Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

Bachelor Thesis

Evaluation of Argumentation Schemas for the Identification of Software-Architecture Knowledge in Developer Communities

Joël Harms

3harms@informatik.uni-hamburg.de

Course of Studies: Software-System-Development

First Supervisor: Prof. Dr. Chris Biemann

Second Supervisor: Mohamed Soliman

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CSV	Comma Separated Value
HTML	Hypertext Markup Language
ID	Identification Number
SO	StackOverflow

1 Introduction

According to a study conducted in March 2017 by the American hard disk manufacturer Seagate and the International Data Corporation, global data volumes will increase tenfold by 2025[1]. It can be assumed that a huge amount of that data is in text form. To handle this huge amount of text data, Argumentation Mining is becoming a hot topic.

Argumentation Mining consists of several subject areas. These include natural language processing, argumentation theory and information retrieval. The aim of Argumentation Mining is the automatic extraction of argument structures from unstructured text in order to provide a data basis for machine processing (Lippi and Torroni 2016). In this way, it will be possible to index and search through huge amounts of text data available on the internet for relevant arguments, whether in online journals, product reviews, blogs or forums. It thus supports the collection of information which is not present in other sources.

Soliman et al. dealt with that kind of problem. The goal of their work was to find out if it is possible to use an internet forum (StackOverflow[2] (SO) in this case) as the basis for research in the area of software architecture solutions. Because SO is generally used by developers who are looking for solutions for coding problems, Soliman, Galster, and Riebisch developed an ontology that identifies architectural knowledge concepts on SO (2017). The data generated by Soliman, Galster, and Riebisch serves as the data basis for the argument annotation process which is conducted in this bachelor thesis (Soliman et al. 2016; Soliman, Galster, and Riebisch 2017).

[1] https://www.seagate.com
[2] https://www.stackoverflow.com

In this thesis, argument annotation is defined as a sub process of Argument Mining in which human annotators annotate a text with the help of predefined guidelines and a given annotation schema in order to create training data for supervised machine learning algorithms. In this thesis we annotate the same data that was used by Soliman et al. (Soliman et al. 2016) with one established argumentation schema - the claim premise schema. The data is evaluated in order to examine the usability to cover Solimans, Galsters, and Riebisch's ontology (2017). This would allow the automatic extraction of Solimans, Galsters, and Riebisch's schematized information on new data using established reasoning argument mining algorithms (2017).

It is assumed that:

> *The argument structures that have been found by the annotators overlap with the ontology classes, for example benefits and drawbacks, which were found by Soliman, Galster and Riebisch in the architecture-relevant posts on SO* (2017).

The aim of this work is to corroborate the hypothesis.

Chapter 2 covers the related work including argumentation theory in general, the argumentation mining process, the results of Soliman et al. and introduces some common argumentation schemas. In Chapter 3 the methodology is presented, starting with the preparation process for the data, followed by the annotation process and the best practices. Lastly the finalized guidelines are presented. In Chapter 4 the annotation process is evaluated. Thereby the inter annotator agreement is illustrated. We show the overlap with the ontology classes developed by Soliman, Galster and Riebisch with our data. Chapter 5 summarizes the results and discussed them. Finally a prospect for future work is given.

2 Related Work

In this chapter the related work is presented starting with the argumentation theory in general following with the argumentation mining process and the results of Soliman, Galster and Riebisch. At the end of this section some common argumentation schemas are introduced.

2.1 Argumentation Theory

This chapter provides an introduction to the topic of argumentation.

> *"Argumentation is a multidisciplinary research field, which studies debate and reasoning processes, and spans across and ties together diverse areas such as logic and philosophy, language, rhetoric and law, psychology, and computer science"* (Lippi and Torroni 2016).

So argumentation is a highly complex interdisciplinary topic.

Van Eemeren, Grootendorst and Snoeck Henkemans describe argumentation as a verbal activity that is done in an ordinary language in order "to state, question or deny something, to respond to statements, questions or denials and so on" (van Eemeren, Grootendorst, and Snoeck Henkemans 1996). Argumentation can be supported by the use of facial expressions and gestures. It is important to note that argumentation can appear without these nonverbal means of communication, but it is not possible the other way around (van Eemeren, Grootendorst, and Snoeck Henkemans 1996), which is the case in the posts of SO. Because there is only text and no videos or live conferences, the argumentation occurs without gestures and mimic.

To make it a less complicated for the context of this thesis a simplified definition of argumentation is given: "Argumentation is defined as the act or process of forming reasons and of drawing conclusions […]" (Moens 2013).

Argumentation theory is a much-discussed field of research. Via the examples, an insight into and a basic understanding of the topic is given. In the next chapter an overview of the whole argumentation mining process is presented to show where our project is to be placed.

2.2 Argumentation Mining

Argumentation mining is the computational approach to automatically extracting argumentation from text corpora in order to provide data for machine learning processes (Rocha, Cardoso, and Teixeria 2016):

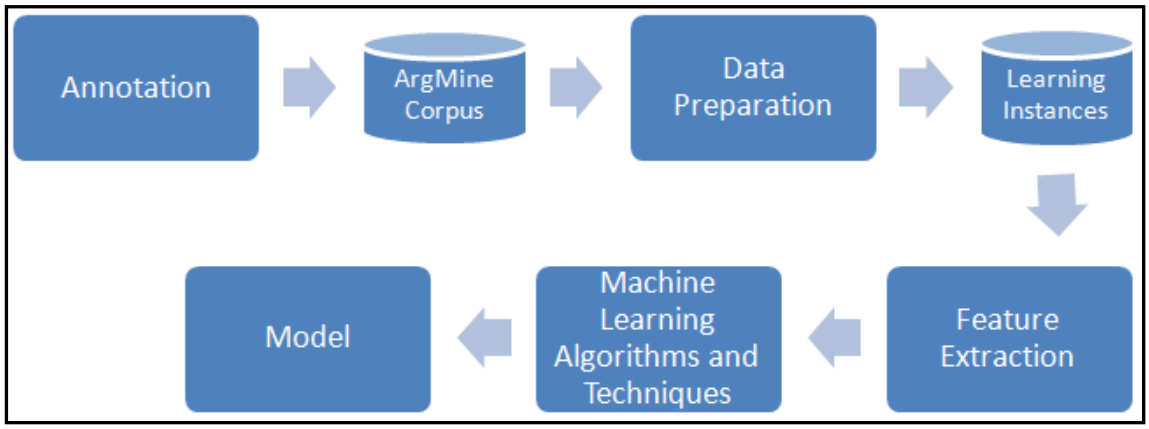


Figure 2.1: The five Steps of the Argumentation Mining process.

As illustrated in Figure 2.1, the whole process of argumentation mining can be classified into five steps. During the annotation phase human annotators annotate any kind of text in order to create a so called corpus. In the next step the data of the corpus is transformed into learning instances. In the third step a set of features is selected that represents the data best. During the next phase several machine learning algorithms and techniques are used to create a model of the argumentation.

In this bachelor thesis we focus on the first step, presented in Figure 2.2. We annotate posts that contain software architecture relevant knowledge by using

the claim-premise schema. Furthermore we try to figure out whether the argumentation overlaps with the ontology classes that have been found by Soliman, Galster, and Riebisch (2017). In the next chapter we present the work of Soliman et al. (2016/2017).

2.3 **Argumentation Mining Datasets**

In this section an overview of the papers by Soliman et al. is given (2016, 2017). In their first paper "Architectural Knowledge for Technology Decisions in Developer Communities" from 2016, Soliman et al. tested a new approach to gather architecture knowledge from a popular online developer community named SO. There are several reasons for their approach. The most important one is that "Architectural decisions have a big influence on basic properties of a software system […]". Another aspect is that the software architects have to gather the information about architecture knowledge manually because there are not enough sources to acquire the knowledge.

SO was chosen because it is the largest software developer community. Posts on SO follow a question and answer structure. "The quality of the knowledge on SO is ensured through evaluation of posts from users". Furthermore there is the possibility for users to rate the answers that were given. This possibility ensures the quality as well. The posts are constantly updated, which leads to the information being current.

Because SO is primarily used for software relevant questions, Soliman et al. conducted an empirical study to identify posts that contain technology related information (2016). The overview they have received is supposed to be used to perform further analysis steps such as automatic mining and classification of architecture relevant posts.

They identified that architecture relevant posts are based on two dimensions:

1. The purpose of the question
2. The solution type of the question

For the purpose dimension they classified the following sub-types:

1. Solution Synthesis:

   "[C]oncerned with searching for suitable technology solutions, which have certain characteristics […]; address a design problem or context".

2. Solution Evaluation:

   "[C]oncerned with assessing one or more proposed technology solutions. The evaluation of solutions could be done individually or through a comparison between different alternative solutions. In addition, several concepts are considered during evaluation, such as technology features, benefits and drawbacks, suitable use cases, and quality attributes".

3. Multi-purpose:

   "[T]his type of ARP [architecture relevant posts] comprise both types of posts, solution evaluation and synthesis. Several questions are asked within a single post".

For the solution type dimension, the following types were defined:

1. Technology Feature:

   "[F]ocus on specific features of a technology[…]".

2. Technology Bundle:

   "[C]onsider the technology as a single architecture solution without referring to the features within the technology".

3. Architecture Configuration:

   "[C]oncerned with the components and connectors design configurations".

4. Combined Solution:

   "[C]oncerned with different solution types".

In summary, it can be said that Soliman et al. demonstrated the relevance of SO as a source for software architecture knowledge in their 2016 paper.

