INSTITUTE FOR SOFTWARE

OST Ostschweizer Fachhochschule

Visual OO Debugger

Bachelor Thesis

Department of Computer Science
OST – University of Applied Sciences
Campus Rapperswil-Jona

Spring Term 2022

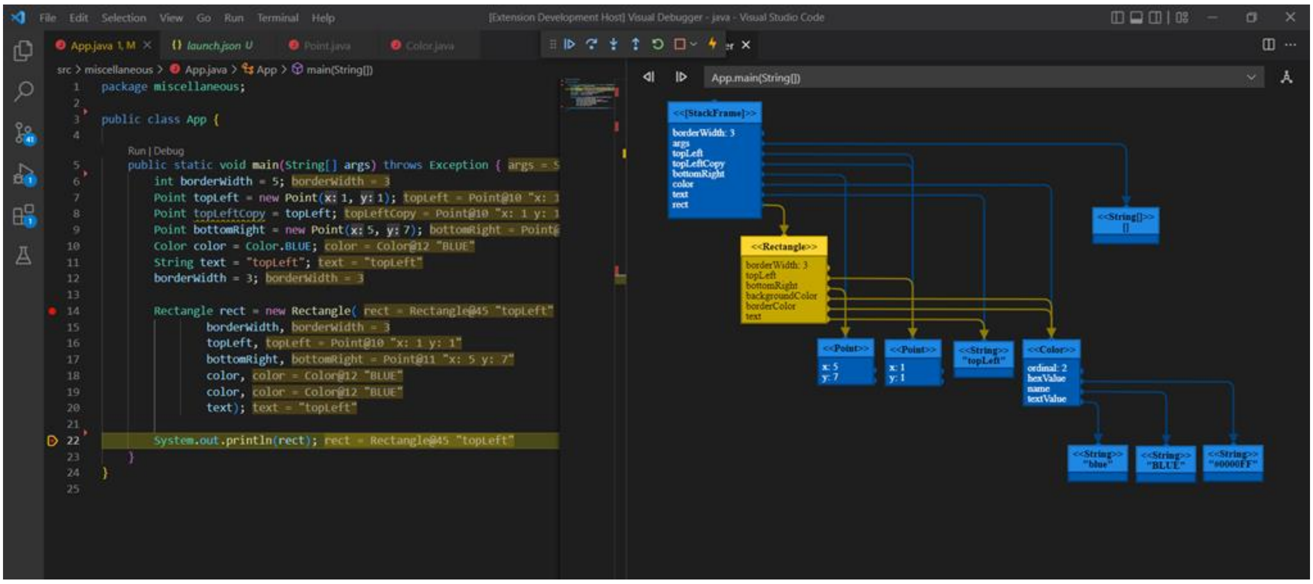
Author(s):	Gino Cardillo, Alexandre Lagadec, Pascal Schürmann
Advisor:	Prof. Mirko Stocker
Project Partner:	Institute for Software
External Co-Examiner:	Leo Büttiker
Internal Co-Examiner:	Prof. Frank Koch

Abstract

Object-oriented programming can be a challenge for unexperienced or new developers. The relations between objects, variables, and the concept of call-by-reference in methods is difficult to comprehend for a lot of people, sometimes even for more experienced developers. Teaching object-oriented programming can be just as challenging as learning it. One of the best ways to teach this topic is to visualize the relations between objects and variables. In the autumn term of 2021, as our term project, we created the VS Code extension Visual OO Debugger, VOOD for short, whose goal it was to ease the process of learning and teaching the concepts of object-oriented programming. It achieves this by using debugger information at runtime to visualize objects and variables in a graph.

The goal of this project is to extend VOOD with more useful features, as well as adapt the code to facilitate further extension. The current library used for visualization, vis.js, is a great starting point, but it has its limitations. Thus, the main feature of this project is to add the option to change the visualization style. Currently, only Java is supported by VOOD. While the support of other languages is out of scope, it should be possible to add support for them. Since the Java-specific parts are intertwined with the rest of the debugger, the code must be refactored to separate them.

The result was a new version of VOOD with many new features and improvements. As for the main feature, a new visualization was added, utilizing the library JointJS. JointJS was already evaluated in the term project and was deemed fit as an alternate visualization. It offered more flexibility for customization but at the cost of more complexity. Another new feature was the option to choose a stack frame of the call stack in a dropdown and visualize it. Hitherto, the topmost stack frame was always used for the visualization and other stack frames were discarded. With the growth of the graph, it becomes more and more confusing and polluted with information, which the user might not want. Two features were added to counteract this problem. By clicking on a node, it collapses with its referenced nodes, and they form a cluster. Those clusters can be opened either by clicking on them separately or by clicking the left button in the upper right-hand corner to open all clusters at once. The second feature involves the right button in the upper right-hand corner. By dragging a node or cluster of nodes over this button, they can be hidden completely from the visualization. And by clicking that button, all hidden nodes and clusters are displayed again. These are just a fraction of the features and improvements implemented in this project.



```java
package miscellaneous;

public class App {

    Run | Debug
    public static void main(String[] args) throws Exception { args = S
        int borderWidth = 5; borderWidth = 3
        Point topLeft = new Point(x: 1, y: 1); topLeft = Point@10 "x: 1
        Point topLeftCopy = topLeft; topLeftCopy = Point@10 "x: 1 y: 1
        Point bottomRight = new Point(x: 5, y: 7); bottomRight = Point@
        Color color = Color.BLUE; color = Color@12 "BLUE"
        String text = "topLeft"; text = "topLeft"
        borderWidth = 3; borderWidth = 3

        Rectangle rect = new Rectangle( rect = Rectangle@45 "topLeft"
                borderWidth, borderWidth = 3
                topLeft, topLeft = Point@10 "x: 1 y: 1"
                bottomRight, bottomRight = Point@11 "x: 5 y: 7"
                color, color = Color@12 "BLUE"
                color, color = Color@12 "BLUE"
                text); text = "topLeft"

        System.out.println(rect); rect = Rectangle@45 "topLeft"
    }
}
```

Figure 1 JointJS visualization

Table of Contents

Introduction 4
Requirements and Constraints 5
1 Requirements 5
1.1 Requirements Overview 5
1.2 Stakeholders 10
1.3 Existing comparable products 14
2 Constraints 16
3 Quality Requirements 17
3.1 Quality Goals 17
3.2 Quality Tree 19
3.3 Quality Scenarios 20
4 Solution Strategy 25
5 Risks and Technical Debts 27
5.1 Risk Assessment 28
5.2 Risk Matrix 29
Initial Solution 30
6 Cross-cutting Concepts 30
6.1 User Experience concepts (UX) 30
6.2 Development Concepts 31
7 Deployment View 32
8 System Scope and Context 33
8.1 Business Context 33
8.2 Technical Context 34
9 Building Block View 36
9.1 Whitebox Overall System 36
9.2 Level 2 38
9.3 Level 3 40
Development 43
10 Implemented Requirements 44
10.1 REQ-BA:5 Visualization of the individual stack frames with parameters/local variables and references to the heap 44
10.2 REQ-BA:6 Alternative visualization with JointJS 47
10.3 REQ-BA:7 Show objects that are no longer referenced. 54
10.4 REQ-BA:8 Connection with the editor: Click on the variable in the visualization, the variable is highlighted in the source code. 55

10.5 REQ-SA:3.5 Usability [of the system] 56
10.6 REQ-SA:4.3 Source code that is open to extensions. 60
10.7 R:6 API Deprecations by VS Code 62
10.8 R:7 Incompatible 3rd party extensions 62
11 Conclusion 63
11.1 Overview of Changes for the BA 63
11.2 Usability Test 65
11.3 Target Achievement 66
11.4 Outlook 67
Indices 68
12 Glossary 68
13 List of Figures 69
14 List of Tables 70
15 Bibliography 71

Introduction

This document describes the development process and the results of the project "Visual OO Debugger", short "VOOD". The document is divided into three main parts.

The part "Requirements and Constraints" discusses the requirements and constraints of the project. The part "Initial Solution" provides an overview of the implemented solution of the SA. The part "Development" consists of a detailed description of features that were implemented during the BA.

The first two parts were created based on the SA report of the previous semester. The text provides enough information to distinguish which passages are adopted from the SA and which are newly created for the BA.

This document is based on the arc42[1]-template and was modified to better differentiate between the work done in the SA and the BA.

[1] (arc42, 2022)

Requirements and Constraints

This part discusses the requirements and constraints of the project.

1 Requirements

In this chapter, an overview of the requirements of this project is given. Furthermore, the main quality goals and stakeholders are listed.

1.1 Requirements Overview

Our requirements-engineering methodology is loosely based on the principles of the Sophist group[2]. Partial requirements are derived by syntactically and semantically breaking down higher-level requirement descriptions. Parent items are italicized in tables and have dotted borders in mind map diagrams. The index indicates the relative position and nesting level in the hierarchy.

We have adopted the first set of requirements from the SA. These requirements can be found in the SA Requirements sub-section. The second set of requirements is specific to the BA. These requirements can be found in the BA Requirements sub-section.

1.1.1 SA Requirements

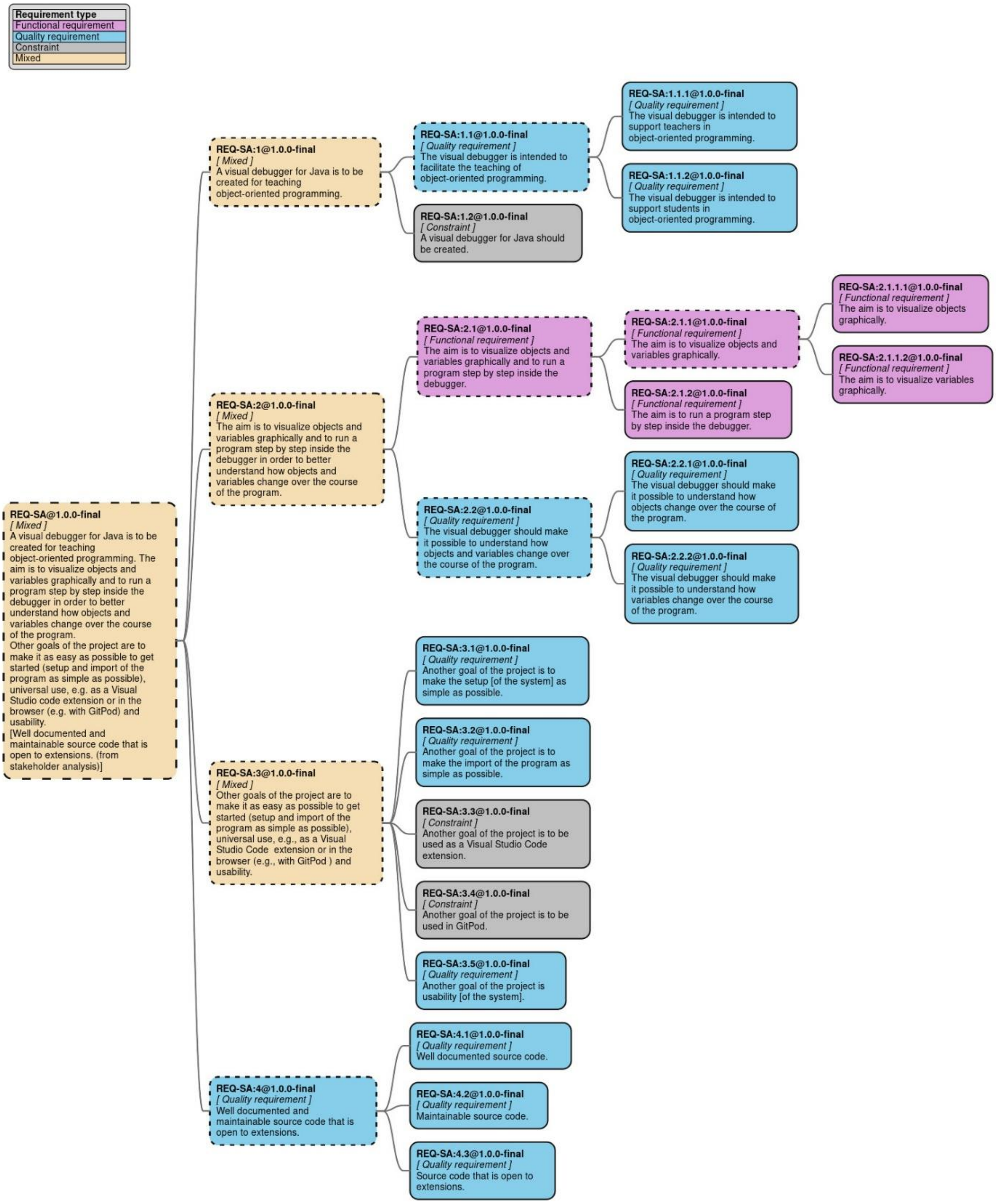
From the initial SA assignment, we inherited the following set of requirements (see Figure 2 and Table 1). All requirements were derived from the assignment with the exception of the requirement *REQ-SA:4@1.0.0-final* and its sub-requirements, which were derived from the stakeholder analysis (see section 1.2.2). For the BA the requirements were categorized in functional requirements, constraints, and quality requirements.

ID	Description
REQ-SA@1.0.0-final	A visual debugger for Java is to be created for teaching object-oriented programming. The aim is to visualize objects and variables graphically and to run a program step by step inside the debugger in order to better understand how objects and variables change over the course of the program. Other goals of the project are to make it as easy as possible to get started (setup and import of the program as simple as possible), universal use, e.g., as a Visual Studio Code extension or in the browser (e.g., with GitPod) and usability. [Well documented and maintainable source code that is open to extensions. (from stakeholder analysis)]
REQ-SA:1@1.0.0-final	A visual debugger for Java is to be created for teaching object-oriented programming.
REQ-SA:1.1@1.0.0-final	The visual debugger is intended to facilitate the teaching of object-oriented programming.
REQ-SA:1.1.1@1.0.0-final	The visual debugger is intended to support teachers in object-oriented programming.

[2] (Sophist GmbH, 2022)

REQ-SA:1.1.2@1.0.0-final	The visual debugger is intended to support students in object-oriented programming.
REQ-SA:1.2@1.0.0-final	A visual debugger for Java should be created.
REQ-SA:2@1.0.0-final	The aim is to visualize objects and variables graphically and to run a program step by step inside the debugger in order to better understand how objects and variables change over the course of the program.
REQ-SA:2.1@1.0.0-final	The aim is to visualize objects and variables graphically and to run a program step by step inside the debugger.
REQ-SA:2.1.1@1.0.0-final	The aim is to visualize objects and variables graphically.
REQ-SA:2.1.1.1@1.0.0-final	The aim is to visualize objects graphically.
REQ-SA:2.1.1.2@1.0.0-final	The aim is to visualize variables graphically.
REQ-SA:2.1.2@1.0.0-final	The aim is to run a program step by step inside the debugger.
REQ-SA:2.2@1.0.0-final	The visual debugger should make it possible to understand how objects and variables change over the course of the program.
REQ-SA:2.2.1@1.0.0-final	The visual debugger should make it possible to understand how objects change over the course of the program.
REQ-SA:2.2.2@1.0.0-final	The visual debugger should make it possible to understand how variables change over the course of the program.
REQ-SA:3@1.0.0-final	Other goals of the project are to make it as easy as possible to get started (setup and import of the program as simple as possible), universal use, e.g., as a Visual Studio Code extension or in the browser (e.g., with GitPod) and usability.
REQ-SA:3.1@1.0.0-final	Another goal of the project is to make the setup [of the system] as simple as possible.
REQ-SA:3.2@1.0.0-final	Another goal of the project is to make the import of the program as simple as possible.
REQ-SA:3.3@1.0.0-final	Another goal of the project is to be used as a Visual Studio Code extension.
REQ-SA:3.4@1.0.0-final	Another goal of the project is to be used in GitPod.
REQ-SA:3.5@1.0.0-final	Another goal of the project is usability [of the system].
REQ-SA:4@1.0.0-final	Well documented and maintainable source code that is open to extensions.
REQ-SA:4.1@1.0.0-final	Well documented source code.
REQ-SA:4.2@1.0.0-final	Maintainable source code.
REQ-SA:4.3@1.0.0-final	Source code that is open to extensions.

Table 1 SA requirements



Requirement type
Functional requirement
Quality requirement
Constraint
Mixed

REQ-SA@1.0.0-final
[Mixed]
A visual debugger for Java is to be created for teaching object-oriented programming. The aim is to visualize objects and variables graphically and to run a program step by step inside the debugger in order to better understand how objects and variables change over the course of the program.
Other goals of the project are to make it as easy as possible to get started (setup and import of the program as simple as possible), universal use, e.g. as a Visual Studio code extension or in the browser (e.g. with GitPod) and usability.
[Well documented and maintainable source code that is open to extensions. (from stakeholder analysis)]

REQ-SA:1@1.0.0-final
[Mixed]
A visual debugger for Java is to be created for teaching object-oriented programming.

REQ-SA:1.1@1.0.0-final
[Quality requirement]
The visual debugger is intended to facilitate the teaching of object-oriented programming.

REQ-SA:1.1.1@1.0.0-final
[Quality requirement]
The visual debugger is intended to support teachers in object-oriented programming.

REQ-SA:1.1.2@1.0.0-final
[Quality requirement]
The visual debugger is intended to support students in object-oriented programming.

REQ-SA:1.2@1.0.0-final
[Constraint]
A visual debugger for Java should be created.

REQ-SA:2@1.0.0-final
[Mixed]
The aim is to visualize objects and variables graphically and to run a program step by step inside the debugger in order to better understand how objects and variables change over the course of the program.

REQ-SA:2.1@1.0.0-final
[Functional requirement]
The aim is to visualize objects and variables graphically and to run a program step by step inside the debugger.

REQ-SA:2.1.1@1.0.0-final
[Functional requirement]
The aim is to visualize objects and variables graphically.

REQ-SA:2.1.1.1@1.0.0-final
[Functional requirement]
The aim is to visualize objects graphically.

REQ-SA:2.1.1.2@1.0.0-final
[Functional requirement]
The aim is to visualize variables graphically.

REQ-SA:2.1.2@1.0.0-final
[Functional requirement]
The aim is to run a program step by step inside the debugger.

REQ-SA:2.2@1.0.0-final
[Quality requirement]
The visual debugger should make it possible to understand how objects and variables change over the course of the program.

REQ-SA:2.2.1@1.0.0-final
[Quality requirement]
The visual debugger should make it possible to understand how objects change over the course of the program.

REQ-SA:2.2.2@1.0.0-final
[Quality requirement]
The visual debugger should make it possible to understand how variables change over the course of the program.

REQ-SA:3@1.0.0-final
[Mixed]
Other goals of the project are to make it as easy as possible to get started (setup and import of the program as simple as possible), universal use, e.g., as a Visual Studio Code extension or in the browser (e.g., with GitPod) and usability.

REQ-SA:3.1@1.0.0-final
[Quality requirement]
Another goal of the project is to make the setup [of the system] as simple as possible.

REQ-SA:3.2@1.0.0-final
[Quality requirement]
Another goal of the project is to make the import of the program as simple as possible.

REQ-SA:3.3@1.0.0-final
[Constraint]
Another goal of the project is to be used as a Visual Studio Code extension.

REQ-SA:3.4@1.0.0-final
[Constraint]
Another goal of the project is to be used in GitPod.

REQ-SA:3.5@1.0.0-final
[Quality requirement]
Another goal of the project is usability [of the system].

REQ-SA:4@1.0.0-final
[Quality requirement]
Well documented and maintainable source code that is open to extensions.

REQ-SA:4.1@1.0.0-final
[Quality requirement]
Well documented source code.

REQ-SA:4.2@1.0.0-final
[Quality requirement]
Maintainable source code.

REQ-SA:4.3@1.0.0-final
[Quality requirement]
Source code that is open to extensions.

Figure 2 Mind map of the SA requirements

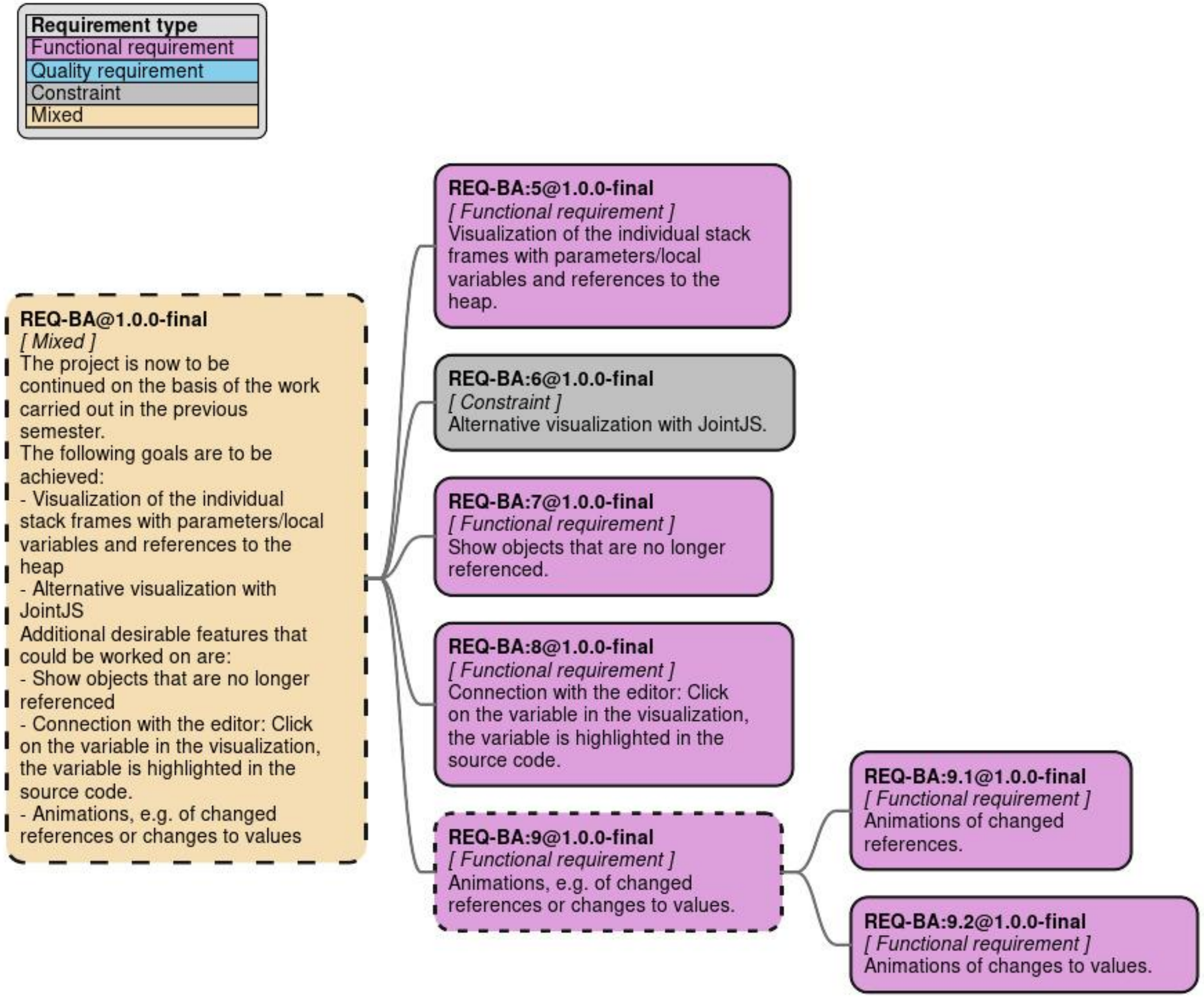
1.1.2 BA Requirements

For the current BA assignment, we received the following additional set of requirements (see Figure 3 and Table 2):

ID	Description
REQ-BA@1.0.0-final	The project is now to be continued based on the work of the previous semester. The following goals are to be achieved: - Visualization of the individual stack frames with parameters/local variables and references to the heap - Alternative visualization with JointJS[3] Additional desirable features that could be worked on are: Show objects that are no longer referenced - Connection with the editor: On click on the variable in the visualization, the variable is highlighted in the source code. - Animations, e.g., of changed references or changes to values
REQ-BA:5@1.0.0-final	Visualization of the individual stack frames with parameters/local variables and references to the heap.
REQ-BA:6@1.0.0-final	Alternative visualization with JointJS[3].
REQ-BA:7@1.0.0-final	Show objects that are no longer referenced.
REQ-BA:8@1.0.0-final	Connection with the editor: Click on the variable in the visualization, the variable is highlighted in the source code.
REQ-BA:9@1.0.0-final	Animations, e.g., of changed references or changes to values.
REQ-BA:9.1@1.0.0-final	Animations of changed references.
REQ-BA:9.2@1.0.0-final	Animations of changes to values.

Table 2 BA requirements

[3] (JointJS, 2022)



Requirement type
Functional requirement
Quality requirement
Constraint
Mixed

REQ-BA@1.0.0-final
[Mixed]
The project is now to be continued on the basis of the work carried out in the previous semester.
The following goals are to be achieved:
- Visualization of the individual stack frames with parameters/local variables and references to the heap
- Alternative visualization with JointJS
Additional desirable features that could be worked on are:
- Show objects that are no longer referenced
- Connection with the editor: Click on the variable in the visualization, the variable is highlighted in the source code.
- Animations, e.g. of changed references or changes to values

REQ-BA:5@1.0.0-final
[Functional requirement]
Visualization of the individual stack frames with parameters/local variables and references to the heap.

REQ-BA:6@1.0.0-final
[Constraint]
Alternative visualization with JointJS.

REQ-BA:7@1.0.0-final
[Functional requirement]
Show objects that are no longer referenced.

REQ-BA:8@1.0.0-final
[Functional requirement]
Connection with the editor: Click on the variable in the visualization, the variable is highlighted in the source code.

REQ-BA:9@1.0.0-final
[Functional requirement]
Animations, e.g. of changed references or changes to values.

REQ-BA:9.1@1.0.0-final
[Functional requirement]
Animations of changed references.

REQ-BA:9.2@1.0.0-final
[Functional requirement]
Animations of changes to values.

Figure 3 Mind map of the BA requirements

1.2 Stakeholders

The initial stakeholder analysis was conducted in the beginning of the SA project. For the BA, this analysis was revisited and updated.

The stakeholders were initially conducted to gain a better understanding of the expectations of VOOD. Some of these expectations were added as additional requirements for VOOD.

The analysis consists of an identification of the stakeholders (Table 3), the stakeholder analysis (Table 4) and the relation map (Table 5).

The analysis shows that there are multiple parties with high influence and motivation for this project. There are multiple parties that can contribute feedback and ideas to the project. Especially during the initial phases of VOOD project, where students of OO lectures were observed to gain ideas for the first design.

Notable changes to the stakeholder analysis for the BA are:

- The addition of Frank Koch and Leo Büttiker as co-examiners. As external examiners they expect an easy to read and well-structured report, which results in more time and effort put into the creation of the documentation compared to the SA.
- The addition of the English lecturer AnneMarie O'Neill as proof-reader.
- The students of OO have completed their course in the previous semester. They can still provide valuable feedback, but they cannot offer the perspective of a student visiting OO for the first time anymore. A new course would be offered again in the following semester. It is therefore difficult to gain first-hand user experience during the BA.

1.2.1 List of Stakeholders

Group	Contact	Goals	Role(s)	Expectations
OO lecturers: Mirko Stocker	mirko.stocker@ost.ch	Successful completion of the project	Adviser Product Owner Lecturer	MVP as a basis for further development. A tool for his students to study the runtime behaviour of OO programs easily.
Initial developers: Gino Cardillo Pascal Schürmann Alexandre Lagadec	gino.cardillo@ost.ch pascal.schuermann@ost.ch alexandre.lagadec@ost.ch	Successful completion of the project	Developer Student	To gain experience. To gain reputation.
FOSS community		To gain experience. To gain reputation.	Developers	Well-documented and maintainable source code that is open to extensions.
Lecturers in OO-related subjects: Thomas Letsch (AD) Silvan Gehrig (PF)	thomas.letsch@ost.ch silvan.gehrig@ost.ch	To demonstrate the runtime behaviour of high-level OO concepts.	Lecturer	To have a tool with which one can demonstrate the runtime behaviour of high-level OO concepts.
Students of OO and related subjects: Patrick Schürmann Alexandre Lagadec	patrick.schuermann@ost.ch alexandre.lagadec@ost.ch	To gain a deeper understanding of OO in general. To gain a deeper understanding of concepts that build on OO.	Student	To have a tool with which one can study the runtime behaviour of OO programs in general. To have a tool with which one can study the runtime behaviour of high-level OO concepts.
English lecturers: AnneMarie O'Neill	annemarie.oneill@ost.ch	Ensure linguistic quality of thesis in English at OST	Lecturer	A solid thesis report in English that reflects the high quality of English courses at OST
Co-examiners: Frank Koch Leo Büttiker	frank.koch@ost.ch leo@buettiker.org	Examine thesis as an unbiased individual	Additional thesis examiner	An easy to read (well-structured and comprehensible) and interesting thesis report

Table 3 List of Stakeholders

1.2.2 Stakeholder Analysis

Group	Cooperation	Influence	Motivation (s)
OO lecturers	Very Positive	Very High	Very High
Initial developers	Very Positive	Very High	Very High
FOSS community	Positive There is a vast amount of FOSS projects. Those who choose to contribute to our project are likely to be supportive.	None The project is not accepting contributions by the broad public yet.	Medium There is a vast amount of FOSS projects. Those who choose to contribute to our project are likely to be motivated to provide at least simple feature requests or bug reports.
Lecturers in OO-related subjects	Positive	Very High Lecturers in OO-related subjects can provide valuable insights on what makes learning high-level OO concepts and OO in general challenging	High Lecturers in OO-related subjects would probably like to demonstrate the runtime behaviour of high-level OO concepts.
Students of OO and related concepts	Positive	High Students of OO and related subjects can provide raw feedback on what makes learning high-level OO concepts and OO in general challenging	High Students of OO and related subjects would probably like to study the runtime behaviour of high-level OO concepts and OO in general.
English lecturers	Very positive	High The linguistic quality of written thesis report has a significant influence on the final grade	High AnneMarie O'Neill offered to proof-read our thesis report
Co-examiners	None A co-examiner will probably try to challenge the project team during the thesis presentation	High The final grade will be influenced by the co-examiner	High Academic interest

Table 4 Stakeholder analysis

1.2.3 Relation map

	OO lecturers	Initial developers	FOSS community	Lecturers in OO-related subjects	Students of OO and related concepts	English Lecturers	Co-examiners
OO lecturers	-	Very Good Successful kick-off meeting Weekly meetings planned	Unknown	Good Silvan and Mirko even share some lectures	Default Patrick attended OO lecture	Unknown	Unknown
Initial developers	-	-	None	Default Alexandre attended AD and PF lectures; no active discussion yet	Good Patrick is Pascal's brother	Good The project team and the English lecturers enjoyed the shared English classes at OST	Unknown
FOSS community	-	-	-	Unknown	Unknown	Unknown	Unknown
Lecturers in OO-related subjects	-	-	-	-	Default Alexandre attended AD and PF lectures; no use case yet	Unknown	Unknown
Students of OO and related concepts	-	-	-	-	-	Unknown	Unknown
English Lecturers	-	-	-	-	-	-	Unknown
Co-examiners	-	-	-	-	-	-	-

Table 5 Relation map of the stakeholders

1.3 Existing comparable products

One of the initial tasks done for VOOD was to test different products that also offer visualizations of a program during debugging. This was done to understand what possible visualization techniques already exist and what features VOOD could provide.

The result of this research is listed in Table 6. Based on this research the following features were aimed to be included in the final solution:

- Dynamic rendering
- Plant UML export
- Back-stepper function

During the mid-term presentation of the BA, the learning IDE BlueJ[4] was briefly discussed. The IDE was further analysed by the team, but no tangible ideas were adopted from it.

[4] (BlueJ, 2022)

Title	Authors	Organization	Category	Features	References
Visual Tracing for the Eclipse Java Debugger	- Bilal Alcala - Peter Bodesinsky - Alexander Gruber - Silvia Miksch	TU Wien	Eclipse plug-in	- Tracking - Temporal scaling - Search for variables	Paper (TU Wien) Paper (IEEE) Paper (Research gate) YouTube
Mirur Visual Debugger	Brandon Borkholder	Brandon Borkholder	Eclipse plug-in	- Plots for numeric arrays	Eclipse Marketplace
JIVE	Support: - Demian Lessa - Lead JIVE Developer - Jeffrey K. Czyz - Eclipse/JIVE Developer - Paul V. Gestwicki - Stand-alone JIVE Developer - J. Swaminathan - JIVE Plug-in Developer	University at Buffalo	Eclipse plug-in	- 'Reverse stepping' - Based on UML model -	University at Buffalo
OCL-based Runtime Monitoring of JVM hosted Applications	Lars Hamann (H-Man2), Martin Gogolla, Mirco Kuhlmann	Universität Bremen	Stand-alone?	- Based on UML model	Stack overflow TU Berlin SourceForge
Visual Debugger	Tim Kräuter	-	IntelliJ plug-in	- Uses native IntelliJ debugger as data source	Tim Kräuters Webseite JetBrains Marketplace Visual Debugger GitHub UI GitHub
Debug Visualizer	Henning Dieterichs	Microsoft (VS Code)	VS Code plug-in	- Dynamic rendering	Visual Studio Marketplace GitHub

Table 6 Existing comparable products

2 Constraints

The following requirements are considered to add a constraint in the freedom of the design of VOOD.

ID	Constraint	Consequences
REQ-SA:1.2@1.0.0-final	A visual debugger for Java should be created.	
REQ-SA:3.3@1.0.0-final	Another goal of the project is for it to be used as a Visual Studio Code extension.	The constraints and guidelines for VS Code extensions apply for the entire project. VS Code must be used for testing.
REQ-SA:3.4@1.0.0-final	Another goal of the project is for it to be used in GitPod.	
REQ-BA:6@1.0.0-final	Alternative visualization with JointJS.	- JointJS[5] must be integrated as an alternative visualization to vis.js[6]. - The user must be able to switch the visualization dynamically. - The underlying view model must be enhanced.

Table 7 Constraints

[5] (JointJS, 2022)
[6] (vis.js, 2022)

3 Quality Requirements

This chapter contains a list of all quality requirements. These quality requirements were further grouped in a quality tree and then specified in quality scenarios.

3.1 Quality Goals

We adapted all given quality requirements as quality goals. All quality requirements were found in the initial SA requirements set. During the BA the initial requirements where revisited and further developed.

We assigned ISO/IEC 25010 quality (sub-)characteristics[7] to each of these goals (see Table 8).

Requirement ID	Description	ISO/IEC 25010 quality characteristic
REQ-SA:1.1@1.0.0-final	The visual debugger is intended to facilitate the teaching of object-oriented programming.	Functional Suitability :: Functional appropriateness
REQ-SA:1.1.1@1.0.0-final	The visual debugger is intended to support teachers in object-oriented programming.	Functional Suitability :: Functional appropriateness
REQ-SA:1.1.2@1.0.0-final	The visual debugger is intended to support students in object-oriented programming.	Functional Suitability :: Functional appropriateness
REQ-SA:2.2@1.0.0-final	The visual debugger should make it possible to understand how objects and variables change over the course of the program.	Functional Suitability :: Functional appropriateness
REQ-SA:2.2.1@1.0.0-final	The visual debugger should make it possible to understand how objects change over the course of the program.	Functional Suitability :: Functional appropriateness
REQ-SA:2.2.2@1.0.0-final	The visual debugger should make it possible to understand how variables change over the course of the program.	Functional Suitability :: Functional appropriateness
REQ-SA:3.1@1.0.0-final	Another goal of the project is to make the setup [of the system] as simple as possible.	Portability :: Installability
REQ-SA:3.2@1.0.0-final	Another goal of the project is to make the import of the program as simple as possible.	Usability :: Operability
REQ-SA:3.5@1.0.0-final	Another goal of the project is good usability [of the system].	Usability

[7] (ISO 25000 Portal, 2022)

REQ-SA:4@1.0.0-final	Well documented and maintainable source code that is open to extensions.	Maintainability
REQ-SA:4.1@1.0.0-final	Well documented source code.	Maintainability :: Analysability
REQ-SA:4.2@1.0.0-final	Maintainable source code.	Maintainability
REQ-SA:4.3@1.0.0-final	Source code that is open to extensions.	Maintainability :: Reusability

Table 8 Quality goals derived from quality requirements

3.2 Quality Tree

A quality tree was created as a visual reference for the quality goals (Figure 4).

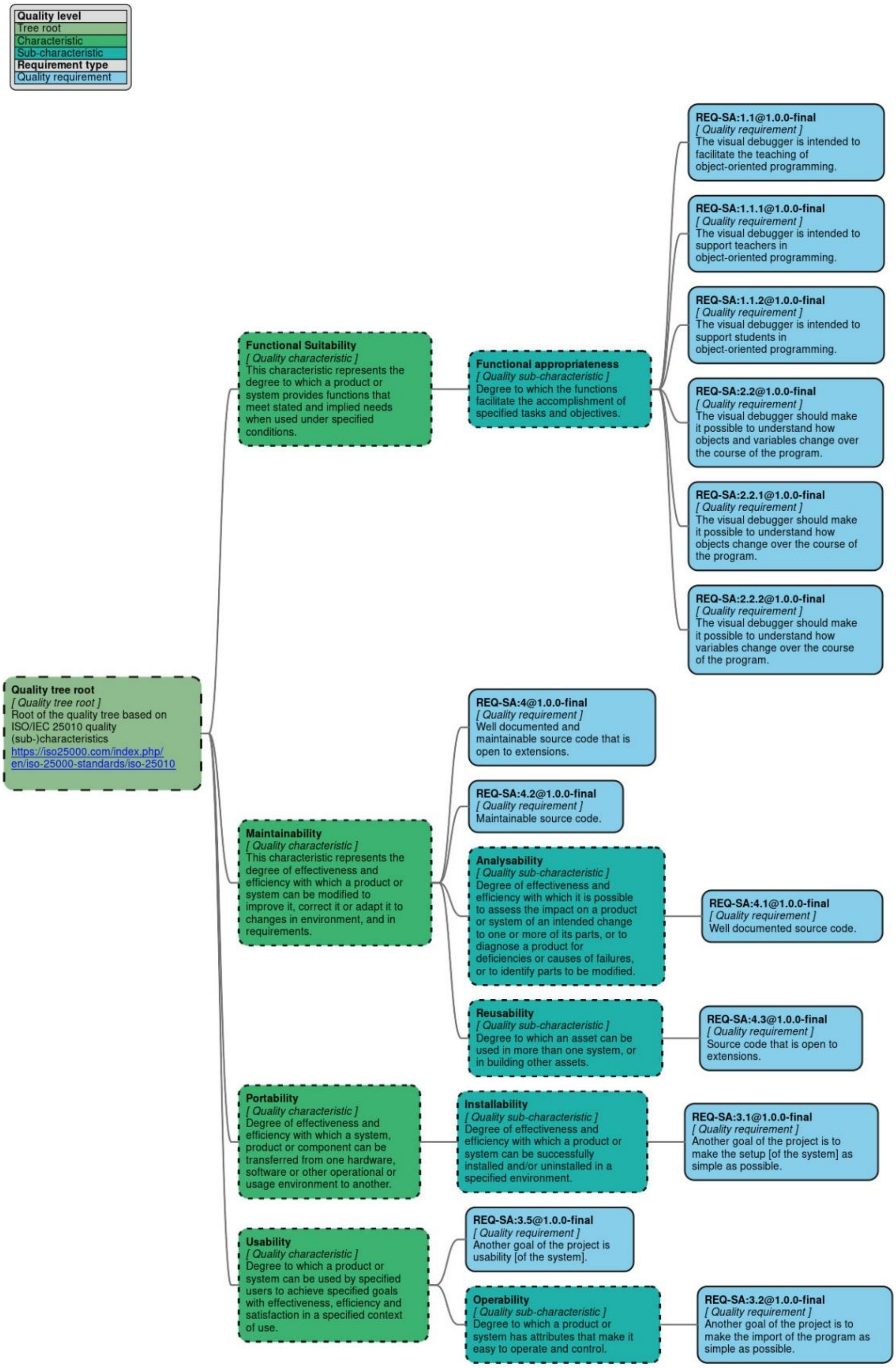

Quality level
Tree root
Characteristic
Sub-characteristic
Requirement type
Quality requirement
Quality tree root
[Quality tree root]
Root of the quality tree based on ISO/IEC 25010 quality (sub-)characteristics
https://iso25000.com/index.php/en/iso-25000-standards/iso-25010
Functional Suitability
[Quality characteristic]
This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions.
Functional appropriateness
[Quality sub-characteristic]
Degree to which the functions facilitate the accomplishment of specified tasks and objectives.
REQ-SA:1.1@1.0.0-final
[Quality requirement]
The visual debugger is intended to facilitate the teaching of object-oriented programming.
REQ-SA:1.1.1@1.0.0-final
[Quality requirement]
The visual debugger is intended to support teachers in object-oriented programming.
REQ-SA:1.1.2@1.0.0-final
[Quality requirement]
The visual debugger is intended to support students in object-oriented programming.
REQ-SA:2.2@1.0.0-final
[Quality requirement]
The visual debugger should make it possible to understand how objects and variables change over the course of the program.
REQ-SA:2.2.1@1.0.0-final
[Quality requirement]
The visual debugger should make it possible to understand how objects change over the course of the program.
REQ-SA:2.2.2@1.0.0-final
[Quality requirement]
The visual debugger should make it possible to understand how variables change over the course of the program.
Maintainability
[Quality characteristic]
This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements.
REQ-SA:4@1.0.0-final
[Quality requirement]
Well documented and maintainable source code that is open to extensions.
REQ-SA:4.2@1.0.0-final
[Quality requirement]
Maintainable source code.
Analysability
[Quality sub-characteristic]
Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
REQ-SA:4.1@1.0.0-final
[Quality requirement]
Well documented source code.
Reusability
[Quality sub-characteristic]
Degree to which an asset can be used in more than one system, or in building other assets.
REQ-SA:4.3@1.0.0-final
[Quality requirement]
Source code that is open to extensions.
Portability
[Quality characteristic]
Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.
Installability
[Quality sub-characteristic]
Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment.
REQ-SA:3.1@1.0.0-final
[Quality requirement]
Another goal of the project is to make the setup [of the system] as simple as possible.
Usability
[Quality characteristic]
Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.
REQ-SA:3.5@1.0.0-final
[Quality requirement]
Another goal of the project is usability [of the system].
Operability
[Quality sub-characteristic]
Degree to which a product or system has attributes that make it easy to operate and control.
REQ-SA:3.2@1.0.0-final
[Quality requirement]
Another goal of the project is to make the import of the program as simple as possible.

Figure 4 Quality tree

3.3 Quality Scenarios

This section further defines the quality requirements using scenarios.

3.3.1 REQ-SA:1.1.1

Scenario	OO lecturer prepares a course for a semester.	
Business Goals	The visual debugger is intended to support teachers in teaching object-oriented programming.	
Relevant Quality attributes	Functional suitability	
Scenario Components	**Stimulus**	OO lecturer runs VOOD with codes samples for the OO lecture.
	Stimulus Source	OO lecturer
	Environment	VOOD is installed as an extension in VS Code. The OO lecturer has opened the demo code for the lecture.
	Artifact	Export functions, PanelView
	Response	The debugger visualizes the demo code in a way that it could be used for an explanation during the lecture.
	Response Measure	The lecturer can include exported slides in the presentation without having to edit the export.

Table 9 Quality scenario: OO lecturer prepares a course for a semester

3.3.2 REQ-SA:1.1.2

Scenario	Students analysing a sample code for the OO course.	
Business Goals	The visual debugger is intended to support students in object-oriented programming.	
Relevant Quality attributes	Functional Suitability	
Scenario Components	**Stimulus**	Student has opened a sample project in VS Code.
	Stimulus Source	Students of OO and related concepts
	Environment	VOOD is installed as an extension in VS Code.
	Artifact	PanelView
	Response	The student can run the sample in the debugger. VOOD visualizes the output of the debugger in a way that helps the user to gain a deeper understanding of the code.
	Response Measure	The student feels that the visualization of VOOD helped him to understand the sample

Table 10 Quality scenario: Students analysing a sample code for the OO course

3.3.3 REQ-SA:2.2.1

Scenario	Student uses VOOD during an OO exercise.	
Business Goals	Functional Suitability	
Relevant Quality attributes	The visual debugger should make it possible to understand how objects change over the course of the program.	
Scenario Components	**Stimulus**	Student runs extension during an OO exercise.
	Stimulus Source	Students of OO and related concepts
	Environment	VOOD is installed as an extension in VS Code. The OO exercise is opened in VS Code and the debugger was started.
	Artifact	PanelView
	Response	The system visualizes the debugging steps to help the user understand the mechanics of objects.
	Response Measure	The student feels that VOOD helped him understand the OO exercise.

Table 11 Quality scenario: Student uses VOOD during an OO exercise

3.3.4 REQ-SA:2.2.2

Scenario	Student uses VOOD during an OO exercise.	
Business Goals	VOOD should make it possible to understand how variables change over the course of the program.	
Relevant Quality attributes	Functional Suitability	
Scenario Components	**Stimulus**	Student runs extension during an OO exercise.
	Stimulus Source	Students of OO and related concepts
	Environment	VOOD is installed as an extension in VS Code. The OO exercise is opened in VS Code and the debugger was started.
	Artifact	PanelView
	Response	The system visualizes the debugging steps to help the user understand how objects are assigned to variables.
	Response Measure	The student feels that VOOD helped him understand the OO exercise.

Table 12 Quality scenario: Student uses VOOD during an OO exercise

3.3.5 REQ-SA:3.1

Scenario	Student participates in their first exercise of the OO course and has no IDE installed for developing Java.	
Business Goals	Another goal of the project is to make the setup [of the system] as simple as possible.	
Relevant Quality attributes	Installability	
Scenario Components	**Stimulus**	Student wants to use VOOD.
	Stimulus Source	Students of OO and related concepts
	Environment	VS Code is not installed locally on the Computer of the Student.
	Artifact	Installation
	Response	VS Code and VOOD can easily be installed or used in a web browser with GitPod.
	Response Measure	Student can get a running version of VS Code with VOOD with a short instruction text in under 15 minutes.

Table 13 Quality scenario: Student participates in their first exercise of the OO course and has no IDE installed for developing Java

3.3.6 REQ-SA:3.2

Scenario	User installs VOOD.	
Business Goals	Another goal of the project is to make the import of the program as simple as possible.	
Relevant Quality attributes	Operability	
Scenario Components	**Stimulus**	VOOD is installed and started for the first time in VS Code.
	Stimulus Source	Students of OO and related concepts
	Environment	A published version of VOOD available on the VS marketplace.
	Artifact	-
	Response	VOOD can be installed from the VS marketplace. The introductory text on the VS marketplace page should instruct the user on how they can work with the extension.
	Response Measure	The user can install and use VOOD within reasonable time.

Table 14 Quality scenario: User installs VOOD

3.3.7 REQ-SA:3.5

Scenario	User uses system to debug a simple solution.	
Business Goals	Another goal of the project is usability [of the system].	
Relevant Quality attributes	Usability	
Scenario Components	**Stimulus**	User runs command “VOOD: Open debugger view” and starts debugging.
	Stimulus Source	Users in General
	Environment	VOOD is installed as extension in VS Code. The user has read the VS marketplace page.
	Artifact	PanelView, Settings
	Response	The system visualizes the debugging steps in a way that is understandable for the user.
	Response Measure	The user can use the functions of the extension without a problem after they read the VS marketplace page.

Table 15 Quality scenario: User uses system to debug a simple solution

3.3.8 REQ-SA:4.1

Scenario	New developers want to contribute to VOOD extension.	
Business Goals	Well documented source code	
Relevant Quality attributes	Analysability	
Scenario Components	**Stimulus**	A new developer wants to contribute to VOOD and therefore needs to understand how VOOD works.
	Stimulus Source	Future developer of VOOD
	Environment	VOOD is available on a public repository.
	Artifact	
	Response	New developers should get an understanding of the inner workings of VOOD within a reasonable amount of time.
	Response Measure	The source code is self-explanatory. The CONTRIBUTING.md file exists.

Table 16 Quality scenario: New developers want to contribute to VOOD extension

3.3.9 REQ-SA:4.2

Scenario	After the BA is finished, a feature of VS Code used by VOOD is no longer supported. A new developer is tasked with fixing the issue.	
Business Goals	Maintainable source code	
Relevant Quality attributes	Maintainability	
Scenario Components	**Stimulus**	A new developer needs to fix an issue of VOOD.
	Stimulus Source	Future developer of VOOD
	Environment	VOOD is available on a public repository.
	Artifact	
	Response	The developer should be able to fix the issue and replace deprecated components with ease.
	Response Measure	The components of VOOD are well documented. The architecture allows for logical components to be replaced.

Table 17 Quality scenario: After the BA is finished, a feature of VS Code used by VOOD is no longer supported. A new developer is tasked with fixing the issue

3.3.10 REQ-SA:4.3

Scenario	After the bachelor thesis is finished, the need for a new visualization option arises.	
Business Goals	Source code that is open to extensions	
Relevant Quality attributes	Reusability	
Scenario Components	**Stimulus**	A new developer needs to add a visualization for VOOD.
	Stimulus Source	Future developer of VOOD
	Environment	VOOD is available on a public repository.
	Artifact	
	Response	The developer should be able to extend the functionalities of VOOD.
	Response Measure	The architecture of VOOD should allow for new components to be added.

Table 18 Quality scenario: After the bachelor thesis is finished, the need for a new visualization option arises

4 Solution Strategy

To outline our solution strategy, we first mapped goals (based on fine-grained requirements) to solution approaches.

Goal/Requirement	Description	Solution approach
REQ-SA:1.1.1@1.0.0-final	The visual debugger is intended to support teachers in object-oriented programming.	The visualization can be exported as a PNG, animated GIF, PlantUML or GraphViz for further use in teaching.
REQ-SA:1.1.2@1.0.0-final	The visual debugger is intended to support students in object-oriented programming.	The dynamic and highly interactive visualization of object graphs allows for playful exploration and learning.
REQ-SA:1.2@1.0.0-final	A visual debugger for Java should be created.	The outputs of a Java debugger extension are captured and visualized.
REQ-SA:2.1.1.1@1.0.0-final	The aim is to visualize objects graphically.	External matured graphics libraries will be used for visualization. Various common formats will be used for exports.
REQ-SA:2.1.1.2@1.0.0-final	The aim is to visualize variables graphically.	External matured graphics libraries will be used for visualization. Various common formats will be used for exports.
REQ-SA:2.1.2@1.0.0-final	The aim is to run a program step by step inside the debugger.	The history of the visualizations is saved and made navigable.
REQ-SA:2.2.1@1.0.0-final	The visual debugger should make it possible to understand how objects change over the course of the program.	Transitions between object states are animated in the view.
REQ-SA:2.2.2@1.0.0-final	The visual debugger should make it possible to understand how variables change over the course of the program.	Transitions between variable states are animated in the view.
REQ-SA:3.1@1.0.0-final	Another goal of the project is to make the setup [of the system] as simple as possible.	The finished extension will be made available on the VS marketplace.
REQ-SA:3.2@1.0.0-final	Another goal of the project is to make the import of the program as simple as possible.	This is done by VS Code.
REQ-SA:3.3@1.0.0-final	Another goal of the project is to be used as a Visual Studio Code extension.	The extension will be written in TypeScript and will adhere to best practices of VS Code extension development.

REQ-SA:3.4@1.0.0-final	Another goal of the project is for it to be used in GitPod.	As a VS Code extension, the product will be usable by GitPod as well.
REQ-SA:3.5@1.0.0-final	Another goal of the project is usability [of the system].	Various quality-of-life improvements increase usability. - Clustering of nodes - Hiding of nodes - Customizable colours - Visual improvements - Easier access to exports
REQ-SA:4.1@1.0.0-final	Well documented source code.	The source code is kept as simple as possible, and comments are added where necessary.
REQ-SA:4.2@1.0.0-final	Maintainable source code.	The source code is well structured.
REQ-SA:4.3@1.0.0-final	Source code that is open to extensions.	The source code is modular, which allows for easier extensions.
REQ-BA:5@1.0.0-final	Visualization of the individual stack frames with parameters/local variables and references to the heap.	The stack frame is selectable via a dropdown.
REQ-BA:6@1.0.0-final	Alternative visualization with JointJS.	Use the JointJS[8] library to add an alternative visualization.
REQ-BA:7@1.0.0-final	Show objects that are no longer referenced.	Not to be implemented. See section 10.3.
REQ-BA:8@1.0.0-final	Connection with the editor: Click on the variable in the visualization, the variable is highlighted in the source code.	Make a connection between the clicked variable and its position in the source code. Then move the cursor to that location.
REQ-BA:9.1@1.0.0-final	Animations of changed references.	Newly added and changed references between debugging steps use a configurable, dedicated colour set (background colour, border colour, and font colour) across all dynamic visualizations.
REQ-BA:9.2@1.0.0-final	Animations of changes to values.	Newly added nodes and nodes with changed values between debugging steps use a configurable, dedicated colour set (background colour, border colour, and font colour) across all dynamic visualizations.

Table 19 Solution strategy

[8] (JointJS, 2022)

5 Risks and Technical Debts

The following risks were identified for the bachelor thesis:

ID	Category	Title	Description	Expected effects	Prevention	Behaviour on entry
R1	PM	Bad work package ordering	The order in which work packages are processed is causing the project to stall.	Delays	Analyse inter-package dependencies during planning.	Assign all available team members to blocking work packages.
R2	PM	Poor requirements analysis	Requirements are not properly understood, approved, and prioritized.	Delays	Regular validation of the requirements internally and with stakeholders.	Re-evaluate erroneous requirements with stakeholders.
R3	CI/CD	Blocking build failure	A build failure in the CI/CD pipeline blocks the project.	Delays	Developer guidelines: wait for CI / CD results every day before finishing work.	Responsible developers fix build problems immediately and notify all blocked team members.
R4	Dev	Poor plug-in development	Functionalities and best practices related to the chosen plug-in framework are unknown, leading to unnecessary efforts and/or delays.	Delays, unnecessary complexity	Study plug-in development tutorials.	Perform design reviews on a regular basis and maintain a knowledge base for key insights.
R5	Team	Illness	Team members are not able to work at full capacity due to illness.	Delays	-	-
R6	Tech	API deprecations by VS Code	VS Code core APIs are suddenly deprecated, requiring implementation changes.	Loss of functionality	-	Re-implement critical parts.
R7	Tech	Incompatible 3rd party extensions	Third-party extensions suddenly behave differently (version pinning is not available), requiring implementation changes.	Loss of functionality	-	Re-implement critical parts.

Table 20 Risk identification and management plan

5.1 Risk Assessment

All parameters and calculation methods are based on vague estimates and negotiations within the project team.

