Beurteilung von Software Qualität stärken durch Ontologien und Wiederverwendung von Wissen

Stefan Strell[1], Vladimir A. Shekhovtsov[2] und Heinrich C. Mayr[3]

Abstract: Angemessene Qualitätsbeurteilung bei Software erreichen wird beeinträchtigt wenn Beteiligte kommunizierte qualitätsbezogene Informationen unterschiedlich wahrnehmen. Wir beabsichtigen diesen Kernpunkt durch Einführung eines breiteren Kontexts von Quality-Aware Software Engineering zu adressieren und diesen Zusammenhang durch Einführung des entsprechenden Software-Toolset, das es erlaubt bereits existierendes Wissen wiederzuverwenden und Werkzeuge zur Hilfe und Förderung der Qualitätsbeurteilung für SoftwareentwicklerInnen zur Verfügung zu stellen, beispielhaft einzuführen. Wir führen das spezifische Beispiel der Entwicklung des Tool-Support für qualitätsbewusste Softwareentwicklung basierend auf den Ergebnissen des Forschungsprojekts QuASE ein. Diese Lösung unterstützt Verständlichkeit und Wiederverwendung von kommunizierten Informationen sowie die Qualität von Entscheidungen basierend auf solchen Informationen durch Verwertung entsprechender Ontologien und Knowledge Bases. In diesem Artikel legen wir den Schwerpunkt auf die Komponenten um Wissensverwaltung des QuASE-Systems, speziell auf seine Knowledge Base (KB), die aus zwei Ontologien besteht: Eine *Site Ontology*, die das Site-spezifische Kommunikationsumfeld definiert, und eine Ontology die das gesamte notwendige Wissen zur Unterstützung der qualitätsbezogenen Kommunikation enthält. Wir beschreiben die gesamte Architektur des Systems und die Komponenten, die für Unterstützung der Wissensverwaltung zuständig sind, im Detail und stellen Informationen über bereits eingesetzte Applikationen dieser Softwarelösung zur Verfügung.

Keywords: Quality-Aware Software Engineering; Software Qualität; Knowledge Reuse; Ontologie; Knowledge Base.

1 Einleitung & Motivation

Seitdem es die Softwareentwicklung als eigenständiges Fachgebiet in der Informatik gibt, ist die Qualität und vor allem die Verbesserung ein oft diskutiertes Thema. Alle ExpertInnen sollten das Ziel haben, Software nicht nur für Maschinen und Computer zu entwickeln, sondern vor allem um den BenutzerInnen, also den Menschen, zu helfen besser zurecht zu kommen. Die Informatik und die Softwareentwicklung sind allerdings keine isolierten Fachgebiete, sondern hier arbeiten zum Teil ExpertInnen in den verschiedensten Bereichen, wie Industrie 4.0 und IoT, zusammen um gemeinsam Ziele,

[1] Alpen-Adria-Universität Klagenfurt, Institute for Applied Informatics, Universitätsstraße 65-67, A-9020 Klagenfurt am Wörthersee, sstrell@edu.aau.at
[2] Alpen-Adria-Universität Klagenfurt, Institute for Applied Informatics, Universitätsstraße 65-67, A-9020 Klagenfurt am Wörthersee, Volodymyr.Shekhovtsov@aau.at
[3] Alpen-Adria-Universität Klagenfurt, Institute for Applied Informatics, Universitätsstraße 65-67, A-9020 Klagenfurt am Wörthersee, Heinrich.Mayr@aau.at

die sog. *„Grand Challenges“* [GI16] zu erreichen. Zu diesen zählt nach [GI16] auch die Verlässlichkeit von Software, eines der wichtigen Qualitätskriterien der Softwareentwicklung und zugleich eines der umstrittensten. Nach Hoffmann in [Ho13] gibt es acht Kategorien um Software Qualität einzuteilen, wie in Abb. 1 veranschaulicht.

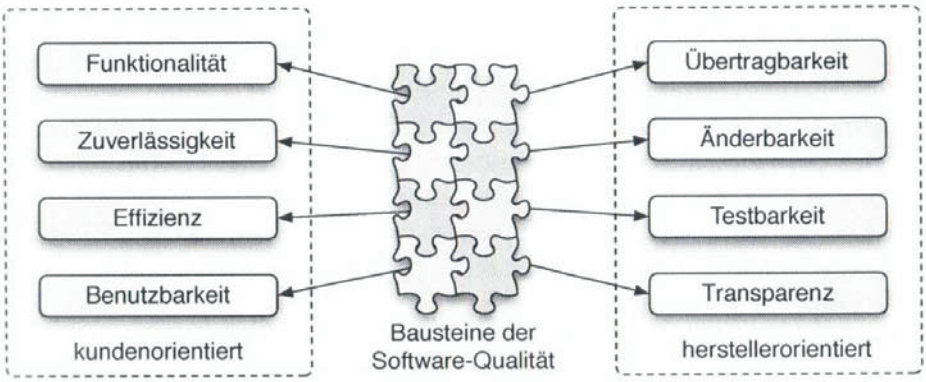

Abb. 1: Qualitätsmerkmale eines Software Produkts (aus [Ho13], S. 7)

Wie in [St16] von einem der Autoren bereits angemerkt gibt es keine eindeutige Definition für Software Qualität. Dieser Umstand macht es mitunter schwierig die Kommunikation zwischen den einzelnen Beteiligten in einem Softwareentwicklungsprojekt auf gleicher Ebene durchzuführen und ist ein wichtiger Aspekt im Forschungsprojekt QuASE[4]. Es wurde in [St16] eine Arbeitsdefinition verwendet, welche in Kapitel 3.1 ersichtlich ist. Das Themengebiet um Software Qualität befasst sich u.a. mit den funktionalen und nicht-funktionalen Anforderungen an ein konkretes Softwareprodukt. Zu den funktionalen Anforderungen werden die spezifischen Funktionen der Anwendung gezählt, also was die Software später auszeichnet (vgl. [St14]). Die nicht-funktionalen Anforderungen sind in der Regel komplexer und nicht einheitlich beschreibbar. Zu diesen zählen Sicherheit, Flexibilität, Usability, Performance und Interoperabilität (vgl. [CP09, St14]). Vor allem diese Anforderungen werden in der klassischen Softwareentwicklung nicht so stark berücksichtigt, bzw. immer wieder neu definiert, allerdings sind sie für die Qualitätsbeurteilung unerlässlich. Um diesen Umstand zu reduzieren und vor allem die Kommunikation zwischen den Beteiligten in einem Softwareprojekt zu verbessern, wurde das Forschungsprojekt QuASE initiiert.

In diesem Artikel zunächst werden die wichtigsten Begriffe definiert, die in der weiteren Folge zur Verwendung kommen und anschließend einleitende Worte über die qualitätsbewusste Softwareentwicklung vorgestellt. In Kapitel 3 wird das Forschungsprojekt QuASE und dessen Proof-Of-Concept erläutert. Kapitel 4 beschäftigt sich mit den Ontologien und der Knowledge Base im Forschungsprojekt QuASE. In

[4] Offizielle Webseite des Forschungsprojekts: http://quase-ainf.aau.at/

Kapitel 5 wird auf weiterführende Literatur verwiesen. Der Artikel schließt mit einer Zusammenfassung und einem Ausblick, sowie einer kurzen Wiederholung zur Beantwortung der Forschungsfrage aus [St16] ab.

2 Quality-Aware Software Engineering

2.1 Definitionen & Grundlagen

Ontologien & Knowledge Base

Der Begriff „Ontologie“ ist im Gegensatz zur „Knowledge Base“ nicht nur in der Informatik zu finden, sondern auch Bestandteil der Wissenschaft der Philosophie. Hier ist es vor allem die Metaphysik, die diesen Begriff geprägt hat (vgl. [Gu05] S. 51 ff., [St16], S. 9 ff.).

Die Knowledge Base (KB) ist das Konzept zur Digitalisierung, Repräsentation und Verwertung von Wissen. Diese werden entweder über komplexe Datenbanken oder mit Hilfe von Ontologien aufgebaut (vgl. [St16], S. 17). Anwendungen, die KBs verwenden, nutzen dafür zumeist komplexe Metriken und SPARQL als Query-Language.

Arbeitsdefinition von Software Qualität

Der Begriff „Software Qualität“ lässt sich nicht konkret mit genau einer Definition beschreiben. Das liegt vor allem an den vielen unterschiedlichen Qualitätsmerkmalen eines Software-Produkts. Denn jede/r SoftwareentwicklerIn und auch jede/r EndbenutzerIn hat eine andere Auffassung, welche Kriterien für ihn bzw. sie als Qualitätsmerkmale in einer Software vorhanden sein müssen. Nach der ISO/IEC-Norm 9126 wird der Begriff Software Qualität wie folgt definiert:

„Software Qualität ist die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte Erfordernisse zu erfüllen.“ ([Ho13], S. 6, vgl. [ISO00])

Da dies nur eine von vielen Definitionen über den Begriff „Software Qualität“ ist wurde in [St16] eine Arbeitsdefinition erstellt, die auf den gängigen ISO-Standards beruht:

„Die Qualität von Software wird zum einen durch die Fähigkeit zur Ausführung der gestellten Anforderungen und zum anderen durch die effektive und effiziente Projektdurchführung gekennzeichnet.“ ([St16], S. 29)

2.2 Qualitätsbewusste Softwareentwicklung

Die qualitätsbewusste Softwareentwicklung (engl. *Quality-Aware Software Engineering*) hat zum Ziel die Qualitätsbeurteilung in der Softwareentwicklung zu stärken. Vermehrt

wird dieser Ansatz in der Entwicklung von Automatisierungs- und modellbasierten Systemen eingesetzt (vgl. [St16], S. 47, [Ab14]). In [St16] wird der Begriff „Quality-Aware Software Engineering“ folgendermaßen definiert:

„Die qualitätsbewusste Softwareentwicklung unterscheidet sich vom klassischen Software Engineering vor allem dadurch, dass die Software Qualität und die nichtfunktionalen Anforderungen im Vordergrund stehen.“ ([St16], S. 47)

