1 Proof of Concepts - Dokumentation

1.1 Regel-Engine

Datenstrukturen Es wurden zunächst grundlegende Datenstrukturen im xml Format entwickelt um die Dokumentdaten, Regelbedingungen und Regelattributierungen in das Programm einzulesen und das Ergebnis zu Speichern. Dabei wurde unter Beachtung der strategischen Ziele angestrebt mithilfe der Anwedungsdomaene die Komplexität der Datenstruktur möglichst anzunähern.

Dokumentdaten, data.xml, wird auch für den Export genutzt

Regelbedingungen, conditions.xml

Attributierungen, attributions.xml

Entwicklung & Programmablauf Das Programm wurde entlang der spezifizierten Anforderungen im .NET Framework entwickelt und nutzt die Bibliotheken System.IO, System.Thread und System.IO.Xml.

1. es wird in einem Intervall auf in einem Importverzeichnis auf neu importierte Verzeichnisse gehorcht, wenn eine
2. falls ein Verzeichnis importiert wird, werden die Informationen der darin enthaltenen data.xml, conditions.xml und attributions.xml jeweils in eigenständige `List<string attribut, string wert>` Strukturen überführt
3. Die Dokumentdaten werden auf die Regelbedingungen überprüft
   - das Attribut Volltext wird darauf geprüft ob es den Bedingungsstring enthält
   - alle anderen Attribute werden auf exakte Übereinstimmung geprüft
4. wenn die Bedingungen zutreffen werden Dokumentdaten und Attributierungen in eine Export Struktur zusammengeführt
5. die Exportstruktur wird als `\\Export\\guid\data.xml` gespeichert
6. wieder zu Schritt 1

Anforderungserfüllung

Nr.	Anforderung	Anforderungen erfüllt
1	das Programm horcht auf einen Importverzeichnis	ja
2	wenn ein neues Quellverzeichnis im Importverzeichnis vorhanden ist, werden, sehr zeitnah, die enthaltenen xml Dateien eingelesen	ja
3	die für eine Regelanwendung nötige Anzahl der zu importierenden Dateien wird verarbeitet	ja, es werden data.xml, conditions.xml und attributions.xml verarbeitet
4	Ausgangsdaten werden in entsprechende Datenstrukturen überführt	ja, es wird die eine Struktur `List<string Attribut, string Wert>` genutzt
5	Attributierungen werden korrekt nach Regelbedingungen ausgeführt	ja
6	die Exportdatenstruktur wird als xml Datei in einem eigenen Verzeichnis in einem Exportverzeichnis gespeichert	ja, in `\\Export\guid\data.xml`
7	das Importverzeichnis der verarbeiteten Dateien wird gelöscht	ja

Fazit: Der PoC erfüllt alle spezifizierten Anforderungen. Für die prototypische Realisierung müssen weitere Test insbesondere mit unvollständigen, fehlerhaften und konfliktären Dateneingaben durchgeführt werden. Ebenfalls sollte das Programms in einer besseren Klassenstruktur organisiert werden und es muss evaluiert werden eine andere Importstruktur als Verzeichnisse, zb. komprimierte Archiv-Dateien, zu nutzen.

Quellcode: github.com/thuascgn/EISWS1516/MS3/PoCs/RegelEngine

1.2 Desktop Client

Programmentwicklung Zum Test wurde ein Programm nach dem WIMP Paradigma nativ für die Windows Plattform unter Verwendung des .Net Frameworks in der Entwicklungsumgebung Visual Studio von Microsoft entwickelt. Als grundlegenede Basis Struktur wurde ein Template für die WCF Windows Communication Foundation gewählt mit einem Model View Struktur.
In der Entwicklungsumgebung können Interaktionselemente der graphischen Benutzungsschnittstelle per Eingabe in eine xml Struktur definiert oder per Drag & Drop aus der enthaltenen Standardbibliothek in das angelegte Programm importiert werden. Beide Möglichekeiten wurden genutzt.
Auch bei der Zuweisung von Methoden zu den Interaktionselementen wurden zwei Verfahren genutzt, erstens über Doppelklick auf ein Element in der graphischen Ansicht. Daraufhin wird automatisch ein Methodenkopf im Controller definiert und dem zuvor doppelgeklickten Element als Methode bei einem Klick Ereignis zugewiesen. In der angelegten Methode kann dann der Code geschrieben werden kann. Die zweite Möglichkeit ist, zunächst eine Methode zu schreiben und diese Methode den Ereignislistenern der Elemente zuzuweisen. Auf diese ist es möglich ein Methode an unterschiedliche Interaktionslemente zu binden wenn diese ein ähnliches oder gleiches Verhalten aufweisen sollen.

Anforderungserfüllung

Nr.	Anforderung	Anforderungen erfüllt
1	das Programm ist auf der Zielplatform der Clients PC mit Windows Betriebssystem lauffähig	ja
2	es folgt dem WIMP Paradigma	ja
3	Interaktion soll mit folgenden Interaktionselementen möglich sein	
	Buttons (Schaltflächen)	ja
	Formulareingabe von Text und Zahlen	ja
	Checkboxen	ja
	Radiobuttons	ja
	Auswahl aus DropDown Menus	ja
4	Zusammenführung und Ausgabe	ja, Ausgabefeld das die eingegebenen Daten zusammengeführt ausgibt

Status: durchgeführt

Fazit: Der PoC erfüllt alle Anforderungen. Aufbauend auf diesem PoC sollte zudem der PoC Clientkommunikation durchgeführt werden.

