Bisherige Probleme:

Ein Benutzer sucht nach einer bestimmten Information, die Suche verläuft erfolglos. Bislang gibt es keine Möglichkeit zu erfassen, ob auch andere Benutzer diesen Suchbegriff verwendet haben und somit ggf. eine Wissenslücke im Unternehmen zu identifizieren.

Anforderungen:

Das Wissensmanagement System soll Suchanfragen statistisch erfassen und das Nutzerverhalten auswerten. Zusätzlich soll es den Benutzer direkt zur Wissensartikulation auffordern, wenn eine Suchanfrage erfolglos war. Sobald Wissen zu dem gesuchten Begriff angelegt wurde, sollen die Benutzer, die diesen Begriff erfolglos gesucht haben, informiert werden (Reportfunktion).

- **Persönliches Wissensmanagement**

Bisherige Probleme:

Häufig erfassen die Mitarbeiter ihr Wissen unstrukturiert und erstmal nur in ihrem privaten Bereich. Bislang ist es nicht möglich, dass bereits erfasste Wissen leicht in eine allgemein zugängliche Wissensbasis zu überführen. Stattdessen muss das Wissen erneut artikuliert werden, dies erzeugt eine ökonomische Barriere, die dazu führen kann, dass Wissen im Unternehmen nicht geteilt wird.

Anforderungen:

WAVES soll die systematische und strukturierte Artikulation von persönlichem Wissen unterstützen. Zusätzlich soll die persönliche Wissensbasis mit den organisatorischen Wissenssystemen verknüpft sein und die Möglichkeit bieten, einfach und schnell ausgewählte Bereiche des persönlichen Wissens in eine unternehmensweite Wissensbasis zu überführen.

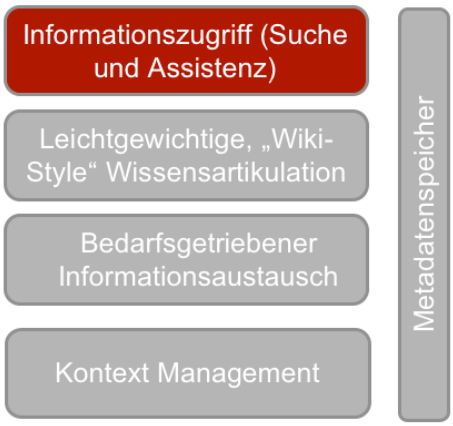
4. Projektergebnisse

Die Entwicklung komplexer Produkte und Dienstleistungen ist angewiesen auf hochgradig spezialisierte, arbeitsteilig organisierte Entwicklungsschritte. Einzelne Entwickler können den detaillierten Gesamtaufbau eines solchen Systems nur noch rudimentär nachvollziehen. Um ihre eigene Arbeit effizient zu erledigen, benötigen sie Kenntnisse der angrenzenden Schnittstellen.

Diese Beschreibung trifft insbesondere auf die Entwicklung von Softwaresystemen zu, da durch die relativ leichte Wiederverwendung der Schnittstellen niedrigerer Systemschichten, Funktionsbibliotheken und nicht zuletzt beliebig verfügbarer Dienste eine extrem hohe Systemkomplexität entsteht. Sie erfordert von Entwicklern eine kontinuierliche Wissensaufnahme. Andersherum müssen sie aber auch entsprechendes Wissen über die von ihnen verantworteten Komponenten bereitstellen.

Während die im Abschnitt 4.1.1 vorgestellte **Waves Integrierte Suche** den Informationszugriff und erleichtert, ermöglichen die Werkzeuge **HKW** (Abschnitt 4.2.2) und **Wiquila** (Abschnitt 4.2.1) eine benutzbare und effiziente Wissensartikulation. Das Werkzeug **KISSy** (Abschnitt 4.1.2) fokussiert auf ein Spezialgebiet des Informationszugriffs und verbindet dabei erstmalig in der Literatur die Suche nach Quelltext mit der Betrachtung von Code-Qualitätsaspekten. Das Ziel vom **Kontextmanagement** (Abschnitt 4.4) besteht darin, dem Nutzer der WAVES-Wissensmanagement-Lösung jeweils seiner Rolle und seinem Arbeitskontext gerechte Inhalte darzustellen und dadurch den Prozess der Wissensnutzung und Wissensartikulation zu beschleunigen. Allerdings adressieren diese Werkzeuge nicht, *welches* Wissen artikuliert werden sollte und *wie* es mit Anderen ausgetauscht werden kann. Anders ausgedrückt - ein effizienterer Wissensaustausch sollte sich am Bedarf einer Organisation orientieren und dabei möglichst zielgerichtet erfolgen. Zu diesem Zweck werden die beiden Werkzeuge **Woogle** (Abschnitt 4.3.1) und **Inverse Suche** (Abschnitt 4.3.2) vorgestellt, die beide auf dem Paradigma des "bedarfsgetriebenen Wissensaustauschs" basieren.

4.1 Informationszugriff: Suche und Assistenz (Arbeitspaket 8)

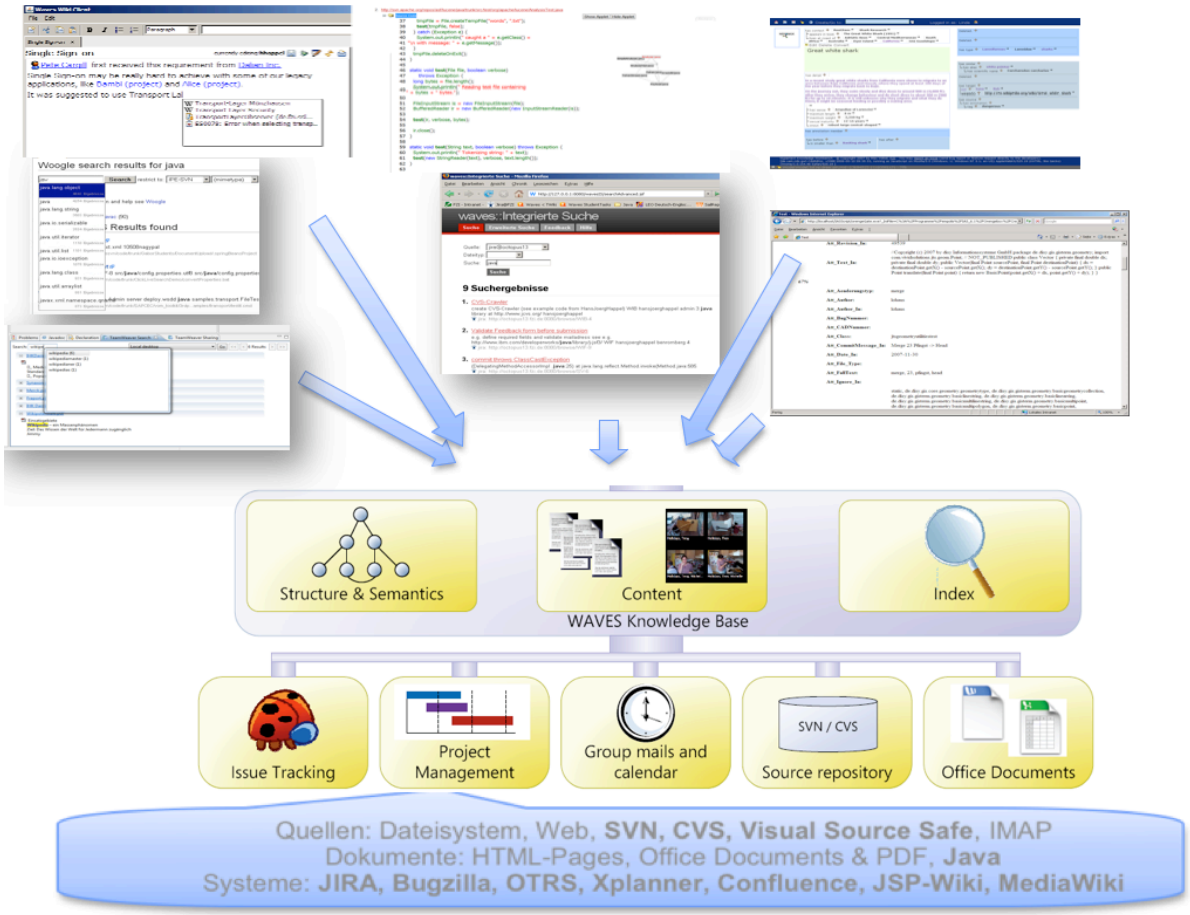


4.1.1 WavesIS: Integrierte Suche von Wissensartefakten

Wie in den Anforderungsszenarien beschrieben, bildet die heterogene Werkzeuglandschaft und die damit verbundene Verteilung von Informationen ein wesentliches Problem für den Informationszugriff. An dieser Stelle setzt die WAVES Integrierte Suche (WavesIS) an, indem sie eine unternehmensweite Suche mit Schwerpunkt auf Software Engineering-spezifische Datenquellen legt.

WavesIS bildet das Kernstück der WAVES Architektur, auf dem viele der weiteren Werkzeuge aufsetzen bzw. komplementäre Funktionalität bereitstellen. Um eine einfache Integration zu ermöglichen, wurde WaveIS als eine dienstorientierte Architektur entworfen.

WavesIS besteht folglich aus einem Backend, welches Dienste für die Indizierung und Such nach Informationen bereitstellt. Einzelne Teilkomponenten sind als Dienste nach dem OSGi-Standard realisiert und können in unterschiedlichen Server-Umgebungen wie einer Eclipse-Instanz oder einem Java Web Application Server ausgeführt werden. Als Benutzerschnitstelle („Frontend“) dient eine Web-basierte Oberfläche, die sich weitgehend an der Gestaltung gängiger Suchmaschinen orientiert. Darüber hinaus existiert eine Eclipse-Erweiterung, mit der sich WavesIS Ergebnisse direkt in einer Eclipse Entwicklungsumgebung oder einer Eclipse RCP-Anwendung anzeigen lassen können. Schließlich kann auch aus einer Woogle-Installation (vgl. Abschnitt 4.3.1) auf ein WavesIS Backend zugegriffen werden.



Backend

Wie bereits angedeutet lag der Fokus bei der Entwicklung von WavesIS auf der *unterstützung heterogener Artefakttypen* mit besonderem Schwerpunkt auf die Domäne Software Engineering. Durch die Wiederverwendung der Open Source Datenextraktions-Komponente „Aperture“ können Extraktoren für Standardformate wie Microsoft-Office-Dateien (.doc, .xls, .ppt), PDF, Outlook und Webseiten einfach genutzt warden.

Darauf aufbauend wurden im Rahmen von WAVES durch das FZI weitere Importer entwickelt, mit denen eine Reihe weiterer Artefakttypen für die integrierte Suche indiziert werden können. Dazu gehören unter anderem:

- Versionverwaltung: SVN, SourceSafe, CVS Changelogs
- (Bug)Tracker-Systeme: Jira, Bugzilla, OTRS (Open Ticket Request System)
- Wikis: JSPWiki, Confluence, MediaWiki
- Xplanner (Projektplanungs-Software), Cobra Adressmanager

- SCRUM-Artefakte

Die Indexierung der Artefakte verläuft inkrementell – d.h. es werden nur die Wissensartefakte indiziert werden, die seit der letzten Indizierung verändert oder neu erstellt wurden. Im Laufe des Indizierungsprozesses extrahiert WavesIS aus den beschriebenen Quellen Volltext-Informationen sowie strukturierte Metadaten. Dies ermöglicht unter anderen eine *Ähnlichkeitsbasierte Suche*. Falls der Nutzer der *WAVES integrierten Suche* nicht genau weiß, wie das zu suchende Wissensartefakt heißt, würde eine Suche nach exakten Matchings keinen Treffer ergeben. Durch die ähnlichkeitsbasierte Suche werden basierend auf Ähnlichkeitsmassen zu dem gesuchten Suchmuster ähnliche Vorschläge angeboten werden.

Weiterhin verarbeitet WavesIS *Implizites Nutzer-Feedback zur Verbesserung der Suchvorschläge.* Abhängig davon, wie präzise ein Suchbegriff eingegeben wurde bzw. wie viele und wie große Datenquellen von der WAVES integrierte Suchen indiziert wurden, kann die Anzahl der Treffer zwischen einigen dutzend und mehreren Tausend variieren. Bei 10-20 Treffern kann der Nutzer in der Regel schnell eine Entscheidung treffen, ob sich das gewünschte Ergebnis in der Trefferliste befindet. Bei mehreren Tausend Treffern verliert aber der Nutzer den Überblick sehr schnell. Um die Suche zu erleichtern wurde eine Heuristik in die *WAVES integrierte Suche* eingeführt. Die Heuristik basiert auf der Beobachtung, dass ein von einem Nutzer abgerufener Treffer evtl. auch für andere Nutzer interessant ist. Diese Idee ist dann sehr viel versprechend, wenn die Nutzer einen ähnlichen Informationsbedarf haben, das beispielsweise bei der Zusammenarbeit am gleichen Projekt de facto gegeben ist. Technisch ist diese Heuristik in der *WAVES integrierten Suche* so realisiert, dass die von den Nutzern aufgerufenen Treffer intern markiert werden. Bei einer Suche zu einem späteren Zeitpunkt werden diejenigen Treffer hervorgehoben, die von anderen Nutzern bereits aufgerufen wurden.

Frontend

Wie bereits erwähnt kann WavesIS mit einer Reihe unterschiedlicher Benutzerschnittstellen kombiniert werden. Neben dem standardmäßigen Browser-Frontend existiert ein Eclipse-Plugin für die Konfiguration und Suche aus Eclipse heraus sowie Erweiterungen für die Wiki-Systeme MediaWiki und Confluence („Woogle“). Darüber hinaus ist eine Integration mit Polarion Enterprise realisiert, durch die Daten aus Polarion in Echtzeit indiziert und über die Polarion-Suchmaske durchsucht werden können.

Die Suchschnittstellen bieten neben einem klassischen Eingabefeld für die Stichwortsuche noch erweiterte Auswahlmöglichkeiten an. Die *Automatische Vervollständigung der Suchmuster* führt im Hintergrund automatisch Anfragen auf der Wissensdatenbank durch und bietet während des Tippens automatisch Vorschläge an, während der Nutzer der *WAVES integrierten Suche* eine Suchanfrage formuliert. Erfahrungen zeigen, dass durch diesen Mechanismus der Nutzer schneller die gesuchten Informationen findet.

Weiterhin können Nutzer Suchergebnisse nach eine Reihe von Kriterien wie Erstellungsdatum, Datentyp oder Quellsystem filtern. Diese Filterung kann vor oder während einer Suche geschehen und ermöglicht es die Suche gezielt einzuschränken. Beide Möglichkeiten sind in den folgenden Bildschirmausschnitten dargestellt.


waves::Integrierte Suche - Wikis i...
WAVES::INTEGRIERTE SUCHE - INTERNAL DEVELOPMENT VERSION
Suche
Erweiterte Suche
Feedback
Hilfe
Wikis im Unternehmen
Suche
?
86 Suchergebnisse
Seite 1 von 9
1. Wiki-Refactoring.ppt
Alpha Anzahl der Wikiseiten Anzahl der Wikinutzer[2] Einsatz von Wikisim Unternehmen Größe des ♦ im Zuge des Wachstums eines Wikis zu messbaren Problemen führen. Probleme können im erschwerten ♦) Manuell Nachfolgend Präventiv Wucherung von Wikis:Befragte Unternehmen Groß (500-1000) Klein (<50
<365d filesystem: file:/U:/IPE/Projekte/Waves/Meetings/20071031_Pro.../Wiki-Refactoring.ppt
qualität wiki refactoring
Refactoring Wikis im Unternehmen wucherung
2. 03_Anforderungsanalyse_CAS.ppt
deswegen an ein Wissensmanagement im Entwicklungsprozess? CAS – ein innovatives Unternehmen: TOP 100
<365d filesystem: file:/U:/IPE/Projekte/Waves/Meetings/20071210_Rev...erungsanalyse_CAS.ppt
no clicks
3. 11_Wissensartikulation_KnowledgeDesktopWikiclient_FZI.ppt
? Integrationsmöglichkeiten Integration ins Kernsystem Aktueller Stand und Planung Rolle von Wikis im Software ♦) Herausforderungen für Wikis im SE Benutzbarkeit Produktivität Qualitätsprobleme Wucherung Veraltete, fehlende ♦ Rolle aus unserer Sicht Wikis im Software-Engineering spielen können. Anschließend stelle ich unseren
<365d filesystem: file:/U:/IPE/Projekte/Waves/Meetings/20071210_Rev...topWikiclient_FZI.ppt
no clicks
Ähnliche Suchbegriffe:
im
wikis
unternehmen
unterstützen
integrierten
unternehmens
integrieren
Suche einschränken:
Quellen:
files (86)
Dateitypen:
Microsoft Word (55)
Adobe Acrobat PDF (19)
Microsoft Powerpoint (11)
application/vnd.ms-office (1)


WAVES::INTEGRIERTE SUCHE
Suche
Erweiterte Suche
Feedback
Hilfe
Quellen:
web (1514)
Spring DM OSGi Docs (1491)
Spring DM OSGi Newsgroup (19)
Spring DM OSGi Web (4)
Dateitypen:
text/html (1503)
text/java (471)
text/jira (410)
text/xml (283)
Zeitraum:
Von:
Bis:
Juli 2008
Heute ist der 29.07.2008
Ähnliche Suchbegriffe:
tests
Dateigroesse:
Byte
Seitenanzahl:
Sortieren nach:
Relevanz
absteigend
test
Suche
?
538 Suchergebnisse
1. Integration test of BundleContextAware support
- [ ] 2.3 BundleContextAware - [ ] test that beans that are BundleContextAware are injected
<365d jira: http://jira.springframework.org/browse/OSGI-25
test
2. externalize test mandatory libraries
The mandatory libraries used by the test suite have to be externalize since a minor version change ♦ requires code updates and recompilation of the test class. OSGI costin costin 4 libraries
<365d jira: http://jira.springframework.org/browse/OSGI-209
no clicks
3. rework OSGi test support
and caching 3. add autowire on the test 4. move waitOnContext synchronization into a single class 5 ♦ . rework the asynch behavior so the user doesn't have to instruct the test to wait for them (make it part ♦ () to run code when the test is bootstrapped inside OSGi and not outside of it. costin
<365d jira: http://jira.springframework.org/browse/OSGI-62
no clicks

4.1.2 KISSy: Strukturierte Suche in Quelltext-Repositories

Im Bereich des Software Engineerings tritt im Verlauf der verschiedenen Entwicklungsphasen einer Anwendung oftmals die Frage auf, ob ein auftretendes Problem bereits andernorts gelöst ist. Aus Sicht des Entwicklers sind dabei mehrere verwandte Fragestellungen von Interesse:

1. Existiert eine Komponente die dieses Problem adressiert?
2. Existiert eine Komponente die dieses Problem teilweise adressiert?
3. Kann eine Lösung oder eine Teillösung zu Bewältigung dieses Problems herangezogen werden?
4. Kann eine vorhandene Lösung ohne Veränderung / mit geringen Veränderungen übernommen werden?
5. Wie kann eine bestehende Komponente übernommen werden?
6. Wie ist die Lösung inhaltlich aufgebaut? (Algorithmus)
7. Falls eine Lösung oder Teillösung existiert, ist es tatsächlich ratsam diese zu übernehmen?

In der Praxis hat sich gezeigt, dass die Entwickler auf diese Fragen oftmals durch entsprechende Suchanfragen an Suchmaschinen verschiedenster Ausprägung reagieren (Singer, J., Lethbridge, T., Vinson, N., Anquetil, N. (1997)). Aus diesem Grund hat sich die Suche nach Information zu einer zentralen Unterstützungsaktivität entwickelt, Studien zu Folge verbringen Beschäftigte im IT-Umfeld stellenweise bis zu 40% Ihrer täglichen Arbeit mit der Suche und dem Zugriff auf Informationen verschiedenster Arten (Henninger, S. (1997), Murphy, G.C., Kersten, M., Findlater, L. (2006)).

Aus Entwicklersicht sind daher besonders Suchmaschinen von Relevanz, die die Suche nach bereits existierenden Code-Fragmenten oder Komponenten unterstützen (siehe Abbildung 7). Bekannte Vertreter solcher Suchmaschinen sind Google CodeSearch, Merobase, Koders oder Krugle. Wenn sich diese Suchmaschinen im Detail doch voneinander abgrenzen, so weisen sie als entscheidende Gemeinsamkeit die Suchmöglichkeit nach Quelltextfragmenten in quelltextoffenen Projekten auf. Diese Suchdienste arbeiten dabei üblicherweise rein textbasiert, vernachlässigen allerdings die Struktur der Software selbst. In Konsequenz ist die Suche dadurch zwar äußerst performant birgt aber zahlreiche andere Nachteile in sich. Besonders auffällig sind dabei das Auftreten stark verrauschter (und daher wenig aussagekräftiger) Ergebnisse, schwieriges bzw. ungenaues Ranking der Treffer und eine fehlende interaktive Navigation der in der Ergebnisliste (Henninger, S. (1997)).

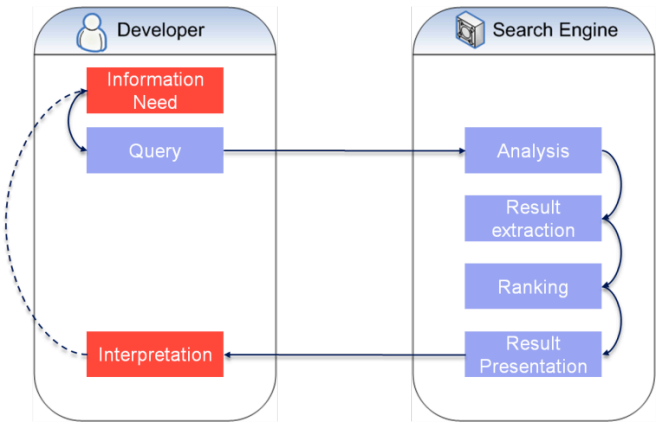


Abbildung 7: Suchanfrage aus Entwicklersicht

Die bislang vorhandenen Einschränkungen der am Markt befindlichen Suchmaschinen führen daher dazu, dass die Suche nach Softwareartefakten durch ein erneutes Überdenken der gesuchten Information und eine mehrfache Reformulierung der Suchanfragen (in Abbildung 7 als gestrichelte Verbindung dargestellt) geprägt wird, die oftmals nicht die gewünschte Information als Teilmenge der Ergebnisse zurückgeliefert wird. Letztendlich führt dies über einen entsprechenden Zeitverlust bis hin zur Nichterreichung der Entwicklerziele.

Unser Ansatz **K**nowledge-based **I**nverstigation of **S**oftware **Sy**stems (kurz: KISSy) stellt eine Methodik dar, die die Softwarequellen nicht nur textbasiert indiziert, sondern auch einer analytischen Betrachtung aus software-technischem Blickwinkel unterzieht. Die Grundidee von KISSy besteht darin, die Struktur (also die vorhandenen Zusammenhänge zwischen den einzelnen Entitäten) der indizierten Software in die Suche einzubeziehen und darauf aufbauend ein qualitatives Ranking der Suchergebnisse zu ermöglichen. Das qualtitative Ranking der Suchergebnisse ist so zu verstehen, dass die Ergebnisse hinsichtlich ihrer Software-Qualität bewertet werden und dementsprechend in einer Reihenfolge geordnet werden. KISSy setzt zur qualitativen Bewertung der Software auf die Ergebnisse des Projekts QBench auf. QBench wurde innerhalb der ersten Auswahlrunde Software Engineering 2006 durch das BMBF gefördert.

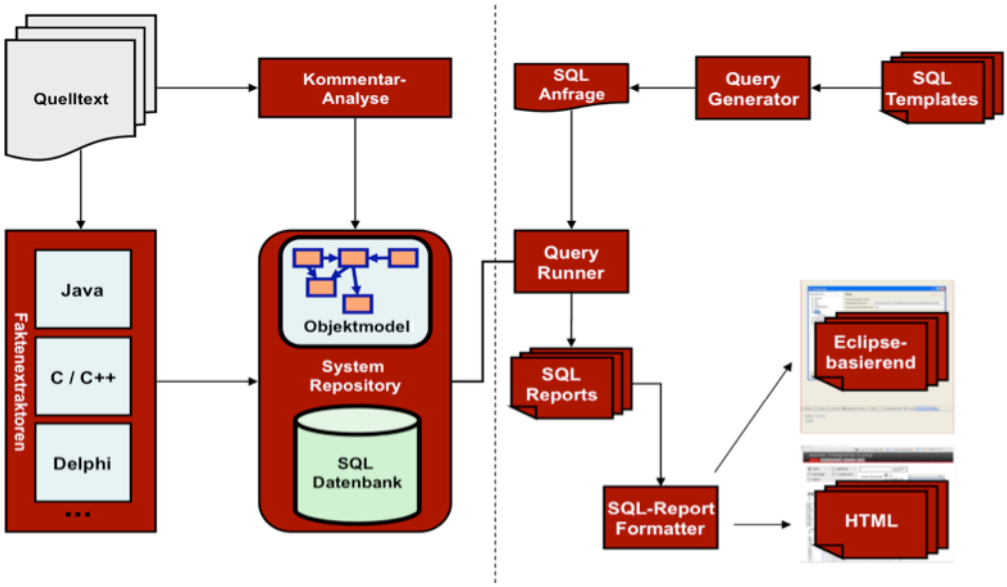


Abbildung 8: KISSy-Systemarchitektur

In Abbildung 8: KISSy-Systemarchitektur wird die Systemarchitektur von KISSy verdeutlicht. Auf der linken Seite ist dabei die Analysekomponente von KISSy, auf der rechten Seite die Anfrage- und Ergebnisaufbereitungskomponenten zu sehen. Wie bereits erwähnt knüpft KISSy nahtlos an die in QBench, mit der Entwicklung des Werkzeuges SISSy, erzielten Ergebnisse an. Wie im Schaubild gezeigt werden Analysekomponenten von SISSy eingesetzt um die Qualität der Software zu bewerten, diese Analyseergebnisse werden in einem Objektmodell und persistent in einer relationalen Datenbank abgelegt (linke Abbildungshälfte). Diese Ergebnisse werden dann im Falle vorliegender Suchanfragen genutzt um ein entsprechendes Ranking der Anfragen zu bilden und dieses darzustellen. Die Darstellung erfolgt derzeit wahlweise webbasiert (ähnlich wie im Falle herkömmlicher Code-Suchmaschinen) oder über eine in eine Entwicklungsumgebung integrierte Sicht.

In mehreren Iterationen und unter Berücksichtigung der Benutzerrückmeldungen durch die Anwendungspartener wurden die folgenden Funktionalitäten in KISSy verwirklicht:

Ranking der Treffer nach Software-Qualität

Eine typische Suchanfrage liefert in der Regel eine Vielzahl an Treffern von Softwareartefakten, man bedenke zum Vergleich beispielsweise an eine generische Suchanfrage, die an Google gestellt wird und zum Schluss mehrere hunderttausend Treffer liefert. Unter der Annahme, dass genügend Softwarequellen indiziert wurden, so liefert eine Code-/Softwaresuche – man denke beispielsweise an die Untersuchung aller quelloffenen und über das Internet zugänglichen Softwareprojekte – eine durchaus vergleichbar lange Trefferliste. Bisherige Rankingverfahren sind im Wesentlichen textbasiert und stützen sich daher lediglich auf das Auftreten der angefragten Stichworte, in Folge ergeben sich die bereits weiter oben angesprochenen Nachteile. Unter zu Hilfenahme geeigneter Ranking-Verfahren, die auf eine software-technische Sichtweise gestützt werden, können die für die Benutzer-

Anfrage tatsächlich relevanten Ergebnisse ausgewählt und bevorzugt gelistet werden. In der Literatur sind bereits einige Ranking-Algorithmen dokumentiert, KISSy verfolgt jedoch hier einen neuartigen Ansatz, in dem das Ranking auf den qualitativen Eigenschaften basiert, d.h. es werden qualitativ besser bewertete Codefragmente in der Ergebnisanzeige bevorzugt. Die qualitative Bewertung der Codefragmente basiert im Falle von KISSy auf der automatischen Ermittlung von so genannten schlechten Entwurfsmustern (Anti-Patterns). Aus der Anzahl der gefundenen Problemmuster in den Codefragmenten lässt sich auf dessen Qualität schließen und kann damit der Ermittlung einer Rangliste der Artefakte dienen.

Konfiguration der Qualitätsanalyse

Um KISSy zur Analyse vorzubereiten, ist es notwendig zahlreiche Konfigurationsdateien zu bearbeiten (z.B. Einstellung der zu indizierenden Quell-Repositories). Um den Konfigurationsaufwand zu reduzieren, wurde für KISSy eine Konfigurationsoberfläche entwickelt. Über diese Oberfläche lassen sich die zu indizierenden Repositories, die Datenbankanbindung für den Index, und das qualitative Ranking konfigurieren.

Abbildung 9: KISSy Datenbank-Konfiguration

Da KISSy mit einer relationalen Datenbank arbeitet ist zur persistenten Speicherung der Analyseergebnisse die entsprechende Einstellung der Verbindung notwendig (dies ist in Abbildung 9) dargestellt. Der angegebene URL muss dabei nicht komplett, entsprechend der Syntax der jeweiligen Datenbank eingetippt werden, sondern wird in der Konfiguration unterstützt.

Der Zusammenhang zwischen den Problemmustern und dem Ranking lässt sich mit dem Konfigurator ebenso anpassen, da verschiedene Benutzer unter Umständen unterschiedliche Problemmuster anders bewerten (siehe Abbildung 10). Wie der Abbildung zu entnehmen ist, können die Problemmuster mit unterschiedlichen Gewichtungen versehen werden und gehen dementsprechend in die Rangfolgenermittlung ein. Da das Bewertungsmodell in einer Datenbank abgelegt ist werden die Problemmuster durch Anfragen an die Datenbank ermittelt, auch diese Anfragen können verändert werden. Letzteres sollte allerdings nur durch erfahrene Benutzer erfolgen, die mit dem Objektmodell bzw. relationalen Schema gut vertraut sind.

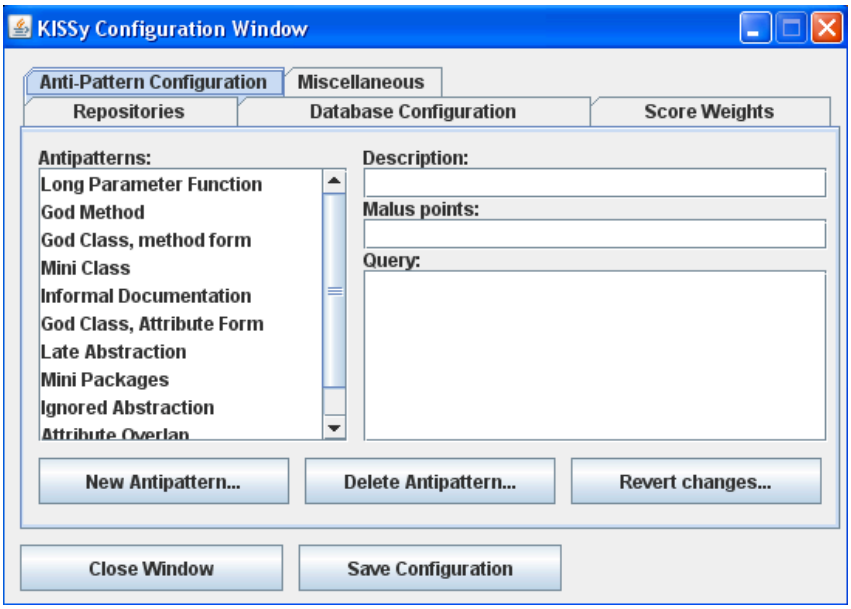


Abbildung 10: KISSy Konfiguration

Fuzzy-Suche

Wie es auch im Bereich herkömmlicher Suchmaschinen bekannt ist, fällt es den Nutzern stellenweise schwer sein Informationsbedürfnis in einer exakten Anfrage zu formulieren, die die gewünschte Information ermittelt (hierdurch entstehen mitunter mehrfache Reformulierungen der Suchanfragen). Mit Hilfe einer unscharfen Suche (oder auch Fuzzy-Suche) werden basierend auf Ähnlichkeitsmaßen nicht nur, die zur Anfrage exakt passenden Ergebnisse geliefert, sondern auch ähnliche Suchergebnisse, die zum Beispiel unter Formulierungen verwandter Anfragen ermittelt werden würden. In der aktuellen Version nutzt KISSy diese Methode um die Formulierung der Suchanfragen zu erleichtern, darüberhinaus werden andere Suchanfragen vorgeschlagen, die der Nutzer ebenfalls gemeint haben könnte. Dieses Vorschlagsverfahren ist eine aus dem Suchmaschinenbereich gängige Methode, durch das Anklicken der Vorschläge erhält der Nutzer automatisch die passenden Ergebnisse gelistet.

Indizierung mehrerer Software-Projekte

Mit Abschluss des vorigen Berichtszeitraums konnte KISSy nur ein Projekt in einem Quelltext-Repository gleichzeitig indizieren. In der Praxis ist eine solche Lösung nicht geeignet, da in der Regel in einem Repository eine Vielzahl von Projekten abgelegt wird, die sogar häufig in gemischten Programmiersprachen implementiert sind. Im Berichtszeitraum wurde KISSy erweitert werden und ermöglicht eine gleichzeitige Indizierung mehrere Software-Projekte in einem oder mehrere Repositories. KISSy untersucht Software-Projekte, die über Subversion, CVS oder Visual-Source-Safe erreichbar sind; die Anbindung weiterer Verwaltungsdienste ist in Zukunft leicht möglich.

Integration weiterer Suchdienste

Im aktuellen Berichtszeitraum konnte KISSy außerdem auch um eine Schnittstelle zu weiteren Suchdiensten ergänzt werden. Hierzu wurde eine zusätzliche Abstraktionsschicht zwischen Benutzerschnittstelle und Suchdienst eingezogen. Benutzer können daher die KISSy-Ergebnisse durch einbinden weiterer Suchmaschinen, wie Google's Codesearch oder Koders, ergänzt werden. Noch zu lösen ist hier allerdings eine genaue Abstimmung der gelieferten Ergebnisse der einzelnen Dienste, diese werden aktuell in verschiedenen Ergebnisslisten angezeigt.

Eclipse-Integration

Seit einiger Zeit hat sich Eclipse als eine der am häufigsten benutzten Entwicklungsumgebungen etabliert, dies gilt inzwischen nicht nur im Java-Bereich, sondern auch für andere Programmiersprachen. Aus diesem Grund wurde für KISSy auch ein Plugin entwickelt das in Eclipse

genutzt werden kann. In Abbildung 11 ist die Anzeige von Suchergebnissen der eclipse-basierten Oberfläche von *KISSy* dargestellt.

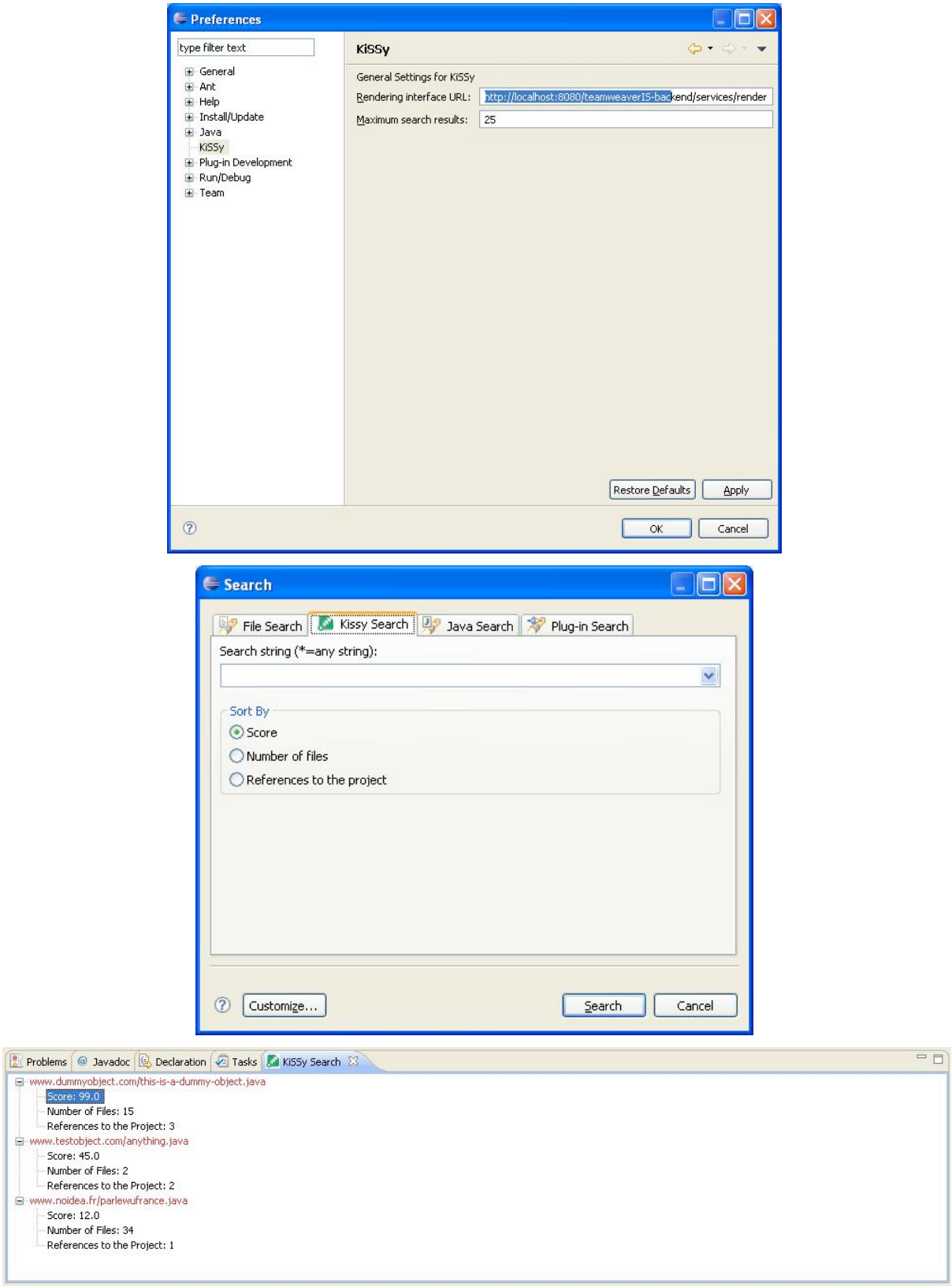


Abbildung 11: Eclipse basierte Oberfläche von KISSy

Wie in Abbildung 11: Eclipse basierte Oberfläche von KISSy zu sehen ist, wird zur Suche mit Kissy die eclipse-eigene Suche erweitert, sodass die Kissy-Suche direkt über den Eclipse-Shortcut

aufgerufen werden kann. Die dargestellten Ergebnisse können dann wie zu sehen ist zunächst über den integrierten KISSy-View durch den Entwickler inspiziert und bei Bedarf direkt, durch Doppelklick auf das jeweilige Ergebnis, in Eclipse geöffnet werden – da es sich unter Umständen um entfernte Resourcen handelt werden diese dazu zunächst heruntergeladen, temporärer gespeichert und dann geöffnet, schließt der Entwickler das Artefakt wieder, wird die Datei verworfen.

Um das Eclipse-Plugin verwenden zu können, muss lediglich der URL des zu nutzenden KISSy-Dienstes spezifiziert werden (siehe Abbildung 11: Eclipse basierte Oberfläche von KISSy). Weiterhin können in den Plugin-Einstellungen auch die Bewertungen der Problemmuster, wie in der bereits beschriebenen grafischen Konfigurationsoberfläche, geändert werden.

Weboberfläche

Es wurde die webbasierte, graphische Oberfläche von KISSy weiter verbessert. Neben einer übersichtlicheren Formatierung der Ergebnisanzeige wird auch die Möglichekti der Interaktion mit den Treffern angeboten (Bsp.: interaktives Navigieren im Quelltext). Ein beispielhaftes Ergebnis einer Suchanfrage an die KISSy-Weboberfläche ist in Abbildung 12 dargestellt. Wie zu sehen ist, kann ein Suchergebnis jeweils durch einen Quellcode-Ausschnitt und eine grafische Projektdarstellung ergänzt werden.

Abbildung 12: KISSy Weboberfläche

Im angezeigten Ergebnis kann das Highlighting des Quellcodes durch Konfiguration (noch nicht in der/einer grafischen Konfigurationsoberfläche integriert) personalisiert werden. Durch anklicken der URL des Ergebnisses kann das gefundene Artefakt geöffnet werden. Ebenfalls interaktiv bedienbar ist die grafische Abbildung des Projektes, das den gefundenen Artefakt enthält, durch anklicken der entsprechenden Ergebnisse, werden diese jeweils geöffnet und komplett angezeigt.

In der Ergebnisdarstellung wurden außerdem die folgenden Änderungen durchgeführt, die vor allem auch durch Benutzerrückmeldungen (von Seiten der Anwendungspartner) in die Darstellung eingeflossen sind:

- Ranking der Ergebnisse

    - Die Ergebnisse werden nach Zugehörigkeit und Art der enthaltenen Artefakte bewertet (zuerst Paket, dann Klassen und dann erst Methoden listen, Ordnen nach Komplexität, Kohäsion usw.)
    - Aggregation der Ergebnisse gleicher Klassen
    - Ranking der Ergebnisse auf Basis der ermittelten Software-Qualität
    - Berücksichtigung der Relevanz in anderen Projekten (Anzahl der Verweise)
- Unterstützung der Benutzer
    - Kontextabhängige unscharfe Suche
    - Interaktive Untersuchung der dargestellten Ergebnisse unter Berücksichtigung der Projektstruktur
    - Wahlweise Darstellung in neuem Suchergebnisfenster oder in bevorzugtem Editor
- Grafische Veranschaulichung der ermittelten Ergebnisse im Kontext des dazugehörigen Projekts
    - Interaktive Querverweise in Projektgraphen
    - Navigation durch den Projektgraphen (z.B. vom Paket zu seinen Klassen und Methoden)
    - Unterschiedliches Darstellungsformat je nach Beziehung (z.B. aufgerufene Methoden innerhalb einer Methode vs. Member einer Klasse)

Im Weiteren sind derzeit folgende Ergänzungen zu KISSy denkbar:

- Unterscheidung des im Code verwendeten Lizenzmodells
- Anzeige eines Trustlevels für die gefundenen Treffer
- Tutorial-Suche nach bestimmten Funktionen/Funktionalitäten durch entsprechende Vorauswahl (anklicken verschiedener Alternativen) durch den Anwender
- Erweiterung der grafischen Konfiguration um die Einstellung des Syntax-Highlightings.
- Anstelle des direkten Öffnens von angezeigten Artefakten im Projektgraphen wäre auch die automatische Formulierung einer Suchanfrage, die diesen Artefakt enthält denkbar
- Eine Kontextbeaobachtung der Entwickler könnte automatisiert Informationen zu Suchanfragen erzeugen, die den Präzisionsgrad der Ergebnisse erhöhen können.

Mit Hilfe von KISSy kann sich der in Abbildung 7 beschriebene Anfragezyklus eines Entwicklers deutlich abändern und vereinfachen. Der neue Anfragezyklus ist in Abbildung 13 dargestellt, er umfasst auch bereits die oben aufgezählten in Zukunft denkbaren Erweiterungen von KISSy.

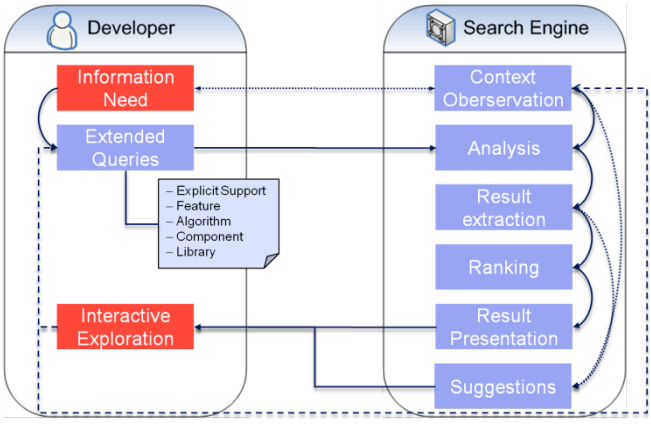
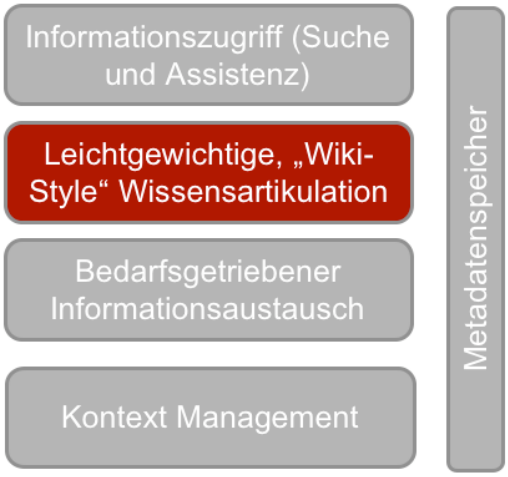


Abbildung 13: KISSy Suchanfragezyklus

Durch eine Kontextbeobachtung und die Spezifikation des gewünschten Anfrageumfangs (Tutorial, Feature, Algorihtmus, Code-Beispiel…) kann es zur Vermeidung mehrfacher Anfrage-Reformulierungen oder zumindest zur Reduktion der mehrfachen Anfrageformulierung kommen, da beide Informationen auf Seite der Suchmaschine genutzt werden können und Einfluss auf die Ergebnisauswahl nehmen.

4.2 Leichtgewichtige „Wiki-Style“ Wissensartikulation (Arbeitspaket AP5)



Eine Plattform, die die vorhandenen Informationsquellen aus Unternehmen, Projekten und externen Wissensquellen miteinander integriert, bietet neue und effizientere Möglichkeiten zur Wissensartikulation und vielfältigere Nutzungsmöglichkeiten des erfassten Wissens. Effizienter, weil viel Information aus dem Kontext ermittelt und vom Benutzer nicht eigens und redundant eingegeben werden muss (bestenfalls aus mehreren Vorschlägen ausgewählt). Die Artikulation kann eindeutiger, strukturierter und damit fehlerfreier erfolgen, ohne den Benutzer zusätzlich kognitiv zu belasten.

Dieses Prinzip wurde im Projekt in zweierlei Hinsicht verfolgt: Ein *evolutionärer* Ansatz bestand in der Verbesserung bestehender *Wikis*, die schon heute in vielen Software-Entwicklungsprojekten die Rolle eines Information Hubs spielen, der für Neulinge als zentrale Anlaufstelle dient, auf die übrigen Quellen verweist, Hinweise zu deren Verwendung gibt, und alle Informationen aufnimmt, die aufgrund ihrer Ad-hoc-Natur oder fehlenden Vorstrukturierung nicht in eine der „richtigen“ Systeme passen. Ein eher *revolutionärer* Ansatz bestand in einer noch radikaleren Unterstützung für die Wissensartikluation in Form von besonders einfach zu erstellenden Ad-hoc-Modellen, hier kombiniert mit dem Fokus auf Persönlichem Wissensmanagement; allerdings greift auch dieser Ansatz altbewährte Muster, nach denen Menschen ihr Wissen organisieren (Listen, Hierarchien usw.) auf.

4.2.1 Der Wiquila-Ansatz

Der im Projekt WAVES entwickelte Wiquila-Client adressiert diese Herausforderungen mit Hilfe von drei Lösungselementen: Einem Rich-Client, der Vernetzung mit existierenden Unternehmensdaten über dynamische Referenzen und mit einer flexiblen Architektur.

Rich-Client

Wiquila liefert einen Rich-Client auf Basis von Java Swing. Dies ermöglicht offensichtlich viel komfortablere Editierfunktionen und sonstige Interaktionsmöglichkeiten. Insbesondere lassen sich Hintergrundprozesse realisieren, die den gerade eingegebenen Text analysieren und zum Beispiel Vorschläge zum Einfügen von Links auf bekannte Ressourcen generieren, ohne dass das normale Arbeiten behindert wird. Auch der Offline-Zugriff ist so kein Problem.

Nun gilt es als einer der Hauptvorteile von Wikis, dass sie ohne weitere Installation im Internet-Browser aufgerufen werden können und zunächst scheint es so, als gäbe unser Ansatz diesen Vorteil auf. Allerdings erscheint uns der Schritt in mehrerer Hinsicht gerechtfertigt: Frühe Hypertext-Clients, einschließlich des ersten WWW-Browsers, hatten immer schon, wenn auch rudimentäre, Editierfunktionen eingebaut. Zweitens sind in den letzten Jahren eine Reihe von Rich-Client-

Technologien bereitgestellt worden, die ein stark vereinfachtes Deployment oder den Start direkt aus dem Webbrowser heraus möglich machen. Und schließlich haben, nicht zuletzt dank dieser Technologien, einige Rich Clients zum Zugriff auf Internetdienste große Verbreitung gefunden, wie Instant Messaging, Peer2Peer-Clients oder Anwendungen zum Verwalten von Online-Fotoalben. Für Leute, die häufig mit Wikis arbeiten, ist eine Client-Applikation die richtige Wahl.

Woran sich dadurch nichts ändert, ist, dass Wikitexte vom Server in einer Markup-Sprache per HTTP über eine URL abgerufen werden können. Ein Zugriff über einen normalen Webbrowser bleibt daher weiterhin möglich.

Für die Wahl der Clientplattform waren folgende Kriterien ausschlaggebend:

- Existierender, erweiterbarer Richtext-Editor (mit Schriftgrößen und –stilen);
- Plattformunabhängigkeit;
- Produktive Entwicklungsumgebung;
- Dokumentation, Stabilität, Große Entwicklercommunity;
- Einbettung in die populäre Entwicklungsumgebung Eclipse möglich, in der auch viele andere Werkzeuge der Projekte WAVES und TEAM ausgeliefert werden;
- „Zero-Install"-Option bzw. Start aus dem Browser heraus;
- Option für lokale Datenablage.

Folgende Alternativen wurden evaluiert und letztlich abgelehnt (im Jahr 2005):

- Eclipse AbstractTextEditor → Kein Richtext;
- Eclipse Richtext-Editor auf Basis des GEF-Frameworks → Noch sehr experimentell;
- AJAX-basierter Editor (z.B. FCKEditor, Kupu) → Schwer zu steuern und zu erweitern;
- Mozilla (Gecko, XUL) → schlecht dokumentiert, wichtige Funktionen nur über C++ erweiterbar (erschwert Deployment), damals schlecht in Eclipse integrierbar;
- Microsoft .NET Framework → Richtext-Editor bietet interessante Features, ist aber noch nicht ausgereift, schlecht dokumentiert, nicht plattformunabhängig;
- Microsoft Silverlight 1.0 → Richtext-Komponente erlaubt noch kein Bearbeiten;
- Adobe Flex → Schöne Darstellung, aber schlecht erweiterbar, noch in frühem Stadium, sollte weiter im Auge behalten werden.

Für die schließlich gewählte Alternative Java Swing (Komponente JTextPane mit HTMLEditorKit) sprach:

- Erweiterbares, objektorientiertes Design
- Open Source
- Unterstützt zwar nur HTML 3.1, ist aber ausreichend für Wiki
- Schlecht dokumentiert, aber gutes Buch verfügbar[1]
- Vielfältige Deployment-Optionen (u.a. Java WebStart aus dem Browser)

Eine gewisse Herausforderung stellte die Einbettung in Eclipse dar – Eclipse verwendet zwar Java, hat aber mit SWT ein eigenes UI-Framework. Eine rudimentäre Bibliothek zur Einbettung von Swing in SWT steht zur Verfügung, und mit einiger Unterstützung der Eclipse-Community gelang es schließlich.

[1] Matthew Robinson, Pavel Vorobiev: Java Swing 2nd edition, Manning

Vernetzung mit Unternehmensdaten über dynamische Referenzen

Der Wiquila-Client ist nicht nur mit einem Wiki-Server (allgemein auch Persistenz-Backend genannt) verbunden, um dort Seiten zu laden und zu speichern, sondern auch mit einer beliebigen Anzahl von Informationsquellen für bereits existierende Informationsressourcen im Unternehmen. Dazu gehören in einem Software-Engineering-Szenario üblicherweise das Ticketsystem für Bugs und Anforderungen, das Quellcode-Versionierungssystem sowie möglicherweise ein Gruppenterminkalender.

Mit Hilfe dieser Informationsquellen können einfache Hyperlinks oder dynamische Referenzen in den Wikitext eingefügt werden. Der Client unterstützt dabei mehrere Methoden: Erstens können Referenzen automatisch während des Schreibens vorgeschlagen werden:

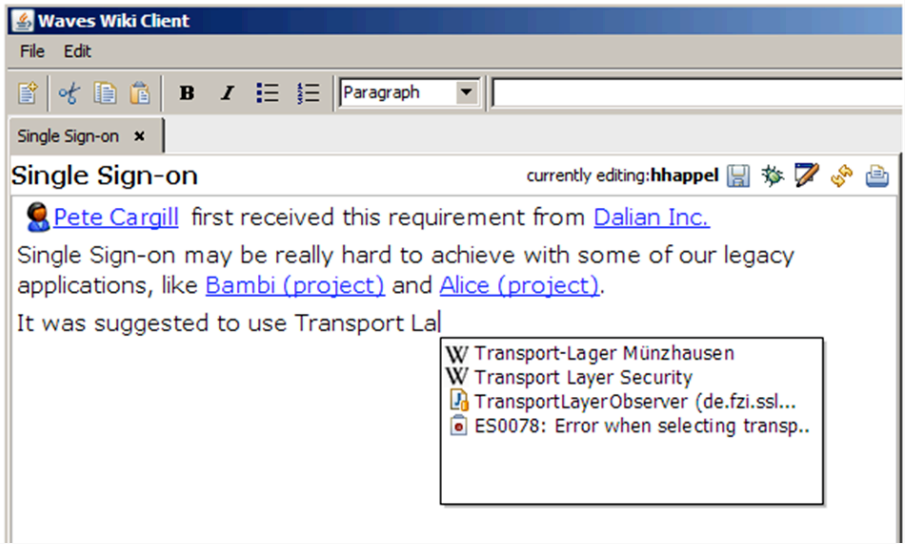


Abbildung 14: Der Wiquila-Client schlägt automatisch Referenzen auf bekannte Ressourcen vor

Der Benutzer kann konfigurieren, welche Quellen hierfür herangezogen werden und damit steuern, wie häufig Vorschläge gemacht werden.

Zweitens können Referenzen aktiv vom Benutzer angefordert werden. Dies ist sowohl mit einer reinen Tastatursteuerung als auch mit der Maus möglich.

Das Anzapfen vorhandener Informationsquellen ist besonders einfach zu konfigurieren, wenn WavesIS im Einsatz ist, da dort bereits alle relevanten Quellen integriert sind und eine schnelle Suche möglich ist.

Viele Referenzen sind dynamisch, d.h. ihre Darstellung kann sich mit der Zeit ändern. So kann die Referenz auf ein Ticket z.B. darstellen, ob das Ticket offen oder geschlossen ist. Es besteht auch keine Garantie, dass eine einmal referenzierte Informationsressource in Zukunft immer verfügbar bleibt. Falls sie verschwindet, muss die Referenz mit einer Warnung versehen werden. Der Wiquila-Client versucht daher regelmäßig, die Referenzen in einer geladenen Seiten zu überprüfen und passt ihre Darstellung gegebenenfalls an.

Flexible Architektur

Die Architektur des Wiquila-Clients ermöglicht große Flexibilität bezüglich des Deployments, des verwendeten Persistenz-Backends, sowie der Informationsquellen für Referenzen.

Die angestrebten Deployment-Varianten wurden bereits angedeutet: Als vollwertige eigenständige Applikation, als Plug-in von Eclipse, als Miniapplikation, die möglichst schnell aus dem Browser gestartet werden kann. Die Kernkomponente, der Wikiseiten-Editor, ist dabei jedes Mal die gleiche, nur Menüs, Toolbars und die weitere Peripherie ändern sich.

Das vollständige Ablösen existierender Unternehmenswikis in einem Schritt erschien nicht realistisch, daher erlaubt der Wiquila-Client generell die Anbindung beliebiger Wiki-Engines (Persistenz-Backends) über ein entsprechendes Plug-in. Ein solches Plug-in wurde z.B. für das Confluence-Wiki der Firma Atlassian entwickelt. Die Art des Backends bestimmt dabei die Features, die im Client aktiv sind. So erlauben viele Backends z.B. außer Wikiseiten auch Attachments zu speichern.

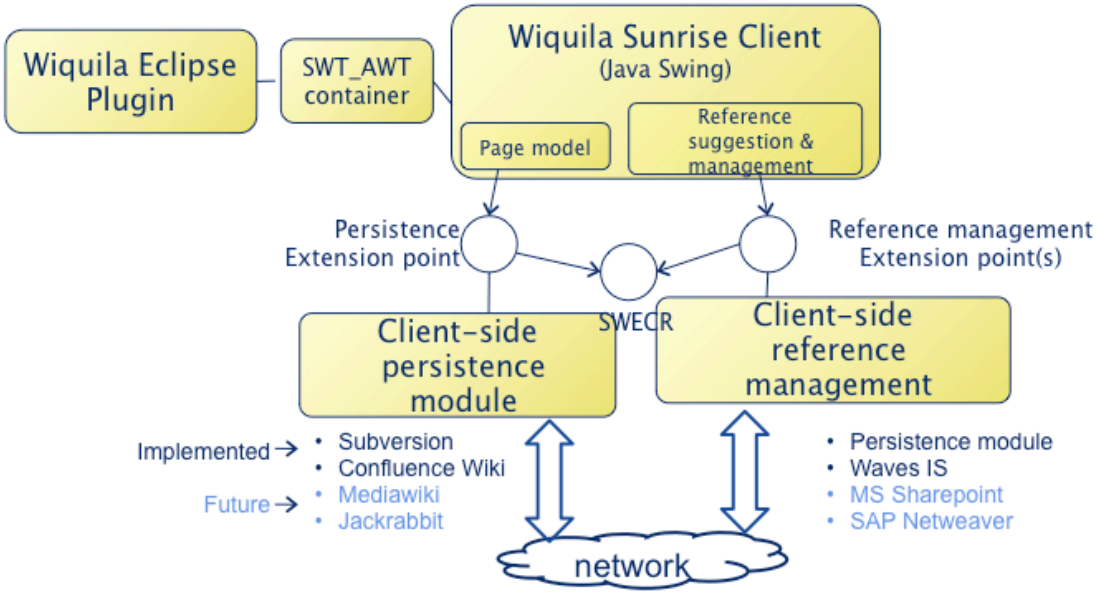


Abbildung 15: Architektur und Erweiterungsmöglichkeiten des Wiquila-Clients

Abbildung 15: Architektur und Erweiterungsmöglichkeiten des Wiquila-Clients zeigt die Architektur im Überblick. Um neue Persistenzmodule oder Typen von Quellen für Referenzen zu verwenden, muss man auf dem Client ein entsprechendes Plug-in bereitstellen. Die Alternative dazu besteht darin, ein Netzwerkprotokoll zu benutzen, für welches schon ein Plug-in existiert, und die nötige Anpassung auf der Serverseite vorzunehmen. An sich wäre der Standard WebDAV ein guter Kandidat für ein standardisiertes Netzwerkprotokoll, da WebDAV den kollaborativen Umgang mit Informationsressourcen zum Inhalt hat, inklusive Erweiterungen für Versionierung und Rechtmanagement. Leider gibt es zum jetzigen Stand weder eine vollständige Implementierung eines solchen Servers noch eine gepflegte Client-Bibliothek, um konforme Nachrichten zu senden und zu empfangen.

Fazit

Der Wiquila-Client entwickelt das Wiki-Konzept evolutionär weiter zu einer produktiveren *Knowledge Workbench*, die mit den vorhandenen Informationsquellen im Unternehmen verbunden ist. Der Client steht als herunterladbare Applikation sowie als Quellcode öffentlich zur Verfügung. Über das Projektende hinaus werden weitere Plug-ins sowie Deployment-Varianten entwickelt.

4.2.2 PKM: Persönliches Wissensmanagement

Primäres Ziel von WAVES besteht darin, in verteilten Softwareprojekten den Aufbau und den Austausch von informellem Wissen zu fördern und seine schrittweise Strukturierung und Vernetzung zu unterstützen. Für einen effizienten Wissensaustausch bei der verteilten Entwicklung von Software ist es notwendig das Erstellen von Artefakten zu erleichtern und zu fördern, damit „überhaupt etwas da ist, was geteilt werden kann“.

Während persönliches Informationsmanagement sich vor allem mit der Verwaltung von bestehenden Wissensartfakten beschäftigt (z.B. Emails, Termine, Kontakte, Bookmarks) konzentriert sich personal knowledge management (PKM) auf selbst erstellte persönliche Notizen. Im PKM kommuniziert der

Nutzer also vor allem über den Umweg eines Werkzeuges mit sich selbst um so kognitive Grenzen (Langzeitgedächtnis, Kurzzeitgedächtnis) zu verschieben. Beim persönlichen Wissensmanagement liegt das Augenmerk also auf der Unterstützung des Einzelnen im Umgang mit digitalen Artefakten. Diese Artefakte sind vom Nutzer selbst geschrieben (Notizen,), oder zumindest selbst gesammelt und eingeordnet (Links, stack-traces, Quellcode-Fragmente, Installationsanleitungen, ...). Wenn der Einzelne, leicht digitale Artefakte erstellen, verwalten und nutzen kann, dann entsteht dabei eine Sammlung digitaler Artefakte. Diese Sammlung besteht zu jedem Zeitpunkt aus teilweise sehr informalen, bruchstückhaften Notizen als auch aus hochwertigen, ausformulierten Notizen. Ab einer gewissen Mindestqualität – die je nach Adressatenkreis sehr unterschiedlich sein kann – können diese Notizen dann mit anderen geteilt und von diesen genutzt werden.

Im Rahmen von WAVES wurde ein Wissensodell und der Prototyp eines Werkzeuges speziell für das Persönliche Wissensmanagement erstellt. Die Ergebnisse werden auch in einer Dissertation (Völkel, 2009) dokumentiert. Der Prototyp wurde unter dem Namen Hypertext Knowledge Workbench (HKW) implementiert und bei vier Anwendungspartnern (Disy, OI, CAS, PTV) evaluiert.

Entscheidend für erfolgreiches PKM ist der Zusammenhang zwischen Aufwand und Nutzen. Aufwände entstehen für das Erstellen einer Notiz, das Organisieren und Formalisieren von Notizen sowie beim Suchen von Notizen. Nutzen entsteht ausschließlich wenn relevante Notizen während einer Aufgabe dem Nutzer helfen, bessere Entscheidungen zu treffen. Maßgeblich für die Suchkosten ist der Strukturierungsgrad der Notizen. Textnotizen ohne jede innere Struktur, ohne Überschriften, ohne Vernetzung mit anderen Notizen, ohne Metadaten (z.B. Erstellungsdatum) und ohne Sortierung in Ordner sind nur schwer wiederzufinden wenn es darauf ankommt. Auf der anderen Seite führen Werkzeuge mit zu hohen Strukturierungsanforderungen (z.B. Datenbanken, Formulare) oft dazu, dass Nutzer ihr Wissen gar nicht eingeben, weil es zu umständlich wäre. Eine ausführliche Diskussion der Kosten- und Nutzenaspekte von PKM findet sich in (Völkel, M., Abecker, A. (2008)).

HKW löst das Problem des Strukturierungsgrades indem es dem Nutzer erlaubt, Notizen in beliebiger Granularität und mit beliebigem Strukturierungsgrad anzulegen, zu verändern und zu nutzen. Notizen können leicht mit anderen Notizen vernetzt werden. In Abbildung 16 ist die webbasierte Oberfläche des Werkzeugs dargestellt. In diesem Beispiel sind die Notizen einer fiktiven Nutzerin dargestellt, sowie deren Verknüpfungen zu anderen Notizen.

In HKW ist ein Wissensnetz aus einer Menge von Notizen angelegt. Innerhalb jeder Notiz kann Wiki-Syntax verwendet werden um Text zu strukturieren und die Notiz mit anderen Notizen zu vernetzen. Konzepte werden ebenfalls durch Notizen repräsentiert. Auch das im Web 2.0 beliebte 'tagging' wird so in HKW realisiert: Der Nutzer verbindet eine Notiz mit einer Tag-Notiz. Eine Tag-Notiz ist dabei eine Notiz die nur den Name des Tags selbst enthält, z.B. "java". HKW unterstützt auch Meta-Modellierung, d.h. jede Verbindung zwischen zwei Notizen wird selbst wieder als Notiz aufgefasst und kann daher annotiert, getaggt, getypt und vernetzt werden.

Technisch vereint HKW damit die Möglichkeiten einer raschen initialen Dateneingabe mit der Möglichkeit einer graduellen Formalisierung und Strukturierung.

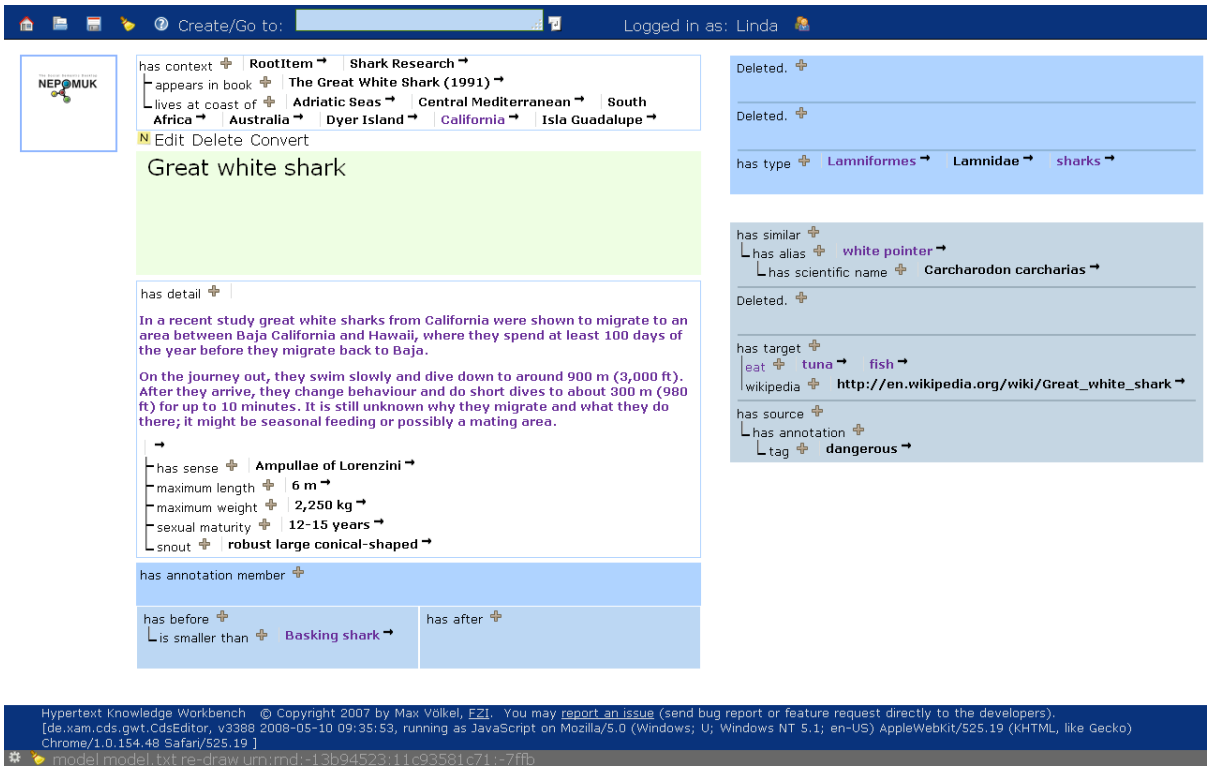


Abbildung 16: Das Werkzeug HKW

4.2.3 Organisationsmodell und Zugriffskontrolle für Wikis (Arbeitspaket AP7)

Überblick

Das Design des Zugriffskontrollsystems besteht aus zwei Teilen, die soweit wie möglich unabhängig voneinander sind: Der erste Teil ist das Organisations- und Zugriffsrechtemodell, welches die verschiedenen Szenarien abdeckt und welches RDF und die Abfragesprache SPARQL als Persistenzschicht verwendet. Der andere Teil realisiert die eigentliche Zugriffsprüfung in verschiedenen Szenarien. Insbesondere kümmern wir uns um Szenarien, wie sie in semantischen Wikis zu erwarten sind.

Design des Organisations- und Rechtemodells

Das Design des Organisations- und Rechtemodells verfolgte u.a. folgende Ziele:

- Das Modell soll möglichst viele der skizzierten Szenarien abdecken. Es soll in organisationsübergreifenden Szenarien funktionieren und gleichzeitig in kleinen, selbstadministrierten Gruppen.
- Das Modell soll sich leicht als RDF-Graph darstellen lassen; mit Hilfe der RDF-Standard-Abfragemechanismen sollte es schnell möglich sein, häufige Anfragen bezüglich Zugriffsrechten zu beantworten.
- Das Modell soll eine kompakte Darstellung der Zugriffsrechte erlauben. Insbesondere ist es nicht wünschenswert, generell für jede einzelne Wissensressource (z.B. Wikiseite) eine eigene Access Control List zu führen; dies würde auch in vielen Anwendungsfällen eine performante Prüfung schwierig machen.
- Das Modell soll es Client-Software ermöglichen, schnell schon im Vorhinein zugelassene und nicht zugelassene Operationen und Ressourcen entsprechend zu kennzeichnen – statt diese Operationen und Ressourcen einfach nebeneinander aufzulisten und erst beim tatsächlichen

Zugriff eine Prüfung durchzuführen. Dazu sollte es der Client-Software auch möglich sein, Teile des Zugriffsmodells effizient zu cachen.