Ziel der qualitätsbewussten Softwareentwicklung und des Forschungsprojekts QuASE ist es, nicht Fehler zu vermeiden, sondern diese früher zu erkennen und aus ihnen besser lernen zu können. Eines der wichtigsten Ziele dieses Ansatzes ist es, *„die Kosten bei der durch ExpertInnen durchgeführten Wartung zu reduzieren“* ([St16], S. 47, vgl. [Ab14]). Hierbei soll nach [St16] und [Ab14] vor allem die Unsicherheit (*uncertainty*) reduziert und aus vorangegangenen Softwareprojekten effizienter gelernt werden (vgl. [St16], S. 47). Um diese Ansätze zu verfolgen werden, wie in [St16] ausführlich beschrieben, vor allem die ISO-Standards 9000 und 9001, sowie eine klare Arbeitsdefinition von Software Qualität hinzugezogen (vgl. [St16], S. 48). In der qualitätsbewussten Softwareentwicklung nutzt man bereits vorhandenes Wissen, das digitalisiert aufbereitet wird, um dieses effizienter zu verwenden. Eine gute Kommunikation unter den Beteiligten und ein klares Entwicklungskonzept zeichnen nach [St16] ebenfalls eine gute Basis dafür aus.

Diese Ansätze nutzt das Forschungsprojekt QuASE, deren Ziel eine Applikation zur Erleichterung der Softwareentwicklung durch eine stärkere Beurteilung von Software Qualität zu entwickeln. In den folgenden Kapiteln werden diese zentralen Themen behandelt und gezeigt, wie Ontologien und Knowledge Bases helfen können, die Qualität von Softwareprojekten zu erhöhen.

3 QuASE-Projekt: Ziele, Architektur & derzeitiger Stand

Das Forschungsprojekt QuASE wurde mit dem Primär-Ziel einen Proof-Of-Concept zu erstellen, der zur Stärkung der Qualitätsbeurteilung durch Wissensverarbeitung dient (vgl. [St16], Kap. 4 ff.). Abb. 2 zeigt das schematische Konzept der Applikation und wie die unterschiedlichen Sichtweisen der Beteiligten eines Softwareprojekts ergänzt und erklärt werden können.

Die automatische Verarbeitung von Wissen mit Knowledge Bases und Ontologien stand hierbei im Vordergrund. Denn nur so können unterschiedlich verwendete Fachbegriffe zusammengeführt und wiederverwendet werden. Die QuASE-Applikation besteht, wie in [St16] dargestellt, aus einer 3-Schichten-Architektur (GUI, Programm-Logik, Daten) (vgl. [St16], S. 59 ff.).

Die Implementierung der Verständlichkeitsverwaltung in QuASE verwendet Methoden zur Verarbeitung von natürlicher Sprache um die Quelle(n) von

Verständlichkeitskonflikten in kommunizierter Information zu lokalisieren, um Konflikte durch terminologische Übersetzung zu reduzieren und um gezielte Erklärungen zu erstellen.

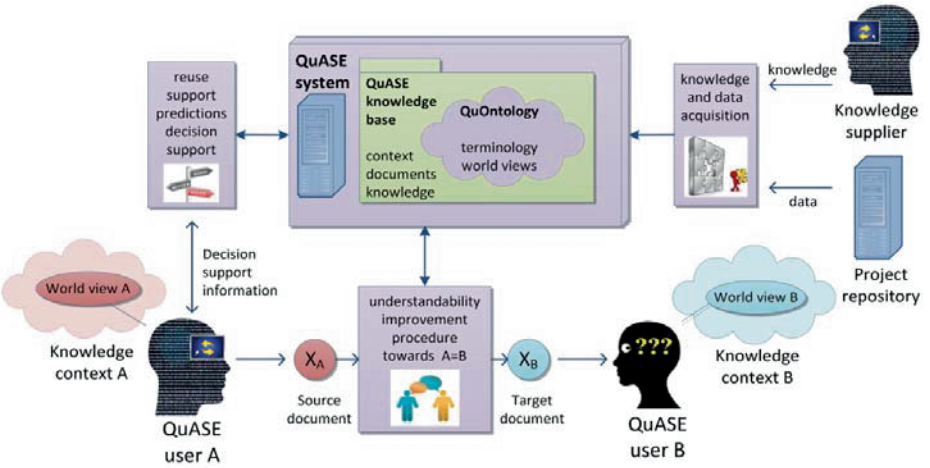

Abb. 2: Das schematische Konzept von QuASE (aus [St16, MS16, SMK15])

Der Teilbereich *Analytical Support* in QuASE verwendet Methoden für maschinelles Lernen für die Menge der Individuals in der QuASE Knowledge Base: Ähnlichkeitssuche für die Wiederverwendung von Informationen, Regressionsrechnung zur Vorhersage der Attribut-Werte und hybrides (teilweise überwachtes) Lernen für auf Klassifizierung basierendes Vorschlagswesen und Entscheidungshilfen.

Der größte Vorteil des QuASE-Ansatzes ist, dass nach auslesen der Daten aus einem Projekt-Repository, wie z.B. Jira (http://www.atlassian.com/jira), diese sofort in Wissen konvertiert werden und für Verständlichkeitsverwaltung und Analysis verfügbar sind. Außerdem erlaubt dies, dadurch dass das Mapping durch die Verwendung einer speziellen DSL flexibel ist, große Mengen von Daten zur Anwendung dieser Methoden.

Das QuASE-System kann als Brücke gesehen werden, die EndbenutzerInnen, die Daten in Projekt-Repositories und die (erweiterbare) Menge von Methoden für maschinelles Lernen und Verarbeitung von natürlicher Sprache verbinden. Diese Methoden werden automatisch nach Beschreibung des Repositories und der Kommunikationsumgebung durch die *QuASE site DSL* verwendbar für die Repository-Daten.

Die QuASE-Architekturkomponenten werden in Abb. 3. Die rot gestrichelte Line zeigt die Abgrenzung der QuASE-Applikation in seiner Umgebung. Die Applikation beinhaltet Client- und Server-Komponenten. In den folgenden Abschnitten werden diese Komponenten im Detail beschrieben.

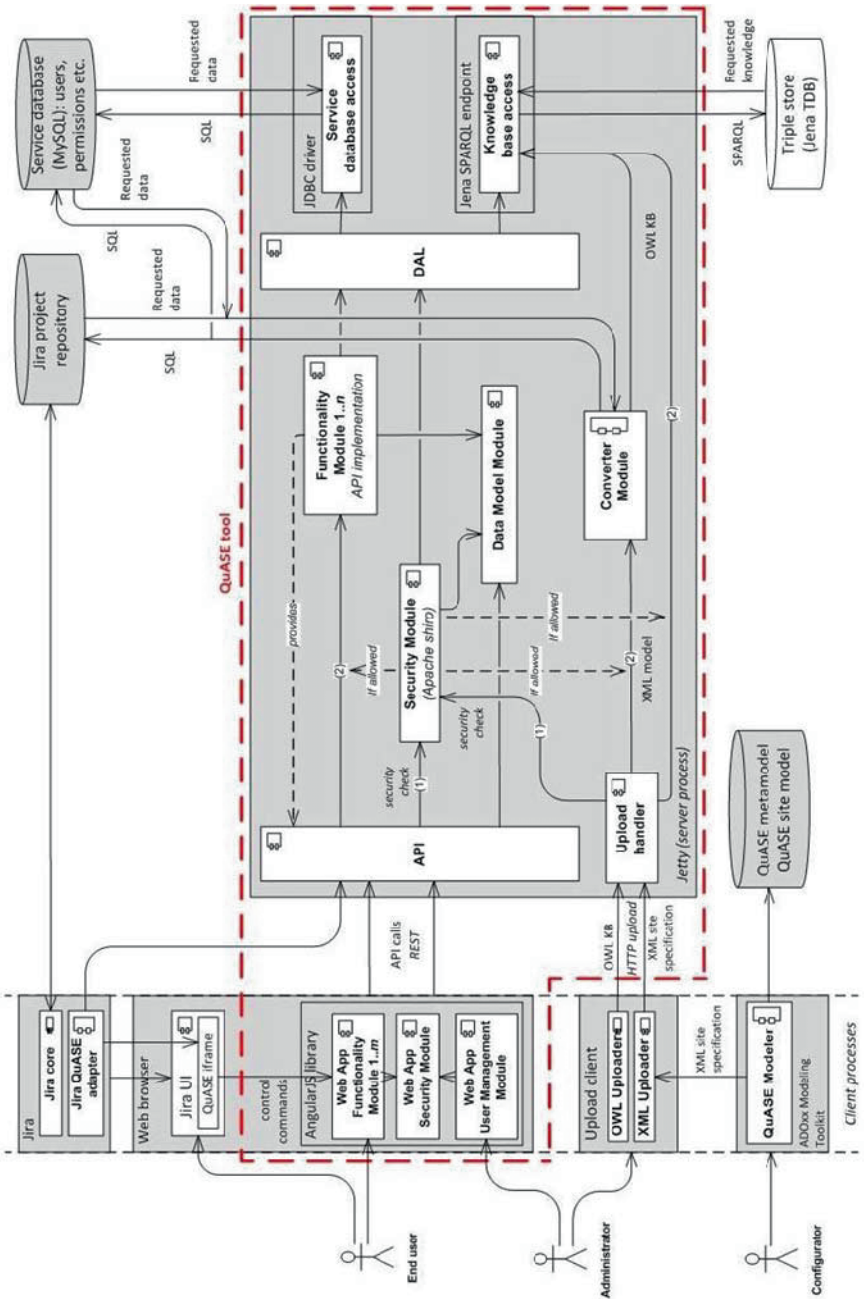

Abb. 3: QuASE System-Architektur

3.1 QuASE Server

Die Hauptfunktionalität der QuASE-Applikation ist im *QuASE Server* implementiert. Es umfasst folgende Funktionalitätsmodule:

1. Sicherheitsmodule;
2. Module für Kontext- und Dokumentenverarbeitung;
3. Verständlichkeitsverwaltungsmodul;
4. Analysis-Module;
5. Module zur Datensammlung;
6. Server-Administrationsmodul.

Diese Module basieren auf den folgenden Komponenten:

1. Die Datenzugriffsschicht (Data Access Layer, DAL) ist durch das DAL Modul implementiert, unterschiedliche Komponenten werden für den Zugriff auf die QuASE knowledge base (gekapselte SPARQL-Queries) und für den Zugriff auf die QuASE service database (gekapselte SQL-Queries) verwendet. Es führt höhere Abstraktionslevels ein, um die Apache Jena TDB und die interne Datenbank, die die Funktionalität komplexe SPARQL-Queries zu konstruieren beinhaltet, zu managen und abzufragen.
2. Die Klassen der Business Logik sind durch das Modul für das Datenmodell (Data Model Module) gekapselt. Der Quellcode zu diesem Modul greift auf externe Speicher durch die DAL zu. Das Datenmodell ist über die Funktionalitätsmodule aufgeteilt. Es stellt eine Menge von Wrappern für Individuals, Daten und Objekt-Properties abgerufen von der Knowledge Base (KB) zur Verfügung.

3.2 Interaktiver QuASE Client

Um EndbenutzerInnen den Zugriff zu ermöglichen ist der QuASE Server durch einen interaktiven *QuASE Client* erreichbar. Der QuASE Client ist in JavaScript implementiert, basierend auf modularer Infrastruktur. Die Infrastruktur beinhaltet den Quellcode zum Einbetten der Security- und API-Zugriffe (API- und Security-Web-App-Modules) und die Menüführung für den Aufruf der Module. Die Menüführung ist mit der Sicherheitsinfrastruktur verbunden, um die Rollen zu spezifizieren, für welche die einzelnen Menü-Items sichtbar bzw. unsichtbar sind.

Die folgenden Kategorien der Web-App-Funktionalitätsmodule sind:

1. Verständlichkeitsverwaltung und -einschätzung;
2. Analysis (Ähnlichkeitssuche, Vorhersage, Empfehlungen und Entscheidungshilfen);

3. Datensammlung (Entscheidungssammlung, Empfehlungsauswertung, Sammlung externer Werte);
4. Web-Administrationsmodul.

3.3 Architektur zur Integration von Jira in QuASE

Die Architektur zur Integration von Jira in QuASE beinhaltet folgende Komponenten:

1. Die Erweiterung der QuASE-Applikation, welche die Befehle von Jira akzeptiert. Diese Befehle beinhalten die Informationen über die aktuell ausgewählte Instanz der Kontext- oder Content-Unit (die Auswahl wird durch den Jira-User über die Jira-UI durchgeführt);
2. Eine Jira-Support-Erweiterung, die die Funktionalität von Jira zur Ausgabe der Kontroll-Anforderung, die zum QuASE-Tool weitergegeben wird, erweitert.

4 Ontologien & Knowledge Base in QuASE

In diesem Kapitel werden zunächst die wichtigsten Begriffe vorgestellt um anschließend auf die Technologien einzugehen, die die Beurteilung der Software Qualität mit Hilfe der Wiederverwendung von Wissen zu stärken. Das Forschungsprojekt QuASE an der Alpen-Adria-Universität Klagenfurt hat diese Technologien als Grundlage für einen Proof-Of-Concept zusammengetragen und eine Software entwickelt die diese zukunftsfähig machen soll.

4.1 QuASE Ontologien

Die Ontologien im Forschungsprojekt QuASE, *QuASE site ontology* und *QuOntology* sind Kernbestandteile der QuASE KB. Die QuOntology stellt dabei den wissensspezifischen Teil der Knowledge Base, während die QuASE site ontology den kontextspezifischen Teil realisiert (vgl. [St16], S. 67). Die konkrete Umsetzung und Anwendung der Ontologien und QuASE KB werden in [St16] und [SMK15] detailliert dargestellt. Abb. 4 zeigt einen schematischen Aufbau der QuASE Ontologien.

Die *QuASE site ontology* beinhaltet folgende Konzepte (siehe auch [St16], S. 71):

1. *site*: Besitzer der gegebenen QuASE-Installation, z.B. ein Softwareanbieter.
2. *context*: Einheiten (*units*) haben spezielle Sichten auf kommunizierte Informationen, z.B. Projekte, Organisationen und deren Abteilungen, beteiligte Stakeholder

3. *content*: Einheiten gestalten kommunizierte Informationen aus den Projekt-Repositories, wie Issues oder Kommentare in Issues;

4. *knowledge*: Einheiten, die kommuniziertes Wissen kapseln, welche ein Gegenstand der Angleichung sind. (vgl. [St16], S. 71/72)

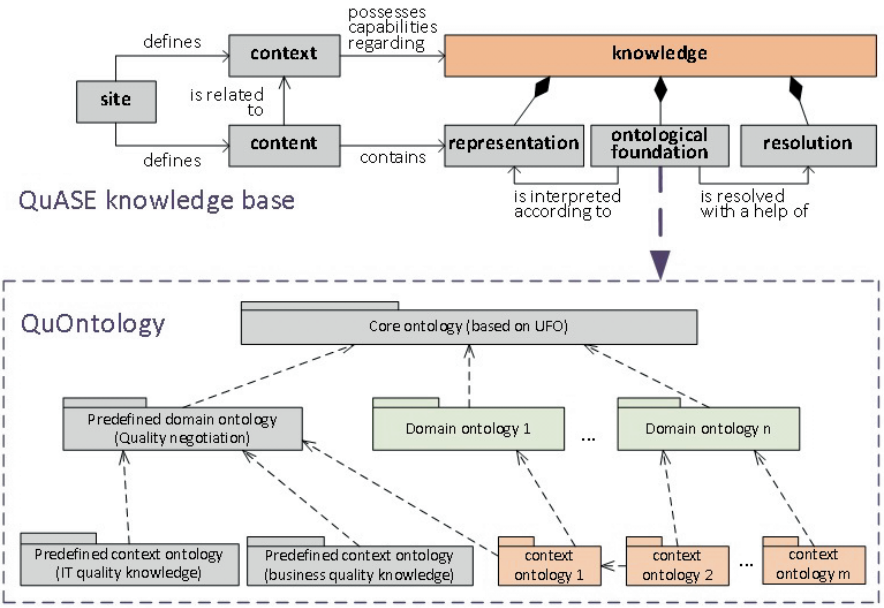

Abb. 3: Konzept der QuASE KB, site ontology & QuOntology (aus [St16, SMK15])

Jede QuASE Knowledge-Unit ist zusammengesetzt aus den folgenden Komponenten:

1. *ontological foundation*: eine Referenz zur Konzeptualisierung der speziellen Teile von Qualität oder Domänenwissen durch die ontologischen Mittel;

2. *representation*: die Repräsentation einer Knowledge Unit in einem Format, das von den Kommunikationspartnern wahrgenommen werden kann (z. B. normaler Text). Representation Units sind in Content Units enthalten.

3. *resolution means*: die Mittel der Auflösung von Verständnis-Konflikten, die zu speziellen Knowledge Units gehören (z.B. Erläuterungen oder externe Referenzen).

Context Units besitzen *capabilities* (deutsch: Funktionen) um mit Knowledge Units umgehen zu können; insbesondere diese Funktionen können sich auf die Fähigkeit des Verstehens einer gegebenen Knowledge Unit (z.B. seine Repräsentation) beziehen; oder eine Knowledge Unit mit einer resolution means erklären. (vgl. [St16], S. 72; [SMK15])

4.2 Wissenswiederverwendung in QuASE

Zur Spezifikation und Wartung der Knowledge Base stellt QuASE folgende Architektur-Komponenten bereit:

1. QuASE-Modellierungstool (*QuASE modeler tool*);
2. Werkzeug für das Hochladen von Modellen (*Model upload tool*);
3. QuASE-Konvertierungsmodul (*QuASE converter module*);
4. QuASE-Komponente zur Unterstützung der Knowledge Base (*QuASE knowledge base support component*).

Diese Komponenten werden in den folgenden Abschnitten beschrieben.

QuASE-Modellierungstool. Dieses Tool, basierend auf dem ADOxx Metamodeling Framework (http://www.adoxx.org), erlaubt es ein *QuASE site model* mit einer graphischen Modellierungsschnittstelle zu erstellen und editieren, sowie die Informationen dieses Modells und die spezifischen Verbindungsinformationen für die zur Modellierung bereitgestellte Organisation, um die QuASE KB auf den QuASE Server zu bereitzustellen.

Dieses Werkzeug unterstützt zwei Arten von Diagrammen:

1. QuASE Site-Modell-Diagramm (*QuASE site model diagram*), welches die Struktur des Kommunikationsumfelds und des kommunizierten Wissens beschreibt. Dieses Modell hängt nicht von site-spezifischen Informationen ab (z.B. Typ oder Adresse der Projekt-Repository-Datenbank, die Adresse des QuASE Server, etc.)
2. QuASE-Arbeitsbereichsdiagramm (*QuASE workspace diagram*), welches spezifische Informationen zu einer bestimmten site beinhaltet und diese Informationen zusammen mit den Referenzen zu den passenden Site-Model-Diagrammen zur Bereitstellung organisiert.

Speziell das QuASE-Arbeitsbereichsdiagramm enthält folgende Elemente:

1. Das Konfigurationselement welches Informationen zu bestimmten Deployment-Site kapselt, dies schließt den Hostnamen und TCP-Port des entfernten QuASE Server, JDBC URLs und benutzerfreundliche Namen zur Verbindung mit dem QuASE Service-DB und die Project Repositories etc. ein;
2. Das Query-Quellelement, welches die spezifischen Informationen für das RDBMS, zuständig für die Implementierung des bestimmten Projekt-Repositories kapselt. Es beinhaltet die Menge der RDBMS-spezifischen Queries, indiziert durch den Namen der Query übertragen durch das Site-Modell. Es können beispielsweise separate Quellen für Oracle- und MySQL-Query-Implementierungen eines bestimmten Repositories existieren, so dass das

Umziehen des Repositories von einem DB-Server zu einem anderen nicht zu einer Änderung des Site-Modells führt;

3. Das Site-Spezifikationselement, welches die Referenzen zur bestimmten Site-Konfiguration und den Query-Quellelementen, sowie das bestimmten Site-Modell beinhaltet, z.B. "die Spezifikation für das Deployment von Unternehmen A (QuASE Server *qqq*) zum Oracle-basierenden Repository bereitgestellt auf Server *xxx*". Dieses Element beinhaltet alle notwendigen Informationen um das Deployment der QuASE KB auf den bestimmten QuASE Server durchzuführen; seine UI stellt den "deploy"-Button zur Initiierung des Deployment-Prozesses zur Verfügung.

Eine Deployment-Aktivität, die vom Modeling-Tool angestoßen wird, bringt den Upload der XML-Datei, die vom ausgewählten Site-Spezifikationselement erzeugt wurde (beinhaltet die Informationen aus Site-Modell, Site-Konfiguration und Query-Quellelement), in den spezifischen QuASE Server, der den Upload Client verwendet.

Upload Client. Der Upload Client ist ein Kommandozeilen-Tool, welches zum QuASE Server verbindet. Es führt die notwendige Authentifizierung durch, und transferiert die bestimmte Datei zu diesem Server. Es ist in der Lage folgende Dateien hochzuladen:

1. OWL Knowledge Base Dateien (z.B. generiert durch Kommandozeilen-Converter),
2. XML Site Spezifikationsdateien (z.B. exportiert durch das QuASE-Modellierungstool).

Der Default-Modus zum Start des Upload Clients ist aus dem QuASE-Modellierungstool. In diesem Fall werden die Modell-Informationen korrespondierend zur aktuellen Site-Spezifikation in eine XML-Datei konvertiert, das mittels des Tools übertragen wird.

Knowledge Base Unterstützung im QuASE Kernel. Einige Architektur-Komponenten unterstützen die QuASE KB, das wichtigste davon ist ein Konverter-Modul (*Converter Module*). Dieses Modul erzeugt die QuASE KB aus dem QuASE Site-Model, interaktive Daten werden in der QuASE Service-DB gespeichert und die Daten für die Site sind verfügbar aus dem Projekt-Repository (z.B. Jira-Datenbank).

1. Ein Konverter-Modul konvertiert zuerst eine XML-Model-Datei in eine OWL QuASE site ontology, die die Struktur des Kommunikationsumfeld beschreibt;
2. Nach Erstellung der QuASE site ontology sammelt dieses Modul die Daten aus dem Projekt-Repository (z.B. der Jira-DB), sowie die vom Benutzer bereitgestellten Daten, die in der QuASE Service-DB gespeichert sind, und konvertiert diese Daten in die Menge der OWL-Individuals entsprechend zur OWL site ontology. Dies formt die QuASE site knowledge base.

Als ein Resultat besitzt die QuASE Site-KB die wiederherstellbare Eigenschaft: z.B. die

komplette KB kann jederzeit basierend auf dem QuASE Site-Model, den Daten aus der QuASE Service-DB und den Daten aus dem Projekt-Repository rekonstruiert werden.

Es existieren zwei Implementierungen des Konverter-Moduls, die im QuASE-System eingebaut sind:

1. Die Implementierung als ein "In-Process"-Modul für den QuASE Server, aufgerufen durch seine Datei-Upload-Handler-Komponente (dies ist der Default Use-Case für den Aufruf des Konverter-Codes);
2. Die Implementierung als ein separates Kommandozeilen-Werkzeug das remote aufgerufen werden kann. In diesem Fall kann die resultierende OWL KB zum Server hochgeladen werden.

Basierend auf der vom Konverter-Modul generierten OWL KB-Datei initialisiert ein spezielles KB-Support-Modul einen Jena TDB Triple-Store (es ist möglich den Triple-Store von Grund auf zu generieren, sowie inkrementelle Updates durchzuführen).

Andere Komponenten unterstützen die KB folgendermaßen:

1. Ein Teil des DAL-Modul weist SPARQL-basierten KB-Zugriff mit Verbindung zum Modell auf und mit der Möglichkeit zum Schreiben von Ad-hoc benutzerdefinierten SPARQL und die resultierenden Daten auszuführen;
2. Ein UI-Modul implementiert den sog. „Internal Endpoint", z.B. das interactive Query-Fenster erlaubt dem Benutzer eigens definierte SPARQL Queries am existierenden TDB-Store auszuführen.

Aktueller Stand des Projekts. Die QuASE-Applikation wurde an den Standorten von zwei Partner-Organisationen installiert und in Betrieb genommen. Es arbeitet gut mit deren Repositories, welche bis zu 26000 Issues bzw. Tickets beinhalten. Die resultierende Knowledge Base beinhaltet bis zu 1,7 Million Axiome ohne signifikante Performance-Probleme.

Die Haupt-Verwendungszwecke des Tools sind wie folgt:

1. Die EndbenutzerInnen bevorzugen das System über die Jira-Integrationsschnittstelle zu verwenden, was ihnen bekannte Möglichkeiten zum Erreichen der speziellen Kontext- und Content-Units bietet;
2. Die vermeintliche Nutzlosigkeit der Understandability-Management-Fähigkeiten der Applikation hängt von der Qualität des terminologischen Wissens in der KB (QuOntology) und zuallererst an der Vollständigkeit dieses Wissens in Verbindung zur bestimmten Applikationsdomäne (z. B. Software Qualität) ab.
3. Die vermeintliche Nutzlosigkeit der analytischen Fähigkeiten der Applikation hängen von der Vollständigkeit der Menge der verfügbaren Metriken für die Analyse ab. Es wird empfohlen qualitative Studien in Zusammenarbeit mit den

UnternehmensvertreterInnen durchzuführen, um die Metriken, welche den Entwicklungsprozess beeinflussen, in das Site-Modell zu integrieren.

5 Related Work

Dieser Artikel basiert zumeist auf den wissenschaftlichen Artikeln und Arbeitsunterlagen des Projektteams QuASE (u.a. [SMK15], [Sh15]) der Forschungsgruppe Application Engineering. Zum Thema Software Qualität bieten Wallmüller [Wa01] und Hoffmann [Ho13] weiterführende Informationen. Guizzardi [Gu05] bietet allgemeine Informationen über Ontologien, [SMK15] und [Sh15] bieten eine Übersicht über die konkrete Umsetzung der Ontologien und Knowledge Bases in QuASE. In [Ab14] wird eine weitere Möglichkeit zur Umsetzung einer Softwarelösung innerhalb des Themengebiets der qualitätsbewussten Softwareentwicklung vorgeschlagen.

6 Zusammenfassung & Ausblick

Die Forschungsfrage in [St16] *„Wie können Ontologien und Wissensdatenbanken die Arbeit der Akteure im Softwareentwicklungsprozess erleichtern und helfen Probleme und Missverständnisse zu reduzieren?“* wird ebenda zunächst auf die beiden Kernaussagen *Stärkung der Qualitätsbeurteilung durch Ontologien und KBs* und *wie dadurch Missverständnisse reduziert und die Arbeit der Akteure erleichtert werden* aufgeteilt und separat beantwortet (vgl. [St16], Kap. 4 u. 6). Um die Software Qualität zu verbessern gibt es bereits seit Jahrzehnten verschiedene Ansätze, wie die Autoren in [Ma95] [Wa01], [Ho13] oder [De01] ausführlich beschreiben. Der Einsatz von Ontologien und Wissenswiederverwendung in diesem Spektrum ist jedoch im Zusammenhang mit Software Qualität und dessen Beurteilung ein relativ junges Forschungsgebiet (vgl. [St16] S. 90 ff.). Die Beteiligten in einem Softwareentwicklungsprojekt müssen zumeist vermehrten Aufwand in administrative Tätigkeiten aufwenden, der mit elektronischen Hilfsmitteln deutlich reduziert werden könnte.

Diese Hilfsmittel, zu der auch QuASE gezählt werden darf, können allerdings nicht nur Vorteile bringen, sondern bedürfen auch einem höheren Aufwand durch Einarbeitung und Pflege der Projektdatenbanken, damit auf längere Zeit gesehen die Vorteile deutlich überwiegen können. Zudem können Synergien zu bereits bestehenden Werkzeugen und Softwaresysteme gebildet werden, die ähnlich gelagerte Probleme beschreiben und lösen können. Auch im Hinblick auf die Weiterentwicklung der IT in Industrie 4.0, Internet der Dinge und Automatisierung von Maschinen kann mit solchen Decision-Support-Systemen die Wissenschaft und Forschung vorantreiben und weitere Anwendungsmöglichkeiten gefunden werden (vgl. [St16], S. 84-86, 103 ff.).

Literaturverzeichnis

[Ab14] Abufouda, M.: "Quality-aware Approach for Engineering Self-adaptive Software Systems". Kaiserslautern, 2014.

[CP09] Chung, L., do Prado Leite, J.: “On Non-Functional Requirements in Software Engineering”, Mylopoulus Festschrift, LNCS 5600, p. 16, 2009.

[De01] DeMarco, T.: Spielräume. München, Wien: Carl Hanser, 2001.

[GI16] GI, Gesellschaft für Informatik e.V., https://www.gi.de/themen/grand-challenges-der-informatik.html, Stand: 16.04.2016

[Gu05] Guizzardi, G.: "Ontological foundations for structural conceptual models," PhD Dissertation, Centre for Telematics and Information Technology, University of Twente, Enschede (NED), 2005.

[Ho13] Hoffmann, D. W.: Software-Qualität, 2nd Edition. Berlin, Heidelberg: Springer, 2013.

[ISO00] Information technology – Software product quality – Part 1: Quality Model, ISO/IEC ISO/IEC 9126-1:2001, 2000/2001.

[Ma95] Mayr, H.: "Software Qualität: Nur eine Frage des Software Engineering?," in ISO 9000 - Softwareentwicklung ; Ethik, Analysen, Tools ; Beiträge vom adi QM/IT Expertentreffen 1994, ed. Münster (GER) [u.a.]: Norbert Ruppenthal, 1995, p. 49 ff.

[MS14] Mayr, H.; Shekhovtsov, V.: QuASE - About the Project, http://quase-ainf.aau.at/ Stand: 05.05.2016

[Sh15] V. Shekhovtsov, H.C. Mayr, S. Ianushkevych, M. Kucko, V. Lubenskyi, and S. Strell: Implementing Tool Support for Effective Stakeholder Communication in Software Development – A Project Report. In: Ausgewählte Beiträge zur Anwenderkonferenz für Softwarequalität Test und Innovation - ASQT 2014, books@ocg.at, Vol.310. Österreichische Computer Gesellschaft, Wien, 2015, pp. 45-58.

[SMK15] V. Shekhovtsov, H.C. Mayr, C. Kop: Facilitating Effective Stakeholder Communication in Software Development Processes. In: Nurcan, S., Pimenidis, E. (eds.): Information Systems Engineering in Complex Environments, Lecture Notes in Business Information Processing 204, Springer, 2015, pp. 116-132

[St14] Strell, S.: „Der Qualitätsbegriff im Requirements und Software Engineering“ (Seminararbeit), unpublished.

[St16] Strell, S.: Ontologien in Quality-Aware Software Engineering. Master thesis, Klagenfurt am Wörthersee, 2016.

[Wa01] Wallmüller, E.: Software-Qualitätsmanagement in der Praxis. München; Carl Hanser, 2001.