Projektangebot

Inhaltsverzeichnis

1. Allgemein

Ziel unseres Projektes ist es die Funktionalität von BOA bezüglich der Verwendung verschiedener Oberflächenformen und Einheiten während der Arbeit mit den Pattern zu verbessern. Unser Programm zeichnet sich auf Grund seiner Architektur(Plug-Ins) besonders dadurch aus, dass es leicht in den gesamten Vorgang von BOA integrierbar ist. Des Weiteren soll durch die Verwendung von Threads die Leistung unseres Programms optimiert werden. Es ist möglich, dass ein Objekt verschiedene Oberflächenformen hat. Ein typisches Beispiel sind hierbei die verschiedenartigen Möglichkeiten ein Datum oder eine Masse in einem Text anzugeben.

2.09.1990	1,5 kg
12. September 1990	1,5 Kilogramm
zwölfter September 1990	0,0015 t
zwölfterSeptember Neunzehnhundertneunzig	1500 g

Unser Framework soll in der Lage sein, für ausgewählte Datentypen und Einheiten diese verschiedenen Formen eigenständig von einer gegebenen Oberflächenform abzuleiten.
Des Weiteren ist es in der Lage in einem Text festgelegte Datentypen zu finden und eine Annotation zu erstellen.
Letztendliches Ergebnis soll ein GUI-Prototyp sein, der auch mittels einiger Beispielsätze die Funktionalität des Programms erläutert.

2. Projektübersicht

Definierte Datentypen, die vom Framework erkannt werden:

- Datum
- Masse
- Längeneinheit
- Temperatur

Das Framework wird in Plug-In Struktur aufgebaut. Es ist beliebig durch weitere Einheiten erweiterbar.
Aus den Grundeinheiten ergeben sich folgende Umrechnungsmöglichkeiten:

- cm, m, km, ft
- mg, g, kg, tonne

- Grad Celsius, Grad Fahrenheit, Kelvin

Auch diese Umrechnungseinstellungen sind beliebig erweiterbar.

2.1 Erstellung von Annotationen

Zur Annotationserstellung wird ein Satz benötigt der einen von uns definierten Datentyp enthält. Wird in dem Satz nun der Datentyp gefunden so gibt unser Programm das Label, den Datentyp und die Stelle an dem das Label gefunden wurde aus.

Beispielsweise sei der Satz:

Toms Geburtstag ist am 19.12.2002.

Unsere Annotation würde wie folgt aussehen:

(19.12.2002) (Datum) (25)

2.2 Erstellung von Oberflächenformen

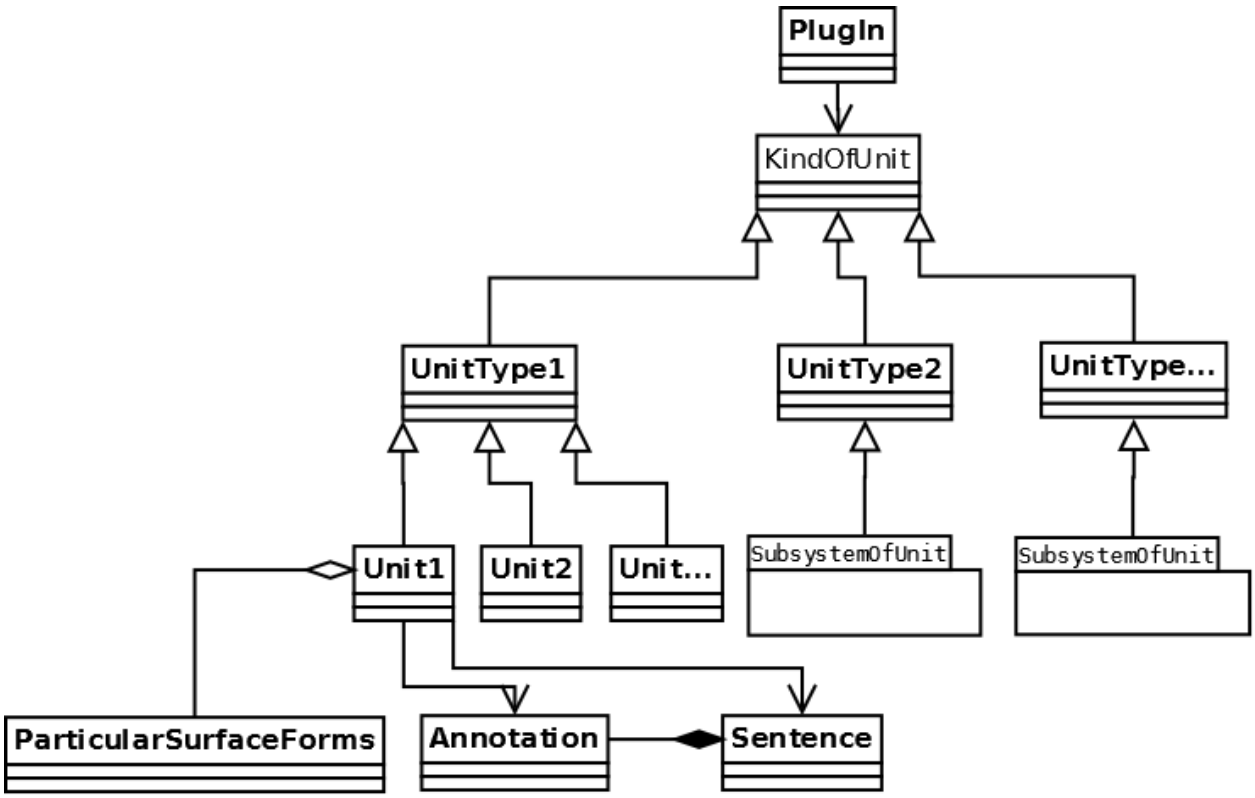
Bei der Erstellung der verschiedenen Oberflächenformen wird unserem Programm ein Label und der Datentyp übergeben. Diese können auch durch die in 2.1 angedeutete Annotationserstellung erstellt worden sein. Auf Basis der einzelnen Datentypen definieren wir die verschiedenen Möglichkeiten in denen ein Datentyp auftreten kann. Bei Einheiten wie Masse, Länge oder Temperatur ist das Umrechnen in andere Einheiten ebenfalls Bestandteil unseres Programms. Für den oben angegebenen Beispielsatz: Geburtstag ist am 19.12.2002. würde zur Oberflächenerstellung nur noch das Label: 19.12.2002 und der Datentyp: Datum benötigt. Daraus ließen sich nun mögliche weitere Oberflächenformen ableiten. Beispielsweise:

Beispielsweise:

- 19. Dezember 2002
- neunzehnter Dezember 2002
- neunzehnter Dezember zweitausendundzwei

3. Grundsätzliches Struktur- und Entwurfsprinzip

3.1 Klassendiagramm



Erklärung zum Klassendiagramm:

Erklärung zum Klassendiagramm: (Klassennamen können sich während des Projektes ändern)

Die Klasse PlugIn ist die Hauptklasse für die Erweiterung. In ihr ist die Schnittstelle für das Hauptprogramm definiert. Zudem werden hier schon die Vorarbeiten für die Hauptfunktionen erledigt, d.h. von hier aus werden die zu überprüfenden Sätze an die verschiedenen Einheiten der Erweiterung weitergeleitet und die Ergebnisse der Extraktionen später wieder zusammengeführt. Zudem werden zusätzlich mitgelieferte Properties ausgewertet und Strings entsprechend zur Weiterverarbeitung weitergegeben.

Die Klasse KindOfUnit ist die Basisklasse für das Unterprogramm. Sie definiert den genauen Ablauf der Suchalgorithmen und enthält unter anderem eine Methode, die den Satz in Tokens aufteilt und jedes dieser Token separat nach speziellen Werten

durchsucht. Diese hängen von der Art der Einheit ab und sind in der zugehörigen Klasse ParticularSurfaceForms abgespeichert.

UnitType bildet eine Subklasse von der KindOfUnit-Klasse und gleichzeitig die Basisklasse für zusammengehörende Unit-Klassen. In ihr sind Werte hinterlegt, mit denen man zwischen den zusammengehörenden Einheiten umrechnen kann, was auch die korrekte Auflösung von Präfixen mit einschließt, und sie vererbt diese dann zusammen mit den von KindOfUnit geerbten Methoden weiter.

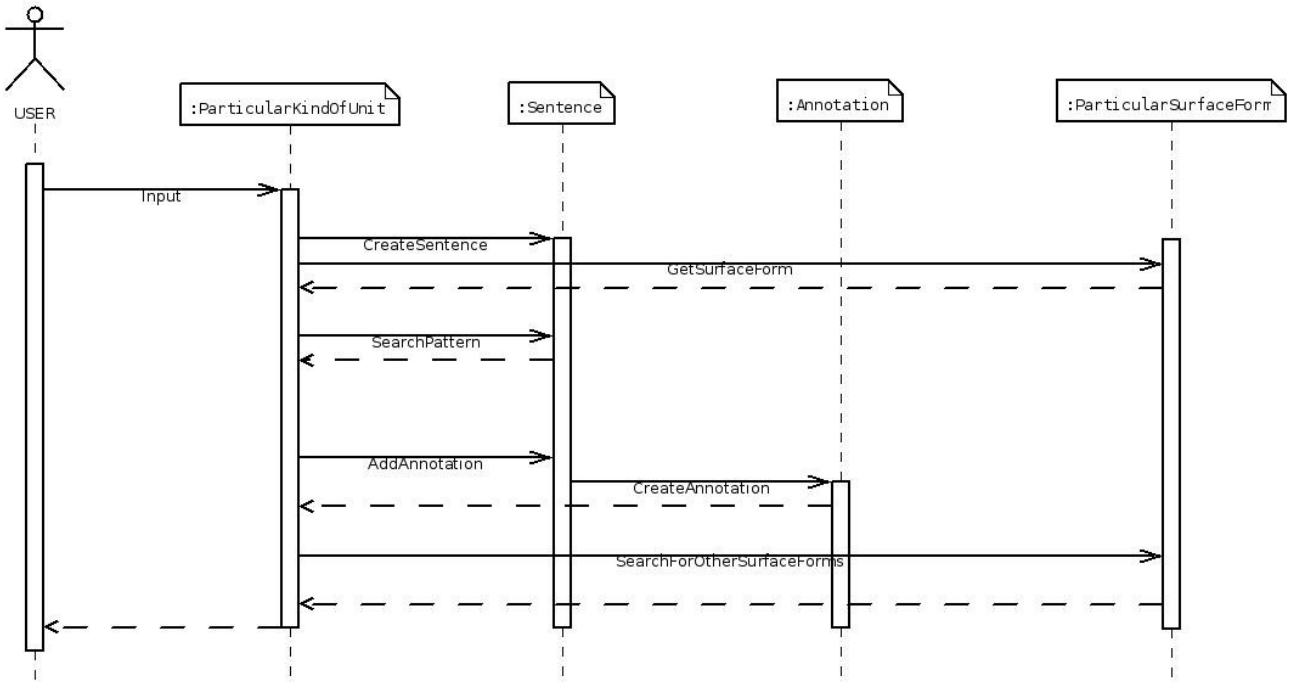
Die Unit-Klasse ist die zentrale Klasse des Unterprogramms. Hier werden die geerbten Methoden ausgeführt. Dazu hat sie direkten Zugriff auf die drei folgenden Nebenklassen: ParticularSurfaceForms, Sentence und Annotation.

Die erste Nebenklasse, Sentence, ist für die Datenspeicherung zuständig. Sie enthält den zu durchsuchenden Satz sowie eine Liste der Annotationen, die beim Durchsuchen des Strings gegebenenfalls erstellt werden.

Die zweite Nebenklasse ist Annotation. Sie stellt Informationen über Label, Typ und Index bereit, sprich Name des Tokens, Einheiten-Typ und Position des Tokens innerhalb der vorher erstellen Token-Liste.

ParticularSurfaceForms bildet die letzte Nebenklasse und ist ein Teil der Unit-Klasse. In ihr sind alle Oberflächenformen der zugehörigen Einheit definiert, die für die effiziente Suche nach der Einheit benötigt werden. Zudem wird sie auch verwendet, um eine Liste von Oberflächenformen eines speziellen Wertes zu generieren, welche entsprechend an die Hauptsteuerung in der Klasse PlugIn zurückgegeben wird.

3.2. Sequenzdiagramm



Für die Annotation von Oberflächenformen eines bestimmten Datentyps verwendet unser Projekt Multithreading. Das obige Sequenzdiagramm zeigt hierbei den Ablauf eines Threads, um eine Oberflächenform eines bestimmten Datentyps zu annotieren.
Zunächst wird vom Benutzer ein (in englisch geschriebener) Satz zum Beispiel über eine Test-GUI eingegeben. Dieser Satz wird zunächst in die Klasse *Sentence* übernommen, wo die Liste der zugehörigen Annotationen als leer initialisiert wird und das Programm den entsprechenden String mit den Oberflächenformen aus *ParticularSurfaceForms* der jeweiligen Einheit vergleicht. Wird dabei eine Übereinstimmung gefunden, so wird für jeden Match eine Annotation der Form `<Label, Typ, Position des Labels im Text>` hinzugefügt. Wurden alle Muster abgeglichen, so wird für jede erstellte Annotation nach weiteren Oberflächenformen des jeweiligen Labels gesucht. Dabei müssen zugehörige Zahlenwerte für Oberflächenformen von anderen Einheiten des gleichen Obertyps umgerechnet werden. Wenn auch dieser Vorgang beendet ist, wird für den nächsten Datentyp ein neuer Thread gestartet und diesem der annotierte Satz sowie die weiteren Oberflächenformen jeder Annotation übergeben.

4. Anforderungen

Die Anforderungen an unser Programm unterteilen sich in zwei Bereiche.
Die zwingenden und die optionalen Anforderungen. Jedem Ziel wird ein geschätzter Aufwand zu geordnet, dieser steht im Verhältnis zum Zeitaufwand den das Gesamtprojekt einnimmt.

4.1 Zwingende Anforderungen

Zwingend soll unser Programm alle gestellten Aufgaben für die Datentypen der Länge , des Gewichtes und für Datumsangaben erfüllen. Für die Umsetzung dieses Ziels planen wir circa 60% des Aufwandes ein. Ein weiteres zwingendes Ziel stellt bei uns die Erweiterbarkeit unseres Programms dar. Das Hinzufügen weiterer Einheiten und Datentypen soll leicht möglich sein.
Dieser Aspekt geht in unsere Aufwandsplanung mit 20% ein.
Die restlichen 20% werden für die Verwendung von Multithreading verwendet.
Multithreading ermöglicht es uns ermöglicht es uns einen Satz gleichzeitig nach mehreren Einheiten bzw. Datentypen zu durchsuchen.

Umsetzung für Gewicht,Längeneinheiten und Datum:	60%
Erweiterbarkeit:	20%
Multithreading:	20%

4.2 Optionale Anforderungen

Eine unserer möglichen Erweiterungen stellt das Hinzufügen der Größen Temperatur und Geschwindigkeit dar. Der Aufwand würde für jede Größe bei 10% liegen. Eine weitere Möglichkeit ist es neben dem Datum, ebenfalls die Zeit als Datentyp zu implementieren. Dies schätzen wir als schwieriger ein, da es sehr aufwendig ist das Auftreten von Zeit zu normalisieren. Deshalb würden wir diesen Aspekt mit 25% Arbeitsaufwand bewerten.
Zu den optionalen Anforderungen ordnen wir ebenfalls die Möglichkeit, einzelne Einheiten Plug-Ins auszuwählen oder abzuschalten ,um die Leistung des Programms zu optimieren,ein.
Des Weiteren wäre es möglich dem Benutzer die Mittel zu geben, selber neue Einheiten zu einem bereits vorhandenen Einheitentyp hinzuzufügen. Dafür müsste er lediglich die Bezeichnung und die Umrechnung in unsere Standardeinheit angeben. Die Implementierung dieser Anforderung liegt bei 5%. In unseren zwingenden

Anforderungen ist bereits eine einfache GUI mit integriert. Eine mögliche Erweiterung dieser sehen wir bei etwa 5% Mehraufwand.

Umsetzung für Temperatur:	10%
Umsetzung für Geschwindigkeiten:	10%
Umsetzung für Zeit:	25%
Ein-Abschalten einzelner Plug Ins:	5%
Möglichkeit für Hinzufügen von neuen Einheiten für den Benutzer:	5%
Erweiterung der GUI:	5%

Letztendlich kommen wir also auf eine Summe von 160% Arbeitsaufwand, daraus lässt sich ableiten, dass es uns nicht möglich sein wird alle optionalen Anforderungen zu erfüllen. Möglicherweise sind diese durch nachfolgende Semester zu realisieren.

5. Entwicklungsphasen

Basierend auf unserem UML-Entwurf lässt sich die Umsetzung des Projekts in folgende vier Bereiche aufteilen:

- Erstellung eines 1-Einheiten-Plugin-Prototyps (ca. 45%)
- Umsetzung der Funktionalitäten für einen kompletten Einheitentyp (ca. 25%)
- Komplettierung und Erweiterbarkeit (ca. 20%)
- Evaluation der (Teil-)Programme (ca. 10%)

Diese Aufzählung ist jedoch nicht streng chronologisch zu verstehen, sondern gibt vielmehr die Reihenfolge an, in der mit dem vorläufigen Abschluss der jeweiligen Projektabschnitte zu rechnen ist. Die Prozentzahl in Klammern entspricht dabei dem geschätzten Arbeitsaufwand gemessen am Gesamtprojekt.

1. Erstellung eines 1-Einheiten-Plugin-Prototyps

Diesem Abschnitt des Projekts liegt die Implementierung der Kernfunktionalität zugrunde. Er beinhaltet das Schreiben eines Plugins, welches zunächst nur für eine ausgewählte Einheit in gegebenen Sätzen entsprechende Annotationen erstellen kann und auf Anfrage weitere Oberflächenformen für das jeweilige Label generiert. Es muss also die Grundstruktur der gesamten Klassenhierarchie bereits umgesetzt werden, die einheitenspezifischen Klassen sollen dabei als Schablone für später zu implementierende Einheiten dienen.

2. Aufbau der Plugin-Hierarchie eines Einheitentyps

Im Zuge dieses Projektteils wird das Plugin in der Hinsicht erweitert, dass es nun mit mehreren Einheiten desselben Einheitentyps (z.B. Gewicht) arbeiten kann. Wichtige Aspekte sind die Koordination des Zugriffs beim Multithreading und die gemeinsame Nutzung der einheitenspezifischen Oberflächenformen sowie die Umrechnung der Einheiten untereinander.

3. Komplettierung und Erweiterbarkeit

Die letzten Schritte der Implementierung entfallen auf das Hinzufügen der restlichen (von uns ausgewählten) Einheiten und die Gewährleistung einer einfachen Handhabbarkeit des Plugins für den Benutzer. Dies beinhaltet auch Mechanismen zum späteren Einbinden weiterer Datentypen und das Erstellen einer oder mehrerer Schnittstellen für Fox oder andere frei zugängliche Extraktionswerkzeuge.

4. Evaluation der (Teil)-Programme

Nebenläufig zur Implementierung der einzelnen Programmkomponenten erfolgt die Entwicklung eines Evaluationskonzepts, welches Inhalt unseres Vorprojekts und deshalb im entsprechenden Dokument auch genauer beschrieben ist. Es ist dabei zu erstreben, das Evaluationsmodul zum Zeitpunkt der Fertigstellung des ersten Prototyps bereits einsetzen zu können, um zu jeder Zeit einen guten Überblick über die tatsächliche Funktionsfähigkeit des Plugins zu haben und gegebenenfalls schon früh Verbesserungen oder Erweiterungen der Kernmethoden in den Projektplan integrieren zu können.