Quellcode:

1.3 Nachrichtenpriorisierung

Um die ersten beiden Kriterien zu erfüllen wurde die Datenbank Redis ausgewählt. Desweiteren bietet Redis mit Sorted Sets eine native Datenstruktur bei der Wert Strings nach einem zugewiesenen Wert , dem Score, priorisiert

werden. Es wurde daraufhin versucht mittels Kommandoeingabe im Shellprogramm der Datenbank schrittweise die gewünschten Bedingungen zu erreichen. Dies zeigen die folgenden Ausschnitten des Shell Protokolls.

1. Anlegen einer simplen Sorted Set Datenstruktur

```
127.0.0.1:6379> ZADD account 0 '{"Absender":"Messebau Schmitz", "rnr":"123456789",
"datum":"25.04.2015", "time":"1000000"}'
(integer) 1
127.0.0.1:6379> ZADD account 1 '{"Absender":"Messebau Howe", "rnr":"456789321",
"datum":"15.03.2014", "time":"1000000001"}'
(integer) 1
127.0.0.1:6379> ZADD account 0 '{"Absender":"Messebau Schmitz", "rnr":"123456790",
"datum":"28.04.2015", "time":"1000010"}'
(integer) 1
127.0.0.1:6379> ZADD account 1 '{"Absender":"Autoverleih Seventh", "rnr":"1787ASD234",
"datum":"01.05.2015", "time":"1002010"}'
(integer) 1
127.0.0.1:6379> ZADD account 2 '{"Absender":"Autoverleih Eigth", "rnr":"1799HG5523X4",
"datum":"07.05.2015", "time":"1002150"}'
(integer) 1
127.0.0.1:6379> ZCARD account
(integer) 5
```

Eine simple Struktur wurde mittels ZADD angelegt und die Anzahl mit ZCRAD überprüft.

2. Testen der Möglichkeiten der Erhöhung der Scores

```
127.0.0.1:6379> ZINCRBY account 1 "*Messebau"
"1"
127.0.0.1:6379> ZINCRBY account 1 "'{"Absender":"Messebau*'"
Invalid argument(s)
127.0.0.1:6379> ZINCRBY account 1 MATCH "{\"Absender\":\"Messebau"
(error) ERR wrong number of arguments for 'zincrby' command
127.0.0.1:6379> ZSCAN account 0 MATCH *Messebau*
1) "0"
2) 1) "{\"Absender\":\"Messebau Howe\", \"rnr\":\"456789321\", \"datum\":\"15.03.2014\",
\"time\":\"1000000001\"}"
   2) "1"
   3) "{\"Absender\":\"Messebau Schmitz\", \"rnr\":\"123456790\", \"datum\":\"28.04.2015\",
    \"time\":\"1000010\"}"
   4) "0"
   5) "{\"Absender\":\"Messebau Schmitz\", \"rnr\":\"123456789\", \"datum\":\"25.04.2015\",
    \"time\":\"1000000\"}"
   6) "0"
   7) "*Messebau"
   8) "1"
```

Die Rückgabe der der ersten beiden Kommandos macht folgende Eigenschaften deutlich: ZINCRBY akzeptiert weder explizite noch impliziten Regex Ausdrücke und die Erhöhung eines Scores ist mit einen Teilstring nicht möglich. Über einen Teilstring können mittels Regex nur die zugehörigen Werte ermittelt werden. Daraus folgt das für eine Veränderung der Priorisierung über die Scores mittels der Kommandos ZINCRBY und ZDECRBY der vollständige Wert, in diesem Fall die vollständige Nachricht, explizit verwaltet, priorisiert und wieder gespeichert werden muss. Damit werden die Bedingungen 3., 5.1 und 5.2 verletzt und dieser PoCs muss als fehlgeschlagen betrachtet werde.

3. Skripting Redis ermöglicht auch die Eingabe von gescripteten Kommandos. Im zugehörigen Protokoll kann eingesehen werden das ein par Versuche unternommen sich dem Problem per Skripting zu nähern. Schnell wurde jedoch deutlich das auch auf diese Weise der ein erheblicher expliziter Priorisierungsaufwand vonnöten sein würde. Die Möglichkeit des Skriptings wurde verworfen, und POCs bleibt fehlgeschlagen.

Anforderungserfüllung

Nr.	Anforderung	Anforderungen erfüllt
1	es soll eine In-Memory Datenbank verwendet werden	ja
2	um größeren Einarbeitungsaufwand zu vermeiden muss die Datenbank den Projektdurchführenden bekannt sein	ja, Redis war durch WBA2 bekannt
3	die Priorisieung soll nach Attributen der Nachrichten mit dem Datentyp String möglich sein	nein
4	Priorisierung und Abfrage durch einen Client müssen voneinander gekapselt sein	ja
4.1	die Priorisierung der Nachrichten darf nicht in die Verarbeitung der Anfrage eines Client injiziert werden	ja
4.2	bei der Verarbeitung einer Client Anfrage sollte vor der Abfrage der Datenstruktur keine separate Priorisierungsstruktur angefragt werden	ja
5	die Datenbank muss adäquate Auswahl- und Sortierungsoperationen zur Verfügung stellen, insbesondere muss vermieden werden für die Priorisierung	teilweise
5.1	eine eigenständig Struktur zur Sortierung entwickeln zu müssen	nein
5.2	explizit durch alle Nachrichten zu iterieren und die gespeicherten Nachrichten zu verändern	nein

Status: fehlgeschlagen

Fazit: Der PoC erfüllt nicht alle Anforderungen, insbesondere die Nichterfüllung der Bedingungen 5.1 und 5.2 gilt als Ausschlusskriterium

Quellcode/Protokoll: