ANTRAG FÜR DIE BETRIEBLICHE PROJEKTARBEIT

1. Projektbezeichnung

Entwicklung einer domänenspezifischen Sprache (DSL) zur Beschreibung von Routingregeln.

1.1. Kurzform der Aufgabenerstellung

Für die ALTE OLDENBURGER Krankenversicherung AG (AO) soll die Beschreibung von Routingregeln, nach denen eingehende Dokumente im Workflow- und Dokumentenmanagementsystem PAM an Sachbearbeiter geleitet werden, von der Programmierimplementierung abstrahiert werden. Hierzu soll eine DSL entwickelt werden, anhand derer der nötige C#-Programmcode für das bestehende Routingprogramm generiert werden kann.

1.2. Ist-Analyse

Als Krankenversicherung erhält die AO täglich Post von Versicherungsmaklern und Kunden, welche je nach Art des Schreibens an Sachbearbeiter oder Abteilungen weitergeleitet werden müssen. So werden Neuanträge für Versicherungen an die jeweilige Antragsgruppe und Leistungsabrechnungen, wie z.B. Arztrechnungen, an die Leistungsabteilung weitergeleitet.
Dies geschieht durch das System BREPL (Beschlagworten Routen Erkennen Prüfen Leisten), welches anhand von vordefinierten Regeln einen Vorgang in PAM anlegt, die Dokumente anhängt und dem zuständigen Sachbearbeiter zuteilt.
Derzeit liegen vorhandene Routingregeln nur in Form von C#-Programmcode vor.
Bei Nachfragen über das Routing von Vorgängen, z.B. warum Vorgang X an Mitarbeiter Y zugewiesen wurde, muss sich derzeit ein Entwickler durch den Programmcode arbeiten, um eine Aussage treffen zu können. Dieses Vorgehen nimmt viel Zeit in Anspruch, da die Komplexität des Programmcodes durch viele Verzweigungen sehr hoch ist.

Neue Regeln und Änderungen an vorhandenen Regeln werden derzeit vom Fachbereich im Ticketsystem der AO schriftlich erfasst und müssen durch einen Entwickler manuell in das Programm eingebaut werden. Dies gestaltet den Programmcode mit jeder weiteren Regel komplexer und das nachträgliche Nachvollziehen von Regeln wird weiter erschwert.

2. Zielsetzung entwickeln / Soll-Konzept

2.1. Was soll am Ende des Projektes erreicht sein?

Durch die DSL sollen Regeln für die Zuweisung von Vorgängen in PAM definiert werden können. Diese Regeln werden in einer Eclipse-Instanz definiert und beim Speichern soll C#-Programmcode generiert werden, welcher der Schnittstelle zum BREPL-System entspricht.
Der generierte C#-Programmcode und der selbst definierte Regelcode müssen versioniert werden können.
Die generierten Regeln müssen durch den Buildserver kompilierbar sein und durch Tests muss sichergestellt werden, dass alle Regeln verwendbar sind und es zu keinen

Doppeldeutigkeiten kommen kann. Ist dies gewährleistet, sollen die Regeln automatisch in das BREPL-System eingespeist werden können.
Die Regeln müssen geloggt werden, sodass jederzeit schnell nachvollziehbar ist, warum ein Dokument eine bestimmte Zuweisung bekommen hat.

2.2. Welche Anforderungen müssen erfüllt sein?

Folgende Anforderungen sollen durch die DSL erfüllt werden:

- Einfache Beschreibung von Regeln
- Die DSL muss testbar sein
- Die Codegenerierung muss testbar sein
- Generierung von C#-Programmcode
- 100% Testabdeckung der Regeln durch automatisierte Tests
- Logging gerouteter Dokumente
- Der generierte Programmcode muss trotz Generierung lesbar sein und innerbetrieblichen Coderichtlinien folgen
- Der generierte Regelcode muss automatisiert deployt werden
- Die Eclipse-Instanz zur Erstellung der Regeln muss automatisiert auf dem neuesten Stand gehalten werden

2.3. Welche Einschränkungen müssen berücksichtigt werden?

Der generierte Programmcode wird über eine im Rahmen des Projektes zu erstellende Schnittstelle in BREPL eingespeist. Das BREPL-System ist ein dauerhaft laufendes C#-Programm, welches nicht zu jederzeit gestoppt werden kann, wodurch das Deployment der Regeln in Form einer DLL nicht im laufenden Betrieb möglich ist.

3. Projektstrukturplan entwickeln

3.1. Was ist zur Erfüllung der Zielsetzung erforderlich?

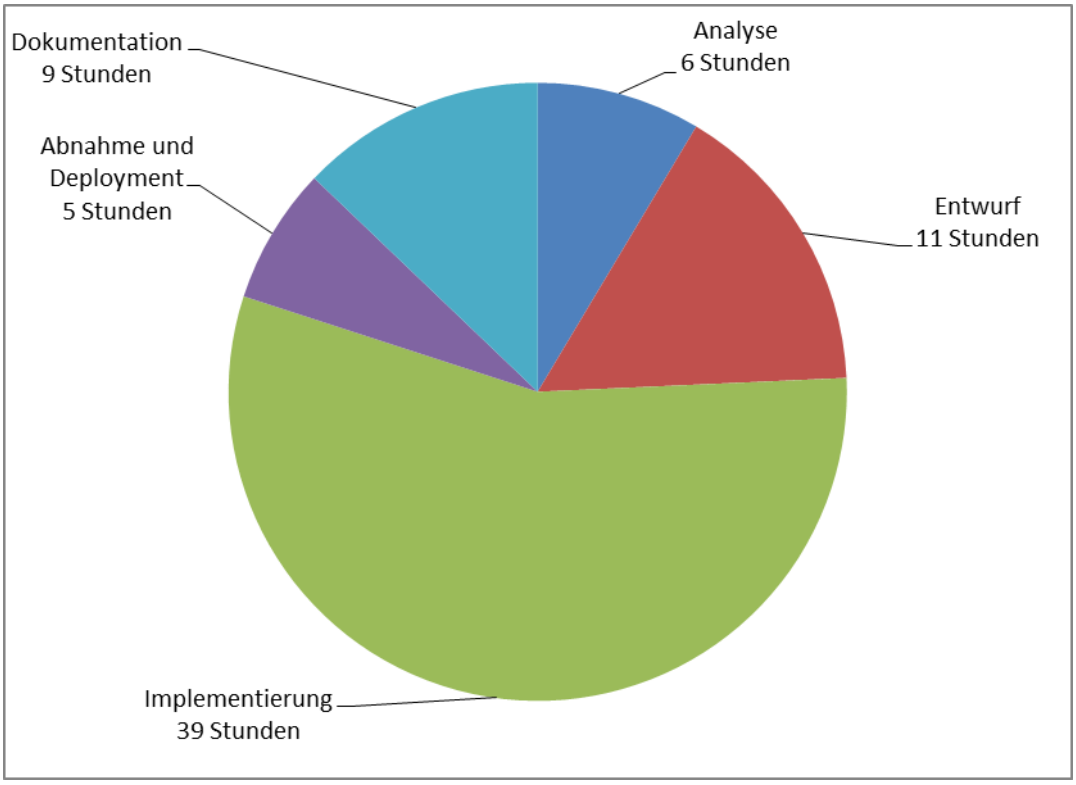
Das Projekt soll agil entwickelt werden. Hierbei wird in kurzen Iterationszyklen Rücksprache mit dem Fachbereich gehalten um Feedback einzuholen. Dieses Feedback ermöglicht es dem Entwickler, schnell auf Änderungswünsche einzugehen und somit Zeit zu sparen. Durch die ständige Einbindung der Endanwender wird auch ermöglicht, dass eine Schulung zur Benutzung der DSL verkürzt wird, da alle Features und Änderungen schon bekannt sind.
Die Implementierung des Projekets wird testgetrieben durch automatisierte Unit-Tests und mit Hilfe eines Buildservers entwickelt, wodurch die Entwicklung einer vom Entwicklersystem unabhängigen Software ermöglicht wird.
Das gesamte Projekt wird in Git versioniert und alle Schritte im Ticketsystem dokumentiert.

3.2. Aufgaben auflisten

- Analyse
    - Durchführung einer Ist-Analyse
    - Durchführung einer Wirtschaftlichkeitsanalyse und Amortisationsrechnung
    - Ermittlung von Use-Cases
    - Erstellung eines Lastenheftes
- Entwurf
    - Entwurf der DSL

    - Erstellung eines Komponentendiagramms
    - Entwurf der Schnittstelle zu BREPL
    - Erstellung eines Deploymentkonzeptes
    - Erstellung eines Pflichtenheftes
- Implementierung
    - Implementierung der BREPL-Schnittstelle mit Tests
    - Integration der Schnittstelle in BREPL
    - Implementierung der DSL mit Tests
    - Implementierung der Codegenerierung mit Tests
- Abnahme und Einführung
    - Bereitstellung von Eclipse
    - Einrichtung des Continuous Integration-Prozesses
    - Konfiguration von automatischen Updates für Eclipse
- Dokumentation
    - Erstellung der Projektdokumentation
    - Erstellung der Entwicklerdokumentation
    - Erstellung des Benutzerhandbuches

3.3. Grafische und tabellarische Darstellung



Phase	Dauer in Stunden
Analyse	6
Entwurf	11
Implementierung	39
Abnahme und Deployment	5
Dokumentation	9
Summe	70

4. Projektphasen mit Zeitplanung in Stunden

Analyse	**6 h**
• Ist-Analyse durchführen (extrahieren vorhandener Regeln aus BREPL-Code)	2 h
• Wirtschaftlichkeitsprüfung und Amortisationsrechnung des Projektes durchführen	1 h
• Unterstützung des Fachbereichs bei der Erstellung des Lastenheftes	3 h
Entwurf	**11 h**
• Komponentendiagramm der Endprodukte erstellen	1 h
• Nutzwertanalyse zur Auswahl des DSL-Frameworks erstellen	2 h
• Aktivitätsdiagramm zum Erstellen einer Regel erstellen	1 h
• Schnittstelle zum BREPL-System entwerfen	3 h
• Deploymentdiagramm erstellen	1 h
• Pflichtenheft erstellen	3 h
Implementierung	**39 h**
• Implementierung der BREPL-Schnittstelle mit Tests	10 h
- Implementierung von Schnittstellen und Basisklassen	3 h
- Implementierung der Routingklassen	7 h
• Implementierung der DSL mit Tests	18 h
- Implementierung der Syntax	5 h
- Implementierung der Grammatik	5 h
- Implementierung der Eingabevalidierung	3 h
- Umsetzung der DSL	5 h
• Implementierung der Codegenerierung mit Tests	10 h
- Implementierung der Codegenerierung für Regelklassen	4 h
- Implementierung der Codegenerierung für Tests der Regelklassen	4 h
- Implementierung eines Konsolenprogramms zum Aufruf der Codegenerierung durch den Buildserver	2 h
• Integration der Schnittstelle in BREPL	1 h
Deployment	**5 h**
• Bereitstellung und Konfiguration von Eclipse	1 h
• Continuous Integration (Einrichten von automatischen Builds)	2 h
• Deployment des Eclipse-Plugins	1 h
• Deployment der C#-Schnittstelle	1 h
Erstellen der Dokumentationen	**9 h**
• Erstellen der Projektdokumentation	7 h
• Erstellen der Entwicklerdokumentation	1 h
• Erstellen der Benutzerdokumentation	1 h

5. Name der Ausbildungsstätte in dem das Projekt durchgeführt wird

ALTE OLDENBURGER Krankenversicherung AG

EDV-Abteilung

5.1. Name des Ausbilders, bzw. Projektverantwortlichen mit Angabe der Tel. Nr.

Ausbilder