Risks are defined as weighted damages. A weight is calculated as the probability of occurrence divided by the probability of detection (a similar approach is used to calculate the so-called risk priority number in the failure mode and effects the analysis process).

The bachelor thesis does not have to satisfy a monetary budget, but a time budget. Damages are therefore quantified using time units instead of monetary units. The maximal damages are calculated as the product of the following factors:

- Estimated number of team members blocked
- Estimated maximal time to resolution in sprints
- Individual mean time budget per sprint (22.5h)

ID	Category	Title	*Blocked team members (max.)*	*Time to resolution (max.) [sprints]*	Damage (max.) [h]	*Prob. of occurrence [1]*	*Prob. of detection [1]*	Weight [1]	Risk [h]
R1	PM	Bad work package ordering	*3*	*1*	67.5	*5%*	*50%*	10%	6.75
R2	PM	Poor requirements analysis	*3*	*2*	135	*10%*	*25%*	40%	54
R3	CI/CD	Blocking build failure	*3*	*1*	67.5	*5%*	*100%*	5%	3.375
R4	Dev	Poor plug-in development	*3*	*2*	135	*20%*	*25%*	80%	108
R5	Team	Illness	*1*	*1*	22.5	*20%*	*100%*	20%	4.5
R6	Tech	API deprecations by VS Code	*1*	*2*	45	*10%*	*50%*	20%	9
R7	Tech	Incompatible 3rd party extensions	*1*	*0.5*	11.25	*5%*	*25%*	20%	2.25
Sum									**187.875**

Table 21 Simplified risk list

After planning risk prevention and containment, we were able to reduce all identified risks to an acceptable level.

The weighted damage adds up to 187.875 hours, which is 17.40% of the time budget of 1080 hours. To save time for dealing with the expected risks, we have increased our time estimates accordingly by roughly 20% during the sprint planning.

5.2 Risk Matrix

All identified risks are still within acceptable limits (taking into account planned risk prevention and treatment measures) as shown in the Risk matrix figure:

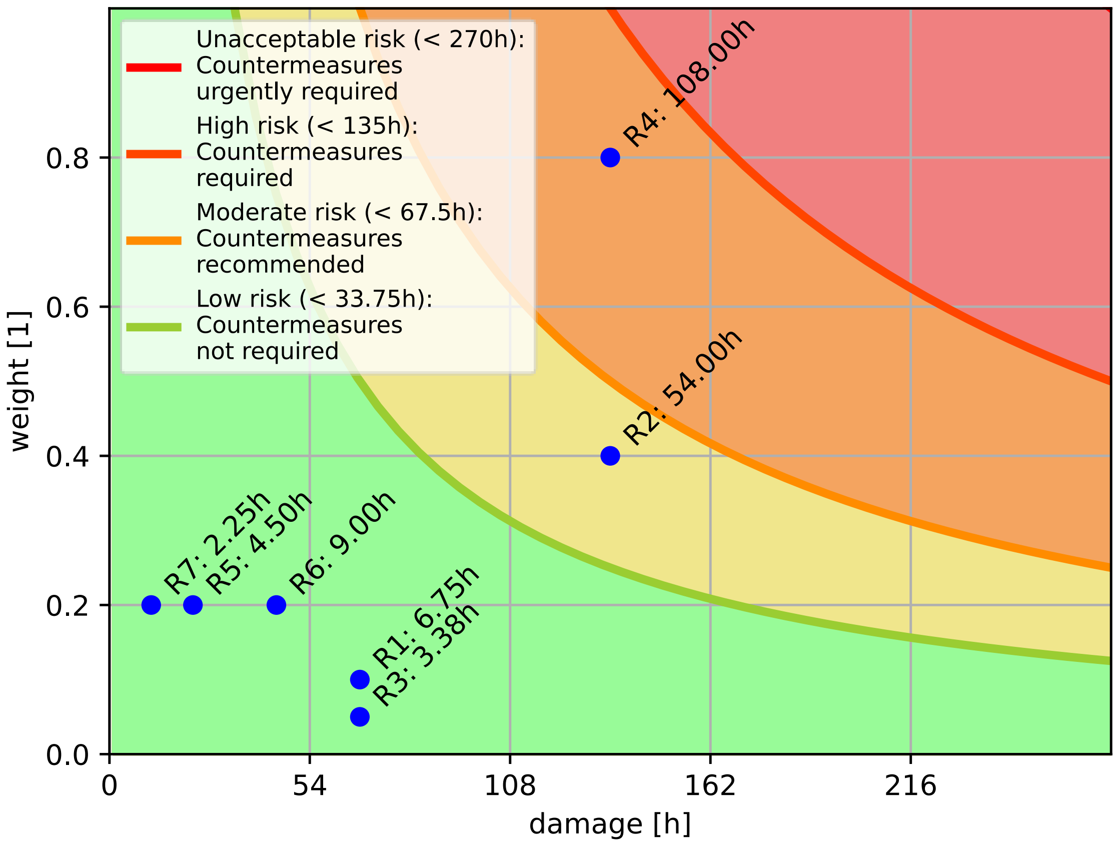


Figure 5 Risk matrix

Initial Solution

This part provides an overview of the fundamental architecture of the SA version of VOOD. The goal of this part is to give an understanding on how the SA version was constructed to provide a basis for the next part “Development”.

This part consists of an overview of the cross-cutting concepts for the solution followed by a description of the system using the C4 model[9].

6 Cross-cutting Concepts

This chapter describes overall principal regulations and solution ideas that were relevant in multiple parts of the system.

6.1 User Experience concepts (UX)

The UI of the extension was designed in a way that is intuitive for the user and gives enough options to freely visualize the current debugging step.

VS Code offers different UI elements for extensions. But the usage of these elements is regulated by the extension guidelines of Visual Studio.

For the Visual OO Debugger, it was assumed that basic commands were sufficient. It was thought that too many options would clutter the UI.

For rendering the visualization, a WebView was used. This approach gave the most flexibility and allowed the use of external visualization libraries.

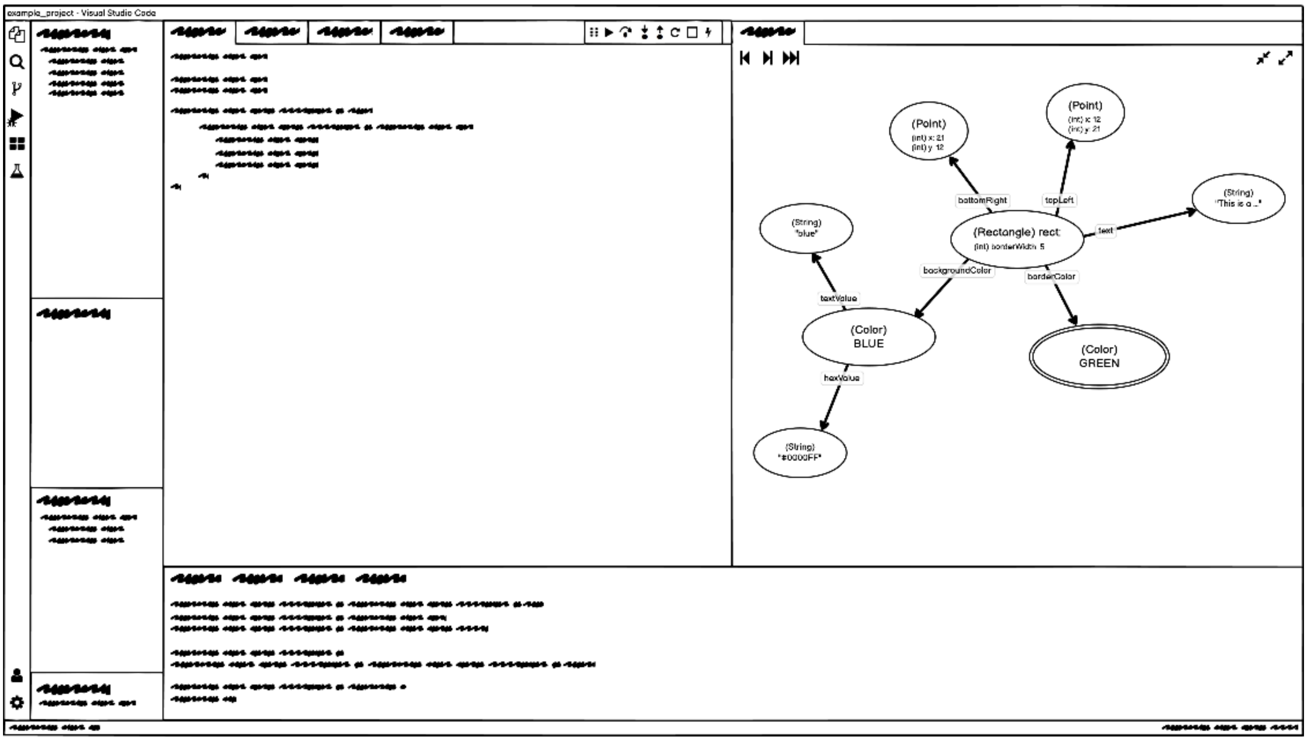


Figure 6 Wireframe of VS Code integration

[9] (C4 model, 2021)

6.2 Development Concepts

This chapter describes concepts that were used during development, with focus on code quality. These concepts implemented during the SA were reused for the BA.

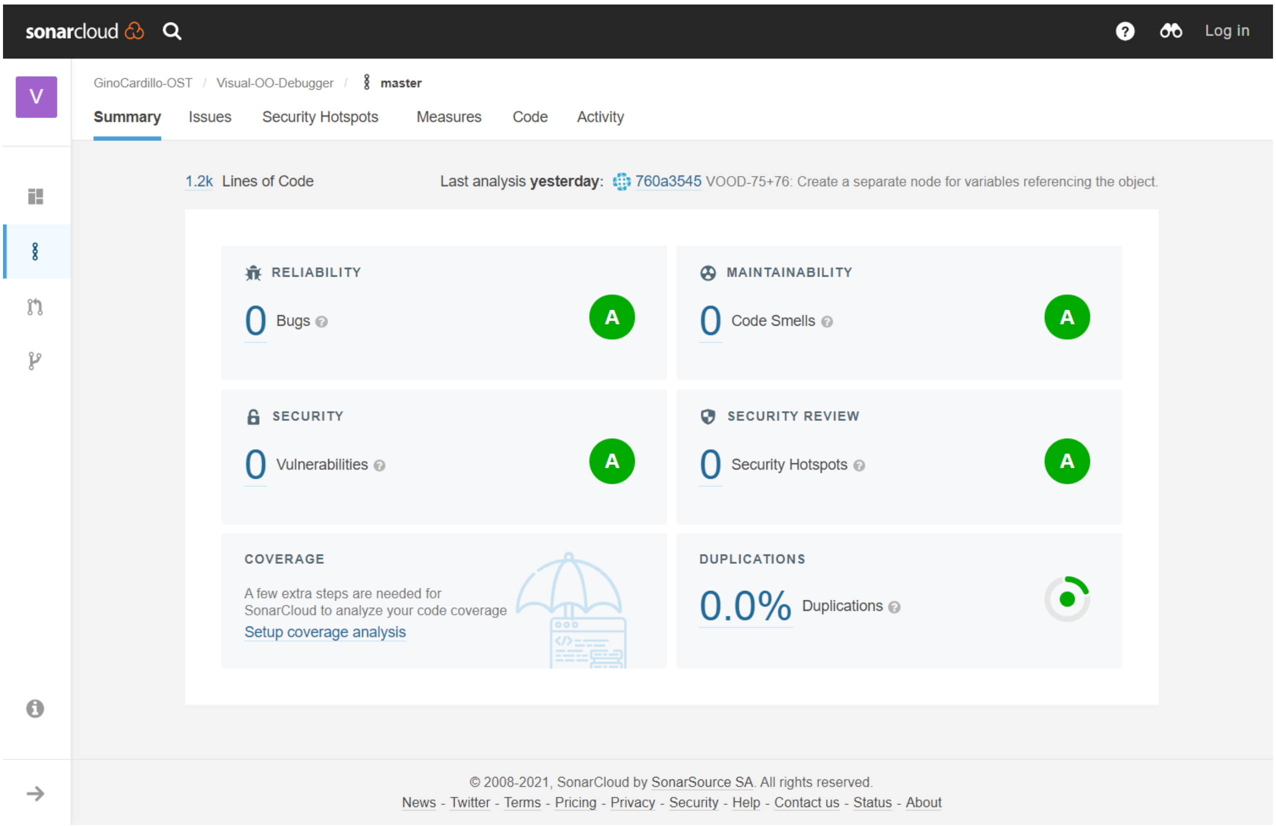
6.2.1 Code Review

For maintaining and developing the code, GitHub[10] was used. New changes to the code will be developed in feature branches. To merge a feature branch into the master branch, a pull request will be created. Only if at least one other team members approves, will they be merged into the master branch.

6.2.2 Code Guidelines

To ensure a coherent code style, prettier will be used.

To ensure a certain degree of code quality, ESLint[11] and SonarCloud[12] will be used.



1.2k Lines of Code

Last analysis **yesterday**: 760a3545 VOOD-75+76: Create a separate node for variables referencing the object.

RELIABILITY
0 Bugs A

MAINTAINABILITY
0 Code Smells A

SECURITY
0 Vulnerabilities A

SECURITY REVIEW
0 Security Hotspots A

COVERAGE
A few extra steps are needed for SonarCloud to analyze your code coverage
Setup coverage analysis

DUPLICATIONS
0.0% Duplications

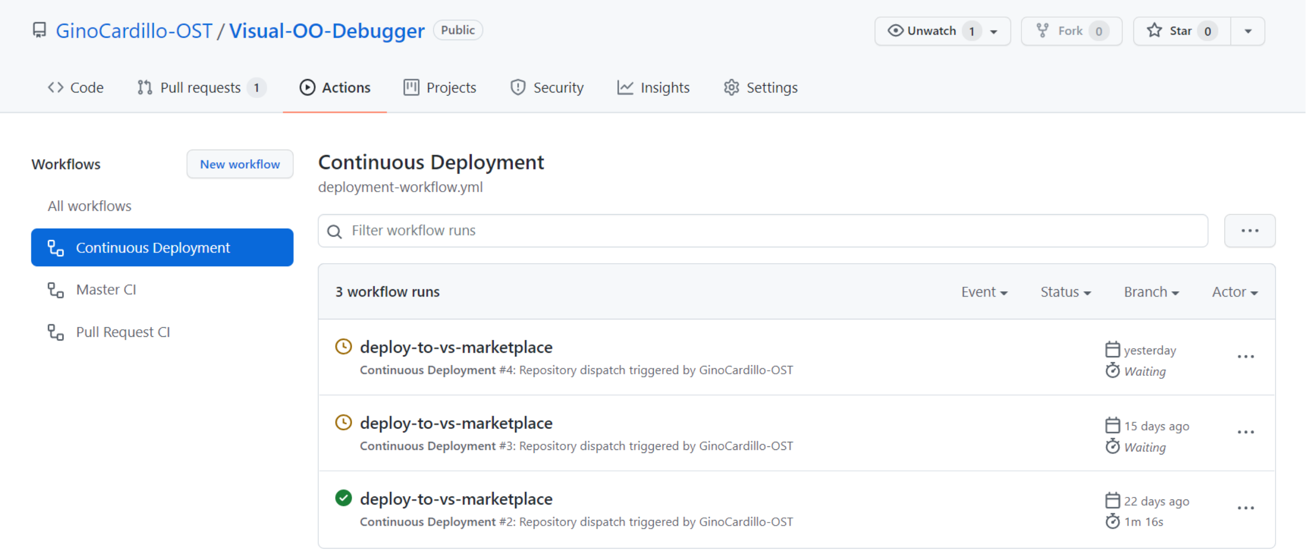
© 2008-2021, SonarCloud by SonarSource SA. All rights reserved.
News - Twitter - Terms - Pricing - Privacy - Security - Help - Contact us - Status - About

Figure 7 SonarCloud summary

[10] (GitHub, 2022)
[11] (ESLint, 2022)
[12] (SonarCloud, 2022)

7 Deployment View

The project makes use of GitHub Actions to enable Continuous Deployment. Three workflows are defined.



Continuous Deployment
deployment-workflow.yml

3 workflow runs

deploy-to-vs-marketplace
Continuous Deployment #4: Repository dispatch triggered by GinoCardillo-OST
yesterday
Waiting

deploy-to-vs-marketplace
Continuous Deployment #3: Repository dispatch triggered by GinoCardillo-OST
15 days ago
Waiting

deploy-to-vs-marketplace
Continuous Deployment #2: Repository dispatch triggered by GinoCardillo-OST
22 days ago
1m 16s

Figure 8 GitHub workflow overview

Pull Request CI

The Pull Request CI workflow is executed whenever a pull request is created or updated. This workflow does a checkout of the code and then runs linting checks, formatting checks, unit tests and integration tests. Only if all those tasks succeed, can the pull request be merged.

Master CI

The Master CI workflow is triggered on every commit to the master branch. First it does the same checks as the Pull Request CI workflow. If that succeeds, another job is started which uses a GitHub action to create a repository dispatch event. This dispatch event will then trigger the Continuous Deployment workflow.

Continuous Deployment

The Continuous Deployment workflow can only be triggered by a repository dispatch event. This workflow only has one job for the deployment, which is executed on the production environment. Any workflows that operate on the production environment must be reviewed before the jobs can start. If approved, the job does a checkout of the code on the master branch, builds it, and publishes it to the VS Marketplace.

8 System Scope and Context

This chapter describes the delimitations of the system from all its communication partners. The visualization is based on the system context diagram of the C4 model.[13]

8.1 Business Context

The business context describes the surrounding system of the solution for VOOD. That includes the User, Visual Studio Code, and the Debug Adapter. The relationships of these components are depicted in the following figure and table.

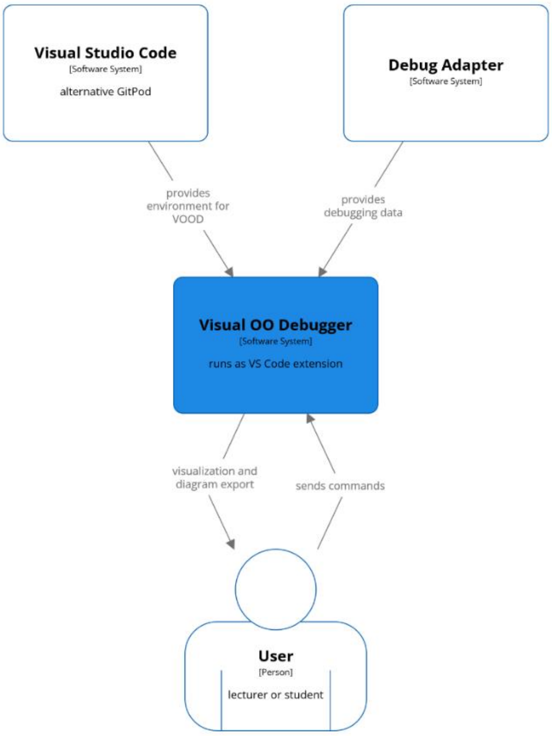


Figure 9 Context diagram

Partner	Communication
User	The user can influence the Visual OO Debugger, either by sending commands or by adjusting the visualization. The user can also export the current visualization.
Visual Studio Code	Besides being the environment for the Visual OO Debugger, Visual Studio Code provides the Visual OO Debugger with: - Redirected user commands - Settings, set by the user - Debugger information
Debug Adapter	The debug adapter manages the communication with the debugger. The debug adapter provides the Visual OO Debugger with the debugging data.

Table 22 Description of the business context

[13] (C4 model, 2021)

8.2 Technical Context

From the business context, two technical contexts can be derived. VS Code and the Debug Adapter. The Visual OO Debugger requires the following API's and protocols to communicate in these contexts.

API	Definition
(VS Code API, 2022)	https://code.visualstudio.com/api/references/vscode-api
(Debug Adapter Protocol, 2022)	https://microsoft.github.io/debug-adapter-protocol/specification

Table 23 Required API's and protocols

VS Code API

The VS Code API allows the Visual OO Debugger to access the functionality and data of VS Code. The following VS Code API features are relevant for the visual OO debugger:

Feature	Description
commands	For registering and listening to commands
Debug	Provides functionalities for accessing the debugger.
DebugSession	Provides access to the current debug session.
ExtensionContext	Provides a collection of utilities private to the extension.
Uri	A universal resource identifier for representing a resource
ViewColumn	To specify a location of a window inside VS Code
Webview	To display html content inside VS Code
WebviewPanel	For handling a window containing a Webview
window	Namespace of the currently active window
workspace	Gives access to the current workspace.

Table 24 Used VS Code API features

Debug Adapter Protocol

Visual Studio Code communicates with the debugger through the debug adapter. The debug adapter is an intermediate component that normalizes the access to different debuggers. It is possible for the Visual OO Debugger to send requests to the debug adapter using the debug adapter protocol. These requests allow access to the following resources:

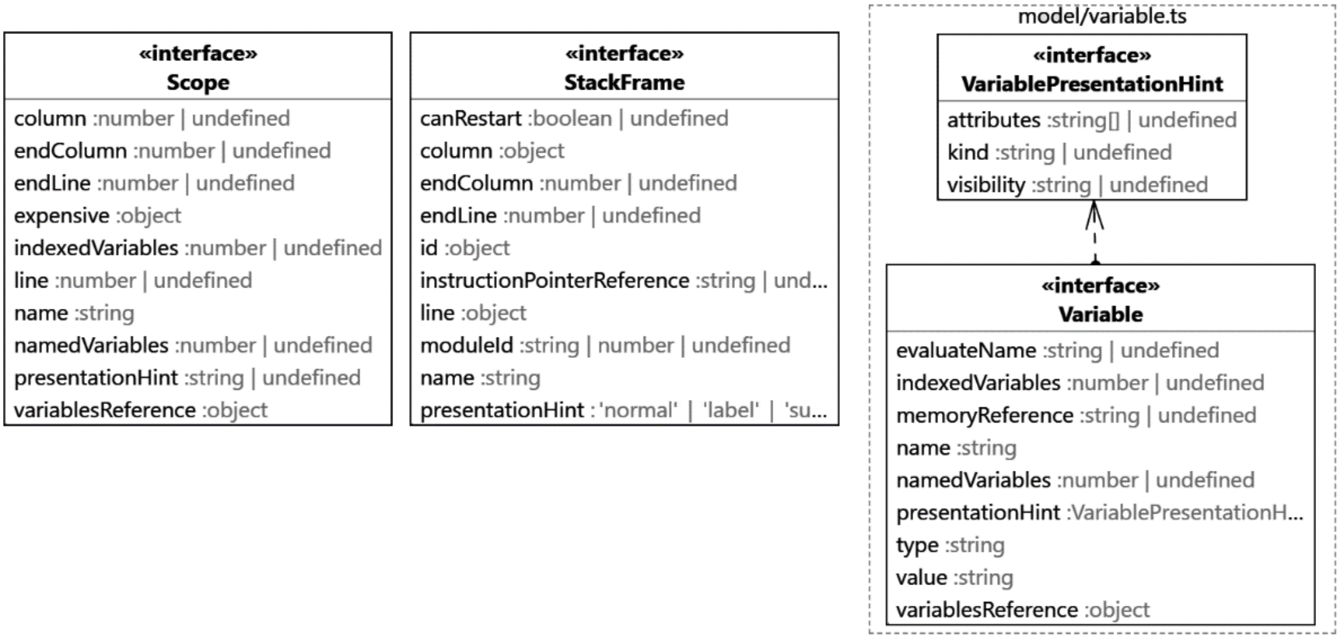

«interface»
Scope
column :number | undefined
endColumn :number | undefined
endLine :number | undefined
expensive :object
indexedVariables :number | undefined
line :number | undefined
name :string
namedVariables :number | undefined
presentationHint :string | undefined
variablesReference :object
«interface»
StackFrame
canRestart :boolean | undefined
column :object
endColumn :number | undefined
endLine :number | undefined
id :object
instructionPointerReference :string | und...
line :object
moduleId :string | number | undefined
name :string
presentationHint : 'normal' | 'label' | 'su...
model/variable.ts
«interface»
VariablePresentationHint
attributes :string[] | undefined
kind :string | undefined
visibility :string | undefined
«interface»
Variable
evaluateName :string | undefined
indexedVariables :number | undefined
memoryReference :string | undefined
name :string
namedVariables :number | undefined
presentationHint :VariablePresentationH...
type :string
value :string
variablesReference :object

Figure 10 Class diagram of debug adapter protocol models

9 Building Block View

The building block view shows the static decomposition of the system into building blocks as well as their dependencies. To illustrate the building blocks, the “Container” Diagram of the C4 model was used. This decomposition shows the initial architectural decision taken for the SA project.

9.1 Whitebox Overall System

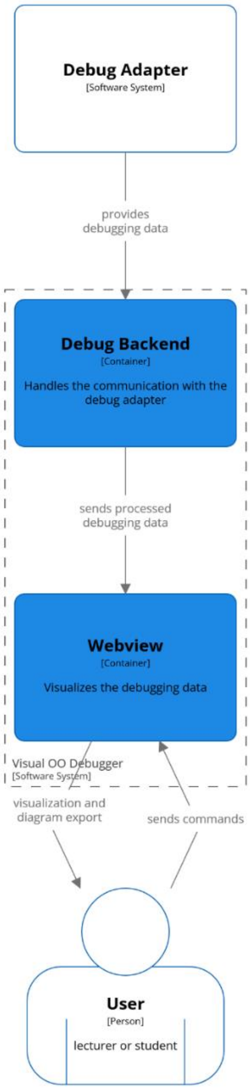


Figure 11 Class diagram of debug adapter protocol models

The Visual OO Debugger can be roughly divided in two parts, the debug backend and the webview. This split was made to separate the logic for visualizing the data and retrieving and processing the debugging data. Furthermore, this solution simplified the distribution of work inside the project team.

9.1.1 Debug Backend

Responsibility

The debug backend handles the communication with the debug adapter of VS Code. This goal can be split into three separated tasks:

- Handling debugging events
- Retrieving the data from the external debug adapter
- Process the received debugging data for the webview

Interfaces

- The debug backend communicates with the debug adapter via the debug adapter protocol.
- The debug backend is given an instance of a webview class which is called whenever a debugging event is triggered.

9.1.2 Webview

Responsibility

The webview is responsible for rendering the visualization of the debugging data. Besides the visualization, the webview handles user interactions. This includes:

- User interactions using commands
- User interactions with the visualization, either by repositioning elements or by using the back-stepper function

Interfaces

- The user can send commands to the Visual OO Debugger to open the webview panel.
- The user can trigger an export by sending a command.
- The user can interact with the visualization on the webview panel.
- An instance of a webview object is given to the debug backend. When the debug backend detects a debug event, an update function will be triggered.

9.2 Level 2

Level 2 specifies the inner structure of the building blocks in the overall system utilizing the “Components” diagram of the C4 model.

9.2.1 White Box Debug Backend

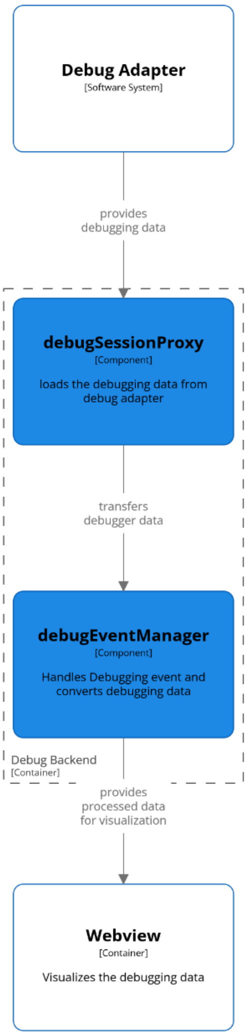


Figure 12 Component diagram of debug backend

The debug backend consists of two components, the debug session proxy, and the debug event manager.

The debugSessionProxy handles the communication with the debug adapter and receives debugging data.

The debugEventManager handles the debugging events. If the debugEventManager detects that the debugger has stopped, it will request the debugging data from the debugSessionProxy. The debugging data received by the debugSessionProxy will then be processed before it is sent to the webview for the visualization.

9.2.2 White Box Webview

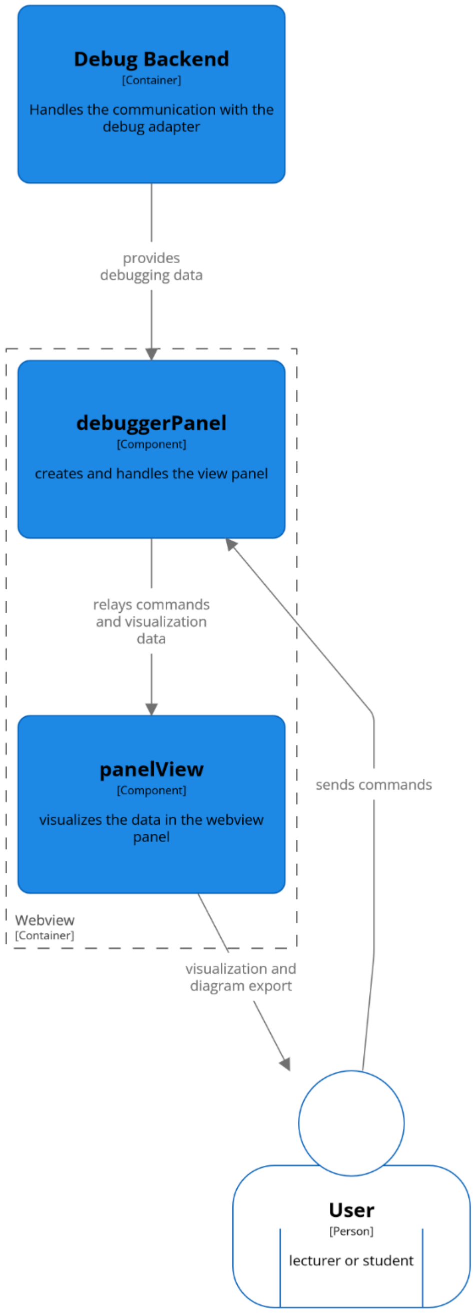

Debug Backend
[Container]
Handles the communication with the debug adapter
provides debugging data
debuggerPanel
[Component]
creates and handles the view panel
relays commands and visualization data
panelView
[Component]
visualizes the data in the webview panel
sends commands
Webview
[Container]
visualization and diagram export
User
[Person]
lecturer or student

Figure 13 Component diagram of webview

The webview can be split into two basic components, the debuggerPanel and the panelView.

The debuggerPanel creates the view panel and handles incoming commands for the visualization.

The panelView renders the visualization, provides the possibility for the user to reposition elements and handles the export of a diagram.

9.3 Level 3

Level 3 specifies the inner structure of the building blocks in level 2.

9.3.1 White Box DebugEventManager

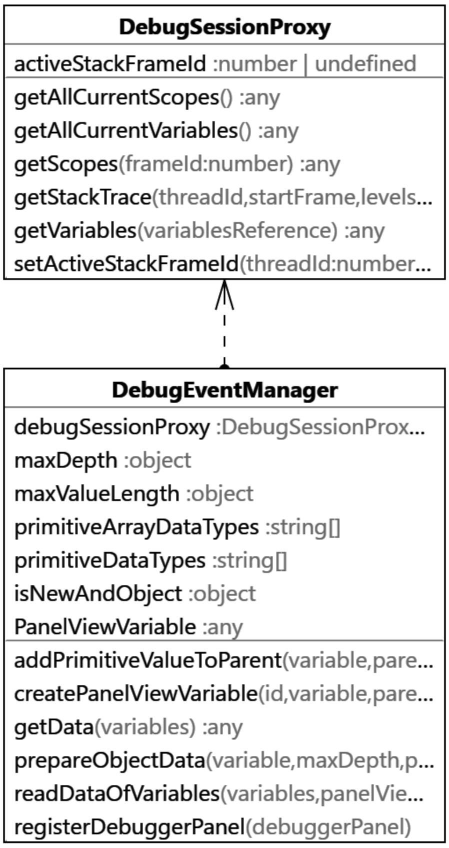

DebugSessionProxy
activeStackFrameId :number | undefined
getAllCurrentScopes() :any
getAllCurrentVariables() :any
getScopes(frameId:number) :any
getStackTrace(threadId,startFrame,levels...
getVariables(variablesReference) :any
setActiveStackFrameId(threadId:number...
DebugEventManager
debugSessionProxy :DebugSessionProx...
maxDepth :object
maxValueLength :object
primitiveArrayDataTypes :string[]
primitiveDataTypes :string[]
isNewAndObject :object
PanelViewVariable :any
addPrimitiveValueToParent(variable,pare...
createPanelViewVariable(id,variable,pare...
getData(variables) :any
prepareObjectData(variable,maxDepth,p...
readDataOfVariables(variables,panelVie...
registerDebuggerPanel(debuggerPanel)

Figure 14 Class diagram of DebugEventManager

The DebugEventManager is a single class with a DebugSessionProxy instance to load the debug data.

The DebugEventManager creates the event handler for when the debugger stopped in his constructor. In this case, the event handler uses the debug session proxy to load all current variables. These variables are then processed and made into PanelViewInputs for the webview to visualize.

9.3.2 White Box PanelView

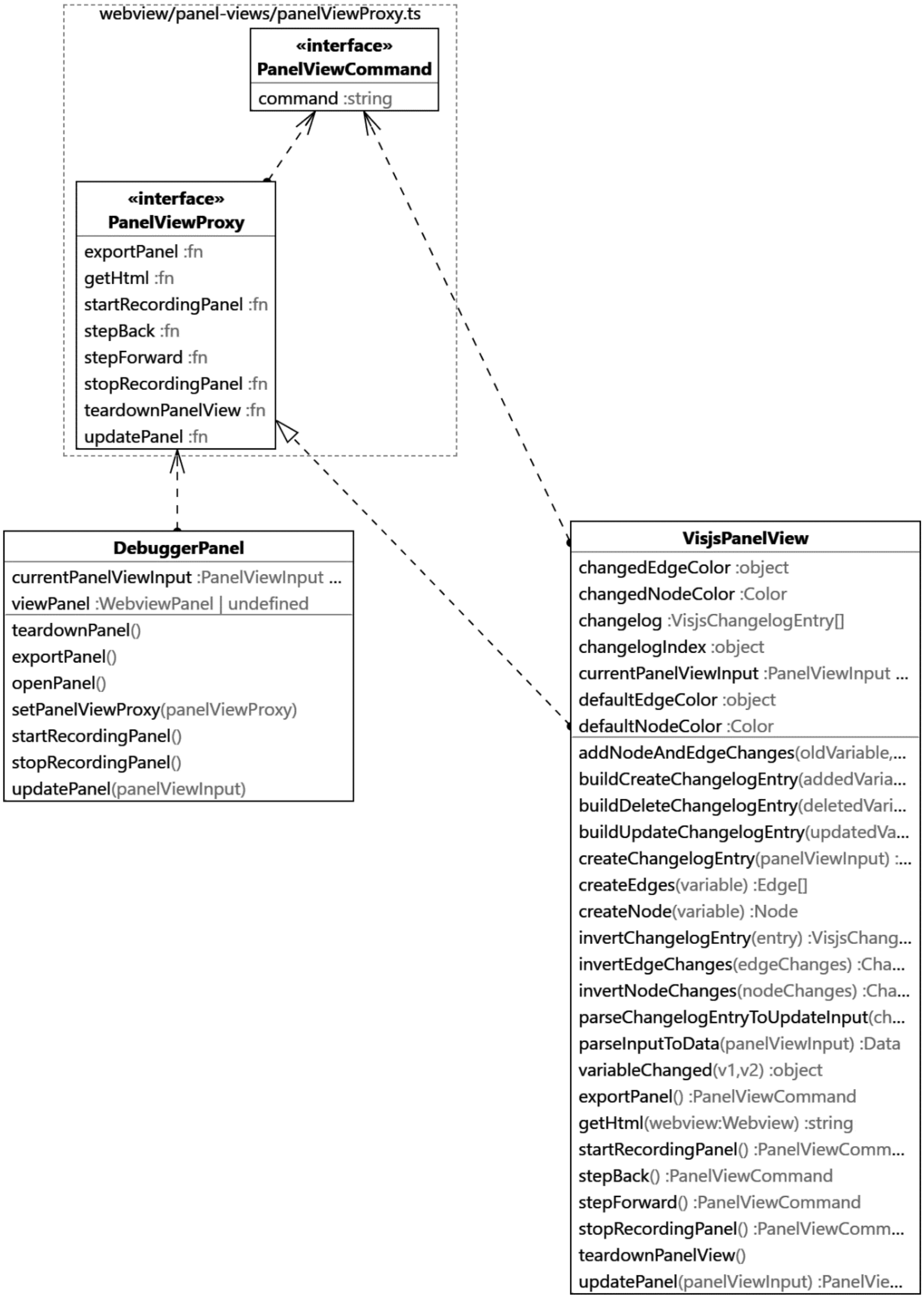

webview/panel-views/panelViewProxy.ts
«interface»
PanelViewCommand
command :string
«interface»
PanelViewProxy
exportPanel :fn
getHtml :fn
startRecordingPanel :fn
stepBack :fn
stepForward :fn
stopRecordingPanel :fn
teardownPanelView :fn
updatePanel :fn
DebuggerPanel
currentPanelViewInput :PanelViewInput ...
viewPanel :WebviewPanel | undefined
teardownPanel()
exportPanel()
openPanel()
setPanelViewProxy(panelViewProxy)
startRecordingPanel()
stopRecordingPanel()
updatePanel(panelViewInput)
VisjsPanelView
changedEdgeColor :object
changedNodeColor :Color
changelog :VisjsChangelogEntry[]
changelogIndex :object
currentPanelViewInput :PanelViewInput ...
defaultEdgeColor :object
defaultNodeColor :Color
addNodeAndEdgeChanges(oldVariable,...
buildCreateChangelogEntry(addedVaria...
buildDeleteChangelogEntry(deletedVari...
buildUpdateChangelogEntry(updatedVa...
createChangelogEntry(panelViewInput) :...
createEdges(variable) :Edge[]
createNode(variable) :Node
invertChangelogEntry(entry) :VisjsChang...
invertEdgeChanges(edgeChanges) :Cha...
invertNodeChanges(nodeChanges) :Cha...
parseChangelogEntryToUpdateInput(ch...
parseInputToData(panelViewInput) :Data
variableChanged(v1,v2) :object
exportPanel() :PanelViewCommand
getHtml(webview:Webview) :string
startRecordingPanel() :PanelViewComm...
stepBack() :PanelViewCommand
stepForward() :PanelViewCommand
stopRecordingPanel() :PanelViewComm...
teardownPanelView()
updatePanel(panelViewInput) :PanelVie...

Figure 15 Class diagram of PanelView

The PanelView is an implementation of the interface PanelViewProxy. VisjsPanelView is one of these concrete implementations which renders the debug data as a vis.js diagram. The visjsPanelView reads an html file, where the rendering of the diagram takes place. The communication between the Visual OO Debugger and the html page is handled with PanelViewCommands.

This html page is rendered in a webview panel, which the DebuggerPanel constructs and manages.

9.3.3 Panel View Variable

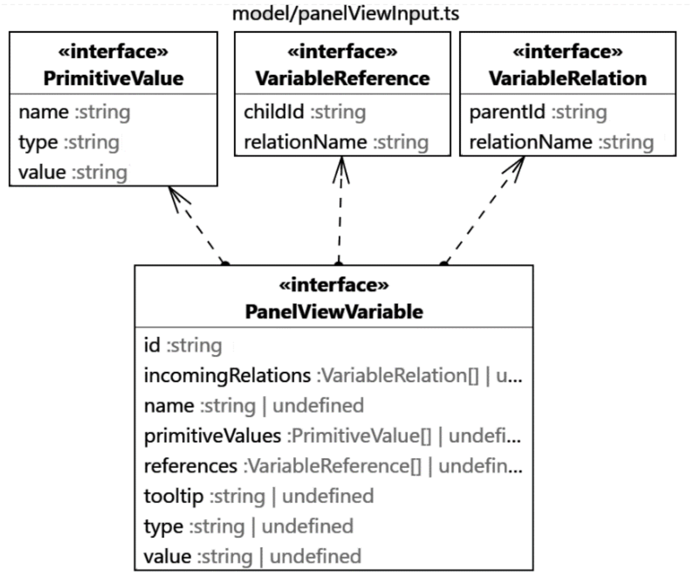


Figure 16 Class diagram of PanelView variables

The DebugEventManager prepares the variables for the webview as PanelView variables.

Development

In this part, all the changes made during the BA are listed in detail. The changes are ordered by their corresponding requirement or risk. For each requirement, there will be a brief explanation of the context of the requirement and what limitations of the environment had to be overcome. A description on how these features were implemented in VOOD will be given.

This part ends with a comparison of the BA version and SA version of VOOD, where further conclusions and an outlook of the VOOD project is given.

10 Implemented Requirements

In this chapter, we go over every requirement and describe how it was implemented or explain why it was not implemented.

10.1 REQ-BA:5 Visualization of the individual stack frames with parameters/local variables and references to the heap.

The debug process for a single thread application can be thought of as a series of debug steps. Each debug step has call stack consisting of stack frames.

