Technische Universität Chemnitz

Web Engineering

Master Thesis

Development of IDE extensions for artificial, schema-driven generation and visualization of RDF A-Box Resources

Wesley Giovanni Obi

Matriculation Number: 1234567

01.01.2010

Supervised by
Prof. Dr. Michael Martin

Assistant Supervisor

Betreuer

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 01.01.2050

. . . .

(Signature) [your name]

Abstract

The Resource Description Framework (RDF) is a standard for describing resources on the web. RDF extends the Web's linking structure by using URIs to name the relationships between resources and the two ends of the link. It is designed for machine readability rather than human readability.

Developers often struggle with the time-consuming and complex task of manually generating example instances for new RDF schemas, which can lower their productivity and the seamless integration of these schemas into software projects.

This thesis aims to improve the user-friendliness visualization of RDF schema, by using a code editor extension that allows not just to better visualize the schema and the relations within but also provides automatically generated example instances for a better understanding.

The main challenge for this project is the User Interface Design. RDF data is structured as a graph of triples (subject-predicate-object), which can be difficult to represent visually. Unlike traditional tabular data, RDF's graph-based nature requires a UI that can effectively display nodes and their relationships. RDF schemas can vary widely in structure and content. Designing a UI that can dynamically adapt to different schemas and data types while remaining clear is a significant challenge. The UI should handle many types of vocabularies and ontologies.

Addressing this challenge will require regular software/web development knowledge but also a deep understanding of the Semantic web and its technologies, like RDFs and SPARQL.

Zusammenfassung

Da die meisten Leuten an der TU deutsch als Muttersprache haben, empfiehlt es sich, das Abstract zusätzlich auch in deutsch zu schreiben. Man kann es auch nur auf deutsch schreiben und anschließend einem Englisch-Muttersprachler zur Übersetzung geben.

Contents

talk to your supervisor if this is needed

List of Figures

List of Tables

1 Introduction

With the rising of the semantic web, the need for a standard for describing resources on the web has become more and more important. The Resource Description Framework (RDF) extends the Web's linking structure by using URIs to name the relationships between resources and the two ends of the link. Different kind of scientist and engineers have been working with RDF:

- **Data Scientists**:
  - RDF aligns with their work on structured data, semantic queries, and knowledge graphs.
  - They use RDF to model relationships, extract insights from interconnected datasets, and support machine learning on graph data.
- **Computer Scientists and Engineers**:
  - They focus on RDF's theoretical foundations, optimization of storage and querying, and application development for the Semantic Web.
  - They are often involved in creating RDF tools like RDF stores and query languages.
- **Knowledge Engineers**:
  - Specialize in designing ontologies and frameworks using RDF to represent knowledge in areas like AI, natural language processing, and expert systems.

Despite all its capabilities, two main challenges must be faced when we work with RDF: Instance Generation and Data Visualization.

The instance generation process for A-Boxes resources is a complex and time-consuming task that requires a deep understanding of the underlying RDF schema and the relationships between resources. Creating few test instances for a small dataset is not an issue for a human, my with really large datasets, time starts playing its role.

Concerning Data Visualization are many grate tools on the web and application that support data visualization for RDF graphs. Protégé is software that can be used to build both simple and complex ontology-based applications and allows to visualise many kind of schema structures [Pro23].

Several web-based tools and services are available for graph visualization, such as isSemantic [?] and RDF Grapher [?]. These services provide an efficient and intuitive approach for visualizing RDF graph data, enabling users to assess the structural correctness of their graphs and verify the validity of relationships between nodes.

Regardless of their extensive feature set, these services often struggle to find their place in an RDF developer's workflow due to their lack of integration with commonly used IDEs and development toolchains.

1.1 Motivation

This motivated me towards ...

1.2 Theses and Scientific Contribution

Objective - What kind of problem do you adress? Which issues do you try to solve? What solution do you propose? What is your goal? 'This thesis describes an approach to combining X and Y... The aim of this work is to...'

1.3 Positioning within the Scientific Context

Scope - Here you should describe what you will do and also what you will not do. Explain a little more specific than in the objective section. 'I will implement X on the platforms Y and Z based on technology A and B.'

Conclude this subsection with an image describing 'the big picture'. How does your solution fit into a larger environment? You may also add another image with the overall structure of your component.

'Figure 1.1 shows Component X as part of ...'

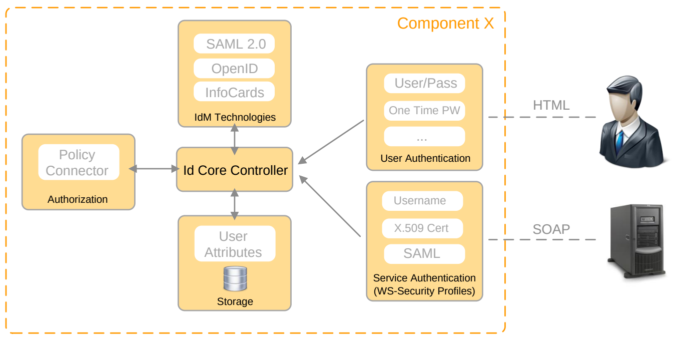


Figure 1.1: Component X

1.4 Structure of the Work (and Methodology)

The 'structure' or 'outline' section gives a brief introduction into the main chapters of your work. Write 2-5 lines about each chapter. Usually diploma thesis are separated into 6-8 main chapters.

This example thesis is separated into 7 chapters.

Chapter 2 is usually termed 'Related Work', 'State of the Art' or 'Fundamentals'. Here you will describe relevant technologies and standards related to your topic. What did other scientists propose regarding your topic? This chapter makes about 20-30 percent of the complete thesis.

Chapter 3 analyzes the requirements for your component. This chapter will have 5-10 pages.

Chapter 4 is usually termed 'Concept', 'Design' or 'Model'. Here you describe your approach, give a high-level description to the architectural structure and to the single components that your solution consists of. Use structured images and UML diagrams for explanation. This chapter will have a volume of 20-30 percent of your thesis.

Chapter 5 describes the implementation part of your work. Don't explain every code detail but emphasize important aspects of your implementation. This chapter will have a volume of 15-20 percent of your thesis.

Chapter 6 is usually termed 'Evaluation' or 'Validation'. How did you test it? In which environment? How does it scale? Measurements, tests, screenshots. This chapter will have a volume of 10-15 percent of your thesis.

Chapter 7 summarizes the thesis, describes the problems that occurred and gives an outlook about future work. Should have about 4-6 pages.

2 Fundamentals and Related Work

This section is intended to give an introduction about relevant terms, technologies and standards in the field of X. You do not have to explain common technologies such as HTML or XML.

2.1 Technologies

This section describes relevant technologies, starting with X followed by Y, concluding with Z.

2.1.1 Technology A

It's always a good idea to explain a technology or a system with a citation of a prominent source, such as a widely accepted technical book or a famous person or organization.

Exmple: Tim-Berners-Lee describes the "WorldWideWeb" as follows:
"The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents." [BL]

You can also cite different claims about the same term.
According to Bill Gates *"Windows 7 is the best operating system that has ever been released"* [Gat] (no real quote) In opposite Steve Jobs claims Leopard to be *"the one and only operating system"* [Job]

If the topic you are talking about can be grouped into different categories you can start with a classification. Example: According to Tim Berners-Lee XYZ can be classified into three different groups, depending on foobar [BL]:

- Mobile X
- Fixed X
- Combined X

2.1.2 Technology B

For internal references use the 'ref' tag of LaTeX. Technology B is similar to Technology A as described in section 2.1.1.

2.1.3 Comparison of Technologies

Name	Vendor	Release Year	Platform
A	Microsoft	2000	Windows
B	Yahoo!	2003	Windows, Mac OS
C	Apple	2005	Mac OS
D	Google	2005	Windows, Linux, Mac OS

Table 2.1: Comparison of technologies

2.2 Standardization

This sections outlines standardization approaches regarding X.

2.2.1 Internet Engineering Task Force

The IETF defines SIP as '...' [RSC+02]

2.2.2 International Telecommunication Union

Lorem Ipsum...

2.2.3 3GPP

Lorem Ipsum...

2.2.4 Open Mobile Alliance

Lorem Ipsum...

2.3 Concurrent Approaches

There are lots of people who tried to implement Component X. The most relevant are ...

3 Design and Specification

This section determines the requirements and design choices necessary for developing the RDF Instance Generator Schema Visualizer and the RDF web service.

3.1 Overview

This project aims to simplify RDF schema exploration, validation, and instance generation. To achieve this the following requirements are defined:

- Functional requirements, for core features like, instance generation, schema validation, and visualization.
- Non-functional requirements, to take into account performance, usability and compatibility.
- Technical requirements, specified for underlying technologies and system design.
- Data privacy and Security, useful to ensure the security and privacy of user data.

The design prioritizes a client-server architecture, and cross-platform compatibility to enhance usability and efficiency.
The Web Service is designed to be compatible with various Browsers and IDEs.
The IDE extension will support users and will seamlessly integrate in their workflow.

3.2 Functional requirements

3.2.1 Instance Generation

The very first requirement for this project is the generation of synthetic RDF instances.
When the user is writing an RDF file, in turtle, N3, xml or other RDF formats, it should be able to generate some synthetic RDF instances that could save him a lot of time avoiding the insertion of testing data.
The application should not just generate compatible triples instances, but also allow the user to modify the automatically generated instances.

3.2.2 Schema Validation

The Program should be designed to validate a user-defined RDF schema by checking the structural integrity and semantic consistency of its triples. Each triple conforms to syntactic constraints (e.g., proper use of IRIs, literals, and blank nodes) and the asserted relationships lead to any logical contradictions or unexpected inferences.
In case of errors in the schema description, feedback should be provided to the user, via UI or Terminal.

3.2.3 Multi-Vocabulary Support

Interoperability and semantic richness are crucial for ensuring seamless data integration, enhancing knowledge representation, and enabling efficient querying and reasoning across diverse RDF datasets. By supporting multiple vocabularies and ontologies such as RDFS, FOAF, Schema.org and others, the system should be able to handle a wide range of RDF data sources and their visualization as graphs.

3.2.4 Import and Export

The software should allow the users to import their custom RDF files, and export the file with the automatic generated instances in the same format.

3.2.5 Schema Visualization

To simplify the exploration and analyzes of an RDF from a human eye, the program should be able to display the described RDF schema with its graphical representation. The displayed graph should have some interactive features like zooming, panning and node collapse.
The graphical representation should adapt based on the schema complexity and hierarchical structure.
In the IDE extension, the user should be able to visualize the graph in a side panel window next to the RDF file their working on.

3.3 Non-functional requirements

3.3.1 Performance

The system should ensure high performance rendering of large RDF graphs to facilitate seamless visualization and interaction. Additionally, operations such as node selection and zooming should be executed with minimal latency to maintain a responsive user experience. The system should optimize rendering pipelines to support real-time exploration of complex RDF schemas without compromising efficiency.

3.3.2 Usability

Both the IDE extension and the web application should feature an intuitive and user-centric interface that enhances accessibility and minimizes the learning curve for users. To achieve this objective, the design should incorporate familiar UI elements or skeuomorphic design principles, ensuring a seamless and intuitive user experience.

3.3.3 Compatibility

To ensure seamless integration across various development environments, the web service should be designed with cross-platform compatibility, enabling effortless adoption in different IDEs with minimal modifications. This approach enhances interoperability, allowing the service to function efficiently within diverse software ecosystems while maintaining consistent performance and usability.
To validate this, minor features in a secondary IDE should be implemented.

3.4 Technical Requirements

3.4.1 Web Architecture

The system should be designed with a client-server architecture. There are no constraints on the specific technologies (e.g. SOAP, REST, gRPC), but it must follow the web principles and use Web technologies, such as HTML, CSS, JavaScript or WebAssembly, and protocols (e.g. HTTP, WebSockets).

3.4.2 IDE extension

The application should be accessible via a web-based client to ensure broad availability; however, primary emphasis should be placed on its integration as an extension within widely used IDEs.

3.5 Data privacy and Security

3.5.1 Security

To ensure the confidentiality and integrity of RDF data, the system should implement secure data transmission protocols, such as HTTPS, to protect sensitive information during transmission.

3.5.2 Confidentiality

Based on the current design, no information should be saved from the Business Logic Layer. All data processing occurs in-memory, either on the client-side or server-side, ensuring that no persistent storage of user data takes place within the system. This ephemeral data handling approach significantly reduces the risk of data retention and unauthorized access. As a result, compliance requirements related to long-term data storage, such as those outlined in the General Data Protection Regulation (GDPR), are minimized.

3.6 Solution strategy

As stated previously in the requirements section, the system is designed with a client-server architecture, where the client and server play distinct roles in the overall functionality.

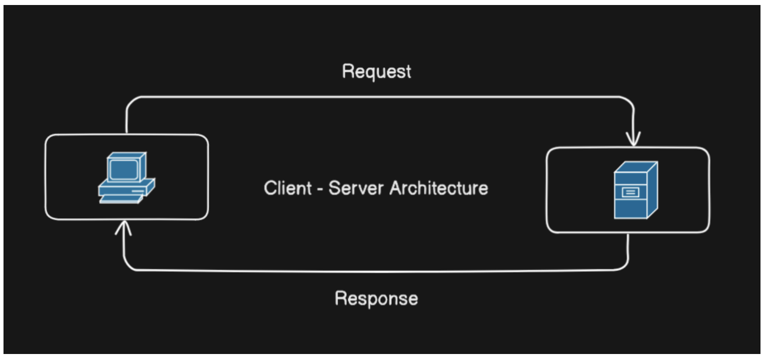


Figure 3.1: Client / Server

3.6.1 Client

To ensure a broad availability of the system, the system is designed to support two type of user-agents:

- Browsers
- Integrated Development Environments

The Browser chosen by the user, should be able to access the web application via HTTP requests protocol. The user will then upload their RDF file and send it to the server. The Client will then receive the processed information and display the resulting graph.

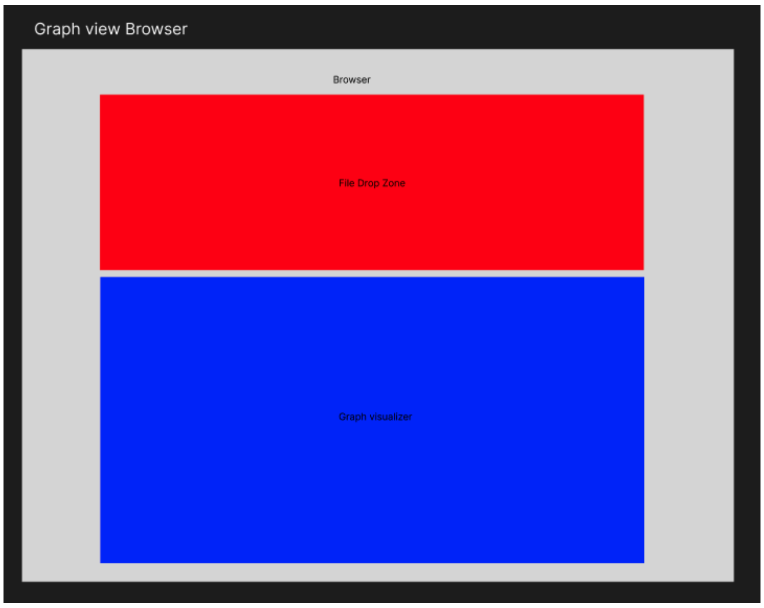


Figure 3.2: Browser file drop zone and graph view

In addition to this, the user should be able to read the RDF file with the generated instances next to its graphical representation.

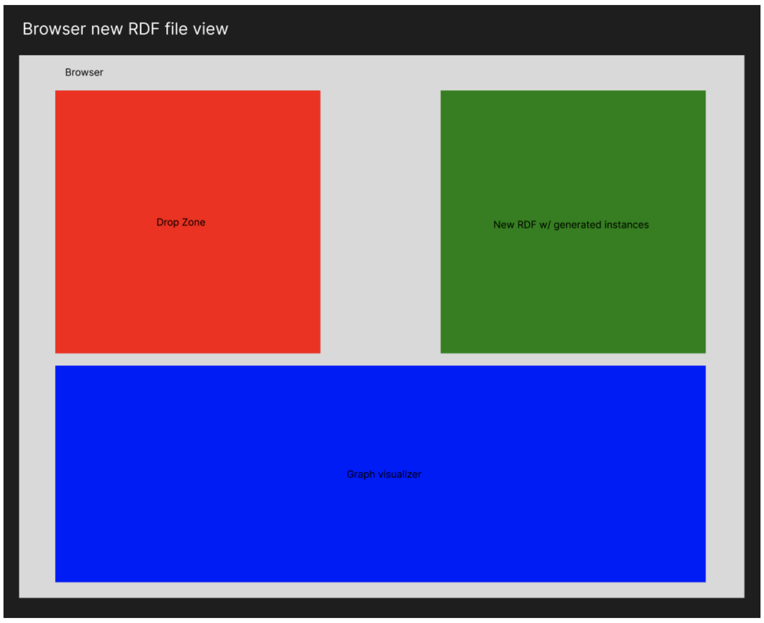


Figure 3.3: Browser new RDF side panel view

Within the IDE, the user experience UX will be slight different from the Browser one. Instead of manually uploading an RDF file, users will interact with the system directly within the IDE. By working on an RDF file and executing a predefined command, a web-based visualization panel of the file on focus will be dynamically launched as a side panel within the IDE. This panel will render an interactive graphical representation of the RDF schema in real time, allowing users to analyze relationships, structures, and dependencies without disrupting their coding environment.

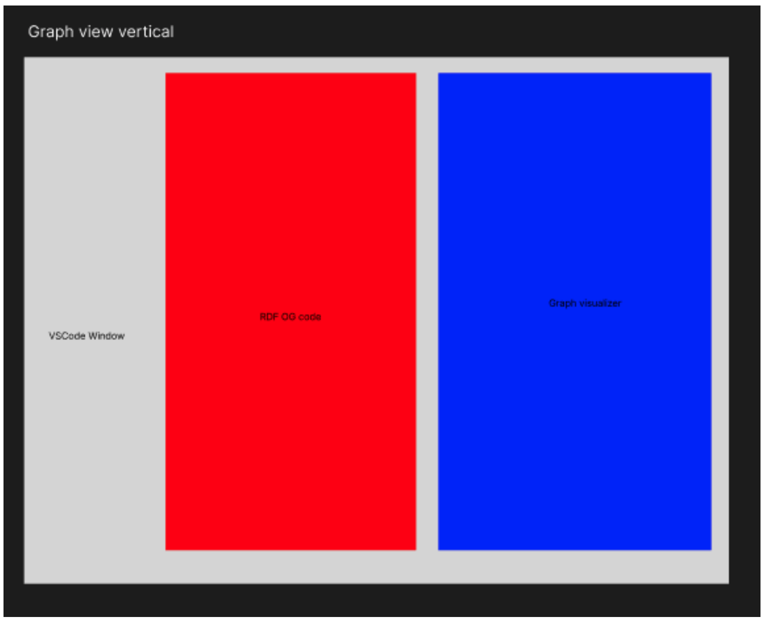


Figure 3.4: IDE Graph side panel view

The extension will not only support the visualization of RDF files in the format requested by the user, similar to its browser counterpart, but it will also enable real-time modification of the file.

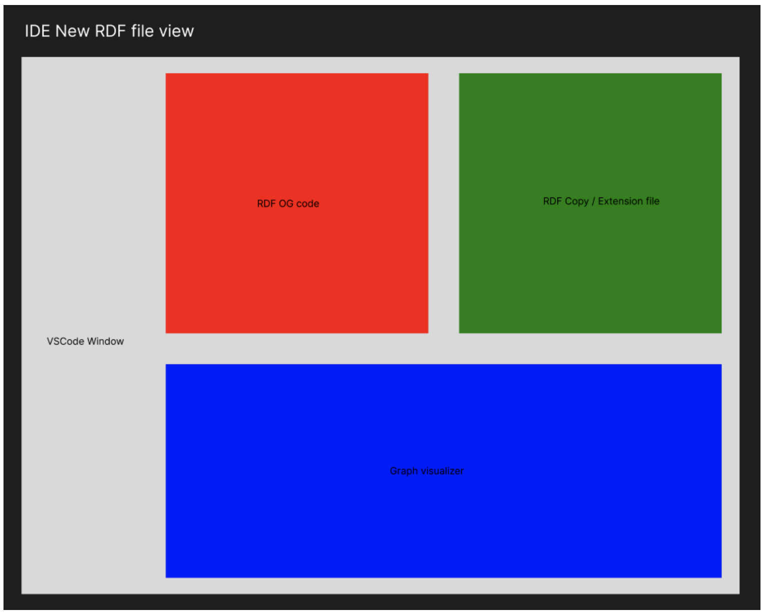


Figure 3.5: IDE new RDF side panel view

3.6.2 Server

To assure system intercompatibility across various device, Operative Systems, and User-Agents, the server is responsible for all computational tasks and data processing. By centralizing calculations and graph data manipulation on the server side, the server minimizes client-side resource consumption and enhances performance consistency.

The server place a crucial role in managing RDF data ensuring schema validation, instance generation, SPARQL query processing, and graph construction.
When the server receives an RDF file, it must scan the file to see potential floss, syntactic or semantic errors. It should be able to handle multiple RDF serialization formats such as Turtle, N3, XML, and JSON-LD.

The service must be able to generate synthetic RDF instances based on a given schema. It must ensure logical consistency between the generated schema and the original one. The system should be able to identify classes and properties within an RDF schema and generate additional instances where necessary to maintain semantic correctness. This is particularly important in cases where a property references an instance of a specific class that has not been explicitly defined in the dataset. For example, if a schema includes a hasCar property that requires an instance of the Car class, but no such instance exists, the system should automatically create one. This ensures that the schema remains logically consistent and that all property constraints are met. By dynamically generating required instances, the system helps maintain data integrity, prevents incomplete assertions, and improves the reliability of reasoning processes within RDF applications.

The additional instance generated should have properties and values in order to be consistent

4 Concept

This chapter introduces the architectural design of Component X. The component consists of subcomponent A, B and C.

In the end of this chapter you should write a specification for your solution, including interfaces, protocols and parameters.

4.1 Sub-component A

The concept chapter provides a high-level explanation of your solution. Try to explain the overall structure with a picture. You can also use UML sequence diagrams for explanation.

Figure 4.1 illustrates the situation between Alice and Bob. (sequence diagram from www.websequencediagrams.com)

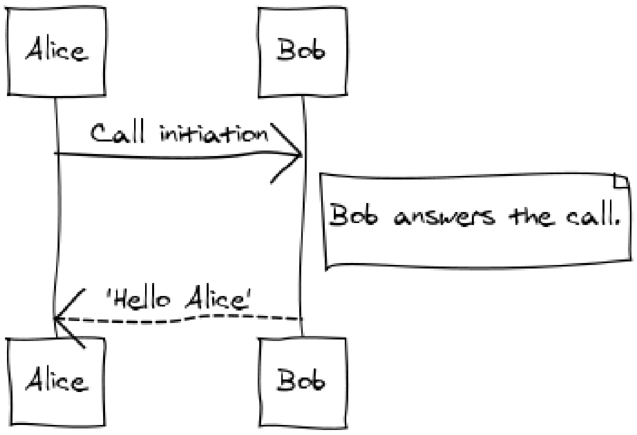


Figure 4.1: Alice and Bob

4.2 Sub-component B

Lorem Ipsum...

4.3 Proposed API

Lorem Ipsum...

4.4 Layer X

Lorem Ipsum...

4.5 Interworking of X and Y

Lorem Ipsum...

4.6 Interface Specification

Lorem Ipsum...

5 Implementation

This chapter describes the implementation of component X. Three systems were chosen as reference implementations: a desktop version for Windows and Linux PCs, a Windows Mobile version for Pocket PCs and a mobile version based on Android.

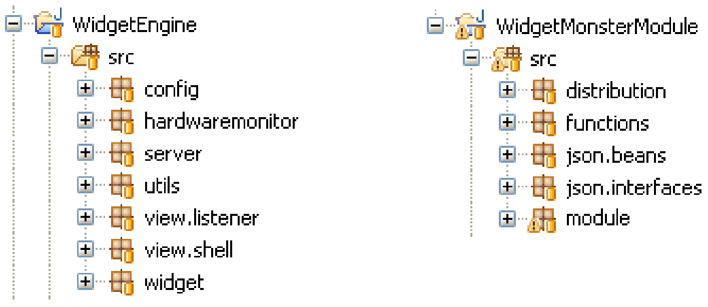
5.1 Environment

The following software, respectively operating systems, were used for the implementation:

- Windows XP and Ubuntu 6
- Java Development Kit (JDK) 6 Update 10
- Eclipse Ganymede 3.4
- Standard Widget Toolkit 3.4

5.2 Project Structure

The implementation is separated into 2 distinguished eclipse projects as depicted in figure 5.1.



```
WidgetEngine
  src
    config
    hardwaremonitor
    server
    utils
    view.listener
    view.shell
    widget

WidgetMonsterModule
  src
    distribution
    functions
    json.beans
    json.interfaces
    module
```

Figure 5.1: Project Structure

The following listing briefly describes the single packages of both projects in alphabetical order to give an overview of the implementation:

config
Lorem Ipsum...

server
Lorem Ipsum...

utils
Lorem Ipsum...

5.3 Important Implementation Aspects

Do not explain every class in detail. Give a short introduction about the modules or the eclipse projects. If you want to explain relevant code snippets use the 'lstlisting' tag of LaTeX. Put only short snippets into your thesis. Long listing should be part of the annex.

Listing 5.1: JSON String Code Snippet

```
{
        id: 1,
        method: "myInstance.getGroup",
        params: ["Teammates", 2, true]
}

{
        id: 2,
        result: [
                  "groupDesc":"These are my teammates",
                  {
                        "javaClass":"src.package.MemberClass",
                        "memberName": "Bob",
                  }
                ]
}
```

You can also compare different approaches. Example: Since the implementation based on X failed I choosed to implement the same aspect based on Y. The new approach resulted in a much faster ...

5.4 Graphical User Interface

Lorem Ipsum...

5.5 Documentation

Lorem Ipsum...

6 Evaluation

In this chapter the implementation of Component X is evaluated. An example instance was created for every service. The following chapter validates the component implemented in the previous chapter against the requirements.

Put some screenshots in this section! Map the requirements with your proposed solution. Compare it with related work. Why is your solution better than a concurrent approach from another organization?

6.1 Test Environment

Fraunhofer Institute FOKUS' Open IMS Playground was used as a test environment for the telecommunication services. The IMS Playground ...

6.2 Scalability

Lorem Ipsum

6.3 Usability

Lorem Ipsum

6.4 Performance Measurements

Lorem Ipsum

7 Conclusion

The final chapter summarizes the thesis. The first subsection outlines the main ideas behind Component X and recapitulates the work steps. Issues that remained unsolved are then described. Finally the potential of the proposed solution and future work is surveyed in an outlook.

7.1 Summary

Explain what you did during the last 6 month on 1 or 2 pages!

The work done can be summarized into the following work steps

- Analysis of available technologies
- Selection of 3 relevant services for implementation
- Design and implementation of X on Windows
- Design and implementation of X on mobile devices
- Documentation based on X
- Evaluation of the proposed solution

7.2 Dissemination

Who uses your component or who will use it? Industry projects, EU projects, open source...? Is it integrated into a larger environment? Did you publish any papers?

7.3 Problems Encountered

Summarize the main problems. How did you solve them? Why didn't you solve them?

7.4 Outlook

Future work will enhance Component X with new services and features that can be used ...

talk to your supervisor if this is needed

List of Acronyms

3GPP	3rd Generation Partnership Project
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
AS	Application Server
CSCF	Call Session Control Function
CSS	Cascading Stylesheets
DHTML	Dynamic HTML
DOM	Document Object Model
FOKUS	Fraunhofer Institut fuer offene Kommunikationssysteme
GUI	Graphical User Interface
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HTML	Hypertext Markup Language
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
I-CSCF	Interrogating-Call Session Control Function
IETF	Internet Engineering Task Force
IM	Instant Messaging
IMS	IP Multimedia Subsystem
IP	Internet Protocol
J2ME	Java Micro Edition
JDK	Java Developer Kit
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
JSR	Java Specification Request
JVM	Java Virtual Machine
NGN	Next Generation Network
OMA	Open Mobile Alliance
P-CSCF	Proxy-Call Session Control Function
PDA	Personal Digital Assistant
PEEM	Policy Evaluation, Enforcement and Management
QoS	Quality of Service
S-CSCF	Serving-Call Session Control Function
SDK	Software Developer Kit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SMS	Short Message Service
SMSC	Short Message Service Center
SOAP	Simple Object Access Protocol
SWF	Shockwave Flash

SWT	Standard Widget Toolkit
TCP	Transmission Control Protocol
Telco API	Telecommunication API
TLS	Transport Layer Security
UMTS	Universal Mobile Telecommunication System
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XCAP	XML Configuration Access Protocol
XDMS	XML Document Management Server
XML	Extensible Markup Language

Bibliography

[BL] Berners-Lee, Tim: *WWW Book (no real book, just an example).*

[Gat] Gates, Bill: *no real citate.*

[Job] Jobs, Steve: *no real citate.*

[Joh03] Johnston, Alan B.: *SIP, understanding the Session Initiation Protocol, Second Edition.* Artech House Publishers, 2003. ISBN: 1580536557.

[Pro23] Protégé: *Protégé: A tool for ontology development and knowledge acquisition*, 2023. `https://protege.stanford.edu/`.

[RSC^{+}02] Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler: *SIP: Session Initiation Protocol.* RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393.

Annex

```
<?xml version="1.0" encoding="UTF-8"?>
<widget>
         <debug>off</debug>
         <window name="myWindow" title="Hello Widget" visible="true">
                 <height>120</height>
                 <width>320</width>
                 <image src="Resources/orangebg.png">
                        <name>orangebg</name>
                        <hOffset>0</hOffset>
                        <vOffset>0</vOffset>
                </image>
                 <text>
                         <name>myText</name>
                         <data>Hello Widget</data>
                         <color>#000000</color>
                         <size>20</size>
                         <vOffset>50</vOffset>
                         <hOffset>120</hOffset>
                 </text>
        </window>
</widget>
```

Listing 1: Sourcecode Listing

```
INVITE sip:bob@network.org SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
Max-Forwards: 70
To: Bob <sip:bob@network.org>
From: Alice <sip:alice@ims-network.org>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Subject: How are you?
Contact: <sip:xyz@network.org>
Content-Type: application/sdp
Content-Length: 159
v=0
o=alice 2890844526 2890844526 IN IP4 100.101.102.103
s=Phone Call
t=0 0
c=IN IP4 100.101.102.103
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000

SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.network.org:5060;branch=z9hG4bK83842.1
;received=100.101.102.105
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
To: Bob <sip:bob@network.org>;tag=314159
From: Alice <sip:alice@network.org>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:foo@network.org>
Content-Type: application/sdp
Content-Length: 159
v=0
o=bob 2890844526 2890844526 IN IP4 200.201.202.203
s=Phone Call
c=IN IP4 200.201.202.203
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Listing 2: SIP request and response packet[Joh03]