- Das Modell sollte für Laien in den meisten sie betreffenden Fällen leicht verständlich sein, und Administratoren die Möglichkeit geben, sich einen schnellen Überblick zu verschaffen. Die Folgen von Flüchtigkeitsfehlern bei der Rechtevergabe sollten möglichst gering gehalten werden.

Bezüglich des letzten Punktes und der Vergabe von Einzelrechten an Einzelressourcen galt uns die im Projekt verwendete Wiki-Software TWiki sowohl als Vorbild als auch als warnendes Beispiel. Sie erlaubt einerseits eine mehrstufige Konfiguration sowohl ganzer Wiki-Spaces als auch einzelner Seiten; Fehlkonfigurationen zu erkennen ist aber äußerst schwierig und sie können den gesamten Server gefährden.

Das schließlich erstellte Modell enthält folgende Elemente:

Spaces und Super-Spaces. Das Objekt, dem Rechte eigentlich zugeordnet werden, ist der Space („Raum"). *Wissensressourcen* befinden sich immer in genau einem Raum. Es ist nicht möglich, Wissensresourcen Einzelrechte zuzuordnen. Räume können, so der Gedanke, von allen Benutzern erstellt werden. Sie sind mit den (Interessen-)Gruppen in Facebook vergleichbar. Es kann z.B. auch Räume geben, für die nur ein Benutzer Schreibrechte hat, aber für den er anderen Benutzern Leserechte gibt – insofern wird strukturell so etwas nachgebildet wie die persönlichen Fotoalben, die man bei Flickr.com oder Facebook einrichten kann.

Die normalen, von Benutzern erstellten Wissensresourcen, wie z.B. Wikiseiten, können nachträglich auch in einen anderen Space verschoben werden. Daneben kann es in der Wissensbasis importierte bzw. virtuell integrierte Ressourcen aus Fremdsystemen geben (wie z.B. Kunden aus einem CRM-System). Ihnen wird ein Space fest zugeordnet, üblicherweise durch den Administrator definiert beim Konfigurieren der Import-Schnittstelle (siehe WavesIS). Bei bestimmten Operationen kann weiterhin die Rechteprüfung des Fremdsystems aktiv werden.

Client-Software, die auf die Wissensbasis zugreift, kann sich beim erstmaligen Zugriff auf einen Space eine Beschreibung der gesamten Zugriffsrechte holen, die der angemeldete Benutzer hat und kann dann bei jeder zukünftigen Ressource in diesem Space sofort die Darstellung entsprechend gestalten (aktiv/inaktiv, etc.).

Jeder Space gehört zu genau einem Super-Space. Standardmäßig sind für eine Organisation z.B. die Super-Spaces *Intern* und *Extern* definiert. Der Super-Space definiert, welche Benutzer in ihm Spaces anlegen dürfen, und welche Benutzer überhaupt Mitglied bei Spaces in diesem Super-Space werden dürfen (zusätzlich müssen sie natürlich vom Administrator des Spaces selbst zugelassen werden). So kann es z.B. nur bestimmten Mitarbeitern gestattet sein, Spaces anzulegen, in denen auch externe Mitarbeiter Mitglied werden können. Der Super-Space bildet daher konzeptionell einen Teil der Funktion der Facebook-*Networks* ab. Nur der globale System-Administrator darf neue Super-Spaces definieren.

Gruppen. Die Abteilungshierarchie eines Unternehmens wird über Gruppen abgebildet. Gruppen können aus Einzelpersonen und anderen Gruppen zusammengesetzt sein (ohne dass sich freilich Zyklen bilden dürfen). Gruppen, so die Erwartung, werden überwiegend durch den Administrator definiert, sind stabil und für die meisten Benutzer sofort durch ihren Namen identifizierbar. Sie dienen vor allem der Wiederverwendung bestimmter Personengruppen über viele Spaces hinweg. Gruppen bilden den anderen – nicht durch Super-Spaces abgedeckten – Teil der Facebook-Networks ab, nämlich die klare Zugehörigkeit von Personen zu Networks, die von ihnen nicht einfach frei gewählt werden kann. *Projekte* und *Arbeitsgruppen* werden üblicherweise durch Spaces, nicht durch die „Gruppen" unseres Rechtemodells abgebildet.

Rollen und Space-Typen. Auch den Subjekten (Gruppen oder Einzelpersonen) werden niemals Einzelrechte direkt zugeordnet, da dies schnell unübersichtlich wird und schwer zu pflegen ist. Stattdessen werden diese Subjekte innerhalb eines gegebenen Spaces *Rollen* zugeordnet. Die Gruppe der *Waves-Entwickler* kann z.B. für den Space *Waves-Architektur* die Rolle der *Autoren* zugeordnet

bekommen. Durch eine entsprechende Benennung der Rolle ist dem Benutzer normalerweise sofort intuitiv klar, welche Einzelrechte dazugehören, auch wenn dazu auf der technischen Ebene viele Einzelrechte gehören. Der normale Benutzer und selbst der Administrator im Alltagsbetrieb müssen sich nie um die genaue Definition der Rollen Gedanken machen. Stattdessen wählt man beim Anlegen eines Spaces einfache einen bestimmten *Space-Typ* als Vorlage. Dieser Space-Typ definiert schon eine sinnvolle Menge von Rollen vor.

So läßt sich z.B. mit einem bestimmten Space-Typ ein skype-ähnliches Modell nachbilden, bei dem jedes Mitglied eines Spaces (Skype-Gruppe) im Normalfall auch andere Mitglieder einladen und zulassen kann, jedoch keines der Mitglieder ein anderes Mitglied ausschließen kann. (In diesem extremen Fall würde der Space-Typ nur eine Rolle, die des Mitglieds, vordefinieren.)

Das hier verwendete Rollenkonzept stellt keine vollständige Implementierung von rollenbasierter Zugriffskontrolle (RBAC) dar – bei letzterer arbeiten Benutzer immer in genau einer der für sie zugelassenen Rollen.

Operationen und Einzelrechte. Die Operationen, um die es bei der Zugriffsprüfung letzlich geht, werden zum größten Teil von der Anwendung definiert. Ihre Menge ist daher einerseits offen, da ständig neue Anwendungsmodule hinzukommen können, andererseits kann sie der Administrator nicht beeinflussen. Um im Falle von später zusätzlich definierten Operationen zu einem vernünftigen Standardverhalten zu gelangen, wenn die Rollen bisher ohne Rücksicht darauf definiert wurden, können Operationen auf generellere Operationen verweisen, so dass eine Generalisierungs-/Spezialisierungshierarchie entsteht. Vorbild hierfür ist der WebDAV-Standard. So kann z.B. eine allgemeine *Lese*-Operation definiert werden, der einzelne, spezielle Lese-Operationen (z.B. für die Ressourcen-Inhalte, für die Metadaten, für das Lesen der Zugriffsrechte etc.) untergeordnet werden. Einzelrechte gewähren also einer bestimmten Rolle innerhalb eines bestimmten Space eine bestimmte Operation, zu der auch alle untergeordneten Operationen gehören, wenn nichts anderes bestimmt ist. Die Prüfung, ob eine bestimmte Operation effektiv zugelassen ist, läßt sich in RDF-Metadatenspeichern effizient über das RDF-Feature der *Subproperties* realisieren.

Das Rechtemodell definiert auch selbst einige Operationen wie z.B. Neuen Space erzeugen (in einem gegebenen Super-Space X), Zugriffsrechte lesen, Subjekt einer Rolle zuordnen usw.

Schnittstellen der Zugriffsrechte-Komponente

Die Zugriffsrechte-Komponente stellt ihr Funktionen nach außen über zwei Schnittstellen transparent bezüglich der Details des Rechtemodells und des Persistenzmechanismus zur Verfügung. Dies stellt sicher, dass das Rechtemodell in Zukunft weiterentwickelt werden kann, ohne dass Änderungen an allen Anwendungen vorgenommen werden müssen.

Die meisten Anwendungskomponenten interagieren nur mit der Schnittstelle zur Prüfung der Zugriffsrechte *(Permission Checking Interface)*. Der häufigste Fall ist dabei die Prüfung, ob eine einzelne Operation auf einer einzelnen Ressource durch den angemeldeten Benutzer zugelassen ist. Daneben stellt die Schnittstelle auch Mengenfunktionen zur Verfügung, mit der die Menge der für eine Operation zugelassenen Spaces ermittelt werden kann, oder die Menge der zugelassenen Operationen für einen Space.

Das Auslesen und Bearbeiten des Rechtemodells (z.B. Vergeben von Rollen) geschieht über die Rechtemodell-Schnittstelle *(Access Control Model Interface).* Ihre Implementierung verwendet wiederum selbst die obige Schnittstelle zur Prüfung der Zugriffsrechte, um z.B. zu überprüfen, ob der angemeldete Benutzer die Rechte eines anderen Benutzers sehen oder ändern darf.

Zugriffskontrolle für semantische, integrierte Wikis

Als Anwendung des Zugriffskontroll-Mechanismus wurde das in WAVES konzipierte erweiterte Wiki im Detail betrachtet. Ein herkömmliches Wiki stellt von der Struktur her zunächst Hypertext dar, d.h. eine Seite besteht aus formatiertem Text mit Links zu anderen Seiten. Dazu kommen bei vielen Wikis noch feste Textbausteine und dynamische Inhalte, deren Darstellung berechnet wird, wie

z.B. Seiteninhaltsverzeichnisse. In solchen Wikis ist die Granularität der Zugriffskontrolle relativ einfach zu definieren, da immer eine Seite im Mittelpunkt steht.

Das in Waves konzipierten Wiki enthält drei Erweiterungen gegenüber herkömmlichen Wikis: Einerseits die Möglichkeit, Links zu anderen Seiten mit Typen zu versehen und damit zu semantisch belegten Aussagen zu machen. Diese Art von Semantik findet sich auch in allen anderen bekannten semantischen Wikis, welche sich zur Zeit langsam in der Praxis verbreiten. Zweitens bietet das WAVES-Wiki auch Unterstützung bei der Referenzierung anderer Ressourcen, die im Unternehmen schon aus anderen Systemen bekannt sind. Um Aussagen über eine Ressource zu machen, muss also nicht nochmal extra eine Wikiseite dafür angelegt werden. Drittens sollte es möglich sein, quer über organisationale Sphähren Wikiseiten zu kommentieren und bei der Darstellung einer öffentlichen Wikiseite auch die privaten Kommentare zu integrieren, auch wenn beide auf unterschiedlichen Servern gehalten werden.

In einem Graphen von semantischen Aussagen ist es nicht mehr so einfach, die Granularität der Zugriffskontrolle festzulegen. Zu einer gegebenen Aussage („Schmidt ist Vorgesetzter von Maier“) gibt es im Allgemeinen eine inverse Aussage („Maier ist Untergebener von Schmidt“) mit der gleichen inhaltlichen Bedeutung.[2] „Gehört“ die Aussage zu Herrn Schmidt oder gehört sie zu Herrn Maier? Wenn sie zu Herrn Schmidt gehört, sind alle Aussagen, die Herrn Schmidt betreffen, vom Zugriff her gleich zu behandeln, oder soll es die Möglichkeit geben, Aussagen bezüglich seiner Untergebenen anders zu behandeln als Aussagen bezüglich seiner Projekte, Kunden, oder Vorgesetzten? Sind Aussagen, die von bestimmten Personengruppen im Rahmen der freien Wissensartikulation über Herrn Schmidt gemacht werden, anders zu behandeln als Aussagen, die aus externen Systemen (z.B. der Personaldatenbank) importiert werden?

Die Herausforderung bestand darin, einen Ansatz zu finden, der

- logisch konsistent;
- in der praktischen Arbeit verständlich;
- auf Basis heute verfügbarer Technologien mit vertretbarem Aufwand implementierbar ist;
- und bei dem Anfragen auf das semantische Netz (die ja den hauptsächlichen Mehrwert von semantischen Wikis und der Datenintegration darstellen) performant beantwortet werden können.

Das Problem der Zugriffskontrolle für RDF-Graphen wurden schon von einigen prototypisch implementierten, und zum Teil recht mächtigen Ansätzen adressiert, die jedoch überwiegend nicht die drei letzten Krtierien erfüllten und zum Beispiel bestimmte Inferenz-Engines voraussetzten.

Der von uns gewählte Ansatz unterscheidet drei Gruppen von semantischen Statements:

1. Metadaten über eine Wissensressource, z.B. Titel, letzter Autor, Datum der Erzeugung einer Wikiseite. Hier richtet sich der Zugriff nach der Wissensressource, über die eine Aussage gemacht wird.
2. Aussagen über realweltliche Objekte. Solche Aussagen müssen immer in einer Wissensressource enthalten sein. Im Falle von frei artikulierten Statements sind sie in einer Wikiseite enthalten. Im Falle von importierten Daten aus externen Systemen muss die Importschnittstelle selbst eine oder mehrere Wissensresourcen definieren, in welche die Aussagen eingebettet werden. Der Zugriff richtet sich nach der Wissensresource, in der die Ausagen enthalten sind, und nicht nach dem Subjekt der Aussage. So kann es zum Beispiel sein, dass ein bestimmter Benutzer keinen Zugriff auf aus dem CRM-System importierte

[2] Das Schließen der einen aus der anderen Aussage ist Sache der verwendeten Ontologie/Regelsprache und der Inferenzmaschine. Diese Technologie sollte im Projekt nicht weiter vertieft, sondern nur angewendet werden. Es ging nur darum, auf welcher Granularitätsebene der Zugriff (z.B. Lese-/Schreibzugriff) auf die genannte Aussage gesteuert werden soll.

Aussagen über einen bestimmten Kunden hat. Nichts hindert ihn jedoch daran, in seiner persönlichen Wissensbasis eigene Statements zu diesem Kunden abzulegen.

3. Kommentare und Annotationen über Wissensresourcen. Auch diese Aussagen müssen immer in einer Wissensresource enthalten sein, welche für die Zugriffssteuerung verwendet wird.

Das Filtern von Abfrageergebnissen auf die Teilmenge, auf die der angemeldete Benutzer Zugriff hat, ist bei diesem Ansatz dann leicht möglich, wenn der RDF-Metadatenspeicher sogenannte Named Graphs bietet und deren gleichzeitige Beauskunftung erlaubt. Dies ist so im SPARQL-Abfragestandard vorgesehen und wird zumindest von der im Projekt verwendeten Sesame-Engine unterstützt. Es werden keine Inferenzmechanismen benötigt, die über den RDF/SPARQL-Standard hinausgehen.

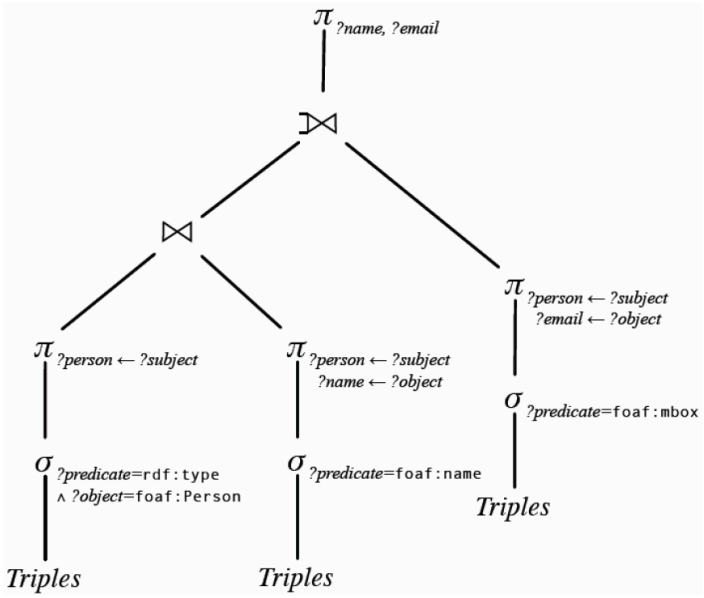
4.2.4 Physikalische Verteilung

Ein weiterer Schwerpunkt im Arbeitspaket Sharing-Engine (AP7) bestand in der Bereitstellung eines Mechanismus zum Zugriff auf physikalisch über mehreren Servern verteilten Wissensbasen. Der Zugriff betrifft sowohl Inhalte (Texte, Bilder, etc.) als auch semantische Aussagen. Da der Zugriff auf Inhalte mit herkömmlichen Techniken gut zu bewerkstelligen ist, soll hier vor allem auf verteilte semantische Abfragen eingegangen werden. Im bisherigen Stand der Technik sind solche verteilte Abfragen nur eingeschränkt möglich und man setzt z.B. voraus, dass überall die gleiche Metadatenspeicher-Software verwendet wird. Im Projekt wurde einerseits eine Technik zur Verteilung der Abfragen über heterogene Metadatenspeicher entwickelt, andererseits eine rudimentäre Technik zur Schätzung von Selektivitäten auf entfernten Metadatenspeichern entwickelt, mit deren Hilfe Join-Operationen optimiert werden können.

Um eine SPARQL-Anfrage auszuführen, ist es zunächst notwendig, die Anfrage zu parsen, sie in einen Syntaxbaum zu überführen und schliesslich als Operatorbaum darzustellen, um auf diesem Optimierungen und die Auswertung durchführen zu können. Cyganiak[3] stellt in seinem Report eine Transformation von SPARQL zu einer relationalen Algebra vor, die als abstrakte Zwischensprache zur Auswertung und Analyse von Anfragen dienen soll. Diese Transformation ermöglicht bestehende Konzepte zur Optimierung und Ausführung aus dem Umfeld der relationalen Datenbanken auf SPARQL zu übertragen. Die folgende Abbildung zeigt eine solche Übersetzung für die folgende Anfrage:

[3] Richard Cyganiak. A relational algebra for sparql. Technical Report HPL-2005-170, Digital Media Systems Laboratory, HP Laboratories Bristol, September, 2005.

```
SELECT ? name ? email
WHERE {
? person rdf: type foaf : Person .
? person foaf : name ? name .
OPTIONAL { ? person foaf : mbox ? email }
}
```



Selektivitätsinformationen werden zur Optimierung von Datenbankanfragen verwendet. Sie dienen dazu, Bestandteile einer Anfrage mit starker Selektivität möglichst früh auszuführen, um die Zwischenergebnisse klein zu halten. Im Folgenden sollen relevante Begriffe und Definitionen dargestellt und im Kontext von RDF betrachtet werden.

Der Begriff der Selektivität und des Selektivitätsfaktors wurde bereits 1979 von Chamberlin und seiner Forschungsgruppe bei IBM zusammen mit dem System R eingeführt. Der Begriff des Selektivitätsfaktors und die damals entwickelten Techniken sind aber immer noch aktuell, wie der Einsatz in heutigen kommerziellen Systemen zeigt. Der Selektivitätsfaktor entspricht dem Verhältnis zwischen allen Tupeln, die ein gegebenes Prädikat erfüllen, und allen in der Datenbank vorhandenen Tupeln. Im Fall von RDF-Metadatenspeichern kann die Selektivität anhand von zurückgelieferten Tripeln anstatt von Tupeln berechnet werden. Bei der Ermittlung des Verhältnisses kann mit Schätzungen anhand von Statistiken gearbeitet werden. Bernstein et. al[4] haben sich mit der Schätzung von Selektivität bei Tripelmustern beschäftigt und Funkionen zur Selektivitätsberechnung vorgeschlagen.

Darauf aufbauend wird eine Kostenfunktion zur Schätzung der Selektivität eines Tripelmusters definiert, welche sich aus den Kostenschätzungen der einzelnen Bestandteile eines Tripels zusammensetzt. Bei der Optimierung kann somit eine Sortierung einzelner Tripelmuster nach ihrer Selektivität erfolgen.

Der Anfragesteller soll auf einer verteilten RDF-Relation so arbeiten können, dass die einzelnen verteilten RDF-Relationen aus Sicht des Anfragestellers transparent sind. Die RDF-Relationen können auf verschiedene Art und Weise auf mehrere Metadatenspeicher aufgeteilt sein. Diese

[4] Abraham Bernstein, Christoph Kiefer, and Markus Stocker. Optarq: A sparql optimization approach based on triple pattern selectivity estimation. Technical report, Department of Informatics, University of Zurich, 2007.

Aufteilung wird im Kontext von relationalen Datenbanksystemen Fragmentierung genannt. Man unterscheidet zwischen horizontaler und vertikaler Fragmentierung. Diese beiden Fragmentierungstypen können miteinander kombiniert werden. Man spricht in diesem Fall von hybrider Fragmentierung. Diese vorgestellten Begriffe wurden auf RDFRelationen übertragen und angewendet. Als Grundlage für die Beispiele in Zusammenhang mit Fragmentierung soll die RDF-Relation WIKI dienen:

?page	*?title*	*?comment*
page:RDF	"Resource Description Framework"	"Important"
page:OWL	"Web Ontology Language"	"Must read!"
page:MF	"Micro Formats"	"See XHTML"

Die vorgestellte RDF-Relation könnte horizontal fragmentiert auf zwei Metadatenspeicher verteilt sein. Der erste Metadatenspeicher könnte dann beispielsweise die Teilrelation WIKI1 enthalten:

?page	*?title*	*?comment*
page:RDF	"Resource Description Framework"	"Important"

Der zweite Metadatenspeicher würde dann, dementsprechend die Teilrelation WIKI2 enthalten:

?page	*?title*	*?comment*
page:OWL	"Web Ontology Language"	"Must read!"
page:MF	"Micro Formats"	"See XHTML"

Um aus den beiden Teilrelationen WIKI1 und WIKI2 wieder die RDF-Relation WIKI zu rekonstruieren, müssen die beiden Teilrelationen vereint werden.

Horizontale Fragmentierung muss bei der verteilten Bearbeitung von SPARQL-Anfragen dann berücksichtigt werden, wenn in der Anfrage ein Tripelmuster auftritt, für das es zutreffende Tripel in mehr als einem Metadatenspeicher zu finden sind. Die Teilergebnisse von den einzelnen Metadatenspeichern müssen dann wie oben beschrieben vereint werden. Das bedeutet, dass nicht eine vollständige RDF-Relation von den Metadatenspeichern abgerufen wird, sondern nur der Anteil, der dem Tripelmuster entspricht. Analog zu dem obigen Beispiel kann eine RDF-Relation auch vertikal fragmentiert sein. Einer der Metadatenspeicher würde dann beispielsweise folgende Teilrelation WIKI1 enthalten:

?page	*?title*
page:RDF	"Resource Description Framework"
page:OWL	"Web Ontology Language"
page:MF	"Micro Formats"

Die zweite der Teilrelationen WIKI2 würde dann dementsprechend wie folgt ausehen:

?page	*?comment*
page:RDF	"Important"
page:OWL	"Must read!"
page:MF	"See XHTML"

Um aus den beiden Teilrelationen WIKI1 und WIKI2 wieder die RDF-Relation WIKI zu rekonstruieren müssen die beiden Teilrelationen vereint werden. Dieser Fall kann bei der verteilten Verarbeitung von SPARQL-Anfragen dann auftreten, wenn mehrere verschiedene Tripelmuster von mehreren Metaspeichern beantwortet werden.

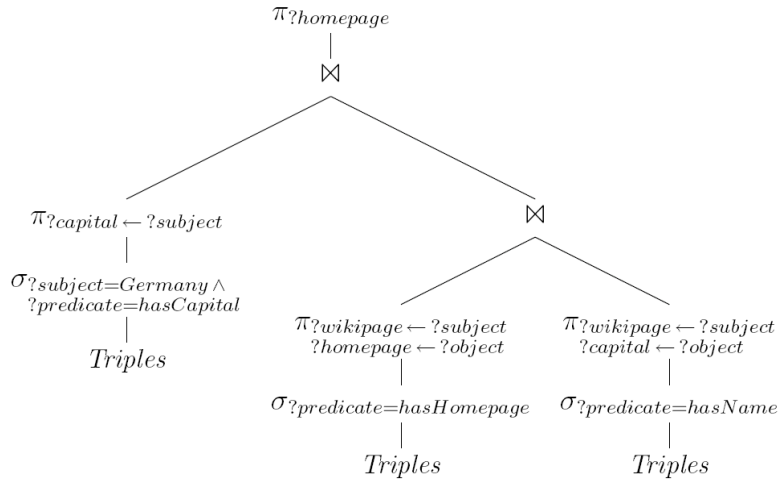
Um eine SPARQL-Anfrage zu verteilen, wird zunächst ein Operatorenbaum der gesamten Abfrage erstellt. Anschließend wird dieser Baum rekursiv durchlaufen, um die einzelnen Knoten mit den Endpunkten, von denen die Informationen abgerufen werden können, zu annotieren. Dies soll am folgenden Beispiel verdeutlicht werden.

Die folgende SPARQL-Anfrage soll verteilt von den beiden Endpunkten E1 und E2

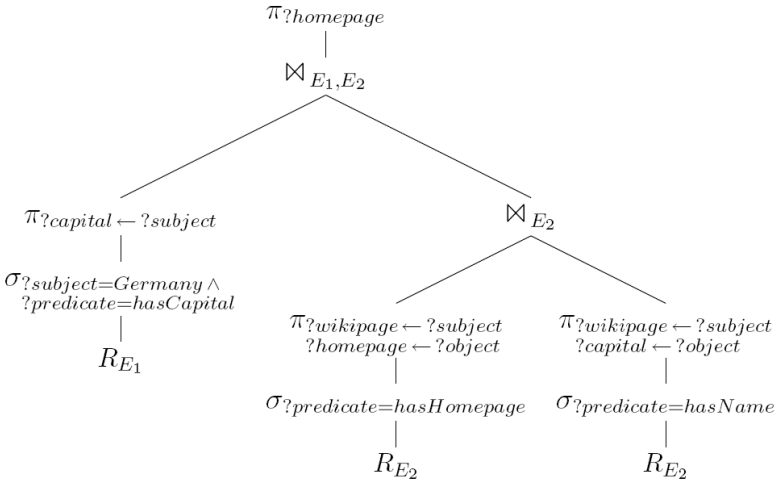
bearbeitet werden:

```
SELECT ? homepage WHERE {
? wikipage foaf : homepage ? homepage .
? wikipage foaf : name ? capital .
db: Germany factbook : hasCapital ? capital .
}
```

Der Endpunkt E1 soll alle Tripel, welche das Prädikat factbook:hasCapital besitzen, beeinhalten. Der Endpunkt E2 hingegen, soll alle Tripel, welche das Prädikate foaf:homepage oder foaf:name besitzen, beeinhalten. Die verteilte Anfrage soll als Ergebnis die Webseiten-URL der Hauptstadt Deutschlands liefern. Eine Zuordnung von Ländern zu Hauptstädten befindet sich in dem Metadatenspeicher mit dem Endpunkt E1. In dem anderen Datenspeicher mit dem Endpunkt E2 befinden sich Wikiseiten mit weiteren Informationen zu einzelnen Städten, darunter auch die Information über URLs der entsprechenden Webseiten. Die folgende Abbildung zeigt die zerteilte Anfrage als Operatorbaum.



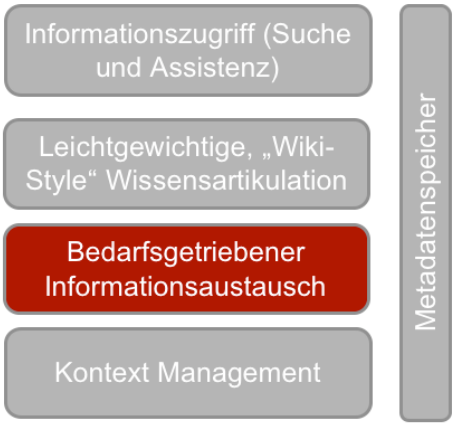
Die nächste Abbildung zeigt das Ergebnis des Algorithmus. Die Knoten des Baumes wurden mit den entsprechenden Endpunkten annotiert. Die angefügten Annotationen können nun bei der Auswertung des Baums verwendet werden. Aus den Annotationen kann abgelesen werden, dass das Tripelmuster (Germany hasCapital ?capital) an den Endpunkt E1 weitergeleitet werden muss. Die beiden Tripelmuster (?wikipage hasHomepage ?homepage) und (?wikipage hasName ?capital) müssen hingegen an den Endpunkt E2 weitergeleitet werden.



Kostenschätzungen kommen bei der Wahl der Ausführungsstrategie für Joins zum Einsatz. Die gewöhnliche Strategie besteht wie eben dargestellt darin, Anfragen parallel an die Endpunkte zu schicken und die Ergebnisse lokal zu vereinen, wobei viele der ursprünglichen Einzelergebnisse

wegfallen können, weil sich kein Gegenstück findet. Ein sehr schlechtes Antwortzeitverhalten kann dann das Ergebnis sein, wenn aufgrund der mangelnden Selektivität einer Teilanfrage zunächst sehr große Datenmengen übertragen werden, die anschließend aussortiert werden. Eben genau diese Situation wird durch die Kostenschätzung nach wenigen durchgeführten Abfragen identifiziert. In diesem Fall wird statt eines lokalen Joins ein sog. Semi-Join durchgeführt; es wird zunächst die Teilabfrage durchgeführt, bei der weniger Ergebnisse erwartet werden. Die Ergebnisse werden dann als zusätzliche Bedingung in der zweiten Teilabfrage verwendet.

4.3 Bedarfsgetriebener Informationsaustausch (Arbeitspaket AP7)



Die Entwicklung komplexer Produkte und Dienstleistungen ist angewiesen auf hochgradig spezialisierte, arbeitsteilig organisierte Entwicklungsschritte. Einzelne Entwickler können den detaillierten Gesamtaufbau eines solchen Systems nur noch rudimentär nachvollziehen. Um ihre eigene Arbeit effizient zu erledigen, benötigen sie Kenntnisse der angrenzenden Schnittstellen.

Diese Beschreibung trifft insbesondere auf die Entwicklung von Softwaresystemen zu, da durch die relativ leichte Wiederverwendung der Schnittstellen niedrigerer Systemschichten, Funktionsbibliotheken und nicht zuletzt beliebig verfügbarer Dienste eine extrem hohe Systemkomplexität entsteht. Sie erfordert von Entwicklern eine kontinuierliche Wissensaufnahme. Andersherum müssen sie aber auch entsprechendes Wissen über die von ihnen verantworteten Komponenten bereitstellen.

In den vorangegangenen Abschnitten wurden diese Aspekte bereits teilweise adressiert. Während die Waves Integrierte Suche den Informationszugriff erleichtert, ermöglichen die Werkzeuge HKW und Wiquila eine benutzbare und effiziente Wissensartikulation. Allerdings adressieren diese Werkzeuge nicht, *welches* Wissen artikuliert werden sollte und *wie* es mit Anderen ausgetauscht werden kann.

Gerade bei Softwareentwicklern sind Tätigkeiten wie Dokumentation und Wissenserstellung relativ unbeliebt, da sie nicht direkt zu "produktiven" Erstellung von Systemfunktionalität beitragen und ihr Nutzen meist nur schwer greifbar ist. Weiterhin ist es ökonomisch wenig effizient bzw. unmöglich, sämtliches entwicklungsrelevantes Wissen explizit zu dokumentieren. Softwareentwickler haben in der Regel aber keine systematischen Informationen darüber, welches Wissen für ihr Projekt/ihre Organisation den größten Nutzen bringt.

Anders ausgedrückt - ein effizienterer Wissensaustausch sollte sich am Bedarf einer Organisation orientieren und dabei möglichst zielgerichtet erfolgen. In diesem Abschnitt werden zu diesem Zweck die beiden Werkzeuge *Woogle* und *Inverse Suche* vorgestellt, die beide auf dem Paradigma des "bedarfsgetriebenen Wissensaustauschs" basieren. Dazu nutzen sie die aggregierten Suchanfragen die bei der integrierten Suche anfallen und priorisieren solche Informationsbedarfe, für die eine schlechte Ergebnisqualität besteht - d.h. nur wenig Wissen vorhanden ist.

Woogle nutzt diese Informationen um Nutzern innerhalb eines Wikis anzuzeigen, wie hoch der Bedarf nach bestimmtem Wissen ist. Dadurch kann zielgerichtet Wissen in einem Wikis erstellt

werden, das besonders stark benötigt wird. Die Inverse Suche hingegen adressiert Dokumente und sonstige Artefakte, die Nutzer lokal auf ihrem privaten Rechner speichern und die demzufolge nicht durch eine unternehmensweite Suche erreichbar sind. Mit Hilfe der aggregierten Informationsbedürfnisse wird eine geordnete Liste solcher Dokumente erstellt, von deren Bereitstellung eine größtmögliche Anzahl von Kollegen profitieren kann.

Beide Werkzeuge - Woogle und die Inverse Suche - werden im Folgenden weiter erläutert. Im Anschluss daran wird darauf eingegangen *wie* Wissen sicher und benutzbar ausgetauscht werden kann - insb. im Hinblick auf Zugriffsrechte.

4.3.1 Woogle: Kombination von Wikis und Suche

Die Suche nach Informationen ist eine der Hauptaktivitäten von Wissensarbeitern. Häufig kann man dabei die Beobachtung machen, dass die Nutzer unzufrieden mit ihren Suchergebnissen sind. Ein Grund dafür ist eine große Treffermenge, welche die Auswahl relevanter Ergebnisse erschwert. Dies ist in Intranets besonders ausgeprägt, da hier automatisierte, statistische Verfahren wie etwa der PageRank Algorithmus von Google schlecht funktionierten und andererseits ist die Annotation der Suchergebnisse mit zusätzlichen Metadaten teuer ist.

Nicht nur das Auffinden, sondern auch das Erfassen von relevanten Informationen fällt ohne entsprechende methodische und Werkzeugunterstützung schwer. Mitarbeiter einer Organisation haben z.B. selten einen Überblick über die “fehlenden” Informationen. Wegen mangelnder Kommunikation zwischen den Mitarbeitern gehen Anfragen häufig verloren.

Grundsätzlich kann man sich zwei Ansätze vorstellen, deren Einsatz man auch in der Geschichte der Internet-Suchmaschinen beobachten kann: Google indiziert Webseiten automatisch im Gegensatz zur Anfangszeit von Yahoo, als Themenkataloge manuell gepflegt wurden.

Das Gestaltungsziel von Woogle ist es den Nutzern Interaktion, Informationsaustausch, gegenseitiges Lernen und Erfassung zusätzlicher Information im Rahmen des Suchprozesses zu ermöglichen. Woogle adressiert daher die unnatürliche Entkoppelung von Suche und Informationsbehebung. Die Grundidee von Woogle ist ähnlich zur Inversen Suche (vgl. folgender Abschnitt) – d.h. Informationsbedürfnisse werden aus Suchanfragen abgeleitet.

Woogle kombiniert die Suche hierbei mit der Wiki-Funktionalität. Es erlaubt das Anlegen einer Wiki-Seite pro Suchbegriff, um einen Informationsbedarf näher zu spezifizieren und um eine leichtgewichtige Wissensartikulation für den Fall zu ermöglichen, wenn noch keine Informationen zu einer Anfrage existieren. Weiterhin können Standardfunktionen des Wikis wie Diskussionsseiten, automatische Benachrichtigungen, das Anlegen von Lesezeichen und die einfache Verlinkung so auf Suchanfragen übertragen werden.

Die Benutzerschnittstelle wird außerdem um Meta-Informationen über die Dringlichkeit eines Informationsbedarfs ergänzt. Damit ermöglicht Woogle eine soziale Interaktion rund um den Prozess der Informationssuche und -Erstellung. Kombiniert mit einer optionalen Anbindung der Waves Integrierten Suche realisiert Woogle so die Vorteile einer automatisierten Indexerstellung und einer kollaborativen, wiki-basierten Beschreibung. Prototypen des Woogle-Konzepts wurden im Rahmen von WAVES für die Wiki-Software MediaWiki und Atlassian Confluence entwickelt.

In der Abbildung 17 ist die prototypische Implementierung von Woogle in MediaWiki dargestellt. In dem dargestellten Beispiel wird nach dem Projekt WAVES gesucht. In dem Beispiel liegen keine Informationen zu der Suchanfrage vor, und Woogle ermöglicht durch das Anlegen von einer neuen Wiki-Seite aus Informationsmangel Informationen zu erzeugen. Ein weiterer Vorteil von Woogle besteht darin, dass es die Akzeptanz von Wikis durch die Integration mit der Suche im Unternehmen erleichtert.

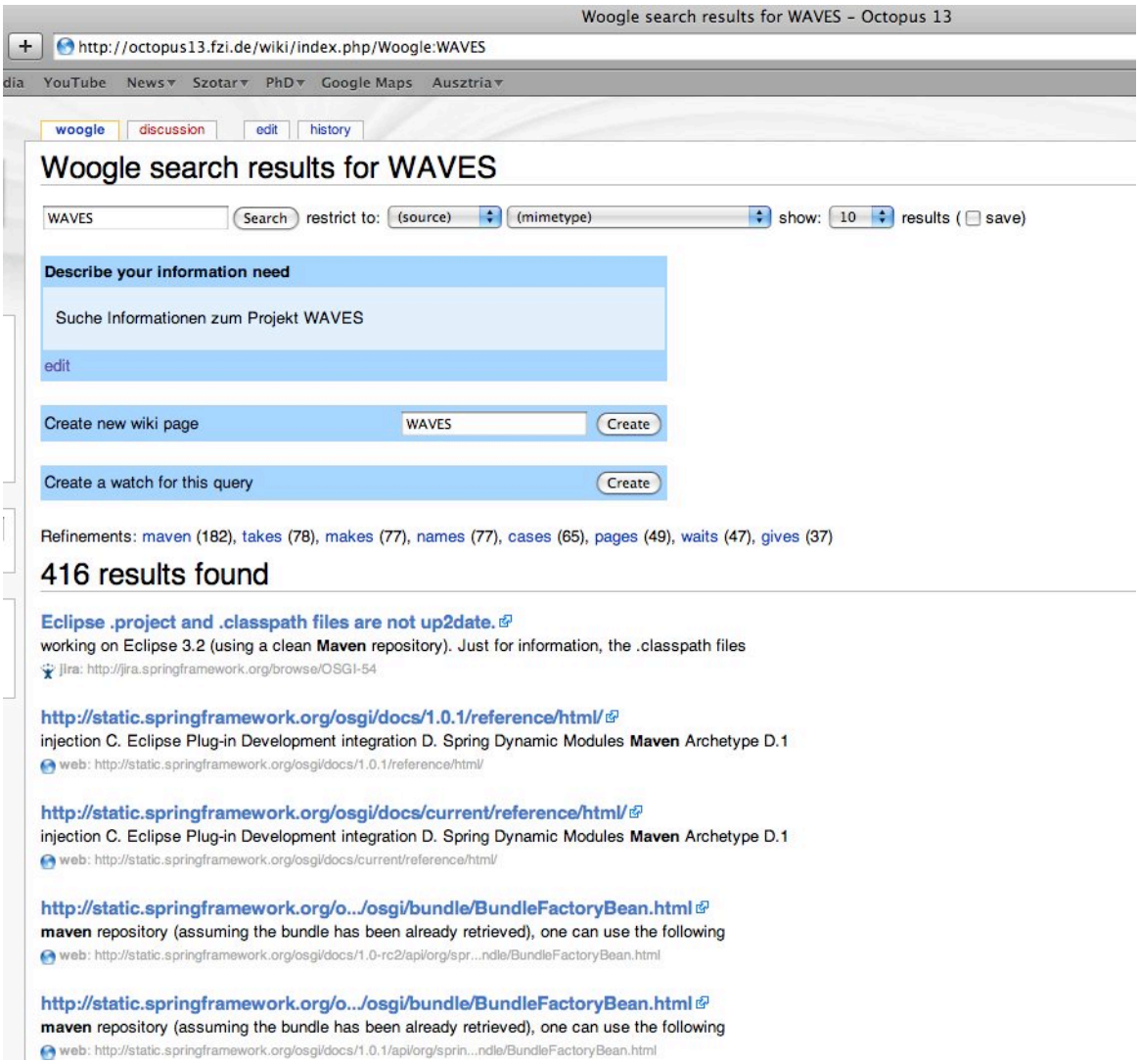


Abbildung 17: Woogle

4.3.2 Inverse Suche: Sharing von lokalen Informationen

Ausgehend von Erfahrungen bei der Arbeit mit der WAVES integrierten Suche (WavesIS) wurde die Idee der inversen Suche abgeleitet. Der Wissensaustausch wird häufig dadurch gehindert, dass viele nützliche Informationen in abgeschlossenen Bereichen verborgen vorliegen (privater Rechner, Desktop, "hidden web"). Die WAVES integrierte Suche ist ein unidirektionaler Prozess, der in der Regel allein auf öffentlich verfügbare Dokumente zurückgreift und zusätzliche versteckte/private Informationen nicht erfasst. Im Suchprozess werden also lediglich die Nutzer mit Informationsbedarf unterstützt. Informationslieferanten werden nicht adressiert, obwohl sie nützliche Informationen bereitstellen könnten.

Das Ziel von der inversen Suche besteht darin, den Austausch von Wissen zu fördern, indem Nutzern Dokumente empfohlen werden, die für andere Mitglieder der Organisation nützlich sind ("Was soll ich austauschen?"). Ein Vergleich zwischen der konventionellen Suche mit der WAVES integrierten Suche und der inversen Suche ist in der folgenden Abbildung dargestellt. Bei der konventionellen Suche stellen die Nutzer eine Anfrage an die gegebene (1) und importieren öffentlich verfügbare Dokumente in ihren privaten Bereich (2). Bei der inversen Suche vergleichen die

Informationslieferanten ihre Dokumente mit einer Menge gegebener Anfragen (3) um Dokumente zu identifizieren, dessen Austausch sich lohnt.

Die Auswahl der auszutauschenden Dokumente basiert auf zwei Werten: Die "Organisatorische Informationslücke" ist groß für solche Begriffe, die relativ häufiger im lokalen Index eines Nutzers auftreten als im öffentlichen Index. Zweitens ist der "Organisatorische Informationsbedarf" höher je häufiger und aktueller ein Begriff angefragt wurde, je mehr verschiedene Nutzer den Begriff dabei verwendet haben und je seltener der Begriff in den lokalen Indizes der Nutzer ist.

Der "Organisatorische Informationsbedarf" ist also die gewichtete Summe der individuellen Informationsbedürfnisse. Die inverse Suche ist daher ein vielversprechender Ansatz für den Wissensaustausch, weil sie die gezielte Freigabe für stark nachgefragte Informationen fördert und demzufolge hilft den Aufwand für den Wissensaustausch zu senken bzw. auf möglichst relevantes Wissen zu fokussieren.

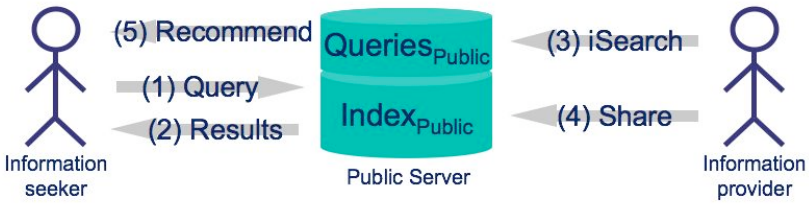
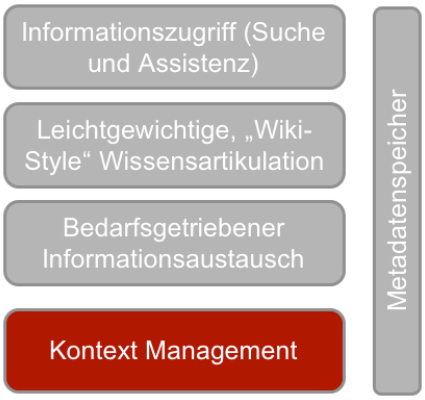


Abbildung 18: Vergleich der konventionellen und inversen Suche

Die *Waves integrierte Suche* dient als Basis-Infrastruktur für die *inversen Suche*. Die Infrastruktur wurde dazu um Anfragemöglichkeiten für Anfragestatistiken erweitert.

4.4 Kontextorientierte Suche und Modellierung (Arbeitspaket AP6)



Im Rahmen von WAVES stellte sich auch die Frage, wie Kontext konkret in einer agilen Softwareorganisation zu modellieren sei und was der konkrete Nutzen sein kann. Hierzu wurde gemeinsam mit der disy GmbH exemplarisch ein solches Modell erstellt und anschließend evaluiert. Dieser Abschnitt fasst die dabei gewonnenen Erfahrungen zusammen.

Kontext kann vielfältig beschrieben werden. Methodisch z.B. durch vergleichende Begriffe, Beispiele oder auch eine generische geschlossenen Definition. Im letzteren Sinne ist Kontext die Gesamtheit der Informationen die eine Situation beschreiben. Dazu gehören im Allgemeinen verschiedenste Aspekte wie

- Wer bin ich?
- Was kann ich?
- Was ist meine Rolle?
- Was ist meine Aufgabe?
- Woran arbeite ich?
- Was brauche ich?
- Wie ist meine Herzfrequenz?

Das Beispiel der Herzfrequenz macht deutlich, dass wirklich die „Gesamtheit der Informationen“ gemeint ist. Das konkrete Beispiel der Herzfrequenz ist ein Indikator für die Stresssituation eines Menschen und als solcher eine wichtige Kontextinformation bei der Bedienung komplexer Maschinen und Anlagen.

Kontextinformation lässt sich grob in menschliche Aspekte und Aspekte der physikalischen Umgebung differenzieren. Die folgende Tabelle zeigt entsprechend kategorisierte Beispiele:

Menschliche Faktoren	Benutzer	Stimmung, Gewohnheiten
	Tätigkeit	Arbeit, Freizeit
	Soziales Umfeld	Personen in der Nähe
Physikalische Umgebung	Ort	objektiv, subjektiv
	Technische Infrastruktur	Geräte
	Verhältnisse	Zeit, Licht, Geräusche, Temperatur

Die Nutzung der Kontextinformation setzt zuerst ihre Erfassung voraus. Diese kann durch Sensorik und Benutzerbeobachtung oder auch explizite Benutzereingaben geschehen. In der Praxis ist die manuelle Eingabe nach wie vor die Regel. Die Verwendung der Kontextinformation besteht dann in der Anpassung des Verhaltens des Systems an die aktuelle Situation. Dabei kann sich die Anpassung auf

- Die Präsentation
- Die Inhaltsstruktur
- Die Inhalte als solche, oder
- Die Auswahl des Inhaltsangebots

beziehen.

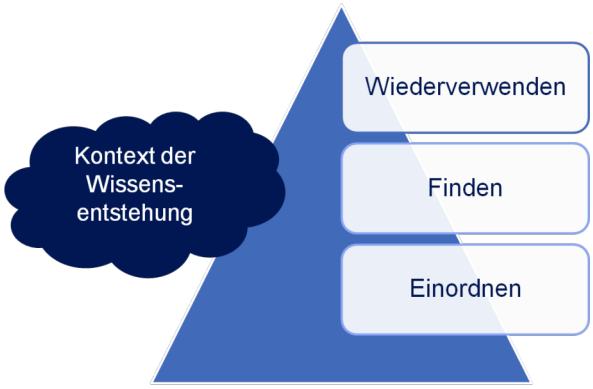


Abbildung 19: Zweck der Kontextnutzung

Der Zweck der Kontextnutzung ist dabei, vornehmlich Wissen einordnen und (wieder)finden zu können und letztendlich durch die Wiederverwendung des Wissens an Produktivität zu gewinnen. Übertragen auf das Software Engineering bedeutet dies heute vor allem Projekterfahrung in Form von Kundenanforderungen, Problemlösungen und anderen Softwareprojektdokumenten wiederverwenden zu können und hierzu den Kontext der Wissensentstehung zu erfassen. Die Wiederverwendung von Code ist erfahrungsgemäß von geringerer Bedeutung. Entsprechend unterstützt das Kontextmodell den Anwender bei der Lösung seines Suchproblems nach wiederverwendbaren Inhalten.

4.4.1 Kontextmodellierung

Es gibt keine geschlossene „Kontexttheorie“. Es kann sie auch nicht geben, da sie das Weltwissen umfassen würde. Kontext muss also von Fall zu Fall modelliert werden. Die Modellierung selbst kann sich dabei auf die erprobten Methoden der Wissensakquisition und -repräsentation stützen – natürlich im Bewusstsein auch der Schwächen dieser Methoden.

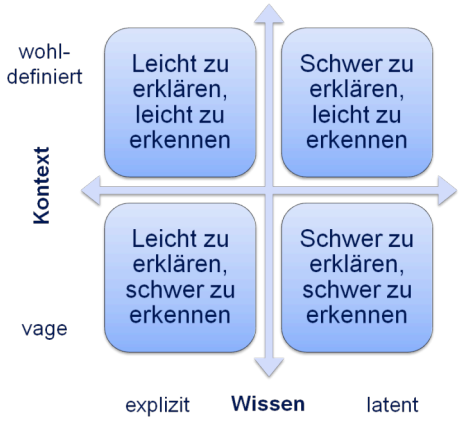


Abbildung 20: Wissen und Kontext

Das resultierende Modell liefert einen Beschreibungsrahmen in dem der Anwender seinen aktuellen Benutzerkontext formulieren kann. Dieser dient dann zur Ergänzung der Beschreibung des aktuellen Informationsbedürfnisses des Anwenders und der Auswahl einer konkreten Suchkonfiguration. Die Modellierung der Kontexte ist dabei ein iterativer Prozess der zu einer kontinuierlichen Sammlung und Erweiterung einer Kontextbibliothek führt. Im Sinne des „Seeding and Growing“ (säen und pflegen) wurde die Bibliothek auf Basis des initialen Modells mit Musterkontexten gefüllt.

Ein Kontextmodell setzt Wissen (Inhalte) und Kontext (Benutzung) miteinander in Beziehung. Daraus ergibt sich ein Spannungsfeld zur Klassifikation von Wissen und Kontext hinsichtlich der Ausdrucksdeutlichkeit das in der obigen Abbildung dargestellt ist.

In der Softwareerstellung sind wir zuerst an Inhalten interessiert die leicht wiederverwendet werden können. Dies sind solche Inhalte, die „leicht zu erklären“ und „leicht zu erkennen“ sind. In der Abbildung ist dies der obere linke Quadrant, d.h. Inhalte die explizites Wissen repräsentieren und einen wohldefinierten Kontext besitzen. Die Abgrenzung von explizitem zu implizitem Wissen ist hinreichend bekannt. Der Begriff des „wohldefinierten Kontext“ Bedarf jedoch selbst der Definition.

Abbildung 21: Wohldefinierter Kontext

Wohldefinierter Kontext soll durch *Attribute* beschrieben sein, die allen Beteiligten bekannt und verständlich sind. Dabei soll die Menge dieser Attribute endlich und über die Zeit betrachtet relativ konstant sein. Die *Wertebereiche* dieser Attribute sollen quantitativ, nominal, aufzählbar und den Anwendern bekannt sein. Letztlich sind sie die Sprache zur Belegung der Attribute. Gemeinsam repräsentieren Attribute und Wertebereiche ein Schema zur Beschreibung von Kontext ähnlich einem Datenbankschema. Beim Vergleich verschiedener Kontexte, d.h. Instanzen des Schemas sollte die *Relevanz* oder besser das verwendete Maß nachvollziehbar sein. Insbesondere so allgemein wie möglich, objektiv und tatsächlich messbar. Soweit darüber hinaus *Beziehungen* zwischen verschieden Entitäten der Kontextbeschreibungen oder Kontexten selbst bestehen sollen diese explizit und in diesem Sinne berechenbar sein.

Ein Kontextmodell, das in diesem Sinne wohldefiniert ist, kann von jedem Anwender erschlossen und damit auch produktiv benutzt werden.

Das stark vereinfachte Beispiel in der folgenden Abbildung erfüllt die Kriterien insbesondere, da die Wertebereiche der Attribute explizit angegeben werden. Sei es durch die nominale Aufzählung oder auch die Angabe von regulären Ausdrücken.

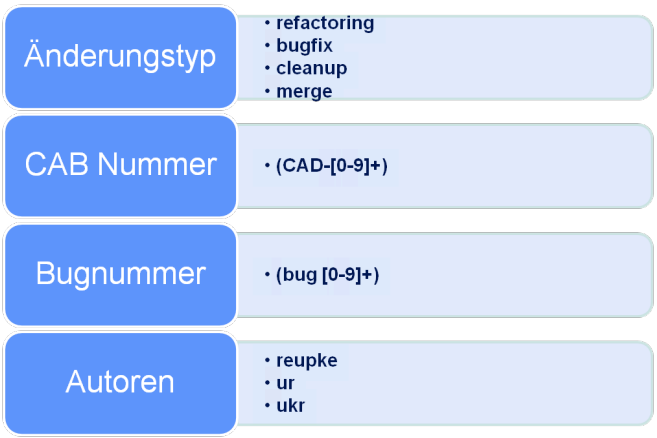


Abbildung 22: Beispiel für ein Kontextmodell

4.4.2 Vorgehensweise bei der Kontextmodellierung und -einführung

Der Prozess der Modellierung und das anschließende Deployment, die Einführung des Systems, waren keine strikt getrennten Phasen sondern Teile eines iterativen Prozesses.

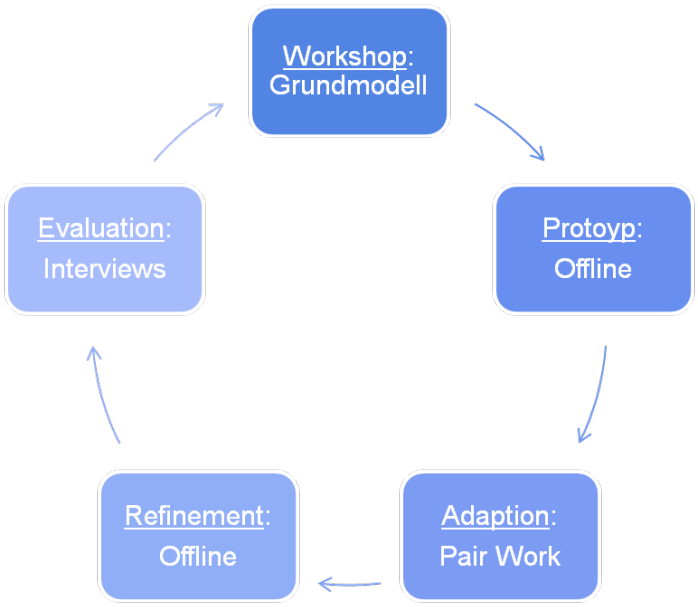


Abbildung 23: Interaktive Kontextmodellierung und -einführung

Die obige Abbildung visualisiert den Prozess: In einem Kick-off Workshop wurde in einem halben Tag ein initiales Grundmodell für die Kontextrepräsentation erarbeitet. Der zweite Schritt des Zyklus war die Realisierung eines ersten Prototyps. Dieser wurde gemeinsam mit einem Mitarbeiter von Disy getestet und verfeinert. Dies war Pair-Modelling im Sinne von Pair-Programming wie aus dem agilen Software-Engineering bekannt. Besonderheit dabei war, ein Pair aus Software und Knowledge Engineer zu bilden.

Im folgenden Schritt wurde das erste operationale System implementiert und systematisch mit Unterstützung der FU Berlin getestet.

Dieses Vorgehen sollte mehrfach iteriert werden, um zu einem vollständigen Modell zu gelangen. Die Anzahl der Iterationen wird dabei von Softwareorganisation zu Softwareorganisation variieren. Im vorliegenden Fall wären vier Zyklen angemessen gewesen von denen zwei durchgeführt wurden.

4.4.3 Praxisbeispiel: das disy Kontextmodell

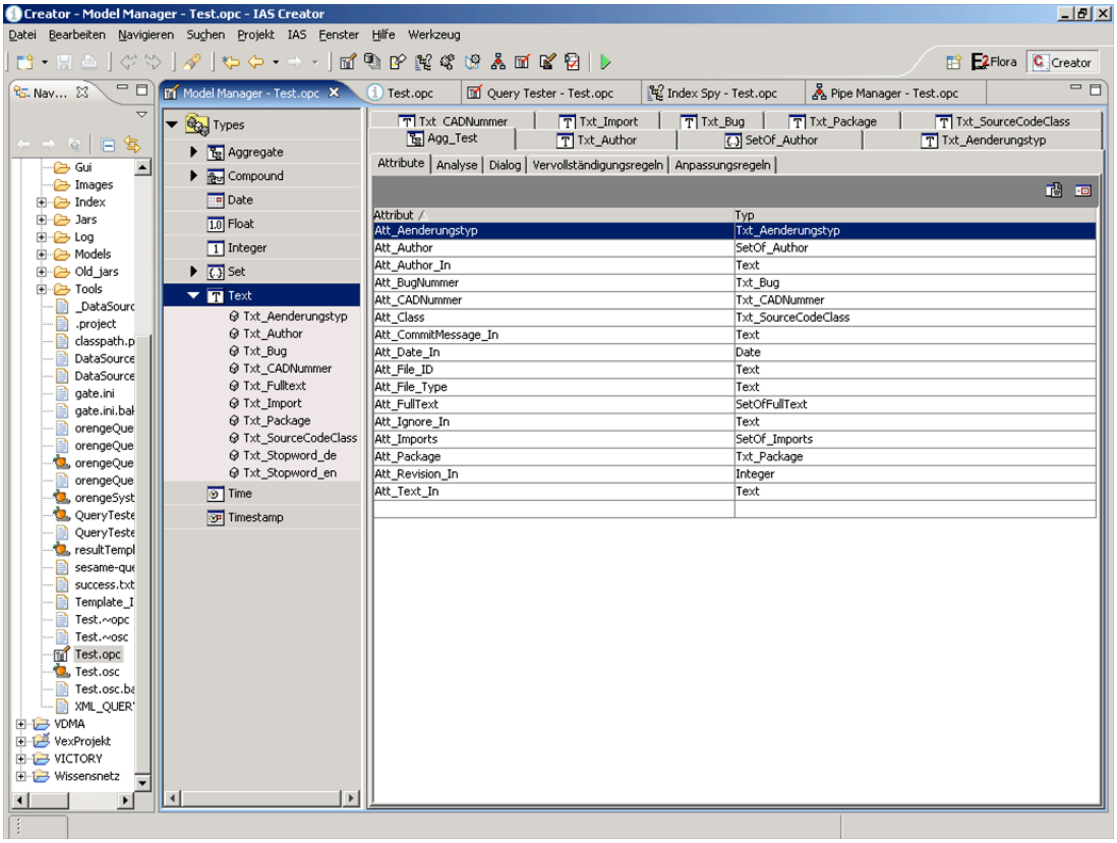


Abbildung 24: Ausschnitt des Kontextmodells im Editor

Das hier mit einigen Screenshots illustrierte Disy-Modell orientiert sich stark an den im Unternehmen vorhandenen Repräsentationen der Projekte. Dies beinhaltet die bestehenden Strukturen und den vorhandenen Sprachgebrauch. Der Mehrwert liegt in der durch die Modellierung hinzugefügte Bewertung und die Explizierung dieser Strukturen. Im besten Sinne des wohldefinierten Kontext.

Eine weitere Vertiefung des Modells hätte vielleicht auch die Verfeinerung der vorhandenen Repräsentationen der Projektdokumentation nahe gelegt. Im positiven Sinn.

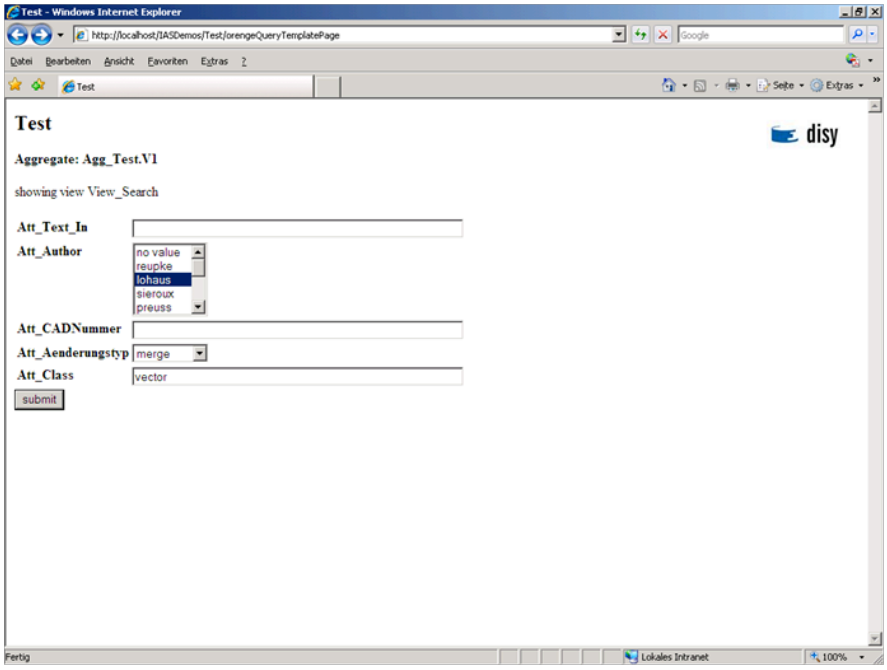


Abbildung 25: Generierte Suchmaske

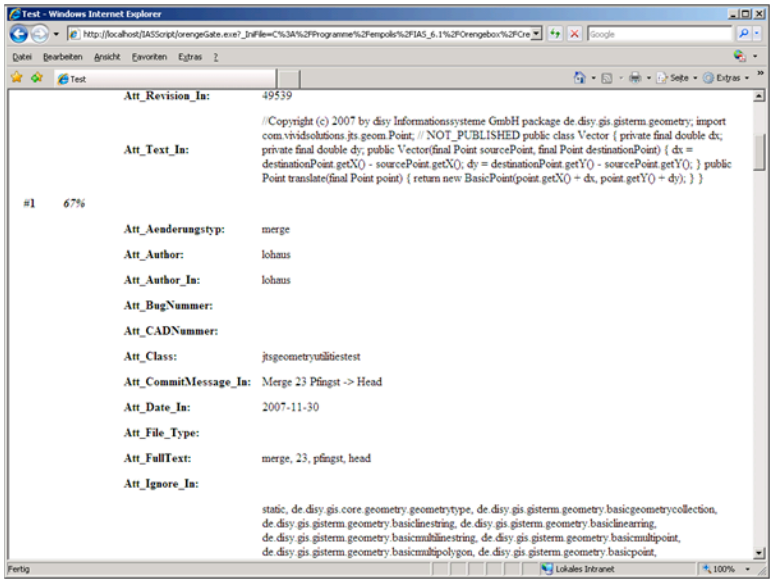


Abbildung 26: Detail eines Suchergebnisses

4.4.4 Nutzen des Kontextmodells

„Es ist schwierig zu entscheiden, was wichtig ist. Es ist ungleich leichter, das Unwichtige wegzulassen.“ Diese auf den ersten Blick vielleicht banal klingende Erkenntnis spiegelt sich auch der Art und Weise der Vorgehensweise der Disy Mitarbeiter wieder. Suche ist ein Prozess der von (gewollt) vielen Ergebnissen hin zum gewünschten Resultat führt. Anders formuliert wird erst der Recall maximiert um anschließend die Precision zu optimieren.

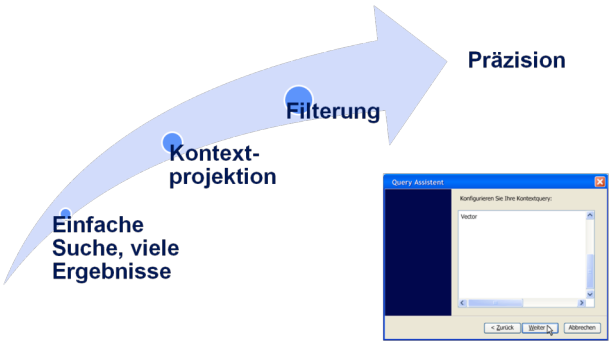


Abbildung 27: Suche ist ein Prozess

In einer Befragung bei Siemens die im Rahmen des SEKT Projektes durchgeführt wurde, haben über 95% der Teilnehmer weniger aber bessere Treffer gewünscht. Das bestätigt das Ziel der Vorgehensweise. Die Vorgehensweise selbst wird von der Technik der Suchmaschinen und unserer Wahrnehmung derselben diktiert: „Erstmal nichts verpassen und dann mal weiter gucken“. Der Suchprozess maximiert zuerst den Recall (nichts verpassen) und fokussiert dann auf die Precision (das Richtige raussuchen).

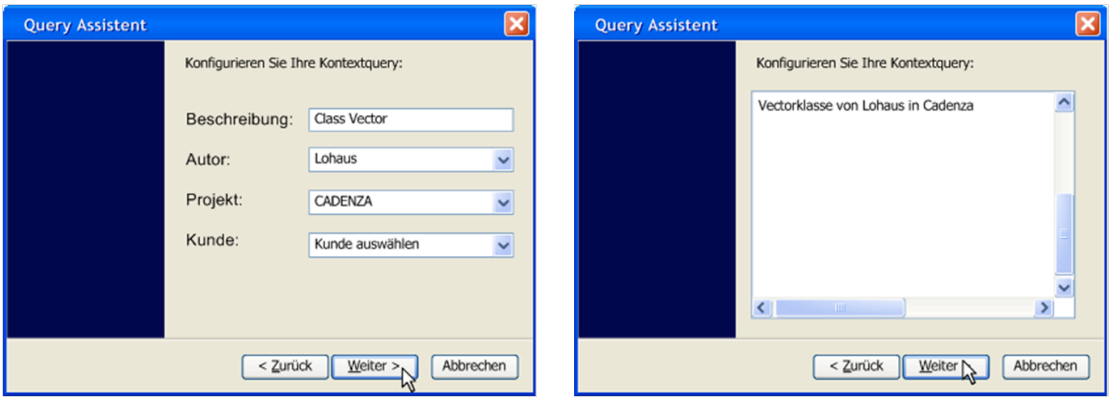


Abbildung 28: Usability - Wer braucht welche Suche?

Was auf den ersten Blick vielleicht trivial erscheint, ist uns längst nicht mehr bewusst. Genauso wie das resultierende Missverständnis zwischen sogenannten Expertensuchen und normalen Suchen. Diese sind oben links (=Experte) und rechts (=Normal) skizziert. Tatsächlich ist das Usabilityverständnis der Anwender genau umgekehrt: Experten wollen keine Struktur (links) sondern Freiheit (rechts). Laien ziehen umgekehrt ihren Nutzen.

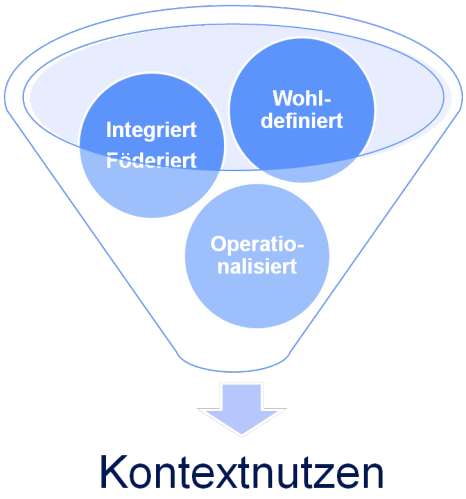


Abbildung 29: Voraussetzungen

Als Fazit kann aus WAVES abgeleitet werden, dass Kontext sinnvoll ist wenn er wohldefiniert ist. Die *Wohldefiniertheit* stellt dabei die Nützlichkeit für die Anwender sicher. Dabei sei vorausgesetzt, dass das relevante Wissen im Sinne der Anwender als Voraussetzung im System auch verfügbar ist. Das resultierende Informationssystem sollte einen *föderierten*, d.h. umfassenden Zugang zu allen Informationen auch in unterschiedlichen Quellen mit einer Anfrage bieten und in diesem Sinne operationalisiert sein. Wenigstens. Eine *Operationalisierung* auch in die Prozesse der Projektierung und Softwareentwicklung ist das eigentliche Ziel – dort wo diese Prozesse auch vorhanden sind.

4.5 Kernsystem (Arbeitspaket AP4)

4.5.1 Metadatenspeicher zum Erfassen von Wissensartefakten

Zunächst wurden die verschienden Anforderungen eines Kernsystems für so unterschiedliche Werkzeuge wie WavesIS, Wiquila und HKW gesammelt. Die Heterogenität an verschiedenen Inhaltstypen ist im Software-Engineering sehr groß. Einige Beispiele:

- Ein *Issue-Ticket* ist ein semi-strukturiertes Anwendungsobjekt und beinhaltet Statusflags, Freitext, Verknüpfung zu anderen Issues, kann aber auch Attachments enthalten.
- Ein *Tag* zum Annotieren von Resourcen besteht i.d.r nur aus einem einzelnen kurzen String.
- Eine *Wiki-Seite* besteht aus einem Seitenname, strukturiertem Text und kann ebenso binäre Attachments enthalten.
- *Quellcode* besteht aus seiner Text-Datei.

- Eine *PDF-Datei* besteht aus einem binären Teil in dem neben den Vektor- und Bitmap.-Daten strukturierte Metadaten (XMP-Standard) codiert sind, z.B. Daten über den Titel, den Autor und das Erstellungsdatum.
- Eine *Person* ist zunächst kein Inhaltstyp, wird aber repräsentiert z.B. durch einen oder mehrere Outlook-Einträge.
- Eine *Idee* ist ebenfalls ein abstractes Konzept. Sie wird z.B: repräsentiert durch eine Wiki-Seite.

Diese unterschiedlichen Inhaltstypen können auf die Grundmerkmale Text, Binär-Daten und Verknüpfungen reduziert werden. Ein Datenmodell auf Basis dieser Beobachtungen wurde unter dem Title *Semantic Web Content Model (SWCM)* publiziert (Völkel, M. (2007)). Eine Übersicht über das Datenmodell ist in Abbildung 30 zu sehen. Inhalte werden dabei in einem *Item* verwaltet, der eigentliche Inhalt befindet sich in einem *Content*-Objekt. *Triples* und *Statements* repräsentieren Verknüpfungen zwischen *Items*.