Building on this knowledge they conducted a qualitative content analysis in order to define an ontology. For this purpose Soliman, Galster, and Riebisch selected a sample of 105 architecture relevant posts. They have identified eleven acknowledgement concept classes which are presented in Figure 2.2.

(ID) Name and Description
(**CONF**) *Architecture Configuration*: represents part of an architectural model, which consists of one or more component names associated with an architecture connector verb or name.
(**CB**) *Component Behavior*: describes the behavior of an architecture component. It gives an overview about the type of implemented logic and complexity. Sometimes internal operations are mentioned during the description.
(**EX**) *Existing System*: describe part of an architecture of an existing software system. It additionally describes the possible problems in the system.
(**DI**) *Design Issue*: users express their design problems through describing the architecture configurations of a planned design, or the architecture configuration design of an existing software system.
(**REQ**) *Requirement and Constraint*: two main types of requirements were found: 1) Quality attribute requirements, and 2) Technology features requirements. In addition, we found three types of constraints: 1) Technical skills constraint. 2) Development time constraint. 3) Solution constraint.
(**UR**) *User Request*: exist in ARP question or title in a form of questions or needs. It complements design issue, requirements and constraints by showing the type of architecture activity (evaluation or synthesis).
(**FEAT**) *Technology Features*: Two main types of technology features: 1) Development features are expressed through certain programming activities (e.g. debugging) or programming features and tools (e.g. code generation), 2) Behavioral features are expressed through technology specific component and class names, as well as their implemented architectural patterns or their relationship with other technologies.
(**ASTA**) *Technology Benefits and Drawbacks*: They are distinguished through the extensive usage of adjectives and adverbs in combination with technology features and quality attributes. The adjectives or adverbs are used to express the advantages or disadvantages of certain technology solutions or features.
(**CASE**) *Technology Use-Cases*: These are either success or failure stories for the usage of technology solutions at certain contexts. The stories could be coming from personal experiences of users, or well-known examples for existing systems. The context associated with stories could include domain description, architecture configurations, infrastructure, and constraints.
(**ADD**) *Recommended Design Decisions*: They are recommendation from users based on their experience or opinion for certain architectural solutions.
(**DR**) *Decision Rules*: Conditional recommendation for architectural solutions. The rule condition might involve other ontology classes such as requirements, constraints and architectural configuration. recommendations involve recommended ADDs for certain technology solution or architecture configuration.

Figure 2.2: Eleven acknowledgement classes found by Soliman, Galster, and Riebisch (2017).

In the next chapter we present different argumentation schemas that can be used to investigate the argumentation of the SO posts.

2.4 Argumentation Schemas

In this section we present some argumentation schemas that can be used in order to annotate web content.

2.4.1 Claim Premise Schema

Besnard and Hunter define an argument as:

> *"[A]n argument is a set of assumptions […], together with a conclusion that can be obtained by one or more reasoning steps […]. The assumptions used are called the **support** (or, equivalently, the **premises**) of the argument, and its conclusion […] is called the **claim** […] of the argument. The support of an argument provides the reason […] for the claim of the argument"* (Besnard and Hunter 2010).

So an argument consists of a set of premises and a single claim. The set of premises can be empty. A claim can support another claim, but this type of argumentation has to be considered a weak one, since it is unclear whether the supporting claim is true. Because there is a premise that can support a claim it logically follows that there is a premise that can attack a claim as well. Therefore the claim premise schema has four basic elements:

- Claim
- Premise
- Support
- Attack

There is an example in Figure 2.3 that illustrates what a claim, a premise, a support and an attack are. The blue box contains a claim. You cannot state whether the weather is nice or not, because everyone has a different opinion of what nice weather is. But you can state if the sun is shining at the moment or not, so the message becomes provable. When that is the case the premise

supports the claim. We can also see an attack in the Figure 2.3. The statement in the red box is an attack on the statement in the blue one. So in this case we have a claim that is supported and attacked by two different premises.

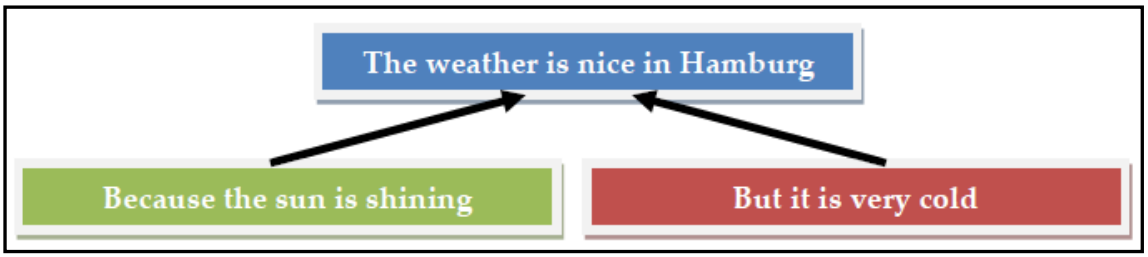


Figure 2.3: A simple example of the claim premise schema.

In the next chapter the Toulmin Model will be presented.

2.4.2 Toulmin Model of Argumentation

In his book "The Uses of Argument" Stephen Toulmin established a model for argumentation which has become rather important in the research field of argumentation theory. This model, the Toulmin Model of Argumentation, basically consists of six main components (Toulmin 2008):

- Claim (or conclusion)
- Fact (or data, ground, evidence)
- Warrant
- Backing
- Qualifier
- Rebuttal

The claim is a statement in which the speaker tries to convince his listener of his point of view. To substantiate the claim the speaker provides facts to convince his counterpart. It must be possible to decide whether a fact is right or wrong. A warrant is the link between the claim and the facts which are related to a claim

by showing the relevance of the facts. This can happen in an explicit or implicit way. The backing offers additional support for a warrant, especially in situations where the warrant cannot confirm a fact by itself.

These four attributes are considered as the basic structure of an argument. The qualifiers and rebuttals occur in more complex argumentative structures. Qualifiers provide conditions under which a claim can be considered true, so they limit the claim. In rebuttals however the speaker tries to refute possible counter-arguments that a listener might have or sets conditions for a the claim to hold (Lippi and Torroni 2016).

A minimum argumentation that follows the Toulmin model of argumentation consists of at least one fact, one warrant and one claim element (Kneupper 1978). The warrant can be implicit or explicit so it might be the case that a fact and a claim are the only two visible arguments in a minimum argumentation object. Figure 2.4 shows an example of an argument containing all elements that are introduced by Toulmin.

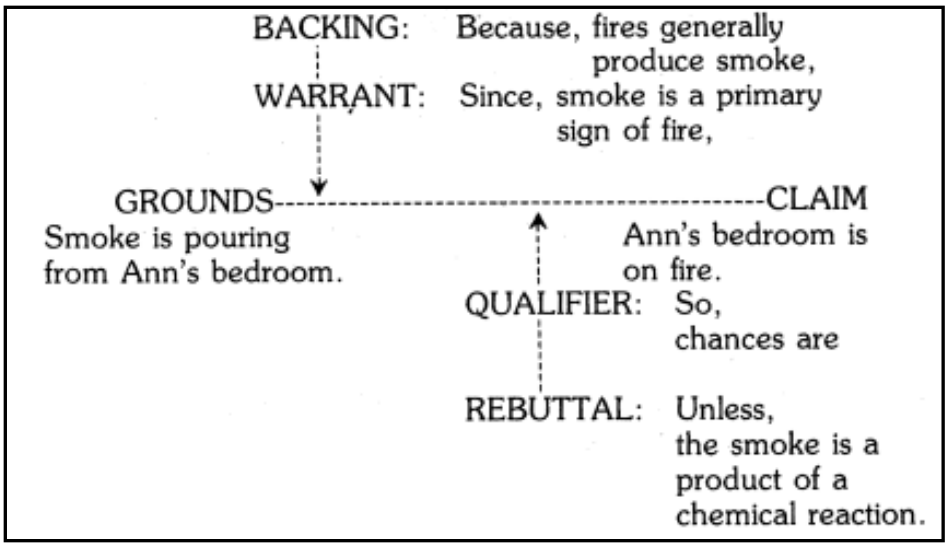


Figure 2.4: All elements of the Toulin Model in action (Karbach 1987).

In the next chapter the stance-based argument mining idea is presented.

2.4.3 Stance-based Argument Mining

Another interesting approach to annotate web content is stance-based argument mining. Stance-based argument mining is an approach that is used to annotate implicit argumentation which often occurs in informal settings like forums (Wojatzki and Zesch 2016). For instance *#JesusOrHell* is a hash tag used in the debate about atheism. It can be assumed that the person who used this hash tag "is against atheism, because the bible says that this will result in a stay in hell after death. However, both claims are never explicitly mentioned" (Wojatzki and Zesch 2016). Since explicit information can be absent, the stance-based argument mining approach follows the idea that the claim corresponds to the overall topic in which the statement was made. In a controversial topic, stance is defined as being in favour of or against a specific theme.

For stance classification it is not only of interest whether someone is for or against a topic. Furthermore the strength of the position of the annotators towards the target of research is significant. Therefore annotators often have to choose from different options. For instance the annotators have to categorise whether they are: "strongly for", "for", "other", "against" or "strongly against" (Sobhani, Inkpen Diana, and Matwin 2015).

3 Methodology

In this chapter we describe the whole annotation mining process that we conducted. We follow the order of the annotation process, beginning with the preparation of the data followed by the annotation process. Finally the best practices and the emergence of the guidelines will be presented.

3.1 Preparation of Data

For this bachelor thesis we received a file from M. Soliman, which was provided in the form of a Comma Separated Value (CSV). This file contains about 100 architecture-relevant posts of SO. The posts are sorted according to their probability to contain architecture knowledge.

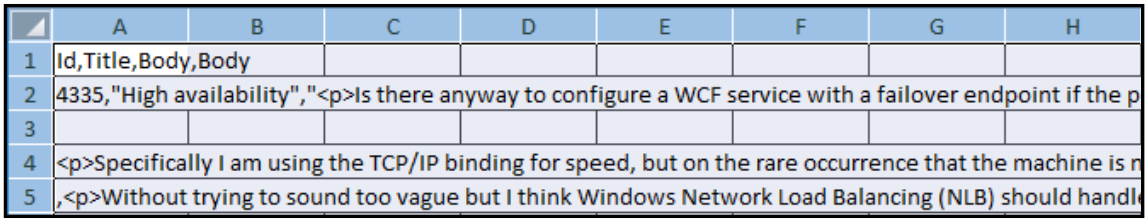


Figure 3.1: Example of the CSV file that we have received from M. Soliman.

The first line of the file that is represented in Figure 3.1 is called the header. The header contains four elements: An ID, a title, a body and another body. The ID is the ID of the post on SO. The title represents the topic of the question that was asked on SO. The first body contains the question itself. The second body is one answer to the question. For each answer on SO there is one line in the CSV file.

In the first step of the data preparation we removed the Hypertext Markup Language (HTML) tags. In the next step we used a tokenizer in order to split

the sentences into tokens and transformed the CSV file into the CoNLL-U format.

The CoNLL-U format contains of ten fields[3]:

1. ID: Word index, integer starting at 1 for each new sentence; may be a range for multiword tokens; may be a decimal number for empty nodes.
2. FORM: Word form or punctuation symbol.
3. LEMMA: Lemma or stem of word form.
4. UPOSTAG: Universal part-of-speech tag.
5. XPOSTAG: Language-specific part-of-speech tag; underscore if not available.
6. FEATS: List of morphological features from the universal feature inventory or from a defined language-specific extension; underscore if not available.
7. HEAD: Head of the current word, which is either a value of ID or zero (0).
8. DEPREL: Universal dependency relation to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one.
9. DEPS: Enhanced dependency graph in the form of a list of head-deprel pairs.
10. MISC: Any other annotation.

We wanted to use the ID, FORM, LEMMA and UPOSTAGs, but some problems occurred while importing our data into our annotation tool WebAnno, a multipurpose linguistic annotation tool, so we decided to use only the ID and the FORM instead (see Figure 3.2).

[3] http://universaldependencies.org

```
# newdoc id = 4335
# newpar id = 4335p2
# sent_id = 4335s2
# text =  Is there anyway to configure a WCF service with a
1   Is    _   _   _   _   _   _   _   _
2   there     _   _   _   _   _   _   _   _
```

Figure 3.2: Example of our data in CoNLL-U Format.

After the import into WebAnno our data looks like the example presented in Figure 3.3:

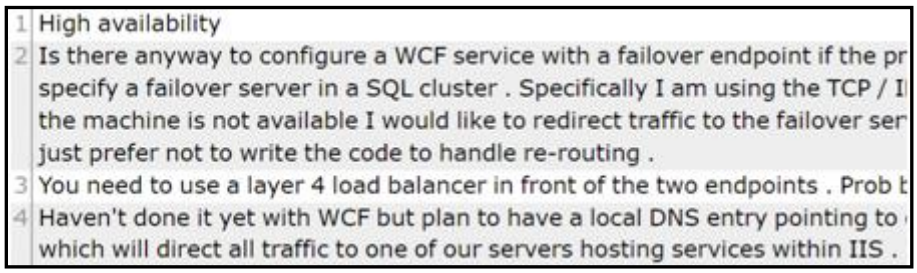


Figure 3.3: Example of our data imported to the annotation tool WebAnno.

In the next chapter we present the process of the guideline development.

3.2 The Annotation Process

We have annotated 14 documents which have a high chance to include architecture knowledge. The documents were annotated by three annotators who have a background in computer science.

For the annotation process we used the tool WebAnno. WebAnno is a web-based tool that enables distributed work. There is no installation effort and a high availability. It also has the possibility to unlock a very large distributed workforce which may be interesting for feature work. Another important aspect is that it is open source so it comes at no costs (Yimam et al. 2013).

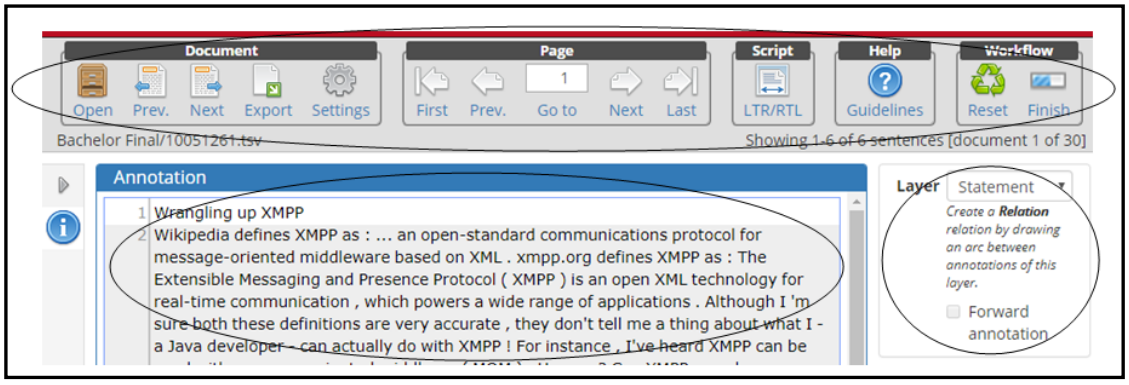


Figure 3.4: Annotator interface of WebAnno.

In Figure 3.4 the configuration of the annotator interface of WebAnno is presented. There are three highlighted areas. The one at the top highlights the annotator menu. The annotator can open a document or skip forwards and backwards through the different documents. It is possible to export a specific text. The annotator has the opportunity to customize the configuration of WebAnno in the settings.

It is possible to navigate through different pages of a document. With the button below “Script” it is possible to switch between a left-aligned and a right-aligned position of the text. With a click on “Guidelines” the annotator can take a look into the guidelines whenever it is needed. It is possible to reset and finish a document.

The highlighted left corner represents the post of SO and the right presents the different layers that can be used.

We used the claim premise schema to analyze the software architecture relevant posts, because it is a comparatively simple but powerful annotation scheme. The Toulmin Model of Argumentation however, is a relatively complex schema, so we decided not to use it. Since the posts on SO are not considered particularly controversial we decided not to annotate with the stance-based argument mining approach either.

In our annotation study following the claim premise schema a complete argumentation consists of a claim or conclusion and at least one premise "that has truth-value" (Rocha, Cardoso, and Teixeria 2016), which provides evidence for the claim. A claim expresses an opinion that either is argued in favour of or against. Support or denial is performed by giving evidence in form of premises. In an argumentation these premises should be related to a claim, because an enumeration of facts without any relation to a claim is not an argumentation that follows our definition presented in Chapter 2.1.

There are three different types of relations that can occur. The relation in which the claim is supported by a premise is named *supporting*. Accordingly, a relation with a claim that is attacked by a premise is called *attacking*. The whole construct of at least one claim and one premise that relates to the claim is called an argument (Figure 3.5).

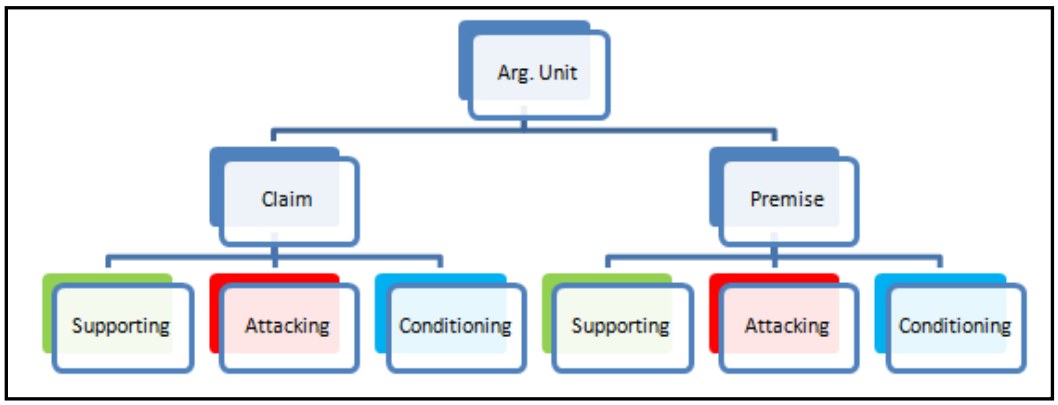


Figure 3.5: A complete argumentation unit consists of one claim and one premise that can have different types: supporting, attacking and conditioning.

As seen in the figure we have added another relation which is called *conditioning*, because in many posts on SO a claim is brought forward that depends on a specific condition. This condition is often expressed through an if-

sentence. Therefore in this annotation study with the claim premise schema the type of a relation can be supporting, attacking or conditioning.

3.3 Best Practices and Emergence of the Annotation Guidelines

During our annotation study we tested several different set ups in WebAnno. We wanted to optimize the annotation speed and the clarity during the annotation.

In our study each sentence of WebAnno contains one part of the post on SO. For example, the first sentence of WebAnno contains the topic of the original post. In the second sentence the question that has been asked on SO can be found. And in the following sentences are located all answers question(s).

The experiences made during the annotation process are presented in the following chapters.

3.3.1 First Approach

In the beginning we started with identifying the statements and defining whether a statement is a claim, a premise or that we are uncertain if it is a claim or a premise.

In the case of uncertainty the annotator has to state a tendency – claim or premise. In the next step we draw the relations between the different statements and specify their types.

Figure 3.6: Example of the first annotation attempt.

We allowed cross answer relations which made it very difficult to draw every relation between the different answers. Due to the presentation of the relations in WebAnno confusion gets created, because if a lot of relations exist from statements that are located at the end of a document to a statement that is located at the beginning, there is a line for every relation and the other statements become obscured. In Figure 3.6 there is an example with only one statement that relates to another one in the first sentences (the yellow marked line, note: the example does not represent a correct relation, it is for presentation purposes only).

Due to the confusion that this attempt caused, a different approach was tested.

3.3.2 Second Approach

To ensure a better clarity during the annotation process we tested another annotation variant. Every statement had its own identification number (ID) starting with one. We replaced the relation marks by assigning a target statement ID in the source statement. The annotator could chose between the same types of statement as before (claim, premise, and uncertainty). The relation receives the type in the span annotation instead of in the relation. So the only layer that we annotated is the statement layer (Figure 3.7).

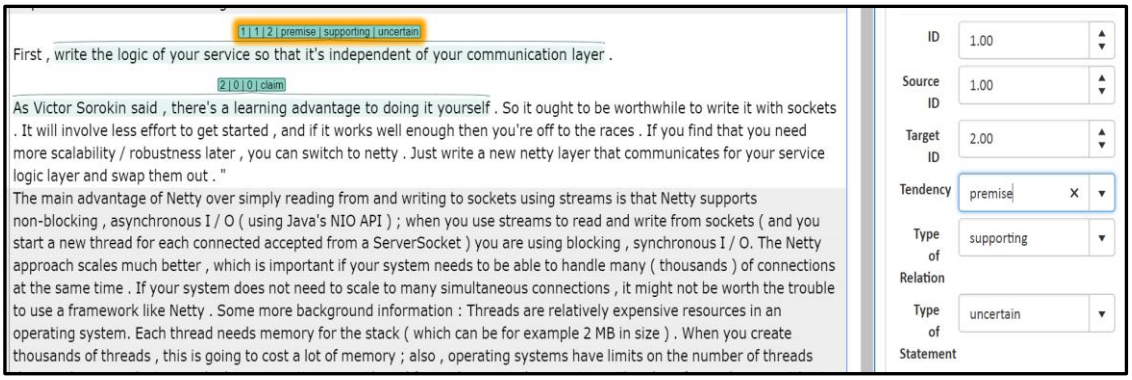


Figure 3.7: Example of the second annotation attempt.

This approach didn't work well. It took a long time to annotate the statements and it gets really confusing with all the ID's.

As a result we decided to keep the annotations within one sentence of WebAnno, respectively within one answer to a question on SO.

3.3.3 Third Approach

Like already mentioned in the previous chapter, we decided to annotate every answer to a question on SO for itself. This led to clear and fast annotation.

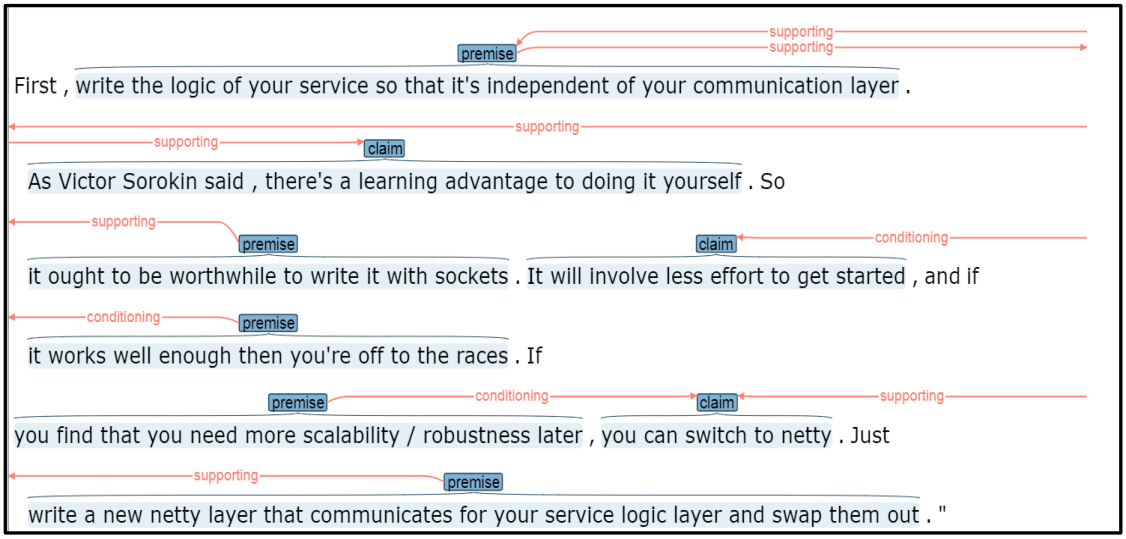


Figure 3.8: Example of the third annotation attempt.

We defined the direction of the relation as following: The annotator has to draw the relation from the statement that relates to another to the statement that is related (Figure 3.8). So it is possible to have multiple relations from one statement to another and they can be bidirectional. The example illustrates that transitive relations of statements which build on each other can exist.

3.3.4 Fourth Approach

We noticed that the intuitive way of defining the type occurs during the process of evaluating which object relates to which statement. This is why we decided to type the statements in the relation. So we mark every statement in the first step. Afterwards we draw the relations and in the relations we type the source

and the target statement (Figure 3.9). Finally every statement that is not part of a relation has to be typed.

We removed the option for uncertainty and decided that in case of doubt the statement has to be typed as a claim.

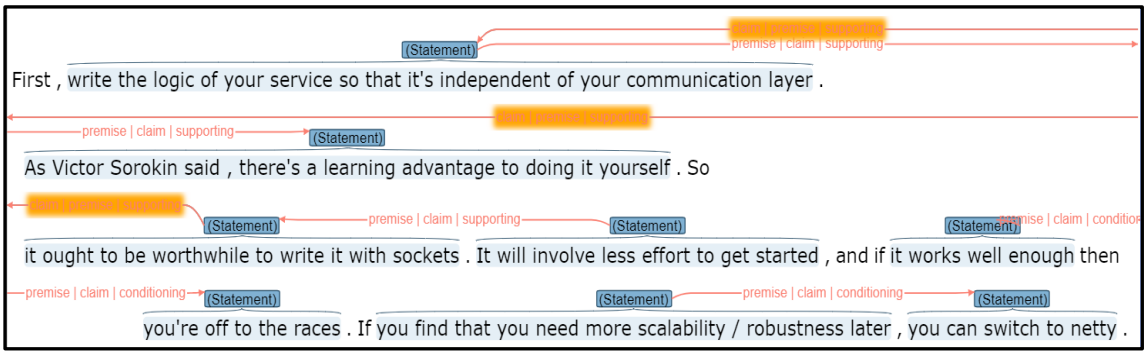


Figure 3.9: Example of the fourth annotation attempt.

3.4 The Annotation Guidelines

This chapter explicates the annotation guidelines. In order to reach the highest annotator agreement possible and to make clear what to annotate and when to annotate, we spent a lot of time to developing the guidelines for the claim premise schema. During that process we tried different approaches in order to get the best possible annotator agreement (see Chapter 3.3).

In the guidelines the motivation for their existence is described. Then the term statement is defined and the boundaries of the statements are stated. Furthermore, definitions are given for the following terms:

- claim,
- premise,
- attacking,
- supporting and
- conditioning.

Finally the order and the process steps that have to be executed during the annotation process are described.

3.4.1 Motivation behind the Guidelines

As already mentioned, internet forums became an adequate place for developers to debate several topics like programming and architecture knowledge. In this study we focus on posts of SO, which consist of a topic, a question and up to several answers to the question. The goal of the guidelines is to define how to identify argument structures in these posts. The annotation guideline is based on the guidelines by Kluge (Eckle-Kohler, Kluge, and Gurevych 2015).

Posts on SO contain several types of statements:

- Controversial statements that are called claims; these segments naturally raise the reader's doubt and need further support.
- Provable statements that are called premises or facts; it can be said whether they are true or false.
- Not every segment in a text is arguable, e.g. when the author presents background information. Such text passages are rather explanatory and not of interest for this study.

In order to maximize the annotator agreement the guidelines try to make absolutely clear which parts of the texts should be annotated.

3.4.2 The Guidelines – Boundaries of the Statements

First we provided a definition for a statement in order to make clear what a statement is and what it looks like.

A statement is a sentence that contains any kind of opinion. So a statement is the main class and claims and premises are sub classes of a statement.

In this study the length of a statement is the smallest possible grammatically correct text passage. Therefore every annotation candidate must contain one subject, one verb and at least one object. An annotation candidate is a statement that we consider for annotation. A sentence would be grammatically correct even without an object. But if there is no object, the statement doesn't argue anything. This is why we decided to include at least one object for the smallest possible annotation. We don't want to annotate expletives or conjunctions that are at the beginning or the start of a sentence, because they are not argumentative, either (see Figure 3.10).

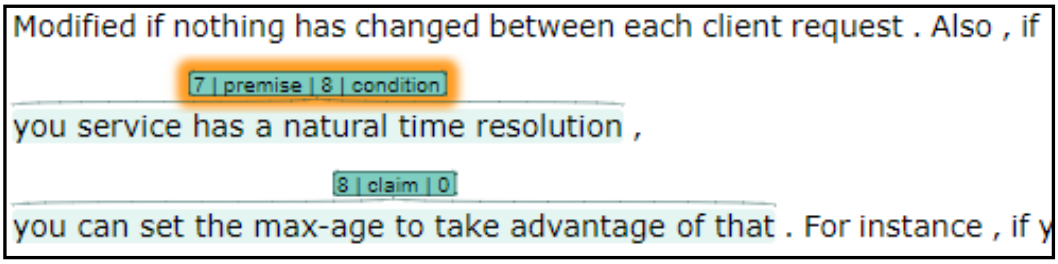


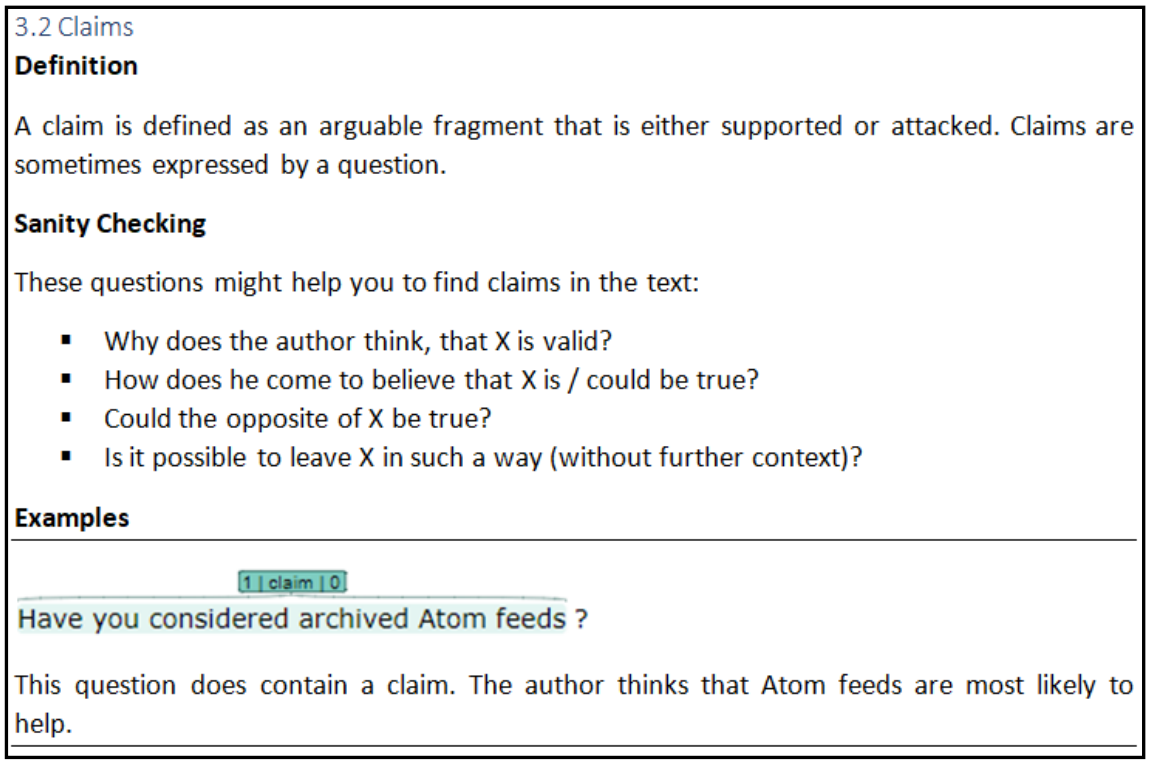
Figure 3.10: An example for an annotation candidate.

If there is a sentence that contains multiple statements, we decided to split it where it makes sense. It is important that the snippets of the sentence are still grammatically correct. The only exception is when a sentence has two statements that are linked with the word "and" and there is only a subject and a verb at the beginning of the sentence. Enumerations instead should be annotated as one statement.

We decided not to annotate punctuation marks and source-code since these elements are not argumentative.

3.4.3 The Guidelines – Definitions

Since most definitions are already given in the Related Work chapter, we only describe the structure of the definitions and add definitions that have not been mentioned yet. The definitions follow the structure presented in Figure 3.11.


3.2 Claims

Definition

A claim is defined as an arguable fragment that is either supported or attacked. Claims are sometimes expressed by a question.

Sanity Checking

These questions might help you to find claims in the text:

- Why does the author think, that X is valid?
- How does he come to believe that X is / could be true?
- Could the opposite of X be true?
- Is it possible to leave X in such a way (without further context)?

Examples

1 | claim | 0

Have you considered archived Atom feeds ?

This question does contain a claim. The author thinks that Atom feeds are most likely to help.

Figure 3.11: Example of a definition in the guidelines.

First the definition for the term is given. After that some sanity checking questions are presented in order to find the occurrences of the term in the posts of SO. Finally there are examples to make it easier for the annotators to understand the use of the definition.

Subsequently two definitions are added. Explanations describe background information for the question in the SO post. They are not argumentative. Therefore we decided not to annotate them. In posts on SO conditions are often formulated for certain statements. This commonly happens via if-sentences.

That is why we decided to add the conditioning to the claim-premise schema presented in Chapter 2.4.1, so we are able to annotate conditions.

3.4.4 The Guidelines – Annotation Process

In the last section of the guidelines, the process of annotating is defined. During the first reading the annotator should read the whole post in order to gather an overview of the post and an idea of the structure of the answers. In the second reading the annotator has to mark every statement that is detected. In the third step the annotator has to draw the relations between the statements in the direction of the object (source) that relates to the related object (target). Then the annotator has to define the type of the source, the target and the relation. Viable types for the source and the target are claim and premise. Viable types for the relation are supporting, attacking and conditioning. During the last reading the annotator has to check the own annotations to verify them.

The complete guidelines are attached to the thesis in Appendix A.1.

4 Results of the Annotation Process

On the one hand this chapter presents the inter annotator agreement. For this purpose, we examined the annotations on the basis of same statements and on the basis of similar statements. In the following we will compare the spans, the different types (claim, premise) and the relations (supporting, attacking and conditioning). To find similar statements we used fuzzy matching in order to increase the number of overlapping spans, which are the basis for the further analysis.

On the other hand the overlap of our annotations and the ontology classes that were found by Soliman, Galster, and Riebisch (2017) is presented. To this end we used the fuzzy matching as well to find the most similar statements. Since there are no relations in Solimans ontology, we first determined which statements belong to which ontology class and then we detected the relations between these classes based on our annotations. Thereby we can confirm that our attempt can expand the results that were found by Soliman, Galster, and Riebisch (2017).

4.1 Inter Annotator Agreement

In this chapter we present the annotator agreement, starting with the statement agreement (the percentage of equal annotated posts). Besides the statement agreement we show the agreement on statement- and relations types.

4.1.1 Statement Agreement

First of all, the statements are compared because they are the basic element of our annotation process. We followed two different approaches to examine the statements. In the first we compared only statements that were exactly the same

(called equal statements in the following), in the second we examined statements that reach a fuzzy matching score of 80 percent (called similar statements in the following). To evaluate the data we chose two different averaging formulas – macro averaging and micro averaging (see Figure 4.1).

$$S_A^d = set\ of\ statements\ annotated\ by\ A\ in\ document\ d$$
$$S_B^d = set\ of\ statements\ annotated\ by\ B\ in\ document\ d$$
$$D = set\ of\ documents$$

$$I_{A,B}^d = set\ of\ common\ statements\ of\ annotator\ A\ and\ B\ for\ document\ d$$

$$I_{A,B}^d = (S_A^d \cap S_B^d)$$

$$\mathbf{I}_{A,B}^{micro} = \frac{\sum_{d \in D} |I_{A,B}^d|}{\sum_{d \in D} |S_A^d|}$$

$$\mathbf{I}_{A,B}^{macro} = \frac{\sum_{d \in D} \frac{|I_{A,B}^d|}{|S_A^d|}}{|D|}$$

Figure 4.1: Micro and macro averaging.

In Table 4.1 the macro agreement for equal statements is presented. The left column represents the gold annotation. So we assume that the annotator has performed the "correct" annotation.

Table 4.1: Macro agreement for equal statements.

macro	A	B	C
A	1	0,33939	0,14141
B	0,48980	1	0,05248
C	0,39773	0,10227	1

Table 4.2 however represents the results for similar statements. The agreement has improved a little, which was expected. It would be possible to increase the agreement even more, but when the equality score gets too low in the fuzzy matching algorithm, we would match statements that aren't the same statements anymore.

Example: "Hello, my name is Leon" and "Good morning, my name is Sarah".

The texts in the example have a match score of 54 percent, so if our score for a match is 50 percent that is considered too low. That is why we decided to choose a score of 80 percent.

Table 4.2: Macro agreement for similar statements.

macro	A	B	C
A	1	0,41212	0,15960
B	0,59475	1	0,07289
C	0,44886	0,14205	1

Table 4.3 shows the results of the micro averaging formula for equal statements.

Since we have a multi class comparison the micro average is preferable. Because in contrast to the macro average, which computes the results for each class independently first and then takes the average, the micro average creates an average metric of all contributed classes.

Table 4.3: Micro agreement for equal statements.

micro	A	B	C
A	1	0,46282	0,70259
B	0,56932	1	0,50464
C	0,64939	0,30706	1

Table 4.4 presents the micro agreement for similar statements. As before, the results are a little bit better, since we match slightly more statements.

Table 4.4: Micro agreement for similar statements.

micro	A	B	C
A	1	0,58741	0,77650
B	0,72707	1	0,69505
C	0,71321	0,43647	1

In the next chapter the results of the type agreement are presented.

4.1.2 Statement and Relation Type Agreement

To evaluate the types of statements (claim or premise) and of the relations (supporting, attacking or conditioning) we used the Fleiss' Kappa which is

basically an advancement of the Cohen's Kappa. The Fleiss' Kappa can be used to measure the annotator agreement for non pair wise annotations (Fleiss 1971). So it is possible to compare the annotations of more than two people. There are three different parameters that are required to calculate the kappa: a number of annotators, the number of subjects (the number of annotated documents) and the number of categories that were examined.

For the statement type we have only two categories whereas for the relation type there are three categories. The number of annotators is three in both cases. The number of subjects is different, because there are more statement types than relation types. Our results for the Fleiss' Kappa are represented in Table 4.5, 4.6 and 4.7.

The overall Agreement is the original Fleiss' Kappa score. The free-marginal kappa was introduced by Warrens (2010) and is an extension to the Fleiss' Kappa. This kappa should be considered when the annotators "are not forced to assign a certain number of cases to each category"[4]. This is the case in our annotation study.

Table 4.5: Fleiss Kappa for statement type.

	Equal Data	Similar Data
Overall Agreement	0,60938	0,60390
Free-Marginal Kappa	0,41406	0,40584

Table 4.6: Fleiss Kappa for relations with same source and target statement.

[4] http://justusrandolph.net/kappa/

	Equal Data	Similar Data
Overall Agreement	0,90909	0,90470
Free-Marginal Kappa	0,86364	0,85714

Table 4.7: Fleiss Kappa for relations with same source type and target type statement.

	Equal Data	Similar Data
Overall Agreement	0,88462	0,90000
Free-Marginal Kappa	0,82692	0,85000

The score for similar data is inferior to the equal data, because a lot of the fuzzy matched statements were typed differently. The Fleiss Kappa and the free-marginal kappa for relations differ only slightly between similar and equal statements.

4.2 Overlap with the Ontology Classes of Soliman, Galster, and Riebisch

To measure the overlap between the ontology classes by Soliman, Galster, and Riebisch (2017) and our results, we used the fuzzy matching as well. For each statement that was annotated by Soliman, Galster, and Riebisch we searched through our annotations and took the best fuzzy matching statement as a

match. This way we achieved an overlapping percentage of 21 % (157 matches of 758 total statements) using the annotations of all three annotators, but with a quite high score (70 %) on the fuzzy matching algorithm. The full list of the overlap is too long to attach to this thesis (see Appendix A.2). It is available on the disk.

The data of Soliman, Galster, and Riebisch didn't consider relations. They only conducted a span annotation like we did with our statements. Since the claim premise schema allowed us to draw relations between different objects, we tried to expand the results by Soliman, Galster, and Riebisch. Therefore we provided a list in which the matching statements, the type of the relation between these statements and the ontology classes are represented (see Appendix A.3).

5 Conclusion

Over the course of this bachelor thesis we dealt with the following hypothesis:

> *The argument structures that have been found by the annotators overlap with the ontology classes, for example benefits and drawbacks, which were found by Soliman, Galster and Riebisch in the architecture-relevant posts on SO (2017).*

In order to test this hypothesis, we conducted an annotation study with the aim of developing an efficient method for analyzing arguments in software architecture relevant posts. First of all we had to choose an argumentation schema that we wanted to annotate with. The options included the claim premise scheme, the Toulmin Model of Argumentation and stance-based argument mining. We selected the claim premise schema because it is much easier to apply than the Toulmin Model of Argumentation. Furthermore the claim premise schema seems to be state of the art. Stance based argument mining is eliminated for the reason that the posts on SO contain relatively few implicit expressions.

In the next step the data was prepared and uploaded to the annotation tool WebAnno. Afterwards the guidelines were engineered in order to reach the highest annotator agreement possible with the minimum amount of time needed for the annotation process. Then the annotation process was performed with three annotators.

The results of the statement overlap have shown that the macro as well as the micro annotator agreement for statements was higher when we looked at similar statements (cf. Table 4.1 – Table 4.4). This is obvious, because the number of matching statements increases significantly when small deviations in the statements lead to a match. The results can't be generalized as the sample of

annotated documents was relatively small. The subjectivity, the different knowledge and language skills of the annotators are factors that might have contributed to further distorting the results. To improve the results, the guidelines can be extended by adding more examples to clarify the boundaries of the statements. In addition, the annotators could have received even more training.

To analyze the types of the statements and of the relations we used the Fleiss Kappa. For the statement types we have reached an overall agreement of 60 % (cf. Table 4.5). At this point, we only examined statements annotated by at least two annotators. A possibility to reach a better inter annotator agreement would be to increase the number of annotators to eliminate strong deviations and to improve the guidelines with more detailed information on what constitutes a claim and what constitutes a premise.

The same applies to the relations. The annotator agreement for the type of relations differs between 88-90 % (cf. Table 4.6 and 4.7). This score seems pretty high, but the total agreement on relations is 36 of 542 relations (~6 %). Even this number can be misleading, since the same or fuzzy matched statements, the same source and target types and the same relation type must be specified for a correct relation. So the number of possible sources of error is comparatively high. For example, assuming only equal statements and the same relation type, the match increases to 17 %. To improve this score the amount of sources for errors has to be reduced. Further improvements and training seems promising in this case too.

In order to corroborate the hypothesis, we compared our statements with those of Soliman, Galster, and Riebisch. As already mentioned in Chapter 4.2, the result is a 21 percent agreement with the data from Soliman, Galster, and Riebisch, but with a quite high score (70 %) on the fuzzy matching algorithm. That leads to the relatively small overlap, because the statements of Soliman,

Galster, and Riebisch are longer than ours so it is impossible for the fuzzy matching algorithm to find matches. In order to increase the score, another matching algorithm has to be tested, because our statements are often substrings of the spans annotated by Soliman, Galster and Riebisch. Since the statements of them are longer than our statements. To reach an even better agreement it is possible to concatenate multiple statements of us and compare the new statement with the data of Soliman, Galster and Riebisch.

Very interesting is that we found relations in our data that link the statements that are part of the ontology classification. This data is represented in Appendix A.3. So we were able not only to find an overlap but also to expand the ontology class of Soliman, Galster, and Riebisch with relations that link the different ontology classes. Therefore, we believe that we have found a good way to examine software architecture relevant posts on SO.

Our work has shown numerous approaches for further investigations. First of all the annotation guidelines could be improved to reach a better overlap than we did. The annotation process can be repeated with more annotators, so that the deviations become negligible. It would be possible to search for more argumentation schemas in order to find a better suiting one.

Apart from possibilities that improve the results of this work our results can be used to expand the ontology classification that was done by Soliman, Galster, and Riebisch with relations. We developed a corpus for machine learning that can be used in order to automatically search through SO and detect software architecture knowledge. This would help many software architects around the globe.

6 References

Besnard, Philippe, and Anthony Hunter. 2010. *Elements of Argumentation.* Cambridge: MIT Press.

Eckle-Kohler, Judith, Roland Kluge, and Iryna Gurevych. 2015. "On the Role of Discourse Markers for Discriminating Claims and Premises in Argumentative Discourse." In *Proceedings of the 2015 Conference on Empirical Methods in Natrual Language Processing*, pp. 2236–2242. Lisbon, Portugal.

Fleiss, Joseph L. 1971. "Measuring Nominal Scale Agreement Among Many Raters." *Psychological Bulletin* 76 (5): pp. 378-382.

Karbach, Joan. 1987. "Using Toulmin's Model of Argumentation." *Journal of Teaching Writing* 6 (1): pp. 81–91.

Kneupper, Charles W. 1978. "Teaching Argument: An Introduction to the Toulmin Model." *College Composition and Communication* 29 (3): pp. 237–241.

Lippi, Marco, and Paolo Torroni. 2016. "Argumentation Mining: State of the art and merging trends." *ACM Transactions on Internet Technolology* 16 (2): pp. 10:1–10:25.

Moens, Marie-Francine. 2013. "Argumentation Mining." In *Proceedings of the 5th 2013 Forum on Information Retrieval Evaluation - FIRE '13*, pp. 10:1-10:6. New Dehli, India.

Rocha, Gil, Henrique Lopes Cardoso, and Jorge Teixeria. 2016. "ArgMine: A Framework for Argumentation Mining." In *Computational Processing of the Portuguese Language*, n. p. Tomar, Portugal.

Sobhani, Parinaz, Inkpen Diana, and Stan Matwin. 2015. "From Argumentation Mining to Stance Classification." In *Proceedings of the 2nd Workshop on Argumentation Mining*, pp. 67–77. Denver, CO, USA.

Soliman, Mohamed, Matthias Galster, and Matthias Riebisch. 2017. "Developing an Ontology for Architecture Knowledge from Developer Communities." In *2017 IEEE International Conference on Software Architecture (ICSA)*, pp. 89–92. Gothenburg, Sweden.

Soliman, Mohamed, Matthias Galster, Amr R. Salama, and Matthias Riebisch. 2016. "Architectural Knowledge for Technology Decisions in Developer Communities: An Exploratory Study with StackOverflow: An Exploratory Study with StackOverflow." In *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 128-133. Venice, Italy.

Toulmin, Stephen Edelston. 2008. *The Uses of Argument.* Cambridge: Cambridge University Press.

van Eemeren, Frans H., Robert Grootendorst, and Arnolda Francisca Snoeck Henkemans. 1996. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments.* Mahwah: Lawrence Erlbaum Associates.

Warrens, Matthijs J. 2010. "Inequalities between multi-rater kappas." *Advances in Data Analysis and Classification* 4 (4): pp. 271–286.

Wojatzki, Michael, and Torsten Zesch. 2016. "Stance-based Argument Mining - Modeling Implicit Argumentation Using Stance." In *Proceedings of the Conference on Natural Language Processing (KONVENS 2016)*, pp. 313–322. Bochum, Germany.

Yimam, Seid Muhie, Iryna Gurevych, Richard Eckart de Casthilho, and Chris Biemann. 2013. "WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations." In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 1–6. Sofia, Bulgaria.

A Appendix

A.1 Annotation Guidelines

Motivation

Internet forums became an adequate place for developers to debate several topics. In this study, we focus on posts on StackOverlow[5]. These posts contain a question and several answers to or comments on it. The goal of this annotation guide is to define how to identify argument structures in these posts. This annotation guide is based on the guidelines by Kluge (Eckle-Kohler, Kluge, and Gurevych 2015).

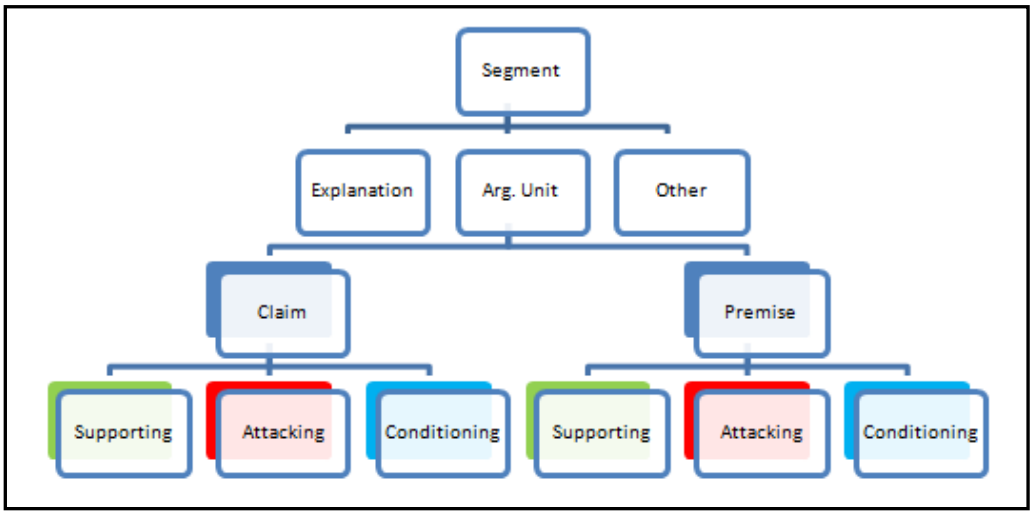


Figure A.1: Taxonomy of terms. In the annotation study, only the colored boxes (claim, premise, support, attack and condition) will be considered.

Posts contain several types of statements:

- Controversial statements are called claims; these segments naturally raise the reader's doubt and need further support.
- Sentences in the context of a claim may either support or attack the claim. Accordingly, these segments are called supports and attacks.

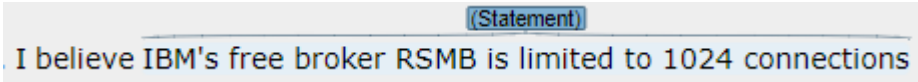
[5] https://stackoverflow.com/

- Not every segment in a text is arguable, e.g. when the author presents background information. Such text passages are rather explanatory and not of interest for this study.

The term argumentation unit generalizes the terms claim, premise and the relations supporting, attacking and conditioning. Premises are provable facts. A statement which we consider for annotation is named annotation candidate.

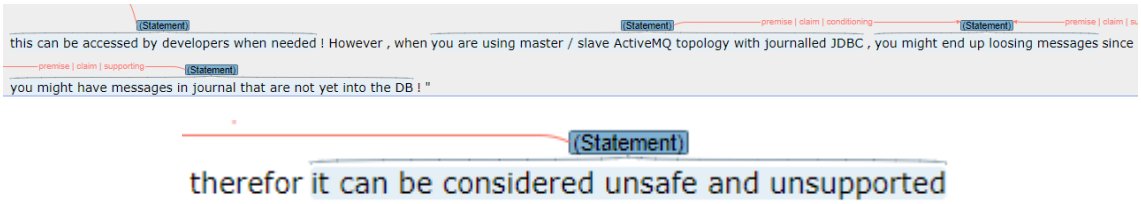
Boundaries of the Statements

We want to annotate the smallest possible grammatically correct passages. Therefore every annotation candidate must contain a subject, a verb and an object.



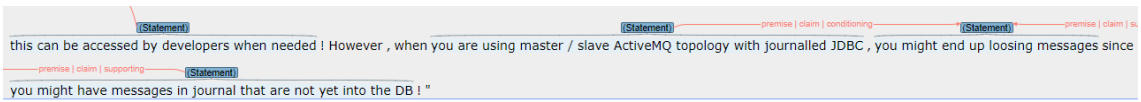
We left out "I believe", because it doesn't matter for the argumentation.

Do not annotate expletives or conjunction.



We left out "therefor".

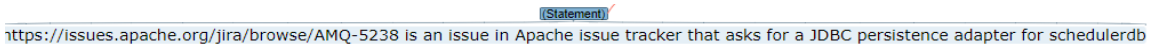
If a sentence contains more statements split them where it makes sense, but keep in mind that every snippet must be grammatically correct.



The sentence that starts with "however" was split into three grammatically correct statements.

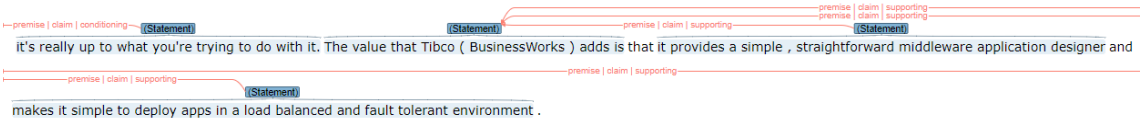
We do not annotate any source-code, because it is not argumentative. Do not annotate any punctuation marks. Try to annotate complete statements of the authors. Don't cut important parts of it.

We do not annotate nested statements. Annotations should make sense when you look at all related statements within an argument. A statement on its own doesn't have to make sense context wise if it is grammatically correct.



The three statements don't make sense by themselves, but when you read the whole sentence, they make sense.

We do not annotate links, unless they are the subject of the sentence. Sometimes there is additional information in a sentence that starts with "that". This should be annotated as one statement.



In this example the link or the information that is linked with the link is the subject of the sentence. Therefore it is annotated. After the "that" there is additional information relating to the main clause. That is why it is annotated as one statement.

If there is an "and" in a sentence that expresses two different aspects, we mark them as two statements. Only in this case, the statements don't have to be grammatically correct. When the "and" appears in an enumeration, then mark the whole enumeration as one statement.

Here is an example where the sentence with an "and" is split into two statements.

Relations should always be drawn to the next claim possible. If the same claim is made twice then the relation should be drawn to both statements.

Suggestions introducing a possible solution are claims.

Statements that are not typed in a relation have to be typed in the statement field. It has to be ensured that every statement has only one type. If you are uncertain whether it is a claim or a premise, then in case of doubt define it as a claim. Ensure that every relation is typed.

Annotation Types

Statements

Definition

A statement is a sentence that contains any kind of opinion.

Claims

Definition

A claim is defined as an arguable fragment that is either supported or attacked. Claims are sometimes expressed by a question.

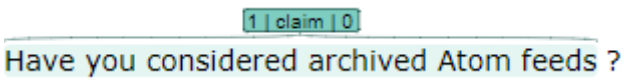
Sanity Checking

These questions might help you to find claims in the text:

- Why does the author think that X is valid?
- How does he come to believe that X is / could be true?

- Could the opposite of X be true?
- Is it possible to leave X in such a way (without further context)?

Examples



This question does contain a claim. The author thinks that Atom feeds are most likely to help.

Premises

Definition

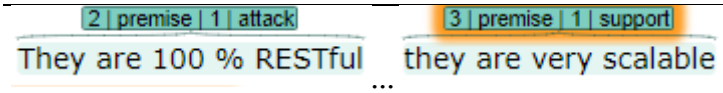
A premise is a provable fact. It is possible to make a clear decision whether it is true or false. Every premise needs a claim that it refers to. Statements about speed are premises, because it is possible to measure speed. Enumerations of tool characteristics are facts and therefore premises as well.

Sanity Checking

These questions might help you to find premises in the text:

- Could this statement be true?
- Is the effort realistic?

Examples



...

The author underlines his statement “Have you considered archived Atom feeds?” by enumerating technical facts.

Explanations

Definition

The posts on StackOverflow often contain explanations or descriptions of background information. We call these segments explanations, because they are not part of the argumentation, we do not annotate them.

Telling apart Explanations and Claims

Because it is difficult to distinguish statements and explanations, here are some questions that might help to decide whether it is a statement or an explanation:

- What did the author do so far?
- What is his background?

