

WAVES – Wissensaustausch bei der verteilten Entwicklung von Software

Volker Kuttruff

FZI Forschungszentrum Informatik an der Universität Karlsruhe
Haid-und-Neu-Straße 10-14
76131 Karlsruhe

Kurzfassung

Ziel des Verbundprojekts „WAVES – Wissensaustausch bei der verteilten Entwicklung von Software“ ist es, in verteilten Softwareprojekten den Aufbau und den Austausch von informellem Wissen zu fördern und seine schrittweise Strukturierung und Vernetzung zu unterstützen. Hierbei soll auch die Integration verschiedener überlappender Wissenssphären unterstützt werden: (1) persönliches Wissen; (2) organisationsinternes bzw. team- oder projektbezogenes Wissen; und (3) öffentliches, allgemein zugängliches Wissen. Um das Ziel zu erreichen, wird WAVES einen Lösungsansatz auf zwei Ebenen verfolgen: der methodischen Ebene und der technischen Ebene. Auf methodischer Ebene wird ein sehr empirischer und fallstudienorientierter Ansatz gewählt, welcher die Ergebnisse der technischen Entwicklungsarbeit, der Anwendungsanalyse und der sozio-ökonomischen Untersuchungen verbindet. Basis für diese Fallstudien und damit ein zentraler Teil von WAVES wird eine zu entwickelnde Open-Source-Plattform sein. Neben der Integration bestehender Open-Source-Komponenten zum Wissensmanagement wie z.B. Wikis werden hier Erweiterungen und Innovationen im Bereich der Wissensartikulation und der Metadatenerzeugung aus unstrukturierten Artefakten sowie der effektiven Wissensnutzung durch kontextsensitive, ontologiebasierte Retrieval-Methoden eine wichtige Rolle spielen. Das Projekt befindet sich derzeit in der ersten Iteration eines iterativen Entwicklungsprozesses und hat die Erfassung der Anforderungen an die WAVES-Plattform sowie die Bestimmung des Stands der Technik weitestgehend abgeschlossen.

1. Einleitung und Vorstellung des Themenkomplexes

Ausgangssituation

Die Produktivität in der Softwareentwicklung basiert auf mindestens den folgenden Elementen, deren Zusammenspiel das Gesamtergebnis bestimmt:

1. mächtige Basistechnologien
2. weitgehende Automatisierungstechniken
3. durchdachte Prozesse und Rollen
4. qualifizierte Entwickler

Die ersten drei Themenfelder sind typische Forschungsthemen im Software Engineering (SE). Bei den Basistechnologien beispielsweise gibt es heute mächtige Bibliotheken, Frameworks und Middleware, so dass Entwickler sich stärker auf den fachlichen Teil der Anwendung konzentrieren können. Code-Patterns, AOP und MDA sind Beispiele für Techniken, die den Aufwand für manuelles Kodieren reduzieren.

Im gleichen Maß, wie die Faktoren (1) und (2) Fortschritte machen, steigen aber die Anforderungen an Wissen und Können der Entwickler im Umgang mit diesen Basistechnologien und Automatisierungsansätzen. Die Herausforderung liegt somit darin, den menschlichen Faktor „Qualifikation und Können“ im Gleichmaß mit den anderen Faktoren zu steigern (s.w.u.). Wissen stellt in der Softwareentwicklung somit einen nicht zu unterschätzenden Produktivitätsfaktor dar.

Projektziel

Ziel von WAVES ist es, in verteilten Softwareprojekten den Aufbau und Austausch von informellem Wissen zu fördern und seine schrittweise Strukturierung und Vernetzung zu unterstützen. Dieser Wissensaustausch ist eine Ergänzung zu gängigen Ansätzen der Wiederverwendung und zielt auf die effiziente Nutzung und Weiterentwicklung eines wesentlichen Wettbewerbsfaktors erfolgreicher Firmen und Projektteams ab, nämlich das in der täglichen Arbeit gewonnene, persönliche Erfahrungswissen in der Domäne, mit dem Kunden und mit spezifischen Werkzeugen und Methoden. Dabei wollen wir einen möglichst transparenten Zugang zu drei überlappenden „Wissens-Sphären“ erlauben: (1) zu persönlichem Wissen; (2) zu organisationsinternem bzw. team- oder projektbezogenem Wissen; und (3) zu öffentlichem, allgemein zugänglichem Wissen.

Unser *methodischer Lösungsansatz* ist in Termini des Wissensmanagements (WM) die Unterstützung von *Communities of Practice* im Software Engineering. Das Projekt ist stark empirisch und fallstudienorientiert angelegt und wird zur Erzeugung praxistauglicher und nachhaltiger Ergebnisse technische Entwicklungsarbeit, Anwendungsanalyse und sozio-ökonomische Untersuchungen verbinden. Wir verknüpfen somit Anwendungsinnovation und empirische Forschung mit technischer Konsolidierung des Standes von Wissenschaft und Praxis, sowie technologischer Basisinnovation.

Hinsichtlich *technologischer Konsolidierung* werden wir eine erweiterbare, für das Software Engineering spezifische, *Open-Source*-Plattform zum Wissensaustausch über Standort-, Projekt- und Unternehmensgrenzen hinweg entwickeln. Diese soll zweckgerichtet und benutzerfreundlich viele, im *Open Source* bereits existierende, Basisfunktionalitäten für online Kommunikation und Austausch, sowie Software Engineering Unterstützung, verknüpfen und Synergien aus ihrem Zusammenspiel ziehen.

Hinsichtlich *technologischer Basisinnovation* werden wir:

- *Wissensartikulation und Metadatenerzeugung* durch erweiterte Wiki-Ansätze sowie die Analyse des aktuellen Arbeitskontexts vereinfachen und teilautomatisieren, und
- effektive *Wissensnutzung* durch kontextsensitive, ontologiebasierte Retrieval-Methoden und die Einbindung aktiver Wissensinhalte präziser und benutzerfreundlicher machen.

Wikis als „leichtgewichtiges“, leicht verständliches und überall verfügbares Werkzeug zum Dokumentieren und Verlinken haben in erstaunlich kurzer Zeit ihren Siegszug durch die Entwicklergemeinden angetreten. Diese sollen weiterführend zum Strukturieren von Information und zum Formulieren formaler Aussagen (Metadaten u.ä.) benutzt werden können. Informationsmanagement und Kontextverarbeitung sollen auf Software Lifecycle *Referenzontologien*

für Informationstypen und Informationsflüsse basieren. Die Wissensnutzung profitiert von der unaufdringlichen, aktiven Präsentation relevanter Inhalte, der Einbindung aktiver Inhalte (Dienste), und der Ergänzung asynchroner Kommunikation (primär über Texte) durch *Social Networking*. Synergien entstehen beispielsweise, wenn Softwareanalyseverfahren Strukturen für die Wiki-Verlinkung zur Verfügung stellen.

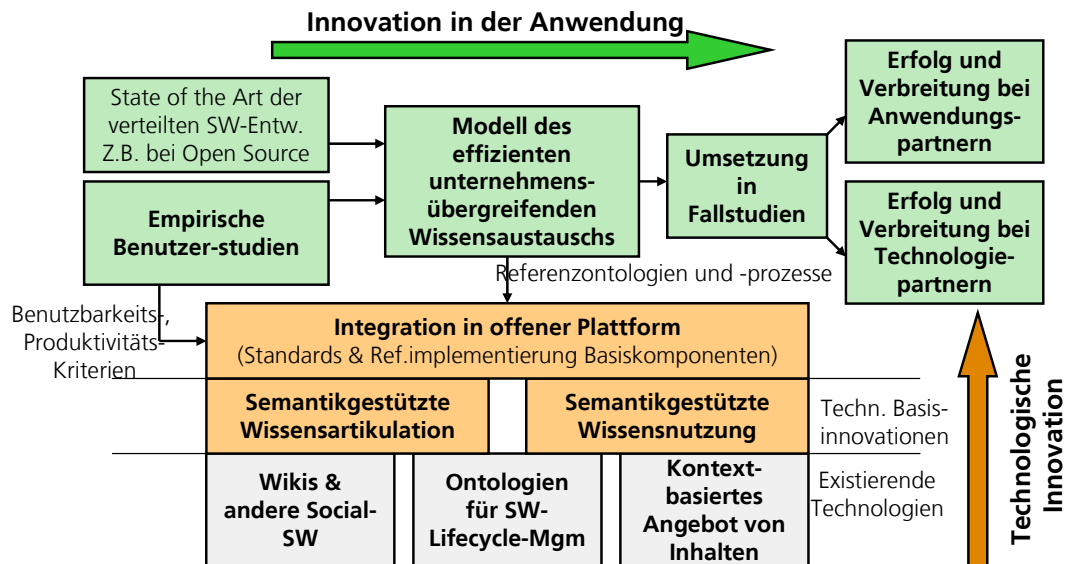


Abbildung 1: Überblick Lösungsansatz

Die *Nachhaltigkeit* unserer Projektergebnisse, deren Wertschöpfung und *Verwertbarkeit* auch über das Projektende hinaus bestehen bleiben, sehen wir in zwei Projektprinzipien begründet:

1. Der Praxisorientierung der Arbeiten mit gleichzeitigem Aufsetzen aussagekräftiger Prototypen und *Show Case* Implementierungen und mit besonderer Berücksichtigung der nicht-technischen, methodischen und sozio-ökonomischen Erfolgsfaktoren.
2. Der Entwicklung einer offenen Plattform, die sowohl kommerzielle als auch *Open-Source*-Module in ihrer Nützlichkeit multipliziert.

Beide Prinzipien sind im Projektkonsortium, in der Ressourcenallokation und im Arbeitsplan angemessen umgesetzt.

Das *Nutzenpotential* unseres Ansatzes für Endanwender lässt sich in folgenden Aspekten zusammenfassen:

- Erhöhung des Vernetzungsgrades (auch standort- und unternehmensübergreifend) von Software-Spezialisten
- damit einhergehend schneller und flächendeckender Aufbau und Verbreitung neuer Expertise (zu neuen Trends, Technologien usw.)
- Verringerung des Initialaufwands beim Übergang zu standort- und unternehmensübergreifenden Entwicklungsprozessen
- Schaffung der Basis für eine unternehmensübergreifende „Wiederverwendungskultur“ – durch mehr informelle, soziale Vernetzung und Vertrauensaufbau.

Die Herangehensweise für unseren Projekt-Arbeitsplan ergibt sich durch eine theoretische und empirische Analyse von *Barrieren und Erfolgsfaktoren* für das Community-basierte Wissensmanagement im verteilten Software Engineering. Diese Barrieren lassen sich in den technisch-strukturellen und in den sozio-ökonomischen Bereich einordnen. Wir arbeiten dann an einer schrittweisen Behebung der identifizierten Probleme; dabei sind die oben angesprochenen Maßnahmen zur Aufwandsreduktion (*Open Source*, Referenzontologien) und Qualitätsverbesserung von Wissensartikulation (Wiki-Ansatz, Kontext), Wissensaustausch (ontologiebasiertes Retrieval, *Social Networking Support*) und Wissensnutzung (kontextsensitive Präsentation, aktive Inhalte) zunächst zentral auf der technischen Seite; gleichzeitig liegt aber auch ein deutliches Gewicht auf der fallstudienorientierten Arbeit an nicht-technischen Faktoren.

Insgesamt soll die freie Implementierung zusammen mit überzeugenden Prototypen die breite Masse an Entwicklern und Softwarefirmen an die neuen Möglichkeiten heranführen, so dass sich ein *Wachstumspfad* hin zu den kommerziellen Angeboten der im Projekt involvierten, aber auch weiterer, Technologieanbieter ergibt.

2. Stand der Technik und Projektstatus

Wissensmanagement und Wiederverwendung. Die Komplementarität unserer Projektidee zu „Mainstream“ SE-Ansätzen (und daraus resultierend die gegenseitige Hebelwirkung von Nutzeffekten) argumentiert z.B. (Griss, 1995): Um eine Wiederverwendungskultur erfolgreich im Unternehmen zu etablieren, sei die Aus- und Weiterbildung der Mitarbeiter eine wichtige Voraussetzung, um mit allgemein steigenden Anforderungen (Kauba, 1996) in den technischen Grundlagen schritthalten zu können. Gerade die Vermittlung genereller wiederverwendungsspezifischer Kenntnisse sei die Grundlage für die Einbettung einer Wiederverwendungskultur.

Bei agilen Ansätzen nach Cockburn sind Kommunikationsbarrieren sowie Zeit- und Energiekosten für Ideentransfer ein großes Manko der verteilten Arbeit. Mit dem Aufbau von Vertrauen durch überzeugende Kompetenz (Paul & McDaniel, 2004) könne diese Hürde überwunden werden. Software-Communités, wie in der *Open-Source*-Gemeinde ansatzweise etabliert (z.B: GNU enterprise (Elliot & Scacchi, 2003)), bieten Entwicklern die Möglichkeit, ihr Wissen zielorientiert aufzustocken und ihr Können unter Beweis zu stellen. Bereits (Gibbs et al., 1990) stellen fest, dass sog. „Software Information Systems“ – Vorreiter der Software Communities - für „large scale class reuse“ erst nützlich werden, wenn man sie *öffentlich*, also *unternehmensübergreifend*, nutzt.

Steigerung von Qualifikation und Können. Schulungsbasierte Ansätze erweisen sich dazu in der Praxis als schwerfällig, schlecht akzeptiert und oft zu unspezifisch für spezielle Situationen und neue Themen. Daher werden Schulungen vor allem bei Neueinsteigern angewandt. Entsprechend ihrem Naturell, den Charakteristika der Themen und den typischen Arbeitskontexten, beziehen dagegen erfahrenere Software Ingenieure ihr Wissen primär durch *apprenticeship learning* und *community-basierte* Kommunikation über ihr soziales Netzwerk bzw. als *on-the-job learning* aus dem Selbststudium (wobei auch hierfür die Impulse oft aus dem Netzwerk kommen). Insgesamt können also *Communities of Practice* (CoP, Wenger et al., 2002) und *Communities of Interest* (CoI) eine enorme Rolle in der Erzeugung, Verbreitung und Aktualisierung von Spitzenqualifikationen im Software Engineering spielen. Dieses Potenzial wird montan aber nur in ersten Experimenten und Überlegungen untersucht (vgl. (Fægri et al., 2005; Dingsøy et al., 2004)).

Tatsächlich tragen *agile Methoden* der Softwareentwicklung dieser Tatsache insofern Rechnung, als sie in ihren Vorgehensmodellen intensiv die Kommunikation und das systematische Kombinieren der Erfahrungen verschiedener Know-How-Träger umsetzen, so dass obige Lernmethoden indirekt zum Tragen kommen. Sie bewirken also lokal, projektbezogen, ähnliche Effekte, wie wir sie hier insgesamt als Ziel verfolgen. Allerdings sind solche Ansätze nicht auf räumlich, zeitlich oder organisatorisch entkoppelte Kooperationen abgestellt, sie nutzen Kooperation nur punktuell, statt die community-weite Wissensbasis weiterzuentwickeln, und sie binden nicht systematisch externe Wissensquellen ein (vgl. auch (Kähkönen, 2005)).

Web-basierte Community Portale. Auf der anderen Seite sind CoPs / CoIs im Wissensmanagement (Davenport, 1998; Probst et al., 1999; Mentzas et al., 2002) etablierte Ansätze, die aber wenig kontext- und projektspezifisch orientiert sind. Diese Communities nutzen schon heute das Internet intensiv. Web-basierte Community Portale stellen online Wissensrepositorien und synchrone wie auch asynchrone Kommunikationsmechanismen bereit, welche Community-Aufbau, -Zusammenarbeit und -Wissensaustausch unterstützen. Semantische Community Web Portale (Hartmann & Sure, 2004; Davies et al., 2004) realisieren ähnliche Funktionalitäten, jedoch mit verfeinerten Informationsintegrationsmechanismen aufgrund ontologiebasierter Metadaten, und mit fortgeschrittenen, ontologiebasierten Informationssuch-, -auswertungs- und -visualisierungsmöglichkeiten.

Außerhalb der Forschungslabore werden viele Community Portale – eher in Form passiver Informationssammlungen – von Herstellern betrieben, andere speisen sich aus *Open-Source-Projekten* wie PHP. Leider ist die Suche in diesen Foren oft rudimentär und ineffizient. Naturgemäß existieren auch keine Integration mit unternehmensinternen Wissenspools und keine speziellen Vernetzungsmöglichkeiten mit anderen Experten.

Erst wenn solche Mechanismen in realen Anwendungsfeldern genutzt werden, werden die schwierigen motivatorischen und rechtlichen Fragen aufkommen, wie man einerseits einen Kreislauf von Geben und Nehmen „wertvollen“ (!) Wissens in Gang setzt, andererseits aber auch das natürliche Interesse von Firmen respektiert, ihren Wettbewerbsvorteil durch genaue Kenntnis von Märkten, Domänen o.ä. zu behalten. Hier liegt ein Schwerpunkt unserer Fallstudienarbeit im vorliegenden Vorhaben.

Wissensmanagement im Software Engineering. Aufbauend auf dem *Experience Factory* Konzept von Basili und Rombach (Basili et al., 1994; Althoff et al, 2001) sind die wichtigsten Arbeiten in diesem Kontext sicher im Umfeld des Fraunhofer IESE und der dort initiierten Workshop-Reihe *Learning Software Organizations* entstanden. Dort werden Lessons Learned und Prozessverbesserungen entlang formaler Prozessmodelle gesammelt. Verfeinerte ontologiebasierte Beschreibungen von Artefakten und Kontexten sind die Basis fallbasierter Retrieval-Verfahren (vgl. z.B. (Decker et al., 2002)). Der Ansatz ist primär unternehmensintern gedacht, während einer unserer Forschungsschwerpunkte gerade das Zusammenspiel von internen und externen Wissensquellen und Austauschmechanismen sein wird. Im indiGo Projekt (Decker et al., 2004) werden moderne Groupware-Instrumente zur kollaborativen Wissens-erzeugung eingesetzt. Auch hier geht es allerdings nur um Lernen über Software-Entwicklungsprozesse, nicht um allgemeine Themen und nicht um externe Quellen, so dass auch Kontextbegriffe hier a priori schon enger eingegrenzt sind. Im Gegensatz zu indiGo sind wir auch mit Wikis und Blogs, aktiven Wissensinhalten, Werkzeuganbindung, Code-Analyse zur Kontextbestimmung und *Social Networking* Software zur Kooperationsunterstützung, vielseitiger und aktueller hinsichtlich der Lern-Instrumentierung. Unser Ansatz kann als Weiterentwicklung dieser Ideen aufgefasst werden und wird auch Elemente des IESE-Ansatzes aufgreifen, gerade im methodischen Bereich und für die Ontologiekonstruktion. Organisatorisch ist

die enge Kooperation durch die Einbindung von empolis im WAVES Konsortium sichergestellt, welches in u. a. in abgeschlossenen Projekten der ersten Förderrunde mit Fraunhofer IESE aktiv war.

Projektstruktur

Das Konsortium umfasst *Forschungspartner* (öffentliche Forschungsinstitute), *Technologiepartner* (Anbieter von Werkzeugen im Bereich Software-Engineering und/oder Wissensmanagement) und *Anwendungspartner*. Die Forschungspartner sind primär für die Entwicklung neuer, innovativer Ansätze zuständig und stellen insgesamt den Bezug zu den neuesten wissenschaftlichen Erkenntnissen sicher. Sie unterstützen die Technologie- und Anwendungspartner in den jeweiligen Entwicklungs- und Fallstudienaktivitäten. Die Technologiepartner stellen ihre Kompetenzen und Erfahrungen in den jeweiligen Entwicklungsaktivitäten der WAVES-Plattform zur Verfügung. Ziel ist hier, die im Rahmen von WAVES entwickelten Ansätze schnellstmöglich geeignet aufzubereiten, um so einen Wettbewerbsvorsprung zu erreichen. Bei den Anwendungspartnern handelt es sich sowohl um größere, erfahrene Unternehmen, so dass eine hohe Professionalität der Fallstudien und das Erreichen einer kritischen Masse an Benutzern und Inhalten sichergestellt sind, aber auch um kleinere Unternehmen, welche insbesondere die Eignung der WAVES-Plattform für KMUs evaluieren sollen.

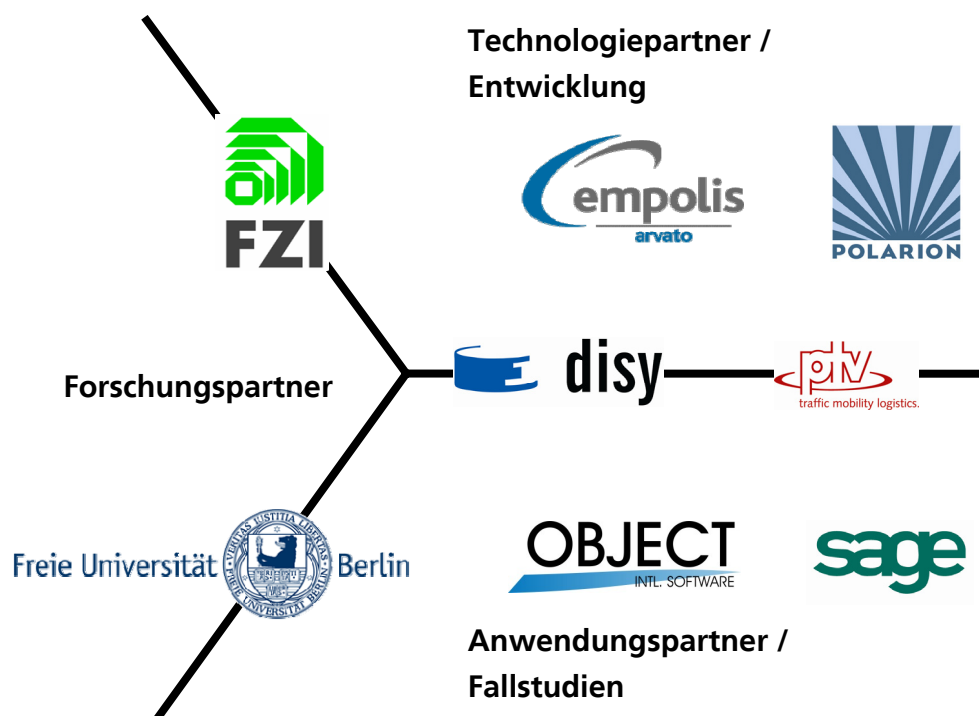


Abbildung 2: Überblick Projektpartner und -rollen

Das Projekt ist auf eine Projektlaufzeit von 32 Monaten hin ausgelegt, die in mehreren Iterationen durchlaufen werden (vgl. Abbildung 3). Am Ende einer Iteration steht ein Release der zu entwickelnden WAVES-Plattform. Die Bearbeitung von Fallstudien beginnt bereits sehr früh, um so die für die WAVES-Plattform notwendigen Praxisanforderungen in einer Rückkopplungsschleife zu ermitteln. Dies sichert eine hohe Relevanz und Praktikabilität der Projektergebnisse und vereinfacht die Einführung der Technologie bei den Projektpartnern.

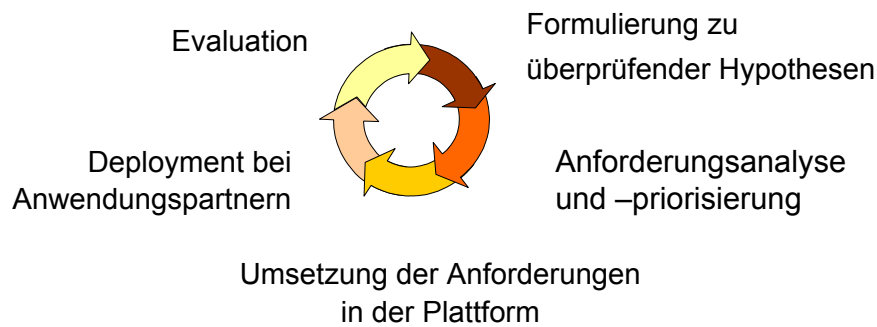


Abbildung 3: Iterative Entwicklung

Lösungsarchitektur

Die folgende Abbildung zeigt die fachliche *Grobarchitektur* der WAVES-Plattform:

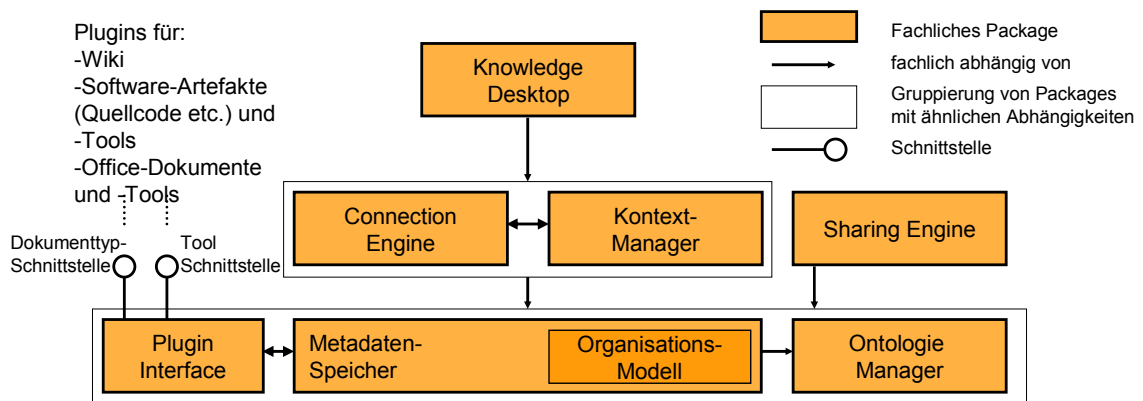


Abbildung 4: Überblick Grobarchitektur

In der (fachlichen) Basisschicht gibt es zunächst den in WM-Systemen üblichen *Ontologie-Manager* zur Verwaltung von Wissensstrukturen und den *Metadaten-Speicher*, der die Eigenschaften von verfügbaren Dokumenten und ihre Beziehung zueinander kennt. Zusätzlich enthält er, im *Organisationsmodell*, Informationen über Benutzer, Projekte, Aufgaben, Teams und externe Partner.

Die Verbindung zwischen dem Kernsystem und den speziellen Tools und Artefakten des SE wird über eine *Plug-in-Schnittstelle* hergestellt. Plug-ins haben sowohl (1) dokumententypspezifische Aufgaben (wie die Dokumentanalyse zum Parsen seiner Struktur und zum Einlesen von Metadaten) als auch (2) eher toolspezifische Aufgaben wie das Navigieren zu einer bestimmten Stelle im Dokument oder der Aufruf einer spezifischen Operation. Eine besondere Rolle kommt dem *Wiki-Plug-in* zu, da in Wiki-Dokumenten der größte Teil des Wissens, seiner Struktur und seiner Vernetzung dargestellt wird und darin bearbeitet wird.

Der *Kontext-Manager* steuert Anfragen an die Plug-in-Schnittstelle (zur Bestimmung des aktuellen Kontexts) und fragt, gesteuert von an die Domäne angepassten Kontextstrategien, aus dem Metadaten-Speicher potenziell nützliche Informationen für den Benutzer ab. Die *Connection Engine* ermittelt umgekehrt neue Verknüpfungen zwischen Wissensobjekten aus

den aktuellen Aktionen des Benutzers. Der *Knowledge Desktop* stellt eine integrierte Benutzerumgebung auf der Zielplattform zur Verfügung.

Die *Sharing Engine* steuert auf Basis des Organisationsmodells die Freigabe und den Zugriff auf Wissen der verschiedenen organisatorischen *Scopes* (persönlich, öffentlich, team- und unternehmensbezogen usw.).

Aktueller Stand

Derzeit befindet sich das Projekt am Anfang der ersten Iteration. Neben einer Erhebung der Technologielandschaft bei den Anwendungspartnern, welche in nicht unerheblichem Maße die zu unterstützenden Softwareartefakte der WAVES-Plattform mitbestimmt, stand die Anforderungsanalyse im bisherigen Mittelpunkt. Diese Anforderungsanalyse ist für die erste Iteration abgeschlossen und resultiert in einer Menge von erwünschten Use-Cases. Diese werden derzeit nach Wichtigkeit und im Hinblick auf die Realisierbarkeit in den einzelnen Entwicklungsiterationen bewertet und priorisiert.

Parallel zur Anforderungsanalyse wurde eine umfassende Ermittlung des Standes der Technik durchgeführt. Hierbei wurde ausgehend von der Grobarchitektur aus Abbildung 4 die existierenden Methoden, Technologien und Werkzeuge für die einzelnen Bereiche ermittelt und im Hinblick auf die Eignung bzgl. der WAVES-Plattform bewertet.

3. Erfahrungen, Bewertungen

Die Anforderungsanalyse hat gezeigt, dass neben der Neuformulierung von Wissen ein besonderes Augenmerk auf die Extraktion und Verknüpfung bereits implizit vorhandenen Wissens gelegt werden sollte. Hierbei sollten alle Arten von Artefakten innerhalb des Software-Entwicklungsprozesses berücksichtigt werden, angefangen bei unstrukturierten Artefakten wie Protokollen, Emails, Anforderungsdokumenten oder How-To-Dokumenten, über schon stärker strukturierte Artefakte wie Architekturmodelle oder UML-Modelle bis hin zu stark strukturierten Artefakten wie zum Beispiel Quelltext. Weiterhin verspricht die Extraktion von Metadaten aus den verschiedenen, im Allgemeinen isoliert zum Einsatz kommenden SE-spezifischen Entwicklungsdatenbanken sowie deren Verknüpfung eine wertvolle, bereits gut gepflegte Wissensquelle. Beispiele solcher Entwicklungsdatenbanken sind Systeme zum Anforderungsmanagement, zur Versionsverwaltung oder zur Fehlerverfolgung. Weiterhin ist bereits jetzt absehbar, dass für eine hohe Akzeptanz der WAVES-Plattform neben den rein technischen Merkmalen der Benutzbarkeit u. a. in Form geeigneter Benutzerschnittstellen große Aufmerksamkeit geschenkt werden muss.

4. Ausblick

Das Ende der ersten Entwicklungsiteration ist für Ende des Jahres geplant. Ausgehend von den erhobenen Anforderungen und der Untersuchung des Standes der Technik wird im Mittelpunkt kommender Aktivitäten die Auswahl geeigneter Technologien und Werkzeuge stehen, welche im ersten Release der WAVES-Plattform integriert werden sollen. Dieses erste Release wird somit im Wesentlichen eine erste Integration der Basistechnologien darstellen und die Basis für die Weiterentwicklung der Plattform in den folgenden Entwicklungszyklen legen.

Bereits auf Grundlage des ersten Release werden empirische Untersuchungen hinsichtlich Nutzerverhalten und Einsatzbarrieren durchgeführt. Diese Untersuchungen werden die An-

forderungen an die WAVES-Plattform innerhalb der zweiten Iteration maßgeblich mitbestimmen. Darüber hinaus stellen die so ermittelten Anforderungen eine hohe Praxisrelevanz sicher.

Neben der technischen Integration der Werkzeuge bzw. Entwicklungsdatenbanken wird die semantische Integration der darin bearbeiteten bzw. abgelegten Entwicklungsartefakte in Form einer geeigneten Referenzontologie ein wertvolles, auch über die WAVES-Plattform hinweg nutzbares Ergebnis darstellen.

Literatur

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

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

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

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

B. Decker, K.-D. Althoff, M. Nick, A. Jedlitschka, C. Tautz & J. Rech (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.

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

T. Dingsøyr, K.-H. Rolland & M.L. Jaccheri (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.

M.S. Elliott, W. Scacchi (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).

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

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

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

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

- T. Kähkönen (2005): Agile Methods for Large Organizations – Building Communities of Practice. In: *Agile Development Conference, Salt Lake City*.
- E. Kauba (1996); *Wiederverwendung als Gesamtkonzept - Organisation, Methoden, Werkzeuge*. In: OBJEKTSpektrum; 1:20-27.
- G. Mentzas, D. Apostolou, R. Young & A. Abecker (2002): *Knowledge Asset Management – Beyond the Product-centric and the Process-centric Approach*. Berlin, New York, Heidelberg: Springer-Verlag.
- D.L. Paul, R.R. McDaniel (2004): *A Field Study of the Effect of Interpersonal Trust on Virtual Collaborative Relationship Performance*. MIS Quarterly: **28**(2):183-227.
- G. Probst, S. Raub & K. Romhardt (1999): *Wissen managen: Wie Unternehmen ihre wertvollste Ressource optimal nutzen* (3. Auflage), Wiesbaden: Gabler.
- E. Wenger, R. McDermott & W. Snyder (2002): *Cultivating Communities of Practice: A Guide to Managing Knowledge*. Boston: Harvard Business School Press.