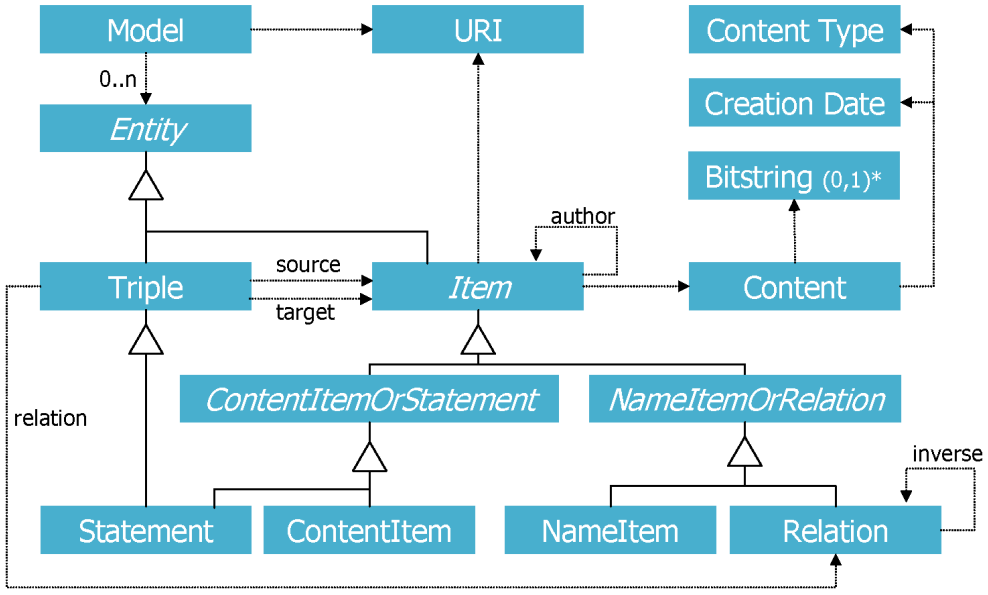


Abbildung 30: Semantic Web Content Modell

In den verschiedenen Werkzeugen konnten verschiedene Nutzungs-Aspekte identifiziert werden:

- **Suchen.**
  Werkzeuge müssen in Texten suchen können (Aspekt der Inhalte) und auch in den Verknüpfungen zwischen Inhalten (Aspekt der Struktur).
- **Persistenz.**
  Für das persistente Speichern bzw. das Replizieren von entfernten Inhalten für eine Offline-Nutzung kann wiederum in verschiedene Speicher unterschieden werden: Texte (Inhalt), Binär-Daten (Inhalt) und Verknüpfungen (Struktur). Verknüpfungen können dabei sowohl aus einem bestehenden Datenmodell folgen, z.B. bei der Replikation der Verknüpfung von einem *Issue* zu einem Projekt. Verknüpfungen können aber auch vom Nutzer frei angelegt werden (Wiquila, HKW).
- **Rechtemanagement.**
  Sowohl beim Suchen (Lesezugriff) als auch beim Speichern in einem geteilten Speicher (Schreibzugriff) können Zugriffsrechte in einer Organisation eine wichtige Rolle spielen.
- **Versionierung.**
  Jede Änderung an Daten kann prinzipiell gespeichert und versioniert werden. Eine Suche

könnte so zum Beispiel auch in „gelöschten“ Daten Resultate finden – was allerdings den Zweck des Löschens auch etwas in Frage stellt. Relevant in der Praxis ist Versionierung daher vor allem um beim kollaborativen Arbeiten an einem Datenbestand den Nutzern die Möglichkeit zu geben, versehentlich gelöschte oder überschriebene Daten wieder herzustellen. Allerdings ist bei einem Datenbestand der aus verschiedenen Quellen zusammengetragen wurde, die Versionierung von Verknüpfungen besonders schwierig. Letzlich muss entweder jede Verknüpfung einzeln als eigene Entität verwaltet und versioniert werden oder das Datenmodell als ganzes.

Verschiedene Werkzeuge haben verschiedene Anfoderungen. So benötigen Suchwerkzeuge bsp. vor allem Volltext-Suche und Verknüpfungs-Suche, Autoren-Werkzeuge hingegen vor allem Persistenz für Texte, Binärdaten und Verknüpfungen. Oft beinhalten Autoren-Werkzeuge auch Suchwerkzeuge, jedoch selten umgekehrt.

Das vorgestellte Datenmodell SWCM wurde in Java implementiert und ist als *Semantic Web Content Repository (SWECR)* als open-source[5] veröffentlicht. Die zentralen Konzepte von SWECR sind hierbei:

- Unterstützung unterschiedlicher Granularitätsstufen der Wissensartefakte, angefangen bei einzelnen Wörtern (wie z.B. ein Wiki-Name) bis hin zu ganzen Dokumenten (z.B. Wiki-Seite). Jedes Artefakt wird eindeutig durch eine URI identifiziert. Eine (Wiki-)Seite besteht dann nicht mehr aus einem monolithischen Text, sondern wird durch eine baumartige Struktur aus URIs beschrieben, welche ihrerseits den dem Anwender zu präsentierenden Inhalt beschreiben.
- Alle Artefakte können annotiert werden, d.h. insbesondere auch einzelne Wörter, Teile einer Seite, oder aber ganze Seiten und (binäre) Dokumente.
- Für alle Artefakte können semantische Aussagen festgelegt werden.

SWECR benutzt intern die Komponenten *binstore* (Eigenentwicklung, da bestehende Binärspeicher in Java keinen wahlfreien Lese- und Schreibzugriff ermöglichen) und einen RDF-Datenspeicher (*OpenRDF Sesame* von *Aduna Software*, open source, angebunden über *RDF2Go*[6]).

4.5.2 WISE: Die WAVES-Ontologie (Arbeitspaket AP2)

WISE ist eine Ontologie für Wissensmanagement in der Software-Entwicklung. WISE steht für ***W****AVES ontology for knowledge management* ***i****n* ***s****oftware* ***e****ngineering*. Die Ziele bei der Entwicklung von WISE waren:

- Nutzbar für den Wissensaustausch in und zwischen Organisationen
- Informationsintegration zwischen Werkzeugen
- Informationsintegration zwischen Menschen (mehrere persönlich genutzte IT-Systeme wachsen zusammen) und Organisationen (mehrere organisationale IT-Systeme wachen zusammen)
- Bessere Suchmöglichkeiten, z.B. durch Verwenden von semantischen Konzept-Hierarchien
- Klein (daher geringer Lernaufwand, leicht zu implementieren)
- Verständlich
- Erweiterbar

[5] Siehe http://www.swecr.org

[6] http://rdf2go.semweb4j.org

Auf den ersten Blick scheint Softwareentwicklung (SE) sich vor allem mit der Software selbst zu beschäftigen, z.B. mit Quellcode oder Modellen. Desweiteren sind im SE zu betrachten: Entwicklungsaktivitäten unterstützende Werkzeugen wie z.B: eine unternehmensweite Suchmaschine, verschiedene organisationale Projekte, den jeweiligen Entwickler-Desktops, Issue-Tickets, persönliche Notizen, Blogs und Foren in denen Entwickler sich gegenseitig helfen. Das WISE-Modell konzentriert sich auf den Prozess der Transformation von vagen Ideen über Dokumente bis hin zu ausführbaren Sofware-Artefakten. Ein typischer Prozess hat z.B. folgende Schritte:

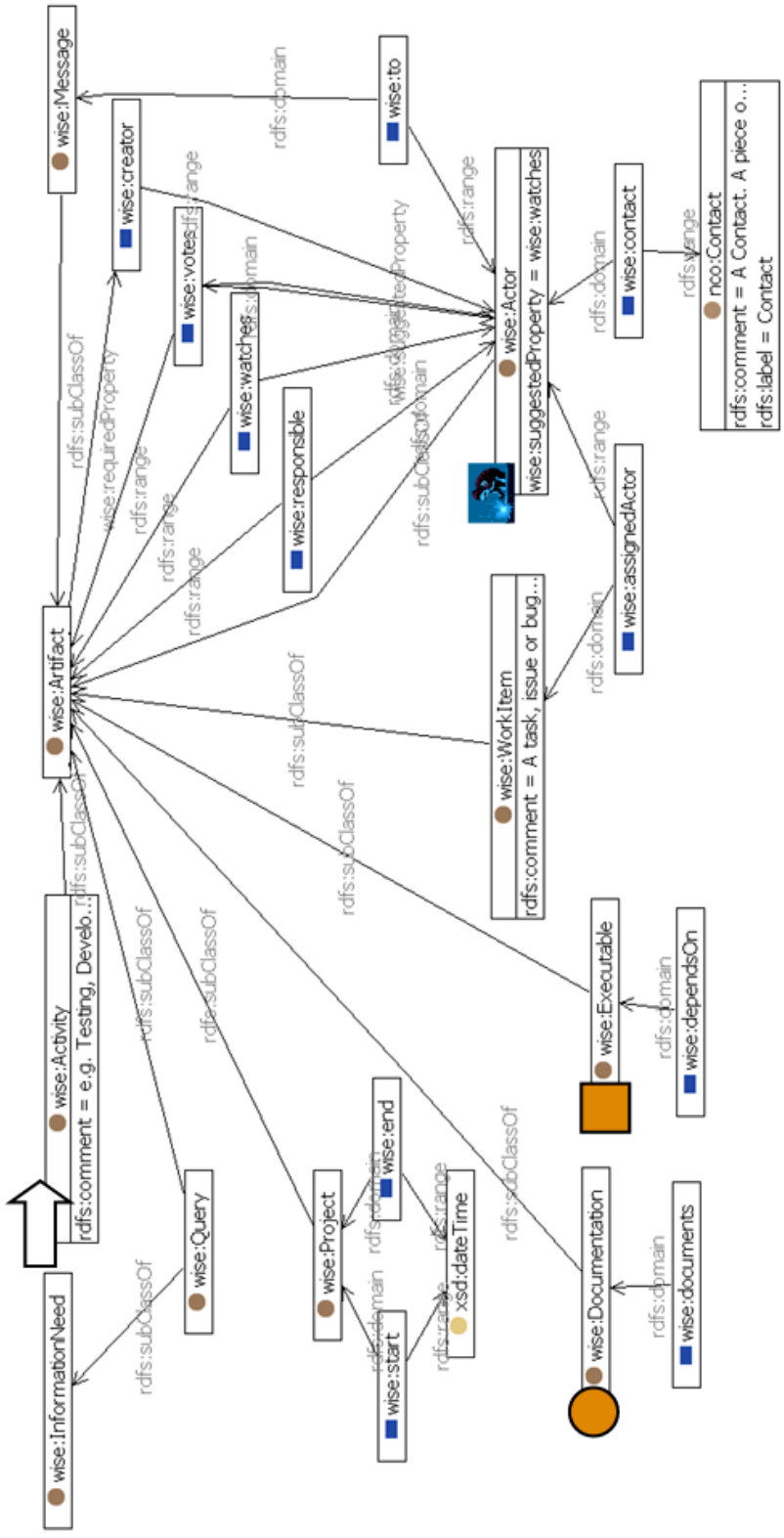
- Jemand dokumentiert eine Idee
- Das Ideen-Dokument wird überarbeitet
- Anforderungen werden prototypische implementiert, gestetet und verfeinert
- Das fertige Software-Artefakt wird ausgeliefert und eingesetzt
- Fehler oder Erweiterungswünsche kommen auf und werden ebenfalls dokumentiert

Der SE-Prozess verwendet also eine große Bandbreite von verschiedenen Dokumentations-Artefakten. Dieser SE-Prozess kann auch rekursiv auf das Feld SE angewandt werden: Entwickler erstellen neue Entwicklungswerkzeuge, dokumentieren Arbeitsprozesse oder erstellen ein neues Informationssystem mit dem sich eine spezielle Art von Informationen wiederum besser managen lässt.

Für die konkrete Entwicklung von WISE wurde eine Reihe von Quellen verwendet:

- Ein allgemeines Metamodell für objektorienterte Software, welches initial im *KISSy/SISSy*-Projekt am FZI Karlsruhe entwickelt wurde. Es wird u.a. zur Berechnung von Qualitätsmetriken verwendet (Anderer, J., Bloch, R., Mohaupt, T., Neumann, R., Schumacher, A., Seng, O., Simon, F., Trifu, A., Trifu, M. (2006)).
- Ein von *Empolis* entwickeltes Kontextmodell. Es basiert auf dem Deutschen V-Modell.
- Eine Analyse der in WavesIS/TeamWeaver indizierten strukturierten Artefakte
- Bestehende Metadaten-Formate zur Beschreibung von SE-Projekten wie z.B. DOAP (http://usefulinc.com/ns/doap) oder das interne Polarion-Modell (http://polarion.org). Das Polarion XML-Schema wurde als Kern von WISE verwendet.
- Im Projekt NEPOMUK (http://nepomuk.semanticdesktop.org) geht es um den Semantischen Desktop. Im Projekt wurde eine Reihe von zentralen Desktop-Ontologien erstellt womit z.B. Outlook-Einträge, Emails oder Dateien repräsentiert werden können.
- Foren und Wiki-Systeme werden mit der SIOC-Ontologie (http://www.sioc-project.org) beschrieben.

Abbildung 31 zeigt einen Überblick über die zentralen Konzepte (classes) und Relationen (properties) zwischen Konzepte. Die wichtigsten Konzepte sind *Actor* (führen Prozesse aus), *Activity* (Prozesse) und *Artifact* (jegliche Entität die in einem SE-Prozess benutzt, umgewandelt oder erzeugt wird).



Überlick über die WISE-Ontologie

Das Klassenmodell ist in der Abbiuldung zu finden. Die zentralen Klassen in WISE sind:

- *Activity (Class)*: Eine Aktivität in der Software-Entwicklung, z.B:
  - *Testing*,

  - *Developing*,
  - *Deploying*,
  - *Coding*,
  - *Refactoring*, oder
  - *Documenting*

- *Artifact (Class)*: Jede Entität in WISE ist ein *Artifact*.
- *Actor (Class):* Jede Aktivität wird von einem *Actor* durchgeführt. Ein *Actor* ist nich notwendigerweise eine a Person, es kann sich z. B. ebenfalls um ein Team oder einen Sub-Unternehmer handeln.
- *Documentation (Class):* Dokumentation ist jeder Artifact, der nicht ausführbar ist.
- *Executable (Class):* Eine ausführbare Entität kann in einer geeigneten Laufzeitumgebung ausgeführt werden. Beispiele für *Executables* und dazugehörige Laufzeitumgebuungen sind:
  - Java class file - Java Virtual Machine
  - Windows Vista application - Windows Vista operating system
  - BPEL model - BPEL engine:
- *Tool (Class)* Werkzeuge sind ausführbare Artefakte die dabei helfen, andere Artefakte zu transformieren
- *Project (Class)*: Ein Projekt ist eine Arbeitseinheit in WISE.
- *Query (Class)*: Eine Anfrage drückt ein abstraktes Informationsbedürfnis aus. Anfragen können Volltext-Suchen oder z.B. SQL-Anfragen sein.

Weitere Klassen in WISE sind:

- *Architecture (Class):* Eine spezielle Sorte von *Model*, spezifiziert eine Software- oder IT-Architektur.
- *Diagram (Class):* Ein Untertyp von *Documentation*. In der Praxis werden z.B. UML-Digramme eingesetzt.
- *Information System (Class):* Ein ausführbarer Artefakt wie z.B. ein Issue-Tracker, Reisebuchungssystem oder Quellcode-Versionskontrollsystem.
- *Model (Class):* Nicht-ausführbare Modelle mit Dokumentations-Character. Ausführbare Modelle (z.B. *executable UML*) sollte als Untertyp von *Executable* modelliert werden.
- *Note (Class):* Jegliche Art von unstrukturierter Dokumentation.
- *Runtime Environment (Class):* Eine Laufzeutumgebung wie z.B. eine *Java Virtual Machine*.
- *Software Class (Class):* Eine kompilierte Klasse, z.B. eine *.class*-Datei in Java.
- *Specification (Class):* Eine Untertyp von *Documentation*.
- *Test (Class):* Tests in WISE sind ausführbare Artefakte.
- *Test Result (Class):* Das Ergebnis eines Tests.
- *Information Need (Class):* Keine Ontologie für Wissensmanagement wäre komplett, wenn sie nur das modellieren würde, was vorhanden ist. Es ist ebenso wichtig die Informationsbedürfnisse von Akteuren zu erfassen und zu modellieren.
- *Message (Class):* Jeder Artefakt der von einem *Actor* zu enem anderen geschickt wird.
- *Test Specification (Class):* Eine *test specification* existiert u.U. noch bevor es den Test selbst gibt. Sie ist konkreter als eine allgemeine Note aber noch nicht ausführbar, wie ein *Test*.

- *Work Item (Class):* Modelliert einen *Issue* in einem *Issue-Tracker* oder ein *Task* in einem Projektverwaltungswerkzeug.

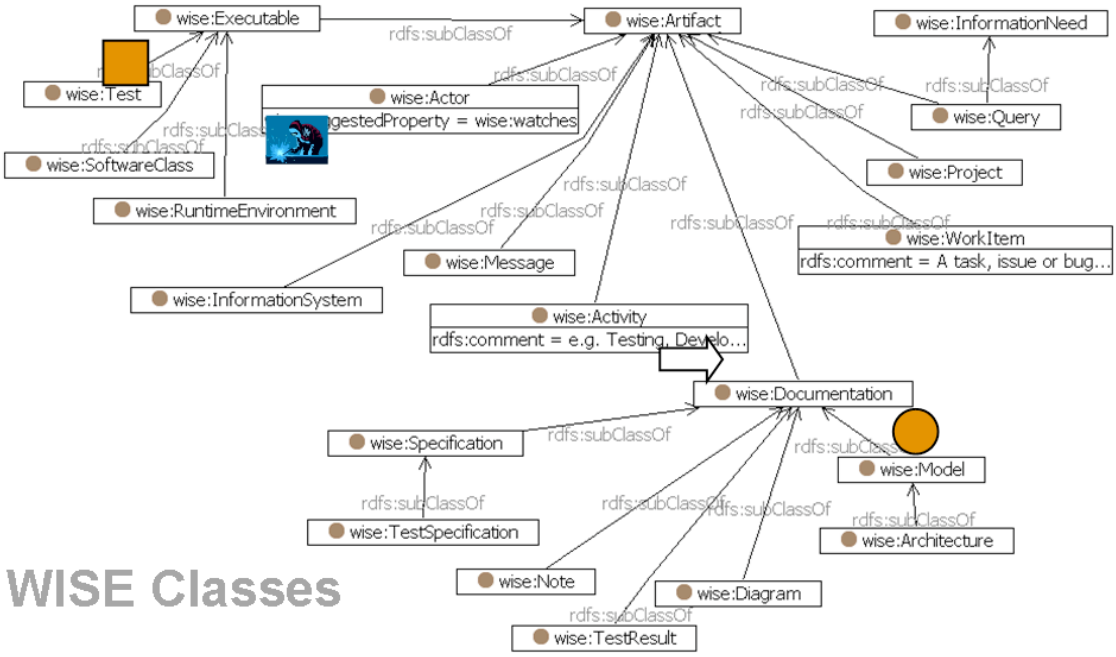


Abbildung 31: WISE Klassenmodell

Das Relationsmodell ist in Abbildung 32 zusammengefasst. Die zentralen Relationen in WISE sind:

- *Contact (Property)*: Das minimale Wissens, welches ein *Actor* über einen anderen benötigt ist eine Kontaktmöglichkeit.
- *Created (Property):* Der Erstellungszeitpunkt eines Artafaktes.
- *Creator (Property):* Der Ersteller eines Artfaktes. Dieser kann im Bedarfsfall kontaktiert werden.
- *Depends On (Property):* Der Kern des Abhängigkeiten-Managements ist dieMöglichkeit Abhängigkeiten zwischen Artefakten (Klassen, Projekten, Dokumenten, Fehlerbeschreibungen, Aufgaben) formal zu modellieren. So können Abhängigkeiten analysiert und dargstellt werden.
- *Documents (Property):* Verbindet asuführbare Artfakte mit ihren Dokumentations-Artefakten.
- *Start (Property):* Jedes Projekt hat einen Startzeitpunkt.
- *End (Property):* Jedes Projekt hat ein Ende.
- *Previous Version (Property):* Als leichtgewichtige Möglichkeit zur Versionierung zeigt diese Relation jeweils von einem Artefakt auf seinen zeitlichen Vorgänger, sofern einer vorhanden ist.
- *Responsible (Property):* Für jeden Artefakt kann ein *Actor* zuständig sein.
- *Assigned Actor (Property*): Der beauftrage *Actor* zur Abbarbeitung einer Aufgabe.
- *To (Property):* Wie bei einer Email beschreibt diese Relation an wen eine Message geschickt werden soll.
- *Votes (Property):* Diese Relation drückt aus, dass ein *Actor* ein Interesse an einem Artefakt hat. Dies wird typischerweise für Fehlermeldungen verwendet.

- *Watches (Property):* Der *Actor* fordert über diese Relation eine Benachrichtigung an, falls der beobachtete Artefakt sich verändert.

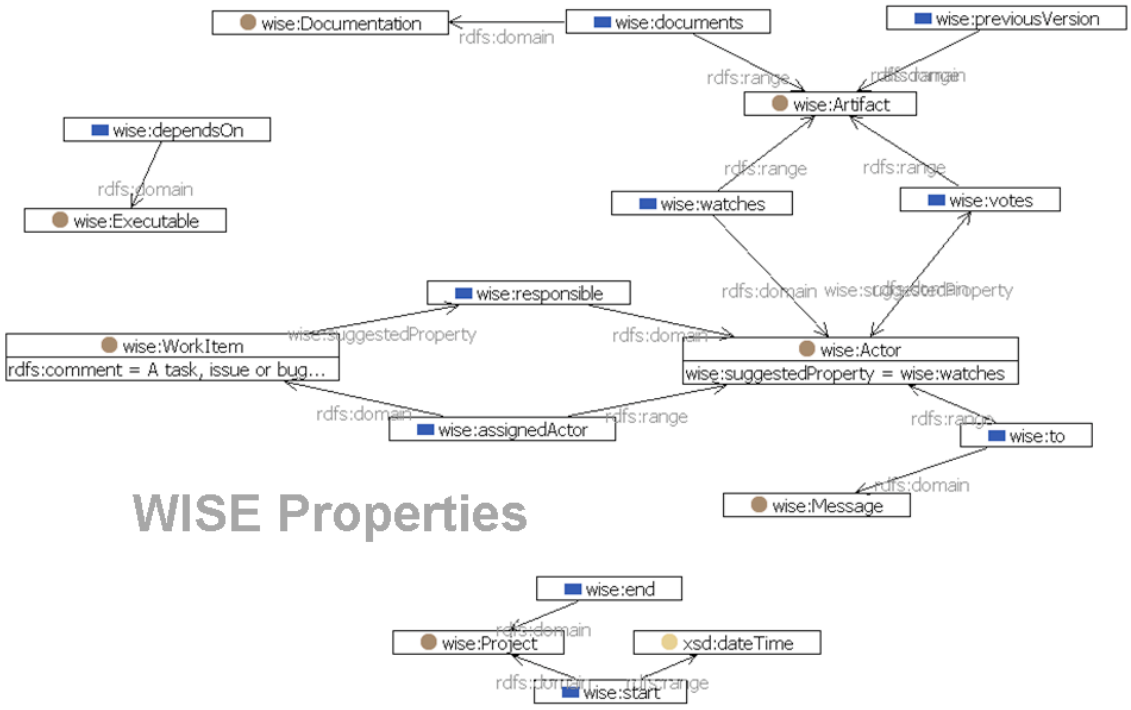
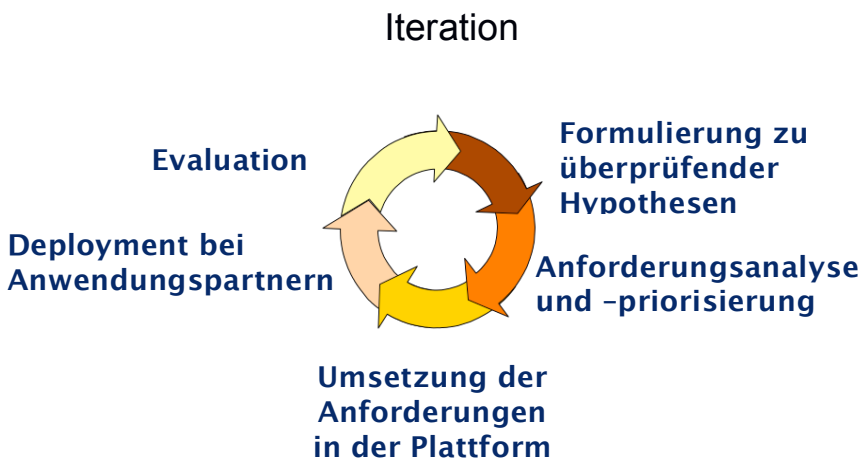


Abbildung 32: WISE Relationsmodell

5. Evaluierung (Arbeitspakete AP1/AP3)

Im Rahmen des Projektes wurden die entwickelten Releases von WAVES kontinuierlich (für jede Iteration) bei den Anwendungspartnern evaluiert. Diese kontinuierliche Evaluation ermöglicht eine iterative Fortschreibung der Anforderungen, eine kontinuierliche Verbesserung des Systems durch regelmäßiges Feedback der Endanwender und eine Bewertung des Nutzens von WAVES. Die nachstehende Grafik zeigt den Ablauf einer Iteration.



In den folgenden Abschnitten wird das Vorgehen bei der Evaluation, die Ergebnisse der Evaluation, sowie die partnerspezifischen Ergebnisse für die einzelnen Anwendungspartner vorgestellt.

5.1 Vorgehensweise bei der Evaluation

Ziel der Evaluation war die kontinuierliche Fortschreibung der Anforderungen, sowie die Bewertung des Nutzens des Systems. Dafür wurden für jedes Release Studien mit den Endanwendern in den Firmen vor Ort durchgeführt.

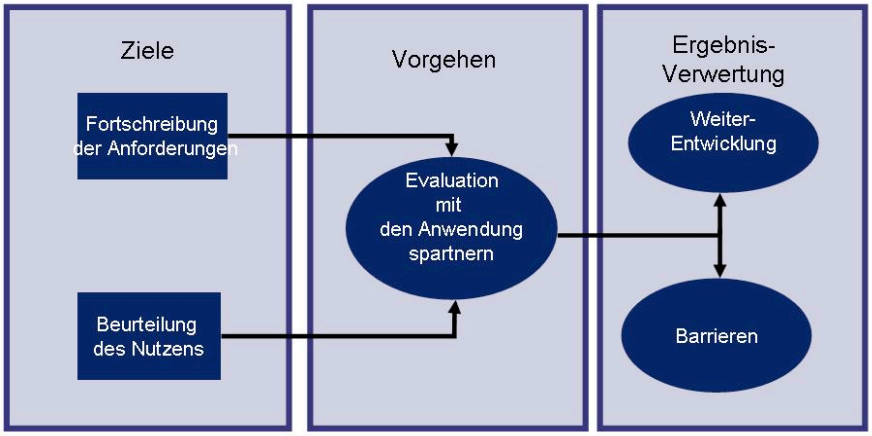
Die Fortschreibung der Anforderungen umfasste folgende Teilaspekte:

- Erfassen von Verbesserungsvorschlägen zur Benutzbarkeit
- Verbesserungen oder Erweiterungen der Funktionalität
- Diskussion und Beurteilung zu den nächsten zu entwickelten Features

Die Bewertung des Nutzens spaltet sich in die folgenden Teilaspekte auf:

- Identifikation von Einsatzszenarien von WAVES in den Unternehmen
- Identifikation von Barrieren (ökonomische, psychologische, technische) und deren Überwindung durch die WAVES Platform

Um diese Ziele zu erreichen, wurden die einzelnen Releases von WAVES jeweils bei allen Anwendungspartnern ausgerollt und von den entsprechenden Endanwendern getestet. Je nach Stand des Systems konnten unterschiedliche Methodiken zur Evaluation eingesetzt werden. Ziel war es immer, den Anwendern ein möglichst realistisches arbeiten mit dem System zu gewährleisten.



Dies bedeutet, dass WAVES bei den Firmen ausgerollt werden musste, um auf die entsprechenden Firmendaten (zum Beispiel für die integrierte Suche) zugreifen zu können.

Im Folgenden werden die einzelnen Maßnahmen zur Evaluation beschrieben, die im Rahmen des Projektes angewendet wurden.

5.1.1 Durchführung von Workshops

Für jedes Release wurde zuerst ein Workshop bei den Endanwendern durchgeführt. Ein Workshop bietet die Möglichkeit einer kleineren Anzahl von Endanwendern das System vorzustellen und es anschließend auf unterschiedliche Weise testen zu lassen. Dabei kann die Akzeptanz der Endanwender dem System gegenüber getestet werden, bevor das System im laufenden Betrieb eingesetzt wird. Dies kann entscheidend sein, da die Erfahrung aus anderen Projekten zeigt, dass für den Fall, dass Anwender, die ein nicht ausgereiftes Werkzeug direkt im laufenden Betrieb einsetzen sollen und damit schlechte Erfahrungen machen, zu einem späteren Zeitpunkt nur wieder sehr schwer davon zu überzeugen sind, das Werkzeug noch ein weiteres mal einzusetzen.

Um die Tests allerdings so realistisch wie möglich zu gestalten, muss das zu testende System idealerweise in die Toollandschaft der Unternehmen integriert werden. Dies führt dazu, dass den Endanwender der Zugriff auf ihre Firmendaten gewährleistet wird.

Folgende Schritte wurden im Rahmen der Workshops durchgeführt:

1) Vorbereitungsphase und Inbetriebnahme

- Bereitstellung der Hardware
- Installation und Inbetriebnahme des Systems
- Anbindung an die realen Daten der Anwendungspartner für realistische Anwendungen während der Tests
- Auswahl der Endanwender (ca. 5 Mitarbeiter pro Anwendungspartner), die das System testen

2) Vorstellung des Systems und der geplanten Erweiterungen bei den Unternehmen vor Ort

- Vorstellung der Funktionalität und Erläuterung
- Erläuterung der möglichen Erweiterungen
- Diskussion über die möglichen Erweiterungen

3) Vorstellung des Vorgehens während der Tests

- Die unterschiedlichen Formen der Test werden im nachfolgenden Text vorgestellt

4) Durchführung der Tests

- Die Tests werden von den Endanwendern einzeln durchgeführt

5) Interview zur Benutzbarkeit und Funktionalität

6) Auswertung der erhobenen Daten

Nach der (1) *Vorbereitungs- und Installationsphase* wird die WAVES-Platform einer kleinen Gruppe von Endanwendern bei jedem Anwendungspartner vorgestellt. Zusätzlich zu der Vorstellung des Systems werden ebenfalls geplante Erweiterungen vorgestellt und diskutiert ((2)*Vorstellung des Systems und der geplanten Erweiterungen*). Dies ermöglicht ein direktes Feedback zu den möglichen Weiterentwicklungen.

Anschließend werden den Endanwendern das Vorgehen während der Tests vorgestellt *((3)Vorstellung des Vorgehens während der Tests)*. Dabei wurden folgende Techniken zum Testen des Systems verwendet.

- **Einsatz von benutzerspezifischen Szenarien**

  Ein effektives Mittel, um Systeme zu testen, ist der Einsatz von benutzerspezifischen Szenarien. Dabei werden typische Abläufe aus dem realen Arbeitsvorgehen definiert und anschließend an dem Testsystem nachgestellt. Somit ist gut zu erkennen, welche Szenarien das System abdeckt, welche nicht und ob das Systeme die Vorgänge adäquat unterstützt. Im Rahmen von WAVES wurden die Szenarien grob vorgegeben, die konkrete Ausprägung der Szenarien war allerdings den Endanwendern überlassen. Bei dem Test der Integrierten Suche wurden beispielsweise folgende Rahmenbedingungen für die Szenarien vorgegeben:

  1. Szenario: Suche in einem aktuellen Projekt
  Sie suchen etwas aus einem Projekt, an dem Sie aktuell arbeiten und in dem Sie sich gut auskennen. Sie wissen, welche Ergebnisse Sie erwarten und können beurteilen, welche Ergebnisse wichtig/nicht wichtig sind.

  2. Szenario: Suche in einem abgeschlossenen Projekt
  Sie benötigen Informationen aus einem Projekt, das bereits abgeschlossen ist, an dem Sie aber mitgearbeitet haben. Sie kennen sich in diesem Projekt aus, können sich aber nicht mehr an alle Einzelheiten erinnern. Sie wissen welche Ergebnisse angezeigt werden müssten, erwarten allerdings dass es weitere relevante Ergebnisse geben müsste. Sie können die Ergebnisse ungefähr nach ihrer Relevanz beurteilen.

  3. Szenario: Suche in einem unbekannten Projekt
  1) Sie wollen sich in ein laufendes Projekt einarbeiten. Sie wissen bislang kaum etwas über dieses Projekt.
  2) Sie suchen etwas in einem abgeschlossenen Projekt, an dem Sie nicht mitgearbeitet haben.

- **Think aloud**

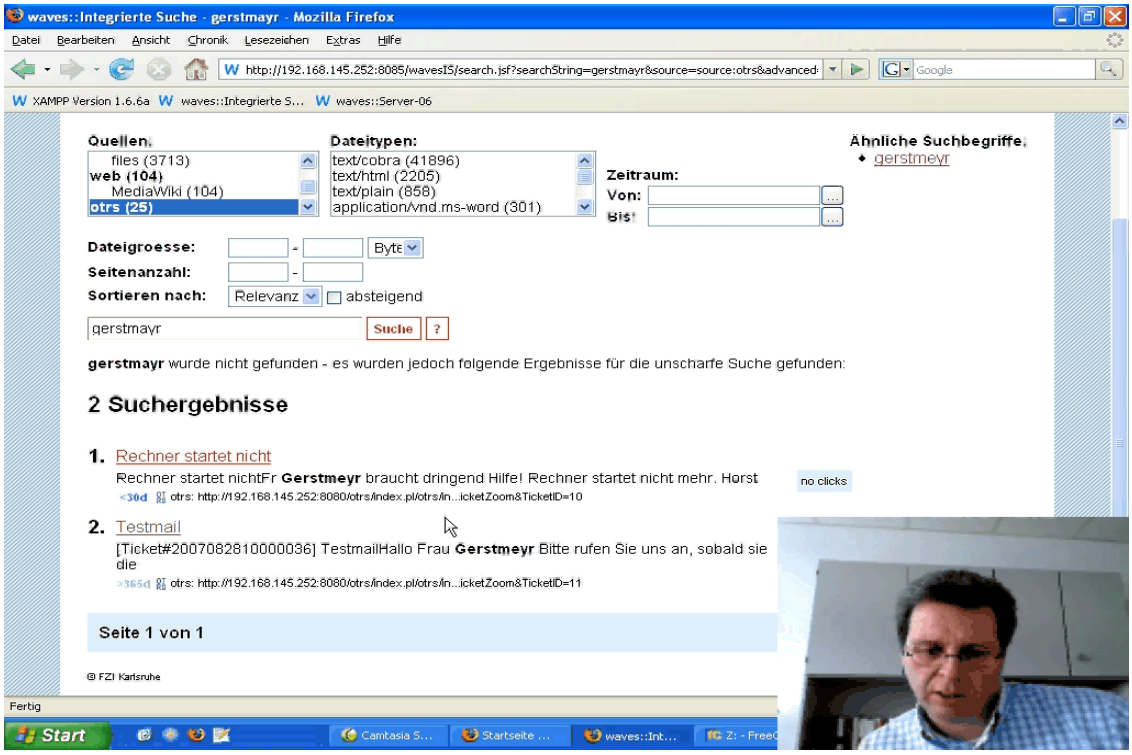
  Bevor die Endanwender mit dem Test beginnen, wird ihnen die Technik des Think aloud erklärt. Hierbei sollen die Benutzer während der Durchführung der Szenarien unter anderem verbalisieren, was sie gerade machen, was gut funktioniert, was sie vom System erwarten und welche Verbesserungsvorschläge sie haben.