Examples

> Need some help figuring out what I am looking for . Basically , I need a service in which the Server dumps a bunch of XML into a stream (over a period of time) and every time the dump occurs N number of clients read the dump . Example : Every time one of a 1000 stocks goes up by 5 cents , the service dumps some XML into a stream . The connecting applications grab the information from the stream . I don't think the connection will ever close , as there needs to be something reading the stream for new data . This needs to adhere to WCF REST standards , is there something out there that I 'm looking for ? In the end , it's just a non-stop stream of data . Update : Looks like the service needs to be a multi-part / mixed content type .

This is a typical explanation where the author describes the background of his topic.

Support and Attack

Claims can either be supported or attacked by premises. A support brings evidence for a claim. An attack rebuts a claim. It is possible that some premises have a characteristic of a claim. A later section will describe how to handle these types of situations. A support or attack can occur before or after the claim it refers to. In our scenario we only mark relations within one answer. Exception: If the author cites another author, then it is allowed to set relations between these two answers. Otherwise we cannot be sure whether the relation to another post was intended by the author or if it is a random phenomenon.

Here are some typical patterns for support:

- *Claim*, because *premise*

- *Premise* leads to *claim*
- Because of *premise claim* is valid
- Out of *premise* follows *claim*
- *Premise*. Therefore *claim*
- *Premise* proves, that *claim* applies
- *Claim* is shown by the fact that *claim* applies

Here are some typical patterns for attack:

- *Claim*, although *premise*
- Although *premise*, it is true that *claim*
- Opposite *premise claim* is
- *Premise.* However claim is
- *Claim. Premise* speaks against it.

Sanity Checking

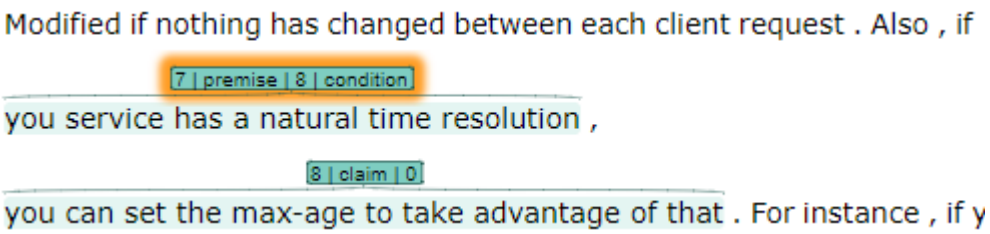
- "Does this support cause me to accept the claim more readily?
- Does this attack foster my doubts against the claim?
- Especially, does the author bring about the premise evidence in order to support/attack the claim?"[6]

Condition

In posts on StackOverflow conditions are often formulated for certain statements. This commonly happens via if-sentences.

The if-sentence is always a premise, the following part is always the claim.

Example



[6]https://www.ukp.tu-darmstadt.de/fileadmin/user_upload/Group_UKP/data/argument-recognition/annotation_guidelinesArgMinNews.pdf

Through the if-clause a condition for the following statement has been given. This is why it is a condition.

Annotation process

This section describes how to annotate a document with our specific WebAnno setup. While the first reading, you should get an idea what the document is about. In the second reading you have to draw relations and type the source and the target statement. In the last reading you should check if the current annotation is correct and follows these guidelines.

First reading: Gather an Overview

Just read the whole document and figure out what it is about. Do not assign any annotation candidates. Try to get an idea of the structure in the answers.

Second reading: Identify statements

During the second reading mark every statement that attracts your attention. We do not annotate the topic and the question of the post, so you have to start in line 3.

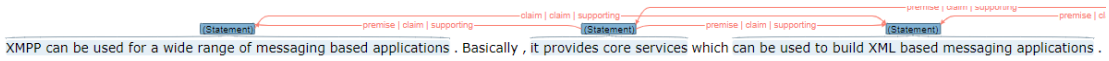
Third reading: Draw relations and type the statements

If you want to, you can type the statement immediately. But this step is optional.

Draw the relation from the direction of the statement that relates to another statement. Define the type of the source statement as premise or claim. Afterwards select premise or claim for the target of the relation. Statements that

are not relating on any other premise or claim remain statements, because a statement alone is no argumentative element.

In transitive relations you have to draw every relation that occurs.



Example for possible objects for relations:

Premise – premise

Premise – claim

Claim – premise

Claim – claim

Fifth reading: Checking

Try to verify the annotation that you made during the last reading.

References

Eckle-Kohler, Judith, Roland Kluge, and Iryna Gurevych. 2015. "On the Role of Discourse Markers for Discriminating Claims and Premises in Argumentative Discourse." In *Proceedings of the 2015 Conference on Empirical Methods in Natrual Language Processing*, pp. 2236–2242. Lisbon, Portugal.

A.2 Overlap of Statements and Ontology Classes

Type	Ontology type	Statement
claim	Development Feature Benefit	where XMPP really shines is in its extensibility
claim	Interoperability Feature Benefit	XMPP integration with Camel is trivial
premise	Technology Pattern Relation	it provides core services which can be used to build XML based messaging applications
premise	Use Technology Relationship	utilizes long-lived TCP connections for communicating
premise	Technology Pattern Relation	Its based on a decentralized client-server architecture
premise	Pattern Use Cases	An incredible amount of use-cases can be elegantly covered by PubSub , from queuing long-running jobs and having workers handle them , to micro-blogging
premise	Development Feature Benefit	Camel's framework allows you to build an application and easily swap out different messaging technologies (JMS , STOMP , mina , etc)
claim	Reliability Feature Benefit, Technology Pattern Relation	XMPP has a very robust and widely available extension to handle PubSub in a standard way
claim	Interoperability Technology Relation	A Camel component that allows integration with Smack API in Camel routes
premise	Development Feature Benefit	described in XEP-0060 and providing out of the box a workflow for handling publishing , subscriptions , notifications and security

A.3 Relations between different Ontology Classes

Source Ontology	Source Statement	Relation Type	Target Statement	Target Ontology
Development Feature Benefit	providing out of the box a workflow for handling publishing, subscriptions, notifications and security.	**supporting**	XMPP has a very robust and widely available extension to handle PubSub in a standard way	**Reliability Feature Benefit, Technology Pattern Relation**
Technology Bundle-Feature Relation	it can be used to wire together applications via XML messaging and XMPP APIs	**supporting**	Camel XMPP - A Camel component that allows integration with Smack API in Camel routes	**Interoperability Technology Relation**
Not Recommend Technology ADD	JAX-RPC or JAX-WS for Java-to-Java communication should be avoided	**attacking**	For a Java to Java application RMI is a good solution	**Technology ADD Context, Decision Rule**
External Constrain	clients are not under your control or might move to another platform	**conditioning**	JAX-RPC or JAX-WS for Java-to-Java communication should be avoided	**Not Recommend Technology ADD**
Architecture Component Behaviour	client is NATTED	**conditioning**	RMI has a much complex underlying network protocol which requires you to open up RMI ports, and also might not work if the client is NATTED.	**Interoperability Feature Drawback**
Use Technology Relationship	they are relying on HTTP only	*	Web Services are more likely to work	**Recommended Technology ADD**
Decision Rule	If you know that this issue is not a problem, you should consider using RMI	**conditioning**	you should consider using RMI	**Recommended Technology ADD**

Source Ontology	Source Statement	Relation Type	Target Statement	Target Ontology
Architecture Configuration	send complex object nets from one application to another	*	it's probably faster with RMI	Performance Feature Benefit
Recommended Technology ADD	I might as well point you to other commercial offerings:CloudAMQP	supporting	CloudAMQP is RabbitMQ as a Service	Use Technology Relationship
Commercial Feature Benefit	an easy to understand and especially published pricing model	supporting	It is complying to nowadays expectations on a Software as a service (SaaS) product,	Technology Pattern Relation
Interoperability Technology Relation	you can integrate with Java web service clients like Spring WS, CXF, and JAX-WS pretty easily.	supporting	give GroovyWS a try	Recommended Technology ADD
Interoperability Feature Benefit	The thing that makes Groovy so great is how easy it is to integrate with Java	supporting	give GroovyWS a try	Recommended Technology ADD
Interoperability Technology Relation	EJB session beans, which are usually invoked either from a web request, a JMS message or a RMI-IIOP remote call	supporting	The business logic is implemented in EJB session beans	Business Requirement Technology Relation
Recommended Technology ADD	you can expose RFC's as Service (WCF) and use it in any application	supporting	Has any one considered Biztalk Adapter service pack , it supports version 4.6c	Recommended Technology ADD
Interoperability Requirement	interoperability	conditioning	i would go for WCF these days	Recommended Technology ADD
Recommended Technology ADD	You could continue to use the SAP Connector for SAP 4.6C	attacking	biggest problem is going to be connecting to SAP 4.6C	Design Issue

Source Ontology	Source Statement	Relation Type	Target Statement	Target Ontology
Wellknown System Experience	I see a lot of big players do recently(for example twitter, digg, facebook, reddit).	**supporting**	I would consider moving away from the relation database in favor of for example Cassandra.	**Recommended Technology ADD**
Architecture Configuration Performance Drawback	This introduces a lot of latency into the application.	**supporting**	Basing the high level design around a set of modules is a good way to manage complexity and structure development	**Recommended Conceptual Solution ADD**
Development Feature Drawback	I think it is way to complex.	**supporting**	I don't like the part about SOAP	**Not Recommend Technology ADD**
Wellknown System Experience	HipHop for php which converts your php code to C code which was a huge boost for facebook.	**supporting**	I would also advice you to have a look at HipHop for php	**Recommended Technology ADD**

Note: The symbol * represents untyped relations. There should be none, but some types were not set.

Statutory Declaration

I declare that I have authored this thesis independently, that I have used no other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the referenced sources.

……. ……

place, date (signature)