  Dies ermöglicht einen besseren Einblick in die Probleme und die Vorteile des Systems, die während der Benutzung auftreten.

- **Videoaufzeichnung**

  Während der Durchführung der Testszenarien wurden die Benutzer in Bild und Ton mittels der aufzeichnungssoftware Camtasia (www.camtasia.de) aufgezeichnet. Zusätzlich wurde auch der Desktop aufgenommen, die untenstehende Abbildung zeigt einen Screenshot aus einer

Videoaufzeichnung. Diese Aufzeichnung ermöglicht eine nachträgliche detaillierte und qualitative Auswertung des Vorgehens der Benutzer und der Kommentare der Endanwender.



Screenshot einer Videoaufzeichnung

In dem Schritt 4 bearbeiten die Endanwender einzeln die Testszenarien((4) *Durchführung der Testszenarien*). Nach den Testszenarien wurde mit den Endanwender noch ein Interview durchgeführt, in dem allgemeine Informationen zum System, dessen Benutzbarkeit, der Funktionalität, den möglichen Einsatzszenarien, sowie dem erwarteten Nutzen abgefragt wurden *((5)Interview zur Benutzbarkeit und Funktionalität)*. Die Auswertung der Daten *((6) Auswertung)* erfolgt qualitativ anhand des aufgezeichneten Videomaterials und des Interviews. Zusätzlich wird die Diskussion über die mögliche Erweiterung ausgewertet. Aufgrund der einzelnen Beobachtung und der Interviews liefert die Studie ein detailliertes Feedback der Endanwender.

Obwohl diese Form der Anwenderstudie nicht die Vorteile des Einsatzes im laufenden Betrieb abdecken kann, sind die Szenarien realistisch und die Ergebnisse für die Weiterentwicklung entscheidend, da sie von den potenziellen Endanwendern auf realen Daten ausgeführt werden. Des Weiteren vermeidet diese Form der Studie die bereits beschriebenen Probleme eines zu frühen Einsatzes in großem Umfang, aufgrund folgender Gegebenheiten:

- Den Endanwendern wird das System erstmal nur zum Testen vorgeführt.
- Aufgrund der einzelnen Beobachtung und der Interviews liefert die Studie ein detailliertes Feedback der Endanwender.
- Auftretende Probleme können vor einem Einsatz in größerem Umfang behoben werden und somit Akzeptanzprobleme umgangen werden.

5.1.2 Einsatz im laufenden Betrieb

Durch eine Studie in einem laufenden Projekt wären folgende Vorteile gewährleistet:

- Realistische Anfragen an das Suchsystem
- Realistische Benutzer

- Einsatz über „längeren“ Zeitraum (je nach Anzahl der Benutzer bzw. Größe des Projektes)
- Erfassung der Einsatzhäufigkeit
- Identifikation von Schwächen/ Stärken der WAVES-Plattform

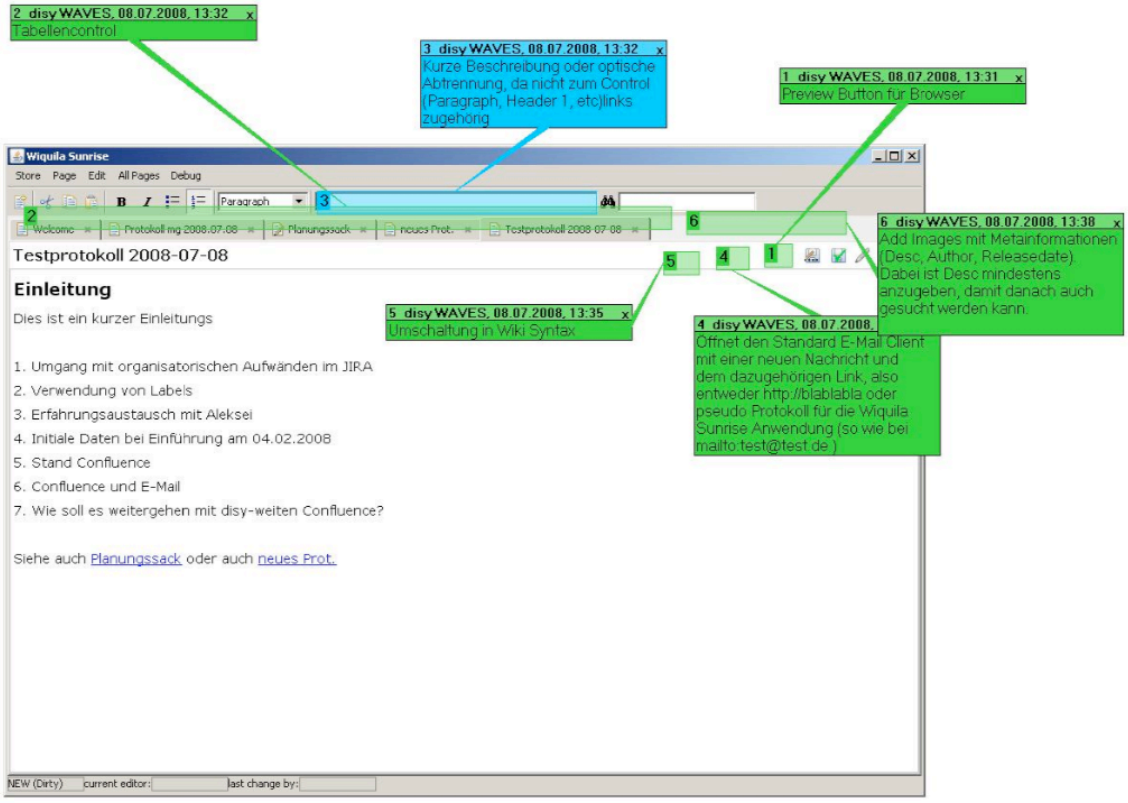
Nach der Durchführung eines erfolgreichen Workshops und der Einarbeitung der Verbesserungen in das nächste Release, wurde im Rahmen des Projektes Kissy und wavesIS im laufenden Betrieb getestet. Der Einsatz im laufenden Betrieb durchläuft ebenfalls die Schritte 1 und 2 des Workshops. Anschließend wird das System allerdings produktiv über einen längeren Zeitraum in den Unternehmen von ausgewählten Mitarbeitern getestet. Der produktive Einsatz von Kissy und wavesIS lief über 2 Monate bei allen Anwendungspartnern, dabei nahmen jeweils 4-5 Mitarbeiter eines jeden Unternehmens an dem Testlauf teil. Entscheidend ist, dass die kritische Menge an Datenquellen, die mittels wavesIS bzw. kissy durchsucht werden können, erreicht werden muss, damit die Mitarbeiter das System auch produktiv nutzen können. Diese Menge wurde gemeinsam mit den Anwendungspartnern und den Mitarbeitern festgelegt. Während des produktiven Einsatzes wurden Daten mittels Interview erhoben. Die Interviews fanden alle zwei Wochen telefonisch statt. Ziel der Interviews war es, die Anforderungen fortzuschreiben und den subjektiven Nutzen der Endanwender und die Erfahrungen mit dem Umgang der Plattform zu ermitteln.

5.1.3 Einsatz von Open Proposal

Für die Evaluation wurde unter anderem auch das Tool OpenProposal eingesetzt, welches im Folgenden kurz vorgestellt wird. OpenProposal (www.openproposal.de) wurde ursprünglich im Rahmen des Projektes CollaBaWü des Forschungsverbunds PRIMIUM entwickelt, das durch das Ministerium für Wissenschaft, Forschung und Kunst des Landes Baden-Württemberg gefördert wurde.

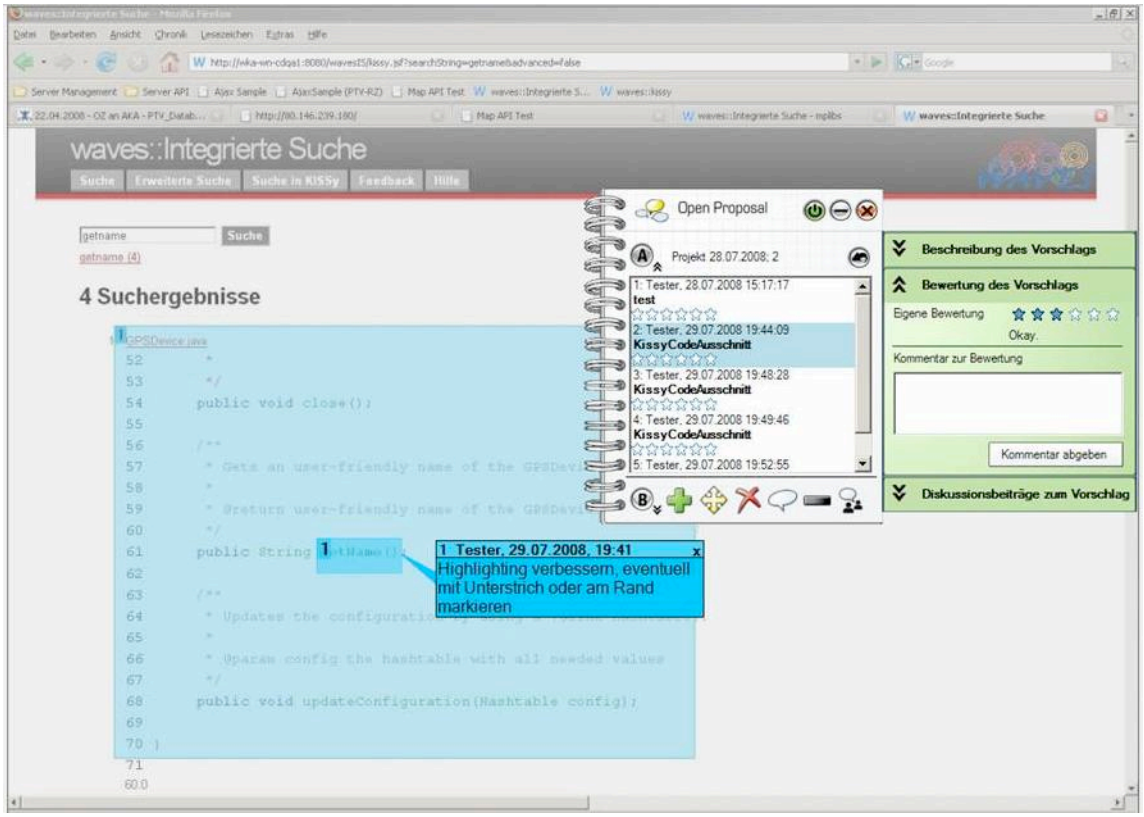
Das Software-Werkzeug OpenProposal ermöglicht nun eine zeitsparende und unkomplizierte Beteiligung der späteren Kunden am Software-Entwicklungsprozess. Es dient als „Übersetzer zwischen Entwickler und Anwender“. Künftige Anwender können sich auf dem Oberflächenentwurf des späteren Programms (dem so genannten Mock-Up), eine grafische Werkzeugleiste einblenden lassen. Mit den Tools dieser Werkzeugleiste beschreiben sie ihre Ideen und Vorstellungen zur Verbesserung der Anwendung direkt auf dem Oberflächenentwurf. Assistenten werden eingeblendet, die sie dabei unterstützen, ihre Vorstellungen und Wünsche so zu formulieren bzw. zu visualisieren, dass die Softwareentwickler sie später eindeutig interpretieren und zielgenau umsetzen können. Die Anregungen (Annotationen) werden auf der Open-Proposal Kollaborationsplattform gespeichert und stehen damit zeitnah allen interessierten Entwicklern und Anwendern gemeinsam zur Verfügung.

In der untenstehenden Abbildung ist beispielhaft ein Screenshot von OpenProposal dargestellt, die bei der Evaluation von Wiquila bei dem Anwendungspartner disy erstellt wurde. Die graphische Oberfläche von Wiquila konnte dabei während dessen Nutzung durch Kommentare, und Verbesserungswünsche aufgezeichnet werden. Diese Anmerkungen können mit der Hilfe von OpenProposal automatisch in das von unserem Projekt verwendeten Issue-Tracker-Werkzeug JIRA (www.jira.com) eingepflegt werden, das für eine verfolgbare Behandlung der Anmerkung auf der Entwicklerseite sorgt.



Das Werkzeug OpenProposal bei der Evaluation von Wiquila

Des Weiteren verfügt OpenProposal über einen Diskussionsmodus, der es ermöglicht die Vorschläge anderer zu diskutieren und zu bewerten. Dies ist speziell in einem verteilten Szenario wie im Projekt WAVES sehr hilfreich. Die Anwendungspartner können so leicht die Vorschläge der anderen Partner einsehen und bewerten.



Diskussionsmodus OpenProposal am Beispiel eines Verbesserungsvorschlages von KISSy

Die bisherigen Erfahrungen über den Einsatz von OpenProposal sind sowohl auf der Entwicklerseite als auch bei den Anwendungspartnern sehr überzeugend. Für die Evaluation wurde OpenProposal in drei verschiedenen Szenarien eingesetzt:

- Im Rahmen der Workshops wurde OpenProposal von den Mitarbeitern für die Formulierung von Verbesserungs-, Erweiterungsvorschlägen genutzt. Dabei kam OpenProposal schwerpunktmäßig bei der Formulierung von Verbesserungen bzgl. der Oberfläche zum Einsatz.
- Im laufenden Betrieb steht OpenProposal ebenfalls den Mitarbeitern zur Verfügung, um Vorschläge direkt während des Einsatzes zu formulieren.
- Ein wichtiger Teil bei der Auswertung der Ergebnisse ist die Priorisierung. Der Diskussionsmodus von OpenProposal ermöglicht, dass die Partner die Vorschläge der anderen einsehen, diskutieren und priorisieren können. Dafür werden die Vorschläge zuerst in der Analyse sortiert bzw. redundante Vorschläge gelöscht, anschließend erfolgt die Diskussion durch die Partner.

5.2 Ergebnisse der Studien

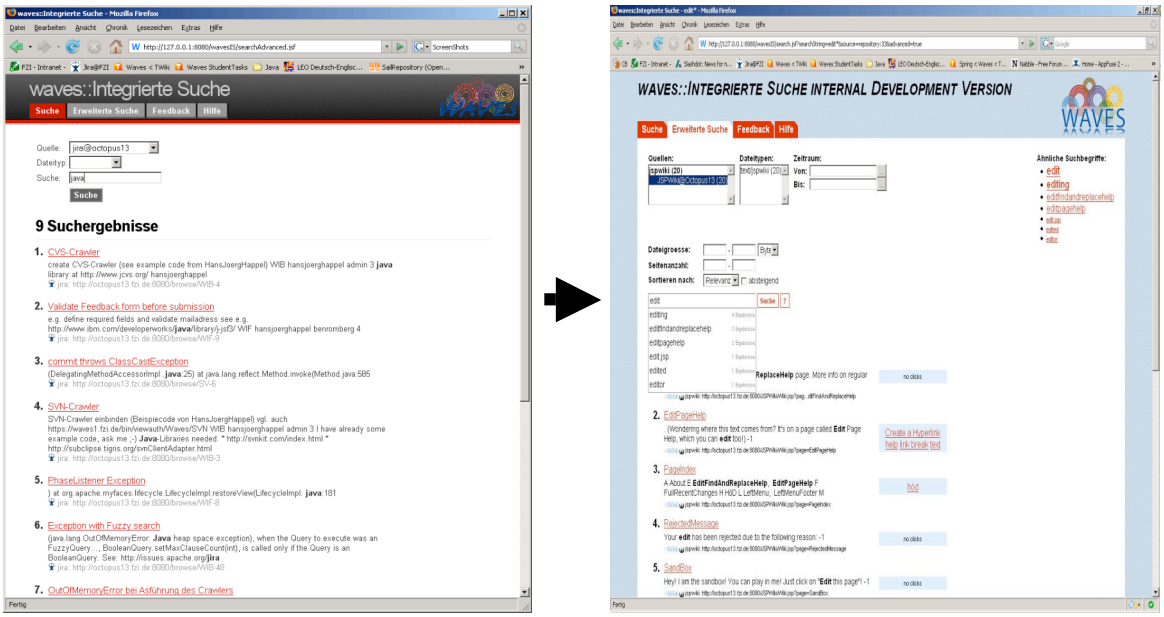
Im Folgenden werden die Ergebnisse der Studien vorgestellt. Hierbei liegt der Schwerpunkt auf der Beschreibung des Nutzens des Systems und primär auf der Integrierten Suche, da diese über einen längeren Zeitraum im laufenden Betrieb getestet werden konnte. Zudem wird im folgenden Abschnitt die allgemeinen Ergebnisse, die alle Anwendungspartner betreffen beschrieben, die partnerspezifischen Ergebnisse werden in den nächsten vier Abschnitten erörtert.

Zuerst soll allerdings kurz auf die Fortschreibung der Anforderungen eingegangen werden, die durch die kontinuierliche Evaluation erreicht wurde. Die Fortschreibung erfolgte iterativ und die aktuellen Anforderungen und die Verbesserungsvorschläge, die im Rahmen der Studien aufgedeckt wurden, wurden jeweils im nächsten Release in der Projektlaufzeit umgesetzt. Dies soll exemplarisch an der nachfolgenden Abbildung deutlich gemacht werden:

Die Abbildung zeigt das Frontend der Integrierten Suche in der ersten und in der zweiten Iteration. Neben den Veränderungen, die die Oberfläche betreffen, sind bzgl. der Funktionalität zum Beispiel weitere Suchraumeinschränkungen eingeführt worden, welche für die Benutzer von entscheidender Bedeutung waren. Des Weiteren ist die Funktion der ähnlichen Begriffe hinzugekommen, so wie eine Autovervollständigung der Suchbegriffe.

.

MR1 MR2



Weiterentwicklung der Integrierten Suche von Release MR1 zu MR2

5.2.1 Integrierte Suche: Ergebnisse aus dem Vergleich: Suche mit und ohne wavesIS

Datenerhebung:

Im Rahmen des Workshops von Release 3 wurde die wavesIS der bisherigen Suchstartegie gegenüber gestellt. Hierfür wurden benutzerspezifische Testanfragen abwechselnd zuerst anhand der wavesIS, dann anhand der bisherigen Suchstrategie und andersherum bearbeitet. An den Testläufen nahmen jeweils vier Mitarbeiter aus allen vier Anwendungsunternehmen teil. Pro Mitarbeiter wurden jeweils vier Anfragen zuerst anhand der wavesIS und vier Anfragen zuerst mittels der bisherigen Suchstrategie bearbeitet.

Ergebnisse:

Grundsätzlich zeigte sich im Rahmen der Studie, dass die Suchstrategie der Benutzer sich durch WAVES verändert.

Während die Benutzer ohne eine integrierte Suche, eher selten auf die Suchmaschinen der einzelnen Tools zurückgreifen, sondern wesentlich häufiger die gesuchte Information mittels Navigation ausfindig machen (häufig sehr zeitaufwendig, speziell wenn nicht klar ist, in welcher Quelle die gesuchte Information liegt), wird beim Einsatz von WAVES wie folgt vorgegangen:

1. Einsatz der Volltextsuche

2. Beurteilung der ersten Treffer (Ranking)

    - Relevanz
    - Qualität

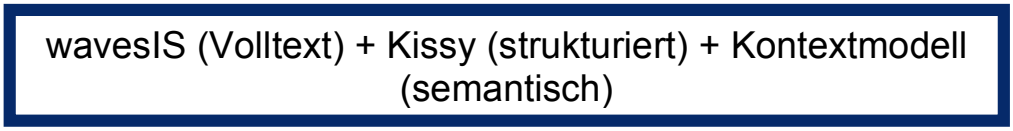
2a. Wenn die Anzahl der Suchtreffer zu groß

-> Suchraum über Suchbegriffe bzw. Quellen einschränken

2b. Wenn die Anzahl der Suchtreffer zu groß

-> Suchraum über Metadaten/Felder einschränken

Für dieses Vorgehen werden unterschiedliche Komponenten von WAVES benötigt:



Das veränderte Benutzerverhalten erklärt auch die Ergebnisse der Studie im Bezug auf drei unterschiedliche Einsatzszenarien von WAVES.

Szenario 1:

Der Benutzer weiß genau, wo die Information steht, die er sucht

Ergebnis:

Die Informationssuche mit der wavesIS dauert im Durchschnitt doppelt so lang wie die Navigation

Szenario 2:

Der Benutzer weiß genau, welches konkrete Artefakt er sucht, weiß aber nicht wo es

abgelegt ist

Ergebnis:

Die Informationssuche mit der wavesIS ist im Durchschnitt doppelt so schnell wie die herkömmliche Suche.

Szenario 3:

Der Benutzer sucht Informationen zu einem Thema, weiß aber nicht, welche Informationen es gibt und wo sie zu finden sind

Ergebnis:

Kein Vergleich möglich, Abbruch mit bisheriger Suchstrategie

Speziell in dem dritten Szenario bietet die wavesIS klare Vorteile. Häufig wurde die Navigation frühzeitig erfolglos abgebrochen, da der Benutzer kein zuverlässiges Abbruchkriterium für die Suche hatte. Folgende Kommentare der Endanwender aus den Interviews unterstreichen die oben beschriebenen Ergebnisse:

„Wenn ich Informationen aus dem Wiki und dem Filesystem suche, hab ich immer die wavesIS eingesetzt, weil die Navigation sonst so lange dauert."

„Ich kann suchen, ohne zu wissen, wo ich die richtigen Suchergebnisse finde."

„Ich weiß, wann ich mit der Suche aufhören kann."

Abschließend kann festgehalten werden, dass speziell für Szenarien, in denen nicht klar ist, in welcher Quelle oder ob Informationen zu einem Suchbegriff vorliegen, die Effizienz der Suche durch die wavesIS verbessert wird.

5.2.2 Integrierte Suche und Kissy: Ergebnisse aus der Studie im laufenden Betrieb und Interviewdaten

Datenerhebung:

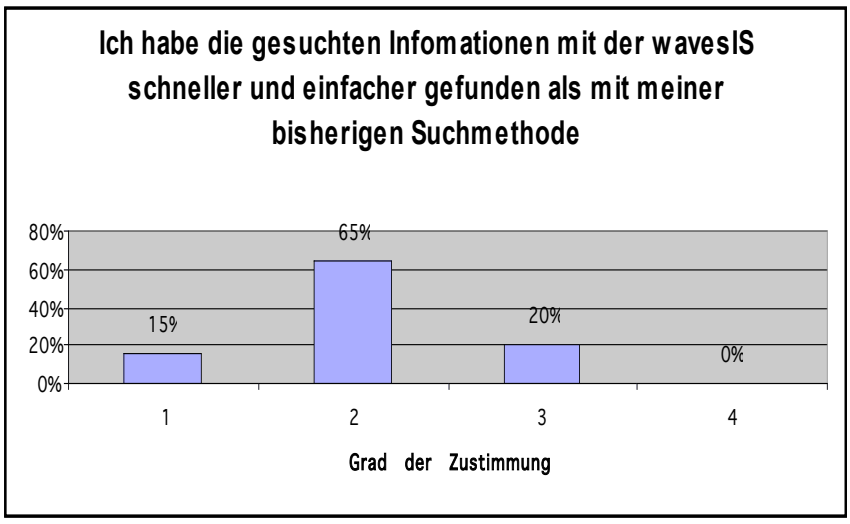
Die Integrierte Suche und Kissy wurden im laufenden Betrieb bei allen Anwendungspartnern über einen Zeitraum von 2 Monaten eingesetzt. In jedem Anwendungsunternehmen nahmen 4-5 Mitarbeiter an dem Testlauf teil. Diese wurden alle jeweils in 2 wöchentlichen Abständen, also jeweils insgesamt viermal telefonisch interviewt. Dabei sollten sie unter anderem bestimmte Aussagen auf einer Skala von 1 bis 4 bewerten, wobei 1 „Ich stimme voll zu“ und 4 „Ich stimme überhaupt nicht zu“ bedeutet. Des Weiteren konnten die Benutzer auch die Option „keine Angabe“ auswählen, falls sie eine Aussage nicht bewerten wollten. Dies führt dazu, dass in den folgenden Statistiken, sich die Summe der Antworten nicht zwingend auf 100% zusammen addieren muss, da das Feld „keine Angabe“ nicht aufgeführt ist.

Die folgenden Ergebnisse basieren also auf den folgenden Daten: Einsatz im laufenden Betrieb der wavesIS bei allen Anwendungspartnern im Zeitraum (leichte Abweichungen je nach Anwendungspartner) Mitte August bis Mitte Oktober 2008 (insgesamt jeweils 8 Wochen). Insgesamt nahmen 18 Mitarbeiter aus den Unternehmen teil, diese wurden jeweils viermal telefonisch je im Abstand von 2 Wochen interviewt. Alle Interviews wurden bei der Auswertung berücksichtigt.

Ergebnisse:

Die folgenden Statistiken zeigen die Ergebnisse der Auswertung der Interview-Daten, die während des Einsatzes im laufenden Betrieb erhoben wurden. Auf der y-Achse ist der Grad der Zustimmung angegeben, wobei 1 „Ich stimme voll zu“ , dann absteigend bis 4 „Ich stimme überhaupt nicht zu“ und 5 (nicht in der Graphik aufgeführt) „keine Angabe“ bedeutet, auf der x-Achse ist jeweils angegeben, wie viel Prozent der Befragten (dabei wurden alle Interviewdaten, also je vier pro Teilnehmer berücksichtigt) sich für den jeweiligen Grad der Zustimmung entschieden haben.

Um die Effizienz und somit den ökonomischen Nutzen der WAVES Plattform beurteilen zu können, wurden, sollten die Endanwender die folgende Aussage bewerten: *„Ich habe die gesuchten mit der wavesIS schneller und einfacher gefunden als mit meiner bisherigen Suchstrategie.“*



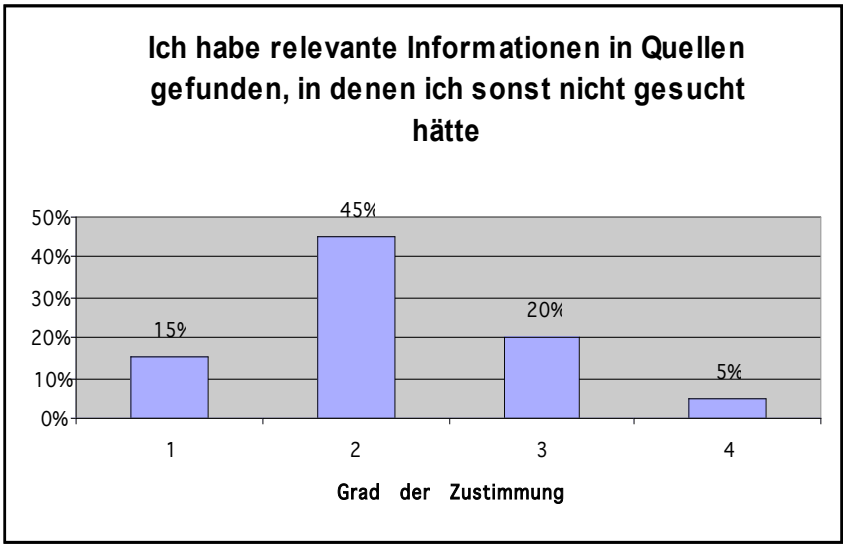
Insgesamt stimmten 80 % der Endanwender dieser Aussage zu. Dieses Ergebnis unterstreicht den ökonomischen Nutzen der Suche. Die wavesIS ermöglicht einen schnelleren Zugriff auf die gesuchten Informationen. Kommentare der Benutzer:

„Die Suche war insbesondere bei der Suche über mehrere Quellen viel schneller.“

„Die Kontextinformationen helfen mir die Ergebnisse schneller zu bewerten. Ich muss nicht alle Dokumente extra öffnen, wenn zum Beispiel die Dateinamen nicht so aussagekräftig sind.“

Die zweite Aussage zeigt, dass neben der schnellen Erschließung mehrer Wissensquellen auch die Funktionalität der Kontextinformationen den Suchvorgang verbessert. Während bei der herkömmlichen Strategie (Navigation) meist nur die Dateinamen durchgeschaut wurden, bietet die wavesIS direkt einen Einblick in das entsprechende Dokument/ die entsprechende Datei, so dass der Kontext in dem der Suchbegriff vorgekommen ist, leicht zugeordnet und bewertet werden kann.

Die Benutzer sollten ebenfalls die folgenden Aussage bewerten: *„ Ich habe relevante Informationen in Quellen gefunden, in denen ich sonst nicht gesucht hätte“.*



15% der Endanwender stimmen dieser Aussage völlig zu, insgesamt 60 % stimmen der Aussage zu, 25 % stimmen der Aussage nicht zu und 15% der Endanwender haben diese Aussage nicht bewertet. . Das Ergebnis zeigt, dass ein Großteil der Endanwender durch WAVES Information aus Quellen erfasst, die bislang nicht für die Wissensnutzung verwendet wurden. Dies bedeutet, dass die Wissensnutzung durch WAVES effektiver wird und dadurch den Reuse von Wissen unterstützt und somit automatisch redundante Wissensartikulation vermeidet. Die folgenden Aussagen wurden von den Endanwendern in diesem Zusammenhang getroffen:

„In den Sourcen und Dokumenten von anderen Teams kann ich ohne wavesIS normalerweise gar nicht recherchieren.“

„Ich habe entdeckt, dass eine andere Abteilung auch schon Dokumente zu dem Thema erstellt hat.“

„So (ähnliche Suchbegriffe) kann man alle Einträge zu einer Person finden, deren Namen oft unterschiedlich geschrieben wird und die Einträge auch direkt verbessern “

Die erste Aussage unterstreicht die Überwindung einer psychologischen Barriere. wavesIS ermöglicht die Erschließung auch unbekannter Wissensquellen. Die letzte Aussage zeigt noch einen weiteren Vorteil der WAVES Plattform auf. Durch die Funktionalität der ähnlichen Begriffe, bietet WAVES die Möglichkeit auch dann Informationen zu finden, wenn der Suchbegriff ggf. falsch oder anders geschrieben wurde. Während der Anwenderstudien stellte sich heraus, dass diese Funktion speziell für das Auffinden aller Informationen eines Kunden (zb. Meyer, Maier, Meier) extrem hilfreich sein kann.

5.3 Einbettung der WAVES-Ergebnisse bei den Anwendungspartnern

5.3.1 CAS Software AG

Aus Sicht der CAS Software AG zeigten sich besonders starke Effekte in folgenden Szenarien:

- Einarbeitung neuer Mitarbeiter: Ein zentraler Zugriff auf das gesammelte Wissen in einem Projekt ist dafür eine große Hilfe. Dies ist besonders in Wachstumsphasen von Unternehmen wertvoll.
- Größere oder langlaufende Projekte: In derartigen Projekten kann ein einzelner Mitarbeiter nicht mehr alles notwendige Wissen überblicken und benötigt deshalb Unterstützung. Ein System zum Management des Wissens wie bei WAVES steigert hier die Effizienz und bringt Wissen schnell zu den richtigen Personen dann, wenn sie es brauchen.
- Spezialwissen zu Bibliotheken / Plattformen: Nicht alles Spezialwissen kann bei jedem Mitarbeiter verfügbar sein. Wurde Spezialwissen gesammelt, sollte es zur Unterstützung des restlichen Teams möglichst schnell verfügbar gemacht werden. Dies ist durch die Werkzeuge von WAVES einfach möglich.

Als sehr nützlich zeigte sich der zentrale und gebündelte Zugriff auf mehrere Informationsquellen (wie Wiki, Dokus, Quellcodes). Damit ließen sich aus verschiedensten Quellen die Informationen zusammentragen und Verbindungen suchen.

Überraschend war der Einsatz von KISSy: Die Einbeziehung der Quelltextqualität in die Suchergebnisse liefert wirklich hilfreiche Ergebnisse. Beispiel: In einem Projekt wurden Redundanzen im Quelltext angezeigt. Ein evaluierender Entwickler kommentierte dies mit „echte Hilfe zur Optimierung!“.

Insgesamt zeigte sich eine sehr positive Aufnahme im Entwicklerkreis.

Als weitere Ergebnisse sind festzuhalten: Die WAVES-Suchmaschine lieferte eine gute Trefferquote mit einer guten Darstellung des „Treffergrundes“. Als sehr hilfreich erwies sich auch die Erschließung weniger bekannter Wissensquellen wie Dokumentationen von Kollegen und nicht direkt projektbezogenes Wissen. Die große Vielfalt von bereits unterstützen Quellen bei der WAVES-Suchmaschine ist besonders hervorzuheben.

Mit der integrierte Suche können tatsächlich wirklich Unterschiede zwischen Artefakten, Beteiligten, Orten nivelliert werden und trotzdem alle Informationen für alle zur Verfügung gestellt werden.

5.3.2 disy

Die fortlaufende Weiterentwicklung der WAVES-Plattform insbesondere der bei disy bereits installierten WAVES integierten Suche, erforderte Installationen und Tests der neuen Meilensteine in regelmäßigen Abständen. Dennoch konnte die Evaluation erfolgreich abgeschlossen werden.

Festgestellte Probleme bei der Installation und Softwarefehler wurden in den WAVES Projekt-Jira eingepflegt und weiterverfolgt. Die Rückmeldung und Problemlösung durch die zuständigen Mitarbeiter des FZI erfolgte sehr zufriedenstellend und zeitnah.

Zur besseren Fehleranalyse und zur Beschleunigung der Fehlerbehebung wurde bei disy auch erstmals ein direkter Zugriff auf den Quellcode der WAVES-Plattform realisiert. Dazu wurde ein spezieller Arbeitsbereich für die von disy genutzte Entwicklungsumgebung Eclipse eingerichtet und eine Verbindung zum zentralen Subversion Dokument- und Quellcode-Repository des FZI aufgebaut. Dies ermöglichte zusätzlich die Erstellung von Distributionen der WAVES-Plattform ohne auf eine Auslieferung des FZI angewiesen zu sein. Dadurch ist es disy gelungen kleinere Fehler im Quellcode zu lokalisieren, zu beheben und sofort im Anschluss eine fehlerbe-reinigte Version in Betrieb zu nehmen. Das endgültige Einpflegen der Bugfixes wurde jedoch den Projektentwicklern des FZI überlassen.

Durch die Ablösung von CVS durch SVN und die Nutzung von Jira zur Zeiterfassung musste die Konfiguration der Datenquellen für die WAVES integrierte Suche angepasst werden:

- Die XPlanner Anbindung wurde entfernt, da dieses Tool nicht mehr benutzt wird und die XPlanner Datenbank abgeschaltet wurde.
- Die Datenquellen CVS und CVS Changelogs wurden ebenfalls entfernt. Die Abschaltung des CVS-Servers steht kurz bevor.
- Die WAVES integrierte Suche unterstützt bereits SVN als Datenquelle, so dass das System nahtlos und mit wenig Aufwand an die bestehende WAVES-Infrastruktur angebun-den werden konnte. Um die WAVES integrierte Suche nicht zu überfordern und Fehler leichter aufdecken zu können, wurde erstmal nur ein kleiner Teil des SVN Quellcode-Repository indiziert.
- Folgende Datenquellen stehen nun bei disy für die WAVES integrierte Suche zur Verfügung: Dateisystem, Bugzilla, JSPWiki, Jira, Confluence, SVN

KISSy wurde bei disy auf zwei verschiedenen Betriebssystemen (Linux und Windows) und zusammen mit unterschiedlichen Datenbanken (HSQLDB und PostgreSQL) installiert. Auf diese Weise konnte KISSy auch im Zusammenspiel mit verschiedenen Systemkonfigurationen getestet werden, so dass ein wichtiger Beitrag zur Stabilität der Software geleistet werden konnte. Für den KISSy Workshop wurde ein Modul aus dem SVN-Repository erfolgreich mit KISSy indiziert, so dass eine Evaluation durch die Teilnehmer des Workshops erfolgen konnte. Der Einsatz von KISSy im laufenden Betrieb ist nach Projektende vorgesehen.

Wiquile wurde auf mehreren Desktop-Computern installiert und erfolgreich getestet. Im Rahmen eines Workshops wurde die Software ausführlich von mehreren Anwendern getestet und bewertet. In einem gemeinsamen Workshop mit der FU Berlin wurde KISSy und der Wiquila einem Anwenderkreis aus Entwicklern vorgestellt. In separaten Sitzungen wurden diese Anwendungen anschließend in einem Usability Test auf ihre Tauglichkeit überprüft und bewertet.

Jeder Teilnehmer hat dazu an einem speziellen Rechner mit Kamera, Mikrofon und einem Screenrecorder ein festgelegtes Szenario bearbeitet. Mit einer Think-Aloud genannten Technik sollten dabei ständig die eigenen Gedanken laut ausgesprochen werden, damit in einer anschließenden, detaillierten Analyse noch Verbesserungspotential aufgedeckt werden kann.

Zusätzlich konnten Verbesserungswünsche, Kommentare und Softwareprobleme mit Hilfe des Werkzeugs OpenProposal direkt am Bildschirm grafisch formuliert und zusammen mit einem automatisch generierten Screenshot gespeichert werden.

Der Workshop zeichnete sich durch lebhafte Diskussionen und einer Fülle von Verbesserungs-vorschlägen sowohl für KISSy als auch für den Rich Wiki Client aus und kann daher als sehr erfolgreich betrachtet werden.

Es wurde innerhalb der ausführlichen Vorhabensbeschreibung bereits ein Verwertungsplan für disy erstellt. Die wirtschaftlichen Erfolgsaussichten werden nach wie vor als gut eingeschätzt. Innerhalb von disy ist durch die bisherigen Aktivitäten das Bewusstsein für Wissensmanagement und damit verwandte Fragestellungen in der gesamten Firma gewachsen.

Darüber hinaus hat gerade die Einführung von Jira und Confluence sowie die Umstellung von CVS auf SVN gezeigt, dass WAVES auch bei größeren Veränderungen im Bereich des Wissensmanagements aufgrund des großen Funktionsumfangs und der einfachen Weiterentwicklung sehr dynamisch an neue Anforderungen anpassbar ist. Dies ist ein Beweis, dass einige der in WAVES entwickelten Techniken und Prototypen in Zukunft wirtschaftlich verwertet werden können.

Die wissenschaftlichen und technischen Erfolgsaussichten werden nach wie vor gut eingeschätzt. disy betrachtet WAVES als eine gute Investition in seine eigene Infrastruktur und erwartet, die Ergebnisse von WAVES direkt für die Verbesserung des Wissensmanagements bei der Softwareentwicklung einsetzen zu können. Die Einführung von Jira und Confluence sowie die Umstellung von CVS auf SVN bedeuten hier einen großen Schritt nach vorne. Durch den Einsatz von WAVES für die

übergreifende Suche in den verschiedenen Datenquellen gewinnt das Projekt damit für disy noch mehr an Bedeutung, da der Wissensverlust durch den Umstieg auf die neuen Systeme durch WAVES minimiert wird.

Durch die Entwicklung und Verbesserung von Methodik und Werkzeugen in WAVES verspricht sich disy nach wie vor auch zukünftig große Vorteile bei zukünftigen Entwicklungen. Das über die WAVES integrierte Suche zur Verfügung stehende Angebot an Datenquellen ist sehr umfangreich und unterstützt viele gängige Artefakttypen. Die reibungslose Umstellung auf SVN als neues System zur Versionsverwaltung und die Einbindung von Jira und Confluence hat gezeigt, dass WAVES auch in Zukunft als Werkzeug zur übergreifenden Suche aktuell bleibt.

5.3.3 Object International

Ausgangsszenario & Status Evaluation

Ausgangsszenario vor WAVES

Der Ausgangsstatus vor WAVES stellt sich folgendermaßen dar: Wissen ist in hohem Maß als implizites Wissen vorhanden und stark an Personen gebunden - die Wissensakquisition ist kein etablierter kontinuierlicher Prozess, des Weiteren sind Aktivitäten im Bereich Wissensakquisition und -management der durch den hohen „On Site“ Anteil bei der Leistungserbringung erschwert. Expliziertes Wissen ist auf eine Vielzahl von Quellen (IT-Systeme und Datenformate) verteilt - als Folgeeffekt ist die Recherche nach Informationsartefakten für Mitarbeiter sehr zeitraubend und für das Unternehmen „teuer“. Ein zentrales Wissensmanagementsystem (beispielsweise ein Wiki-System) existierte nicht.

Kernaspekte aus der Bedarfsanalyse

Nachfolgend sind die im Rahmen der Bedarfsanalyse erarbeiteten Kernaspekte aufgeführt:

Aspekt Wissensakquisition- und transfer: Ausbau und Sicherung der Wissensbasis des gesamten Unternehmens - Schutz vor „Wissensverlust“ durch ausscheidende oder temporär nicht verfügbare Mitarbeiter, Unterstützung bei der Einarbeitung neuer Mitarbeiter und bei der Erschließung neuer Themenbereiche.

Aspekt Steigerung der Produktivität und Useability: Die signifikante Reduktion des Zeitaufwandes bei der Recherche nach Informationen und Daten und damit eine direkte Verbesserung der Produktivität der Mitarbeiter.

Aspekt Bereitstellung und Verteilung von Wissen: Die WM-Plattform soll einen Grossteil der relevanten Datenquellen und Datenformate erschließen und ein kollaboratives Erarbeiten von Wissen unterstützen.

Ökonomische- und technische Aspekte: Die Wissensplattform darf keine hohen Anforderungen an die Zielhardware, die Installation und an den Betrieb stellen. Des Weiteren sollte sich die Lösung gut in die vorhandene IT-Infrastruktur und Toolkette integrieren lassen und möglichst auf Open Source Komponenten (wg. Lizenzkosten) basieren.

Im Rahmen von WAVES evaluierte Softwarekomponenten

Im Rahmen des Projektes wurden bei *Object International* alle verfügbaren Komponenten der WAVES-Toolkette (*WavesIS, KISSy, Wiquila* und *Woogle*) evaluiert - *WavesIS* wurde im 2. Quartal 2007 (MR0) in die IT-Infrastruktur der Object International eingebunden und in den unterschiedlichsten Entwicklungsstufen intensiv evaluiert. Die Komponenten *KISSy* und *Wiquila* wurden im 2. und 3 Quartal 2008 in die IT-Infrastruktur eingebunden und befinden sich im Probebetrieb.

Einsatz und Nutzen der WAVES-Ergebnisse

Da es sich bei WavesIS aus Sicht der *Object International* um die wichtigste Komponente der *WAVES*-Toolkette handelt, wird im nächsten Abschnitt zuerst näher auf *WavesIS* und den Einsatz bei *Object International* eingegangen - anschließend wird anhand der während der Bedarfsanalyse erarbeiteten Use Cases untersucht, in welchem Ausmaß die Anforderungen durch die im Rahmen von WAVES konzipierten Tools abgedeckt werden.

Durch WavesIS indizierte Systeme und Wissensartefakte

Durch die Komponente *WavesIS* wurde die überwiegende Mehrzahl der bei *Object International* eingesetzten Systeme im Laufe des Projektzeitraums für die Informationsrecherche erschlossen. Durch den Einsatz von Open Source Komponenten und der relativ unkomplizierten Adaption bereits existierender Softwarekomponenten (Aperture Importer Komponenten) lassen sich bei Bedarf weitere Datenquellen (IT-Systemes und Datenformate) erschließen.

Die nachfolgende Abbildung gibt eine Übersicht über die indizierten Systeme – ergänzend sind auch diejenigen Systeme dargestellt, die im Projektzeitraum neu in die IT-Infrastruktur der *Object International* integriert wurden.

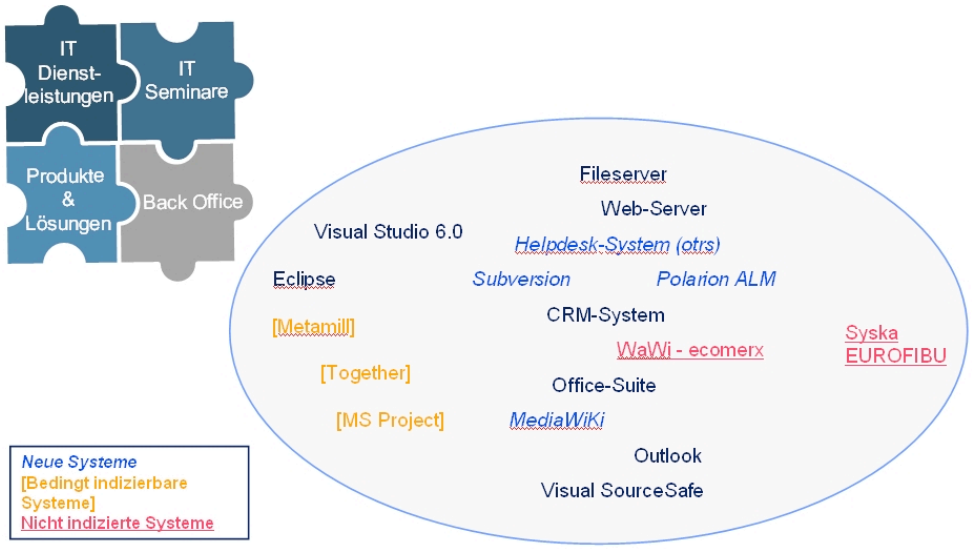


Abbildung 33: Übersicht über die indizierten IT-Systeme und Tools der Object International

Die durch die *Waves integrierte Suche* erschlossenen Wissensartefakte sind unten aufgeführt - wie aus der Abbildung ersichtlich ist, wurden alle relevanten Wissensartefakte im Verlauf des WAVES Projektes durch WavesIS erschlossen - unsere Erwartungen wurden hier deutlich übertroffen. Die als bedingt indizierbare Artefakte gekennzeichneten Wissensartefakte lassen sich über die Umwandlung in ein alternatives Datenformat durch *WavesIS* bei Bedarf erschließen.

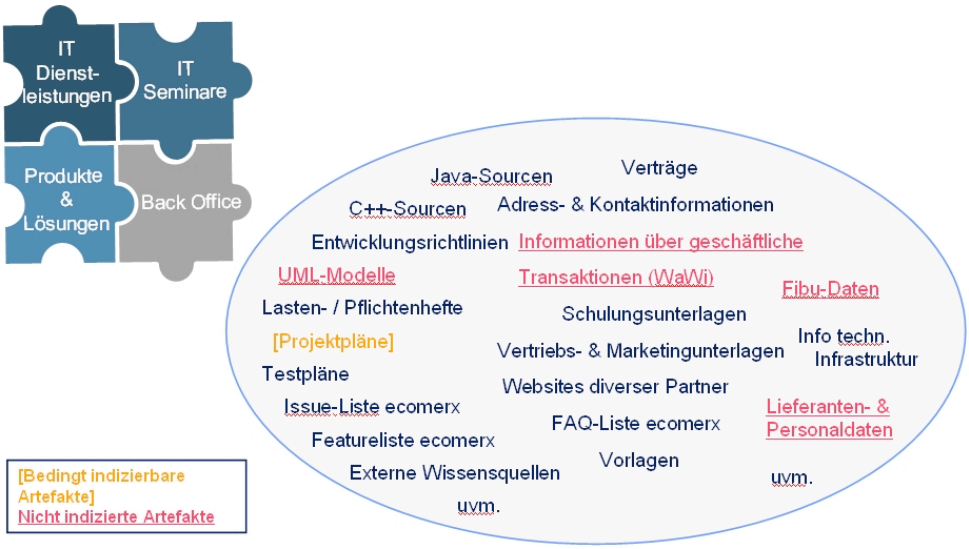


Abbildung 34: Übersicht über die indizierten Informationsartefakte der Object International

Die Kernanwendungsfälle und ihre Abdeckung durch die WAVES-Toolkette

Folgende fünf Anwendungsfälle (Use Cases) waren für *Object International* von besonderer Bedeutung:

- Use Case 1: Räumlich verteilte Wissensnutzung
- Use Case 2: Integrierte Suche über Dokumente und Software spezifische Quelldaten
- Use Case 3: Einführung von Wikis im Unternehmen
- Use Case 4: Informationsbedarf im Unternehmen identifizieren
- Use Case 5: Sharing von Wissen und Motivation zur Wissensartikulation

Die Anwendungsfälle *Räumlich verteilte Wissensnutzung (UC 1)* und *Integrierte Suche über Dokumente und Software spezifische Quelldaten (UC 2)* werden durch *WavesIS* sehr gut unterstützt. Die Informationsbasis des Unternehmens lässt sich durch *WavesIS* in hervorragender Weise erschließen, da bei Bedarf alle relevanten Informationsartefakte für eine Recherche indiziert werden können. Je nach Rolle und persönlichen Vorlieben oder je nach Arbeitskontext kann der (zentrale!) Zugriff auf die gesamte Wissensbasis entweder über den Webbrowser oder über das ebenfalls zur Verfügung stehende Eclipse-Plug-In erfolgen.

WavesIS integriert und erschließt eine Vielzahl unterschiedlicher Quellen - vom Anforderungsdokument über Sourcecode bis zu Adress- und Kontaktinformationen. Die Recherche nach Informationsartefakten kann parallel in allen indizierten Quellen erfolgen. Bei Bedarf - beispielsweise falls die Treffermenge zu groß ausfällt - bietet WavesIS umfangreiche und sinnvolle Möglichkeiten den Suchraum einzuschränken. Des Weiteren ist Funktionalität „Ähnliche Begriffe“ bei der Recherche nach fehlerhaften Namen oder Bezeichnungen äußerst hilfreich.

Durch *WavesIS* wird die „Wissensbasis“ des Unternehmens auch jenen Mitarbeitern erschlossen, die räumlich verteilt arbeiten – beispielsweise weil sie sich beim Kunden vor Ort, im Home-Office oder auf Geschäftsreise befinden. Da es sich bei dem Frontend von *WavesIS* um eine Rich Internet Application auf der Basis von JSF handelt, muss keine zusätzliche Software installiert werden - ein Webbrowser (ab IE 6.x oder Firefox 2.x) genügt völlig. Der Zugang zur *WAVES*-Plattform sollte allerdings durch geeignete Maßnahmen (beispielsweise Anbindung per VPN) abgesichert werden.

*Bewertung***:** *WavesIS* ermöglicht allen Mitarbeiter des Unternehmens - auch diejenigen Mitarbeiter, die „On-Site“ arbeiten - eine effiziente Suche nach Informationsartefakten und erschließt Verbindungen zwischen den unterschiedlichsten Informationsquellen. Die durch *Use Case 1* und *Use Case 2* adressierten Anforderungen werden durch *WavesIS* optimal erfüllt. *WavesIS* verfügt in der aktuellen Ausbaustufe (Version 0.98) über einen beachtlichen Funktionsumfang und weist einen hohen Reifegrad auf - insbesondere im letzten und im aktuellen Berichtszeitraum (Q3 / 2008) wurde eine Reihe wertvoller Funktionen ergänzt.

Die Anwendungsfälle *Einführung von Wikis im Unternehmen (UC 3)* und *Sharing von Wissen und Motivation zur Wissensartikulation (UC 5)* sowie der Anwendungsfall *Informationsbedarf im Unternehmen identifizieren (UC 4)* werden durch die Komponenten *Wiquila*, *Woogle* und *Inverse Suche* gut abgedeckt.

Wiquila ist als Rich-Client implementiert und erlaubt auch Nicht-Experten die Erstellung und Bearbeitung von Wiki-Seiten ohne die etwas „sperrige“ Wiki-Syntax zu erlernen (Verringerung der technischen Barrieren). *Wiquila* erlaubt das formatierte Editieren von Wiki-Seiten (WYSIWYG-Funktionalität) und bietet darüber hinaus die aus den Office-Tools bekannten Drag- und Drop- sowie Copy- und Paste-Funktionalität. Als besonders wertvoll wird die durch *Wiquila* bereitgestellte Auto-Complete-Funktionalität auf der Basis der in der globalen Wissensbasis (*WavesIS* Backend) gespeicherten Wissensartefakte angesehen. Diese Funktionalität verringert die den Wikis eigene Gefahr zur Bildung von „Wucherungen“ signifikant und bietet darüber hinaus die Möglichkeit den Wiki-Text mit allen durch das *WavesIS* Backend erschlossenen Informationsartefakten zu verlinken. *Wiquila* verfügt über eine eigene Datenhaltung und kann daher auch Offline (ohne direkte Anbindung

an eine Wiki-Engine) betrieben werden - damit unterstützt *Wiquila* auch den Anwendungsfall *Räumlich verteilte Wissensnutzung (UC 1)*.

Die Komponente *Woogle* ergänzt MediaWiki um Suchfunktionalität und bietet für den Anwender eine deutliche Verbesserung bei der Artikulation von Wissen im Wiki. Falls auf eine Suchanfrage keine Ergebnisse gefunden werden, kann eine neue Wiki-Seite angelegt werden - zusätzlich erhalten andere Anwender (soweit sie sich zuvor für Suchanfragen notifiziert haben), die am selben Wissensartefakt interessiert sind, eine Benachrichtigung über die Suche *(UC 4*) und den ergänzten *MediaWiki*-Eintrag. Die Hemmschwelle zur Erfassung eines neuen Eintrags wird somit verringert und die Akzeptanz von Wikis damit erhöht.

Die *Inverse Suche* wertet die Nutzeranfragen an die *WAVES*-Plattform statistisch aus und gleicht den lokalen Index (indiziert werden die nicht öffentlich zugänglichen Wissensartefakte des potenziellen Informations-Providers) mit dem globalen Index der Wissensbasis (*WavesIS* Backend) ab. Bei Bedarf - beispielsweise weil in der globalen Wissensbasis keine Informationen verfügbar sind - kann der Informations-Provider nachgefragte Wissensartefakte aus seinem privaten Bereich an die/den jeweiligen Informationskonsumenten weitergeben und in die globale Wissensbasis überführen.

*Bewertung***:** *Wiquila* und *Woogle* reduzieren die technischen und ökonomischen Barrieren für den Einsatz von Wikis signifikant und erhöhen dadurch die Motivation zur Wissensartikulation für alle Mitarbeiter des Unternehmens deutlich.

Woogle und insbesondere die *Inverse Suche* unterstützen die Identifikation von Informationsbedarf und die bedarfsorientierte Bereitstellung von Informationen mit sehr innovativen Konzepten - die durch *UC 3, UC 4, und UC* 5 adressierten Anforderungen werden deutlich erfüllt.

Ökonomische- und technische Aspekte

Die gesamte Toolkette stellt keine hohen Anforderungen an den Zielhardware und kann bei Bedarf auf einem Laptop installiert werden. Die Toolkette lässt sich sehr einfach in die vorhandene Infrastruktur einbetten und in die gewohnte Arbeitsumgebung integrieren - die Installation des Basiskomponente *WavesIS* ist in der aktuellen Entwicklungsstufe ebenfalls relativ einfach. Es fallen keine Lizenzkosten an, da die *WAVES*-Toolkette weitgehend auf Open Source Komponenten aufbaut - des Weiteren lassen sich bei Bedarf Anpassungen an einzelnen Komponenten vornehmen.

Bewertung: Die adressierten nichtfunktionalen Anforderungen an eine Wissensmanagementplattform sind ebenfalls sehr gut erfüllt.

Weitere Ergebnisse

Im Kontext von WAVES wurde die Tool-Landschaft der *Object International* um einige Systeme ergänzt - maßgebliche Kriterien für die Auswahl eines Tools waren hierbei die optimale Integration in die WAVES-Plattform sowie ökonomische Gesichtspunkte.

Nach einer ersten Analyse der frei verfügbaren Wiki-Lösungen wurde im 1. HJ 2007 *MediaWiki* als Wiki-System ausgewählt. *MediaWiki* ist als Open-Source-Software sehr gut verfügbar und kostenneutal, die Syntax und die Administration sind gut bis sehr gut dokumentiert, es gibt ergänzende Komponenten wie beispielsweise das *Semantic MediaWiki* und nicht zuletzt handelt es sich um eine in der Praxis sehr bewährte Wiki-Engine. Das FZI leistete aufgrund der vorhandenen Expertise im Bereich Wikis wertvolle Unterstützung beim Auswahlprozess und bei der Bereitstellung ergänzender Informationen und Studien zu Wiki-Systemen.

Im Verlauf des 1. HJ 2007 wurde das webbasierte Helpdesksystem *OTRS*[7] evaluiert und anschließend in die IT-Infrastruktur eingegliedert - *OTRS* integriert sich aufgrund seiner Technologie ebenfalls sehr gut in die WAVES-Toolkette.

[7] Details zu *OTRS* siehe http://de.wikipedia.org/wiki/OTRS

Im 2. Halbjahr 2007 wurde mit der Evaluierung von Polarion ALM begonnen - auch dieses Tool integriert sich hervorragend in die WAVES-Toolkette.

Redundante Informationen und redundante Ablageorte für Dateien sowie fehlerhafte Bezeichnungen oder Namen konnten mittels WavesIS identifiziert und berichtigt werden - derartige Konsolidierungen sind ohne die durch WavesIS bereitgestellten Funktionen mit sehr großem Aufwand verbunden bzw. gänzlich unmöglich.

Fazit und Nutzungspotential von WAVES nach Projektabschluss

Fazit

Durch den Einsatz der im Kontext von *WAVES* entwickelten Werkzeuge konnte die Effizienz bei der Suche nach Informations- /Wissensartefakten im Projektverlauf bereits deutlich gesteigert werden.

Durch die wertvollen Beiträge der FU Berlin im Rahmen der Anwenderstudien in Verbindung mit dem kontinuierlich zunehmenden Reifegrad und Funktionsumfang der *WAVES*-Plattform konnte das Bewusstsein für Wissensmanagement und den damit zusammenhängenden Möglichkeiten bei allen Mitarbeitern der *Object International* deutlich gesteigert werden.

Die Kontakte zu einzelnen Verbundpartnern wurden im Kontext von *WAVES* intensiviert - insbesondere der Kontakt zum Verbundpartner POLARION wurden im Rahmen des Kooperationsprojektes weiter ausgebaut (intensive Einarbeitung eines weiteren Mitarbeiters in Polarion ALM sowie die gemeinsame Kundenakquisition) - der seit Mitte der 90er Jahre bestehenden Kontakt zum FZI wurde im Rahmen von *WAVES* wieder intensiviert.

Die im Rahmen von *WAVES* entwickelten Werkzeuge kommen auch nach Projektende weiter bei der *Object International* zum Einsatz - daher begrüssen wir es sehr, dass ein Teil der in WAVES entwickelten Tools in die TeamWeaver-Plattform überführt und somit einer breiteren Community zur Verfügung gestellt wird. Aufgrund der im Verlauf des WAVES-Projektes erzielten Resultate wäre aus Sicht der Object International die Weiterentwicklung der bereits prototypisch realisierten Tools im Rahmen eines Anschlussprojektes ebenfalls sehr zu begrüssen.

Nutzungspotential von WAVES nach Projektabschluss

Gemeinsam mit den im Rahmen des WAVES-Projektes erarbeiteten Tools und Methoden erwarten wir uns für den *Geschäftsbereich IT-Dienstleistungen* nach wie vor wertvolle Impulse für Beratungsaktivitäten und Kundenprojekte - insbesondere bei Gutachten und Code-Reviews leistet *KISSy* sehr gute Unterstützung - des Weiteren werden wir die *WAVES*-Plattform interessierten Dienstleistungskunden vorstellen und bei Interesse in die IT-Infrastruktur des interessierten Kunden integrieren.

Nach dem Ende des *WAVES*-Projektes wurde *WavesIS* (Version 0.98) von einem Mitarbeiter mit sehr gutem Erfolg zur Erschließung der Wissensartefakte bei einem Dienstleistungskunden verwendet - der Mitarbeiter verwendete hierzu eine lokale Installation des *WavesIS* Backend auf seinem Laptop.

Im *Geschäftsbereich Seminare* werden die im Verlauf des *WAVES*-Projektes gewonnenen Erkenntnisse sukzessive in diverse Seminarmodule eingearbeitet - hierfür bieten sich insbesondere die Module Requirements-Engineering und Projektmanagement (*WavesIS, Wikis, Wiquila, OpenProposal*) sowie die Software-Engineering-Themen (*KISSy*) an.

Im *Geschäftsbereich Produkte* ist der Einsatz von *KISSy* für die Analyse und Bewertung des Quellcodes der einzelnen Produkte geplant - des Weiteren wird die Installation von *WavesIS* (Fokus Suchfunktionalität) bei interessierten Warenwirtschaftsystem-Kunden intensiv diskutiert.

Der bisherige Prozess zur Erhebung und Bewertung von Anwendervorschlägen wird anhand der im Verlaufe von *WAVES* gewonnenen Erkenntnisse zur Vorgehensweise - Interviews, Think aloud und den Einsatz von *OpenProposal* - entsprechend adaptiert.

5.3.4 Polarion

Polarion Software bietet Lösungen für die gesamtheitliche Planung, Steuerung und Verwalten aller Bereiche einer Anwendungsentwicklung. Die Produktlinie Polarion ALM eignet sich nicht für Softwareprojekte, sondern für alle Ingenieurstätigkeiten, wie Hardwareentwicklung, Embedded-Systeme und angrenzender Gebiete. Kunden setzen Polarion ALM teilweise auch für Risikomanagement, Dokumentenmanagement ein. Die Nutzung einer integrierten, durchgängigen Application Lifecycle Management-Lösung sorgt für einen höheren Durchsatz in der Entwicklung, da alle Beteiligten sich in derselben Umgebung aber unterschiedlichen Sichten arbeiten. Manuelle Überbrückung von Werkzeugschnittstellen entfallen. Durch die Nachvollziehbarkeit über alle Ebenen hinweg, können Entscheidungen über Änderungen viel schneller gefällt werden, da die Auswirkungen transparent sind.

Die grundlegende Motivation für Polarion Software für das WAVES-Projekt lag in der Erschliessung neuer Funktionalitäten, die eine zusätzliche Wertigkeit für die kommerziellen Polarion-Lösungen verspricht. Polarion ALM bietet alles für die Durchgängigkeit, um auf dieser Ebene einen Mehrwert zu bieten. Auf der anderen Seite birgt das Thema Wiederverwendbarkeit von Information ein nicht unwesentlicher Faktor für eine zusätzliche Steigerung der Effizient von Entwicklungsorganisationen. Bei mehreren Hundert Projekten passiert es häufig, dass an identischen oder ähnlichen Themen parallel entwickelt wird.

WAVES bietet Ansätze, bestimmte Themengebiete, die für eine Entwicklungsorganisation von Bedeutung sind, semantisch zu erfassen. Jede Branche hat ihre eigenen Fachtermini und Sprachgebräuche. Gleiche oder ähnliche Formulierungen sind auch mit einer Volltextsuche von Polarion ALM nicht auffindbar.

Ergebnisse

Für Polarion ALM war es zum einen wichtig, eine Anbindung an die WAVES-Infrastruktur (TeamWeaver) zu schaffen, damit der Anwender seine Recherchen nicht „ausserhalb“ des Polarion ALM-Portals durchführen muss. Die Anbindung erforderte im Verlauf des Projekts die Erweiterung der von Polarion veröffentlichen Programmierschnittstelle (Polarion API). Das API steht allen Polarion-Kunden für Ergänzungen und Erweiterungen des Servers zur Verfügung. Die Suche in TeamWeaver erfolgt nun vollständig in Polarion ALM-Umgebung.

Eine weitere Aufgabe galt der Vermarktung der Ergebnisse. Die einschlägigen Entwickler-Gemeinden im Internet wurden über TeamWeaver informiert. Darüberhinaus hat Polarion den öffentlich bereit gestellten Server für Open Source-Projekte um die TeamWeaver-Entwicklung erweitert.

Ergebnis	**Beschreibung**	**Weitere Informationen**
Anbindung Polarion ALM an TeamWeaver	Erweiterung der mit dem Produkt bereit gestellten API für die Einbindung in den TeamWeaver. Alle Artifakte in Polarion ALM können über TeamWeaver, auch anhand der Meta-Information, gefunden werden.	Siehe Zwischenbericht 2 aus dem Jahr 2008.
Vermarktung des TeamWeaver	Hosten des Projekts auf Polarion Community Site	

Polarion bietet TeamWeaver als zusätzliche (kostenfreie) Komponente zu dem darunter liegenden Subversion für erweiterte Suchen und Recherchen zur Verfügung. Die Vermarktung des TeamWeaver über die Polarion Community Site erhöht die Aufmerksamkeit hinsichtlich der Polarion Produkte.

Unternehmen, die auf der Suche nach erweiterten Indizierungsmöglichkeiten für Inhalte, die in Subversion hinterlegt sind, werden auf Polarion aufmerksam.

5.3.5 PTV

Ziel der Evaluierung bei PTV war herauszufinden, wie sich die WAVES-Werkzeuge in Hinblick auf einen verbesserten Informationszugriff ausgewirkt haben. Dazu sollte man nochmals kurz die Ausgangssituation und die Wünsche betrachten:

Als Quellen für Wissen und Information zeigen sich bei der PTV

- Die Köpfe der Mitarbeiter – die richtigen Ansprechpartner zu kennen ist bei PTV essentiell für die Recherche bzw. das Auffinden von technologischem, verfahrensorientiertem, oder Werkzeug- bzw. Prozesswissen.
- Dokumente im Dateisystem – üblicherweise ist viel Wissen über Dokumente im Dateisystem verstreut, die beispielsweise für einzelne Kunden, im Projektkontext, oder als Ergebnis eines Arbeitstreffens formuliert wurden.
- Dokumente im Wiki – insbesondere entwicklungsseitig werden Informationen zunehmend im Wiki-System abgelegt. Für Pflichten- und Lastenhefte, sowie Protokolle und Memos gilt dies jedoch nur eingschränkt.
- Anforderungen, Fehler, Planungen, und Roadmaps werden in der Regel auf Basis von JIRA erstellt bzw. veraltet. Dabei entstehen oft kleine Mini-Spezifikationen, Tipps und Workarounds, die von großem Nutzen für dritte sein können.
- Dokumente und Supportanfragen mit direktem Kundenbezug werden im CRM-System verwaltet (genesisWorld)
- Unzählige Mails in den verschiedenen Mail-Ordnern (auf Servern oder PCs) dokumentieren Wissen über Projekte, Kunden, Bearbeitungsstände, etc.

Daraus ergaben sich bereits in einer frühen Phase verschiedene Hoffnungen an die im Rahmen des Projekts zu erstellende Systemlandschaft: Sie sollte dazu dienen:

- Wissen sollte einfacher zu erstellen sein, um es so leichter aus den Köpfen der Mitarbeiter in eine auffindbare Form bringen zu können.
- Wissen sollte unabhängig von der Quelle innerhalb des Unternehmens leicht auffindbar sein.
- Das alles sollte im Idealfall immer und überall möglich sein, was einen externen Zugang oder eine Offlinefähigkeit erfordert.

Der Fokus wurde dabei schnell auf das Auffinden der Informationen im beschriebenen heterogenen Umfeld gelegt. Dabei wurde der Schwerpunkt bei der Implementierung zunächst auf die in der nachfolgenden Abbildung hervorgehobenen Systeme gelegt: Dateisystem, Wiki und Issue-Tracker. Als primäre Zielgruppen wurden identifiziert: Entwicklung, Produkt-Management und Professional Services (Projektbearbeitung und Support).

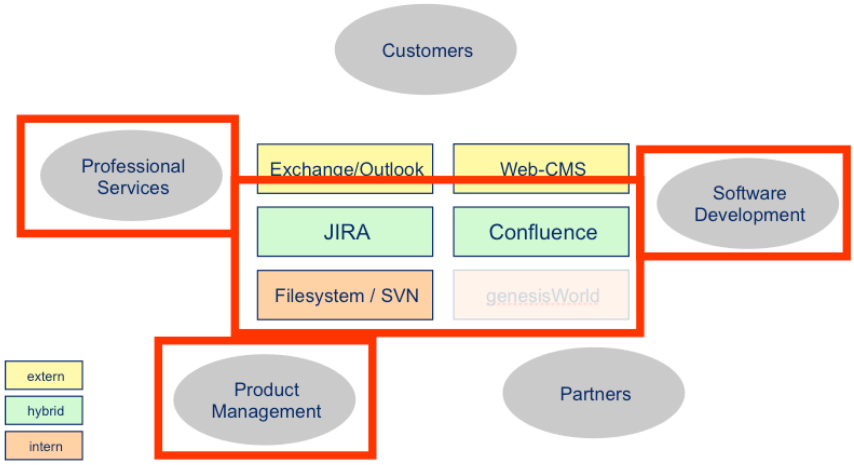


Abbildung 35: Im WAVES-Kontext betrachtete Systeme der PTV

Die Nutzung des Systems durch die einzelnen Zielgruppen zeigte in Summe ein sehr positives Bild, was in den folgenden Abschnitten etwas präziser beschrieben wird:

5.3.5.1 Zielgruppenspezifische Bewertung der WavesIS

Die folgende Tabelle zeigt die Nutzung und Bewertung der WAVES-Infrastruktur bei PTV getrennt nach Zielgruppen:

Zielgruppe	Aufgaben/Anforderungen	Bewertung
Entwickler	• Auffinden von Requirements und Bugs • Auffinden von projektspezifischen Informationen (Quellen: Wiki, JIRA, Dateisystem)	• Suche erleichtert das Auffinden verwandter Informationen zu Projekten und Themen erheblich • Positives Gesamtfeedback • Werkzeugintegration und Zugang zur Suche sollte noch erleichtert werden
Projektleiter	• Auffinden von Requirements und Bugs • Auffinden von projektspezifischen Informationen • Suche in vertrieblichen Dokumenten • Suche von Daten zu Kunden und Partnern (Quellen: Wiki, JIRA, Dateisystem, genesisWorld, Exchange)	• Ebenfalls sehr positives Gesamtfeedback • Werkzeugintegration und Zugang zur Suche sollte noch erleichtert werden • Indizierung der Datenquellen genesisWorld und Exchange unbedingt notwendig.
Entwicklungsleiter	• Auffinden von Requirements und Bugs • Auffinden von projektspezifischen	• Themenspezifische Suche ermöglicht einfachen Zugriff auf komplexe und verteilte Informationen. • Suche deckt viele Synonym-

	Informationen • Suche in vertrieblichen Dokumenten • Suche von Daten zu Kunden und Partnern • Suche nach bestimmten Themenkomplexe um Redundanzen zu verhindern bzw. Synergien nutzbar machen zu können (Quellen: Wiki, JIRA, Dateisystem, genesisWorld, Exchange)	Homonym-Probleme auf • Sehr positives Gesamtfeedback • Werkzeugintegration und Zugang zur Suche sollte noch erleichtert werden • Synonyme und spezielle Glossare sind noch notwendig. • Metainformationen müssen stärker genutzt und Suchanpasungen erweitert werden.
Support	• Auffinden von Wissen zu bekannten Fehlern • Schnelle Suche in Tutorials und How-Tos • Auffinden kundenspezifischer informationen (Quellen: Wiki, JIRA, Dateisystem, genesisWorld, Exchange)	• Suche in Dokumenten und Wiki wird sehr positiv bewertet • Derzeit fehlende Integration von gnesisWorld-Tickets erfordert nbenläufige manuelle Suche • Indizierung von genesisWorld (Tickets) unbedingterforderlich

Da bei der Bewertung ein zentraler Kritikpunkt der etwas schwerfällige Zugang zum System (eigene Suchseite) bemängelt wurde, wurde versuchsweise ein Such-Plugin für die verschiedenen Browser erstellt, das sofort auf großes positives Echo traf.

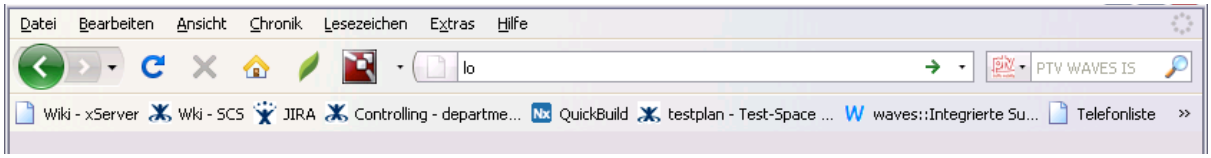


Abbildung 36: Einbettung der WavesIS in einen Browser

5.3.5.2 Flankierende Tätigkeiten im Rahmen WAVES

Da der Fokus im Projekt WAVES zunächst auf der Indizierung von Dokumenten mit dem Ziel der Wissenserschließung lag und die Möglichkeit zur Offline-Bearbeitung von Dokumenten sich zunächst auf spezielle Wiki-Systeme beschränkte, wurde bei der PTV flankierend ein kommerziell verfügbarer Offline-Client für das JIRA Issue-Tracking-System eingeführt.

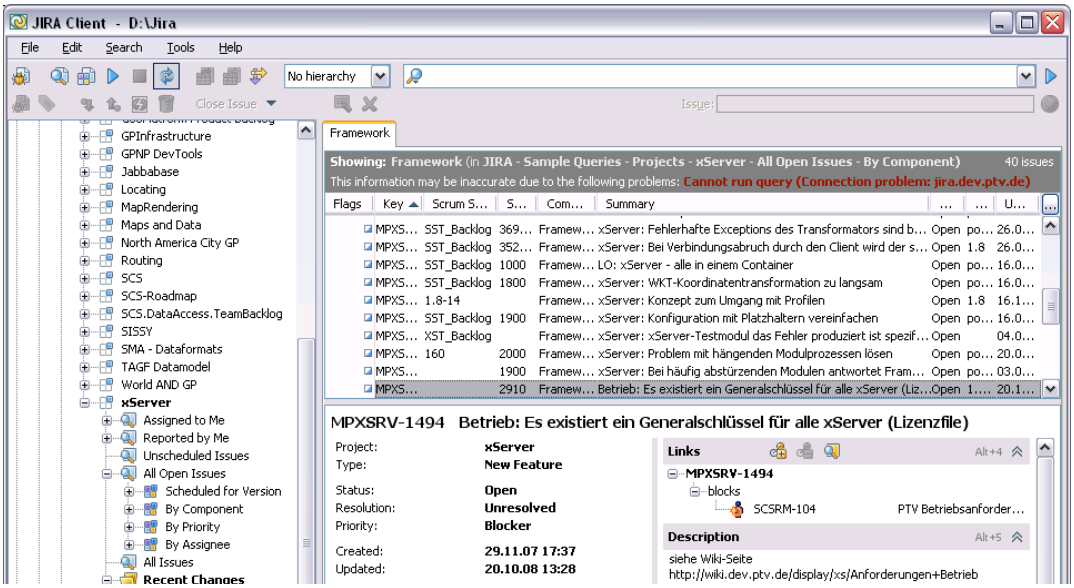


Abbildung 37: JIRA Offline Client

Die Einführung dieses Systems hat für die PTV gezeigt, wir wichtig eine zumindest kurzfristige Offline-Fähigkeit auch für die Erstellung und Bearbeitung anderer Dokumentenarten ist, da mittlerweile eine sehr ausgedehnte Nutzung des Clients im Kontext von Planungs- und außer Haus stattfindenden Strategiemeetings stattfindet. Eine weitere Ausdehnung dieser Nutzung ist deutlich abzusehen.

5.3.5.3 Zusammenfassung und weitere Arbeiten

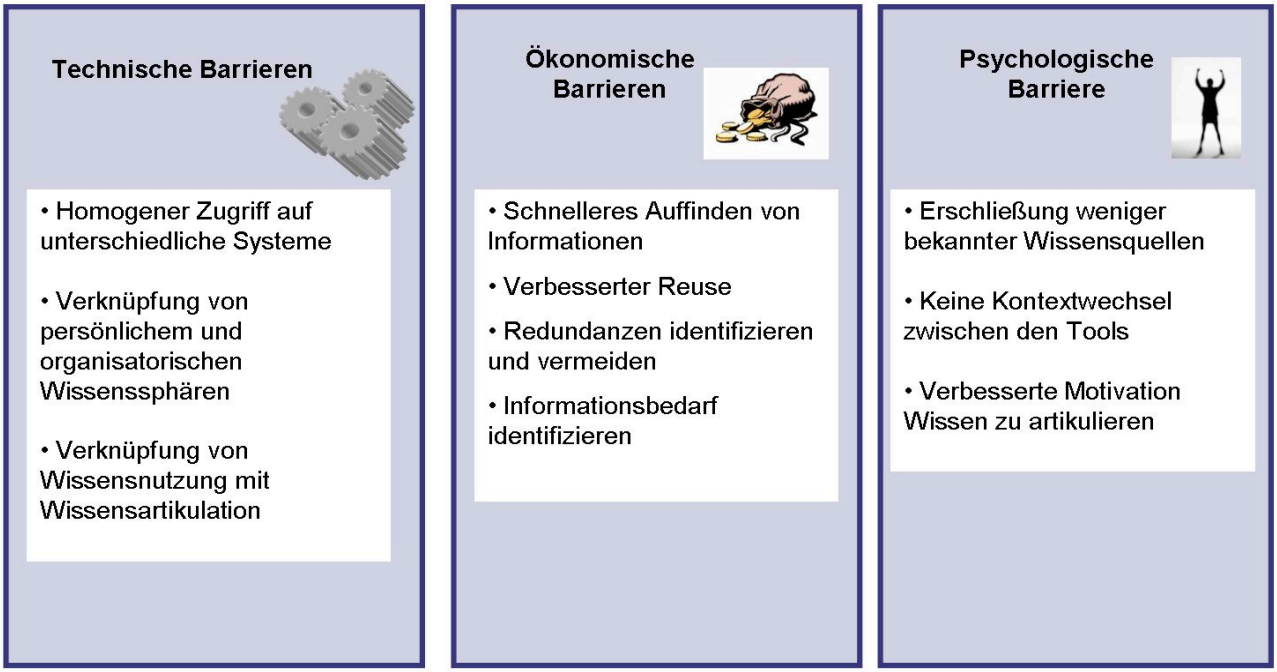
Der Appetit kommt mit dem Essen – so oder so ähnlich lässt sich die Bewertung des Projekts WAVES aus Sicht der PTV kurz zusammenfassen. Die Suchmöglichkeiten wurden in Summe als extrem hilfreich und sehr positiv empfunden. Eine Feinjustierung der Suchparameter, vor allem aber die Integration der noch nicht umfänglich indizierten Systeme genesisWorld und Exchange wurde wiederholt eingefordert.

Zudem hat die Nutzung des Offline-Clients für JIRA, sowie die versuchsweise Nutzung des Offline-Wiki-Editors gezeigt, dass gerade hier ein sehr großes Potenzial für effizientere Wissenserfassung zu liegen scheint.

Dem entsprechend werden in Folge des Projekts bei der PTV weitere Arbeiten an der Such- und Erfassungsinfrastruktur erfolgen. Die Klärung der kommenden Schritte erfolgt derzeit in direkter Abstimmung mit verschiedenen Partnern.

5.4 Zusammenfassung der Evaluationsergebnisse

Zusammenfassend kann festgehalten werden, dass WAVES bei allen Anwendungspartnern sehr positiv aufgenommen wurde und großes Interesse an den Ergebnissen des Projektes besteht. Bereits während den Workshops waren die Diskussionen über die vorhandene bzw. auch noch zu entwickelnden Funktionalitäten sehr intensiv. WAVES bietet mit der Vielzahl der Funktionen die Möglichkeit, sowohl ökonomische, als auch technische und psychologische Barrieren zu überwinden. Im Folgenden werden die einzelnen Vorteile kurz zusammengefasst.



Überblick: WAVES überwindet Barrieren

Sowohl die wavesIS im Zusammenspiel mit Kissy bietet eine Verknüpfung von unterschiedlichen Systemen, aber auch Wiquila, welches bereits bei der Wissensartikulation die unterschiedlichen Systeme mit einander vernetzt. Durch diesen **homogenen Zugriff auf unterschiedliche Systeme** wird die **Erschließung weniger bekannter Wissensquellen** gefördert, da **kein expliziter Kontextwechsel zwischen den Tools** nötig ist.

Die Erschießung neuer Wissensquellen während der Wissensnutzung verbessert den **Reuse von Wissen**. Allerdings fördert WAVES auch bereits während der Wissensartikulation den Reuse beispielsweise durch die Linkvervollständigung über unterschiedliche Systeme, die von Wiquila angeboten wird. Dies führt letztendlich dazu, dass kostenintensive **redundante Wissensartikulation** vermieden werden kann.

Mit Woogle verbindet WAVES die **Wissensnutzung direkt mit der Wissensartikulation** und verbessert dadurch auch direkt die Motivation **neues Wissen zu artikulieren**. Die inverseSuche **identifiziert Informationsbedarf** und motiviert bereits vorhandenes relevantes Wissen zu teilen. Der Prozess des Wissenteilen und somit in die organisatorische Wissensbasis zu überführen wird einerseits durch die inverseSuche, aber auch durch PKM unterstützt. PKM ermöglicht eine systematische und strukturierte Erfassung von persönlichem Wissen und eine einfache **Überführung des persönlichen Wissens in eine organisatorische Wissenssphäre**.

6. Zusammenfassung

Wie in den vorangegangenen Kapiteln dargelegt, hat das Projekt WAVES sowohl aus wissenschaftlicher, aber auch aus praktischer Sicht zahlreiche Ergebnisse zu bieten:

- Verbesserung der existierenden Ansätze im Bereich Wissensnutzung und Wissensartikulation und darauf basierend Entwicklung einer, in der Praxis erprobten Wissensmanagement-Lösung, bestehend aus insgesamt sieben Werkzeugen
  - ein Ansatz und Werkzeug zur qualitativen Quelltextsuche (*KISSy*)
  - eine Enterprise-Suchmaschine zur Verbesserung des Informationszugriffs (*WavesIS*)
  - ein Verfahren zum Erfassen vom persönlichen Wissen (*HKW*)
  - ein leichtgewichtiges, „Wiki-Style“ Authoring-Engine zur Wissensartikulation (*Wiquila*)
  - ein Ansatz zum Umgang mit Kontextinformationen bei der Wissensnutzung (*IAS+*)
  - ein Wiki-basiertes und ein Such-basiertes Verfahren zur Unterstützung des bedarfsgetriebenen Wissensaustauschs (*WAVES inverse Suche*, *Woogle*)
- Kontinuierliche Analyse und Verringerung der mit der Einführung und Nutzung der Wissensmanagement-Lösung verbundenen ökonomischen, technischen und individualpsychologischen Barrieren

Die *Waves Integrierte Suche* integriert und erschließt eine Vielzahl unterschiedlicher Quellen - vom Anforderungsdokument über Sourcecode bis zu Adress- und Kontaktinformationen. Die Recherche nach Informationsartefakten kann parallel in allen indizierten Quellen erfolgen. Bei Bedarf - beispielsweise falls die Treffermenge zu groß ausfällt - bietet WavesIS umfangreiche und sinnvolle Möglichkeiten den Suchraum einzuschränken. Des Weiteren ist die Funktionalität „Ähnliche Begriffe“ bei der Recherche nach fehlerhaften Namen oder Bezeichnungen äußerst hilfreich. Durch WavesIS wird die „Wissensbasis“ des Unternehmens auch jenen Mitarbeitern erschlossen, die räumlich verteilt arbeiten – beispielsweise weil sie sich beim Kunden vor Ort, im Home-Office oder auf Geschäftsreise befinden.

Besonders hervorzuheben ist der revolutionäre Ansatz von *KISSy*, der die strukturbasierte Quelltextsuche mit der Betrachtung von Qualitätsaspekten erstmalig in der Literatur erfolgreich vereinigt. Die in der Projektevaluation involvierten Entwickler haben das Werkzeug sogar als „eine echte Hilfe zur Arbeitsoptimierung“ bezeichnet.

Während die *Waves Integrierte Suche* und *KISSy* den Informationszugriff erleichtern, ermöglichen die Werkzeuge *HKW* und *Wiquila* eine benutzbare und effiziente Wissensartikulation. HKW fokussiert primär auf das persönliche Wissensmanagement und Wiquila geht einen Schritt weiter und unterstützt den Wissensaustausch im Unternehmen. Mit *Wiquila* wurde eine Platfrom geschaffen, die die vorhandenen Informationsquellen aus Unternehmen, Projekten und externen Wissensquellen miteinander integriert, bietet neue und effizientere Möglichkeiten zur Wissensartikulation und vielfältigere Nutzungsmöglichkeiten des erfassten Wissens. Effizienter, weil viel Information aus dem Kontext ermittelt und vom Benutzer nicht eigens und redundant eingegeben werden muss. Die Artikulation kann eindeutiger, strukturierter und damit fehlerfreier erfolgen, ohne den Benutzer zusätzlich kognitiv zu belasten. Als besonders wertvoll wurde die durch Wiquila bereitgestellte Auto-Complete-Funktionalität auf der Basis der in der globalen Wissensbasis (WavesIS Backend) gespeicherten Wissensartefakte angesehen. Diese Funktionalität verringert die den Wikis eigene Gefahr zur Bildung von „Wucherungen“ signifikant und bietet darüber hinaus die Möglichkeit neues Wissen auf das bereits existierende (nicht zwangsläufig Wiki-basierte) Unternehmenswissen effizient aufzubauen.

Allerdings adressieren *Wiquila* und *HKW* nicht, *welches* Wissen artikuliert werden sollte und *wie* es mit Anderen ausgetauscht werden kann. Projektmitarbeiter haben in der Regel keine systematischen Informationen darüber, welches Wissen für ihr Projekt/ihre Organisation den größten Nutzen bringt. Anders ausgedrückt - ein effizienterer Wissensaustausch muss sich am Bedarf einer Organisation orientieren und dabei möglichst zielgerichtet erfolgen. Die beiden Werkzeuge *Woogle* und *Inverse Suche* basieren auf dem Paradigma des "bedarfsgetriebenen Wissensaustauschs". Dazu nutzen sie die aggregierten Suchanfragen die bei der integrierten Suche anfallen und priorisieren solche Informationsbedarfe, für die eine schlechte Ergebnisqualität besteht - d.h. nur wenig Wissen vorhanden ist. Die Komponente *Woogle* ergänzt MediaWiki um Suchfunktionalität und bietet für den Anwender eine deutliche Verbesserung bei der Artikulation von Wissen im Wiki. Die Hemmschwelle zur Erfassung eines neuen Eintrags wird somit verringert und die Akzeptanz von Wikis erhöht. Dank den innovativen Ansätzen von *Woogle* und der *Inversen Suche*, konnte der Stand der Technik im Bereich Wissensaustausch deutlich verbessert werden.

Doch nicht nur aus wissenschaftlicher Sicht, sondern auch aus praktischer Sicht konnten durchschlagende Ergebnisse erzielt werden. Vor allem konnte bereits während der Projektlaufzeit den Anwendungspartnern gezeigt werden, dass die werkzeuggestützte WAVES Wissensmanagement-Lösung in vielen Szenarien den unternehmensübergreifenden Wissensaufbau und –Austausch langfristig für alle Beteiligten ökonomisch interessant machen kann. Das WAVES Wissensmanagement steigert hier die Effizienz und Produktivität der Mitarbeiter und bringt Wissen schnell zu den richtigen Personen, dann, wenn sie es brauchen. Dies ist insbesondere in Wachstumsphasen von Unternehmen eine außerordentlich wertvolle Unterstützung.

Bereits während der Projektlaufzeit wurde ein sehr großer Wert auf die Vermarktung der Projektergebnisse gelegt. Beispielsweise zählen dazu zwei, von WAVES organisierte Eclipse DemoCamps, die dank des sehr großen Besucherkreises eine Vielzahl von Mittelständern in der Region ansprechen konnte. Die WAVES-Wissensmanagement-Lösung wurde unter dem Namen „TeamWeaver“ veröffentlicht und über die Webseite des WAVES-Partners Polarion der breiten Öffentlichkeit zur Verfügung gestellt.

Auch wenn WAVES bereits Fortschritte in vielen Bereichen vorweisen kann, so sind doch an WAVES anschließende Forschungsarbeiten sowohl aus wissenschaftlicher als auch aus der Sicht des Praktikers notwendig. Die Idee der qualitativen Quelltextsuche mit *KISSy* erwies sich als ein äußerst relevantes Gebiet. Um die Praxistauglichkeit des KISSy-Ansatzes weiter zu steigern muss die Analysezeit von Code-Repositories reduziert werden. Dazu bieten sich Verfahren aus dem Bereich inkrementelle Code-Analyse aber auch die Parallelisierung der Analysevorgänge auf mehrere Rechnereinheiten als eine sinnvolle Lösung. Weiterhin wurden im Rahmen des Projektes Ideen zur Verbesserung des Rankings der Treffer abgeleitet. Mittels Betrachtung der Versionsinformationen bzw. durch die Möglichkeit der Auswahl des Lizenzmodells bei der Suche ist eine zusätzliche Verbesserungen bei der Quelltextsuche erzielbar.

WAVES bietet ein breites Spektrum von Lösungsansätzen und eine Werkzeuglandschaft, um den Prozess der Wissensnutzung und Wissensartikulation in der Praxis zu verbessern. Die Werkzeuge sind zur Zeit allerdings lose miteinander gekoppelt und eine bessere Integration in einer Kollaborationsplattform wäre wünschenswert. Dazu bietet sich vor allem die heutzutage bei den Entwicklern so populäre Entwicklungsumgebung Eclipse als eine Basisplatform an.

In der Praxis kann häufig die Beobachtung gemacht werden, dass die seit längerer Zeit wachsenden Wiki-Systeme wegen ihrer steigenden Komplexität schwer gepflegt werden können: die Wikis „wuchern“. Während der Projektlaufzeit wurde die neuartige Idee der Messung von Wiki-Qualität abgeleitet. Einerseits muss anhand einer entsprechenden Qualitätsdefinition für Wikis eine entsprechende Werkzeugunterstützung angeboten werden, um auf die IST-Qualität von Wikis zu schließen, andererseits müssen Verfahren zur (semi-)automatischen Umstrukturierung („refactoring“) von Wikis erforscht werden.

Literatur

Abecker, A. (2004): *Business-Process Oriented Knowledge Management. Concepts, Methods, and Tools.* Dissertation. Institut AIFB an der Universität Karlsruhe.

Ackerman, M.S., Malone, T.W. (1990): Answer garden: a tool for growing organizational memory. In: Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems, New York, NY, USA, ACM 31–39

Alavi, M., Leidner, D.E. (2001): Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS Quarterly 25(1) 107–136

Althoff, K.-D., Decker, B., Hartkopf, S., Jedlitschka, A., Nick, M., Rech, J. (2001): Experience Management: The Fraunhofer IESE Experience Factory. In P. Perner (ed.), *Proc. Industrial Conference Data Mining.* Leipzig: Institut für Bildverarbeitung und angewandte Informatik.

Anderer, J., Bloch, R., Mohaupt, T., Neumann, R., Schumacher, A., Seng, O., Simon, F., Trifu, A., Trifu, M. (2006): Abschlussbericht QBench – Methoden und Werkzeuge zur Sicherung der inneren Qualität bei der Evolution objektorientierter Systeme, Schlussbericht des BMBF-geförderten Forschungsprojekts QBench, ISSN 0944-3037

Basili, V.R., Caldiera, G., Rombach, D. (1994): Experience Factory. In Marciniak, J.J. (Hrsg.): *Encyclopedia of Software Engineering. Band 1.* John Wiley & Sons.

Cockburn, A. (2003): *Agile Software-Entwicklung.* Mitp-Verlag.

Davenport, Th.H., Prusak, L. (1998): *Working Knowledge – How Organizations Manage What They Know.* Boston: Harvard Business School Press.

Davies, J., Duke, A., Sure, Y. (2004): OntoShare - An Ontology-based Knowledge Sharing System for Virtual Communities of Practice. *Journal of Universal Computer Science* 10(3):262-283.

Decker, B., Althoff, K.-D., Nick, M., Jedlitschka, A., Tautz, C., Rech, J. (2002): Die Fraunhofer IESE Experience Factory "Corporate Information Network (CoIN)" - Ein Beispiel für Geschäftsprozessorientiertes Wissensmanagement in Software Organisationen. In A. Abecker, K. Hinkelmann, H. Maus. & H.-J. Müller (Hrsg.), *Geschäftsprozessorientiertes Wissensmanagement.* Springer Xpert.press.

Decker, B., Althoff, K.-D., Rech, J., Klotz, A., Leopold, E., A. Voss (2004): Participative Process Introduction: A Case Study in the indiGo Project. *Journal of Universal Computer Science* 10(3):186-204.

Dingsøyr, T., Rolland, K.-H., Jaccheri, M.L. (2004): The Benefits and Limitations of Knowledge Management in Global Software Development. In: *3rd Int. Workshop on Global Software Development (GSD 2004), Edinburgh.* ICSE Workshop: A co-located event at the Int. Conf. on Software Engineering 2004.

Elliott, M.S., Scacchi, W. (2003): *Free Software Developers as an Occupational Community: Resolving Conflicts and Fostering Collaboration*. Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work GROUP' 03, Sanibel Island (USA).

Falbo, R.A., Arantes D.O., Natali, A.C.C. (2004): *Integrating Knowledge Management and Groupware in a Software Development Environment.* In: D. Karagiannis & U. Reimer, Hrsg.: 5th Int. Conf. on Practical Aspects of Knowledge Management - PAKM'2004. Berlin, New York, Heidelberg: Springer-Verlag. LNAI 3336, S. .94-105.

Fægri, T.E., Dingsøyr, T., Jaccheri, L., Lago, P., H. van Vliet (2005): Exploring Communities of Practice for Product Family Engineering. In: *7th Int. Workshop on Learning Software Organizations, Kaiserslautern.*

Gibbs, S., Tsichritzis, D., Casais, E., Nierstrasz, O., Pintado, X. (1990): *Class Management for Software Communities*. Communication of the ACM: 33(9):90-103.

Griss, M.L. (1995); *Software Resuse: Objects and Frameworks are not enough.* Hewlett-Packard Laboratories, Palo Alto, Kalifornien.

Hartmann, J., Sure, Y. (2004): An Infrastructure for Scalable, Reliable Semantic Portals. *IEEE - Intelligent Systems,* pp. 58-65. IEEE, 2004.

Henninger, S. (1997): Case-based knowledge management tools for software development. Autom. Softw. Eng. 4, 319–340

Irmak, U., Mihaylov, S., Suel, T., Ganguly, S., Izmailov, R. (2006): Efficient query subscription processing for prospective search engines. In: WWW '06: Proceedings of the 15th international conference on World Wide Web. 1037–1038

Kähkönen, T. (2005): Agile Methods for Large Organizations – Building Communities of Practice. In: *Agile Development Conference, Salt Lake City.*

Kauba, E. (1996); *Wiederverwendung als Gesamtkonzept - Organisation, Methoden, Werkzeuge*. In: OBJEKTspektrum; 1:20-27.

Kazakos, W. (2005): *Kontextkonforme Empfehlungen auf der Grundlage verteilter Informationen.* Dissertation. Institut IPD an der Universität Karlsruhe.

Klemke, R. (2002): *Modelling Context in Information Brokering Processes.* Dissertation. RWTH Aachen.

Kukulenz, D., Ntoulas, A. (2007): Answering bounded continuous search queries in the world wide web. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM Press 551–560

Maier, R. (2003): Knowledge Management Systems. Springer

Mentzas, G., Apostolou, D., Young, R., Abecker, A. (2002): *Knowledge Asset Management – Beyond the Product-centric and the Process-centric Approach.* Berlin, New York, Heidelberg: Springer-Verlag.

Murphy, G.C., Kersten, M., Findlater, L. (2006): How are java software developers using the eclipse ide? IEEE Softw. 23(4), 76–83

Paul, D.L., McDaniel, R.R. (2004): *A Field Study of the Effect of Interpersonal Trust on Virtual Collaborative Relationship Performance*. MIS Quarterly: 28(2):183-227.

Prechelt, L., Unger, B. (2001): An Experiment Measuring the Effects of Personal Software Process (PSP) Training. *IEEE Transactions on Software Engineering* 27(5):465-472.

Prechelt, L., Unger, B., Philippsen, M., Tichy, W.F. (2002): Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance. *IEEE Transactions on Software Engineering* 28(6):595-606.

Prechelt, L., Unger, B., Tichy, W.F., Brössler P., Votta, L.G. (2001): A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions. *IEEE Transactions on Software Engineering* 27(12):1134-1144.

Probst, G., Raub, S., Romhardt, K. (1999): *Wissen managen: Wie Unternehmen ihre wertvollste Ressource optimal nutzen* (3. Auflage), Wiesbaden: Gabler.

Raymond, E.S. (2000): *The Magic Cauldron*, http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/.

Schmidt, A. (2004): Kontext-Middleware zur Verwaltung dynamischer und unvollkommener Kontextinformationen. In: H. Höpfner & G. Saake (Hrsg.): *Workshop "Grundlagen und Anwendungen mobiler Informationstechnologie",* Heidelberg. Magdeburg: Otto-von-Guericke Universität.

Schmidt, A., Winterhalter, C. (2004): User-Context Aware Delivery of E-Learning Material: Approach and Architecture. *Journal of Universal Computer Science* 10(1):28-36.

Singer, J., Lethbridge, T., Vinson, N., Anquetil, N. (1997): An examination of software engineering work practices. In: CASCON 1997. Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research, p. 21. IBM Press

Staab, S., Studer, R., Hrsg. (2003): *Handbook on Ontologies.* Springer Series on Handbooks in Information Systems. Berlin, New York, Heidelberg: Springer-Verlag.

Tautz, C., Gresse, C., von Wangenheim (1998): *REFSENO: A Representation Formalism for Software Engineering Ontologies.* IESE-Report 015.98/E. Kaiserslautern: Fraunhofer IESE.

Trifu, M. (2003): Analysis and Extraction of Subsystem Structures. Diplomarbeit. Institut IPD, Universität Karlsruhe.

Yang, B., Jeh, G. (2006): Retroactive answering of search queries. In: WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, ACM Press 457–466

Wenger, E. (1998): *Communities of Practice: Learning, Meaning and Identity.* Cambridge University Press (1998).

Wenger, E. , McDermott, R. & Snyder, W. (2002): *Cultivating Communities of Practice: A Guide to Managing Knowledge.* Boston: Harvard Business School Press.

Wille, C., Abran, A., Desharnais, J.M., Dumke, R.R. (2003): The Quality Concepts and Subconcepts in SWEBOK: An Ontology Challenge. In: *Int. Workshop on Software Measurement (IWSM), Montreal.*

Die WAVES-Publikationen

Geisser, M., Happel, H.-J., Hildenbrand, T., Korthaus, A., Seedorf, S. (2008): New Applications for Wikis in Software Engineering. In *Proceedings of the Multikonferenz Wirtschaftsinformatik 2008 (MKWI'08),* pp -, Munich, Germany.

Happel, H.-J., Seedorf, S. (2006): Applications of Ontologies in Software Engineering. *Workshop on Sematic Web Enabled Software Engineering (SWESE) on the 5th International Semantic Web Conference (ISWC 2006),* Athens, Georgia, 5-9 November 2006

Happel, H.-J. und Schmidt, A. (2007): Knowledge maturing as a process model for describing software reuse. In *Proceedings of the 9th International Workshop on Learning Software Organizations (LSO 2007)*, Potsdam, März 2007.

Happel, H-J., Schuster, T., Szulman, P., Traphöner, R., (2007): Kontext-Strategien im Software Engineering: Stand der Technik. WAVES Projektergebnis

Happel, H-J., Kuttruff, V., Romberg, T., Szulman, P., Völkel, M. (2006): Stand der Technik, Bewertung und Empfehlung zu integrierender Kerntechnologien. WAVES Projektergebnis

Happel, H.-J., Seedorf, S.: Ontobrowse (2007): A Semantic Wiki for Sharing Knowledge about Software Architectures. *Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE),* Boston, July 2007

Happel, H.-J., Stojanovic, L., Stojanovic, N. (2007): Fostering knowledge sharing by inverse search. *Proceedings of the 4th International Conference on Knowledge Capture*, Whistler, BC, Canada, October 28 - 30, 2007. K-CAP '07. ACM Press, New York, NY.

Happel, H.-J., Stojanovic, L. (2008): Analyzing Organizational Information Gaps. *In Proceedings of the 8th International Conference on Knowledge Management (I-KNOW 2008)*

Happel, H.-J., Treitz, M. (2008): Proliferation in Enterprise Wikis. *Proceedings of the 8th International Conference on the Design of Cooperative Systems (COOP'08)*

Happel, H.-J., Schuster, T., Szulman, P. (2008): Leveraging source code search for reuse. In: *Proceeedings of the 10th International Conference on Software Reuse*, 25 - 29 May 2008, Beijing China.

Happel, H.-J., Seedorf, S. (2008): Documenting Service-Oriented Architectures with Ontobrowse Semantic Wiki. In *Proceedings of the Multikonferenz Wirtschaftsinformatik 2008 (MKWI'08)*

Happel, H.-J., Steinbauer, I. (2008): Learning How Developers Search for Source Code with Implicit Relevance Feedback. *Proceedings of the 10th International Workshop on Learning Software Organizations (LSO 2008)*

Happel, H.-J., Maalej, W., Stojanovic, L. (2008): TEAM: Towards a Software Engineering Semantic Web. Workshop on Cooperative and Human Aspects of Software Engineering (CHASE) at *International Conference on Software Engineering (ICSE)* 2008, Leipzig, Germany.

Happel, H.-J., Tim Romberg (2008): Wissensaustausch in der verteilten Softwareentwicklung mit Eclipse, *Eclipse DemoCamp Karlsruhe*, 26.6.2008

Maalej, W., Romberg, T. (2008): Agiler Wissensaustausch für verteilte Software-Teams, *Eclipse DemoCamp München*, 30.6.2008

Panagiotou, D., Maalej, W., Happel, H.-J. (2008): Towards Effective Management of Software Knowledge Exploiting the Semantic Wiki Paradigm. *Tagungsband der Konferenz "Software Engineering 2008" des GI-Fachbereichs "Softwaretechnik",* München 2008

Romberg, T. (2007): Stand der Technik, Bewertung und Empfehlung für die Sharing Engine, WAVES Projektergebnis

Romberg, T., Happel, H.-J., Koser, M., Rohfleisch, F., Sbreszny, P. (2006): Wikis im Unternehmenseinsatz - Strategische Einsatzmöglichkeiten und Potenziale. Vortrag im Rahmen des *Innovationsprogramms Web2.0 - Potenziale des Internets der zweiten Generation*. Pforzheim, Karlsruhe, Heilbronn; Oktober / November 2006.

Romberg, T., Szulman, P. (2008): Agile and Distributed Knowledge Management using Next-Generation Wikis (Project WAVES). *Tagungsband der Konferenz "Software Engineering 2008" des GI-Fachbereichs "Softwaretechnik",* München 2008

Studer, R., Abecker, A., Romberg, T., Happel, H.-J., Kuttruff, V., Szulman, P. (2006): Stand der Wissensverarbeitung in der verteilten Software-Entwicklung am Beispiel des Verbundprojekts WAVES (2006-2008), Beitrag zum *Fachband des Sonderkolloquiums „Entwicklung der Informatikforschung in Deutschland im Zeitraum vom ‚Ersten DV-Programm' bis zu ‚IT-Forschung 2006'" von Prof. Dr. Bernd Reuse*

Völkel, M. (2007): From Documents to Knowledge Models. In Prof. Dr.-Ing. Norbert Gronau (Eds.), *Proceedings of the 4th Conference on Professional Knowledge Management,* volume 2, pp. 209 - 216. GITO mbh, Berlin, März 2007.

Völkel, M. (2007): A Semantic Web Content Model and Repository in *Proceedings of the 3rd International Conference on Semantic Technologies (I-SEMANTICS)*.

Völkel, M., Haller, H., Abecker, A. (2007): Modelling Higher-Level Thought Structures – Method & Tool in *Proceedings of Workshop on Foundations and Applications of the Social Semantic Desktop*. October 2007.

Völkel, M. (2008): Hypertext Knowledge Workbench, In Christoph Lange (ed.), *Proc. of the Third Workshop on Semantic Wikis – The Wiki Way of Semantics*. June 2008.

Völkel, M., Abecker, A. (2008), "Cost-benefit analysis for the design of personal knowledge management systems", in Cordeiro J. and Filipe J. (Eds.), *ICEIS 2008 - Proceedings of the Tenth*

International Conference on Enterprise Information Systems, Volume AIDSS, Barcelona, Spain, June 12-16, Springer, Berlin, pp. 95-105.

Vrandecic, D., Völkel, M., Haller, H., Studer, R. (2007): Semantic Wikipedia in *Journal of Web Semantics*. December 2007.

Sämtliche WAVES-Projektergebnisse können über den WAVES-Webserver http://waves.fzi.de bezogen werden.