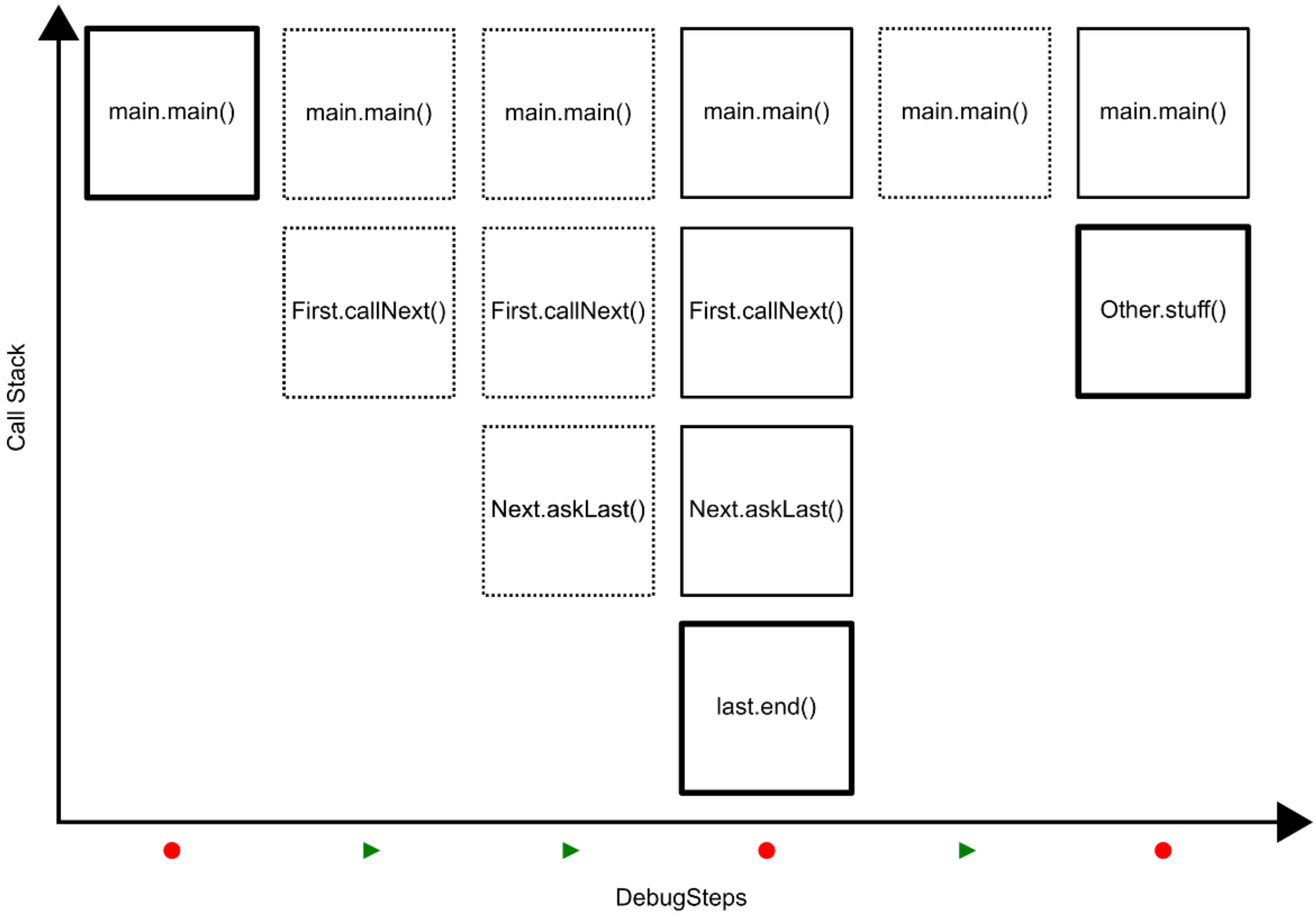


Figure 17 Visualization of the call stack over time

The SA version of VOOD saves the current stack frame (bold outline) in the history, every time the debugging process stops. Otherwise, the data is discarded (dotted outline). The user can then use the back-stepper function to navigate between the saved frames. To fulfil the requirement, an option should be implemented for the user to also navigate through the other stack frames of the call stack for each saved debugging step (normal outline).

10.1.1 Limitations

The Debug Adapter Protocol, which VOOD uses to access the data for each debug step, does not provide an id for a stack frame, and it is unlikely that this feature will be added soon. It is possible to differentiate stack frames by their name and by their position on the call stack, but these are not unique properties.

Therefore, it is not possible for the back-stepper function to navigate with certainty back to a specific stack frame. For example, an implementation where the user chooses a stack frame and then navigates back to an earlier version would not be possible.

10.1.2 Decision

The decision was made that the stack frame can only be changed for the most recent debugging step. When the user steps backward only the topmost stack frame will be rendered.

To change the stack frame on the current debugging step a dropdown will be added to the toolbar of VOOD.

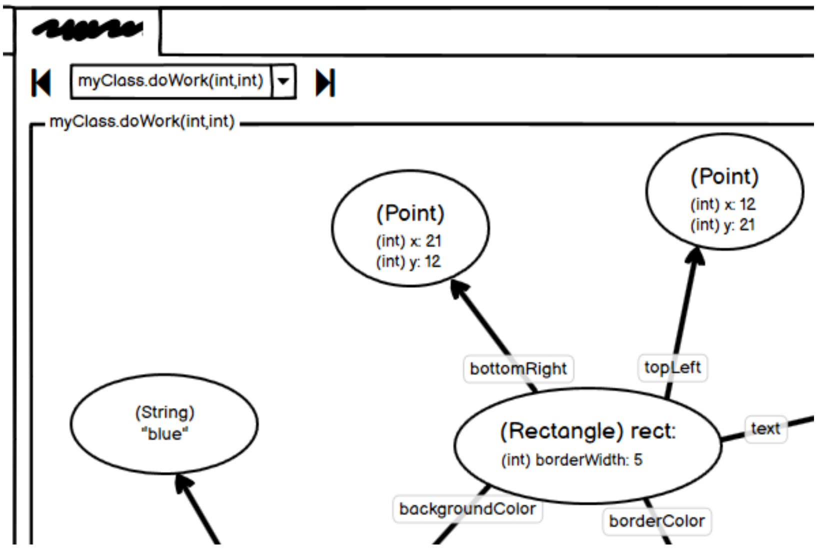


Figure 18 Wireframe for stack frame dropdown

10.1.3 Implementation

The solution implemented the dropdown of the wireframe. Initially, the dropdown was set between the step forward and the step back buttons, with the intention that the dropdown controls the "height" of the view, but this idea was dropped, and the dropdown was implemented as a standalone element.

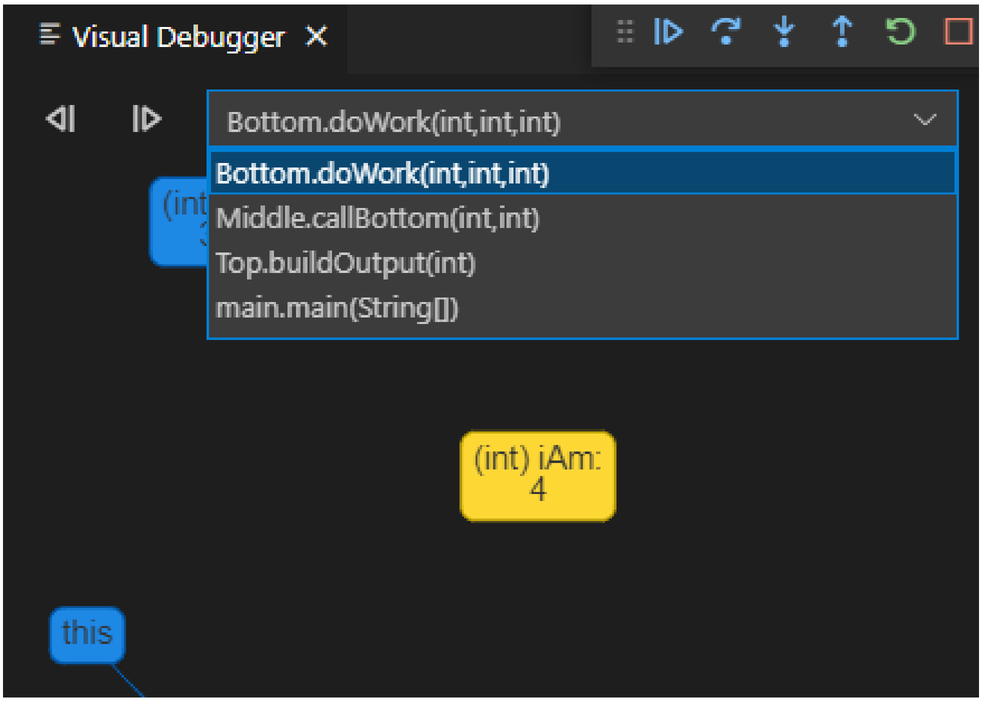

Visual Debugger ✕
Bottom.doWork(int,int,int)
Bottom.doWork(int,int,int)
Middle.callBottom(int,int)
Top.buildOutput(int)
main.main(String[])
(int) iAm:
4
this

Figure 19 Implemented stack frame dropdown

The debugAdapter class already provides the call stack for a debugging step. Therefore, to implant this feature, only the UI had to be updated. There were no major changes in the architecture of the extension.

During the implementation of the dropdown the WebView UI toolkit was installed. This toolkit adds VS Code specific elements and styles to the WebView and ensures a more consistent look inside VS Code. Other components, such as the back/forward buttons were updated to use the new styles.

10.2 REQ-BA:6 Alternative visualization with JointJS.

Already during the term project, JointJS was considered a promising alternate visualization framework to vis.js. While the vis.js framework provides a highly interactive experience that encourages exploration, it lacks the capabilities to display tabular data layouts for stack frame and object fields in a compact manner. Instead, elements that could be rendered as tabular entries are put in separate nodes, which can result in larger-span graphs. This issue has been addressed by the recent addition of node hiding and clustering capabilities.

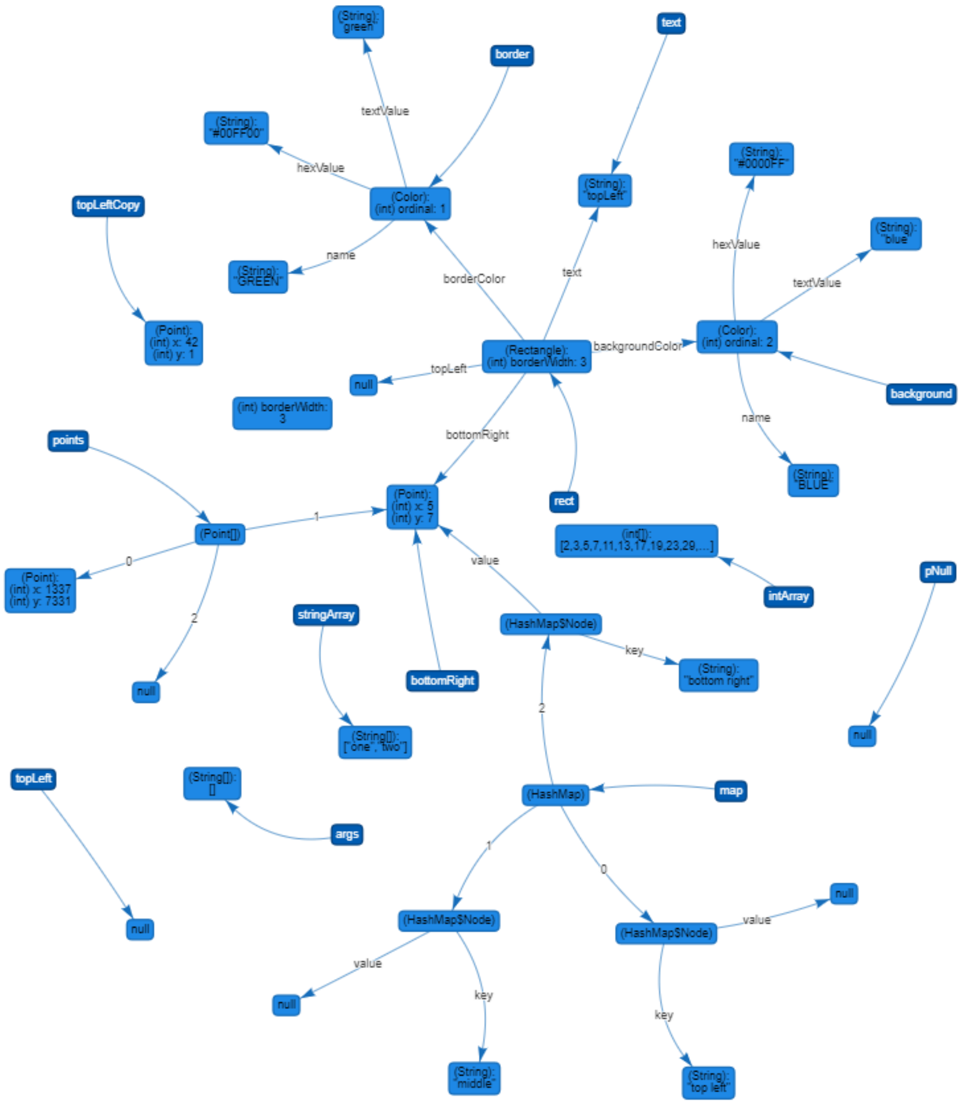


Figure 20 A vis.js-based graph

The markup exports offered a more tabular, but static representation of object diagrams. While a minor code improvement has resulted in a more balanced graph layout for PlantUML exports in general, both types of markup exports (PlantUML and GraphViz) often exhibited undesirable layout trade-offs. Some of these shortcomings are frustrating in that a human editor could easily solve most problems if only the chart elements could be rearranged. Furthermore, the mark-up formats offer only limited post-processing support, particularly the PlantUML format.

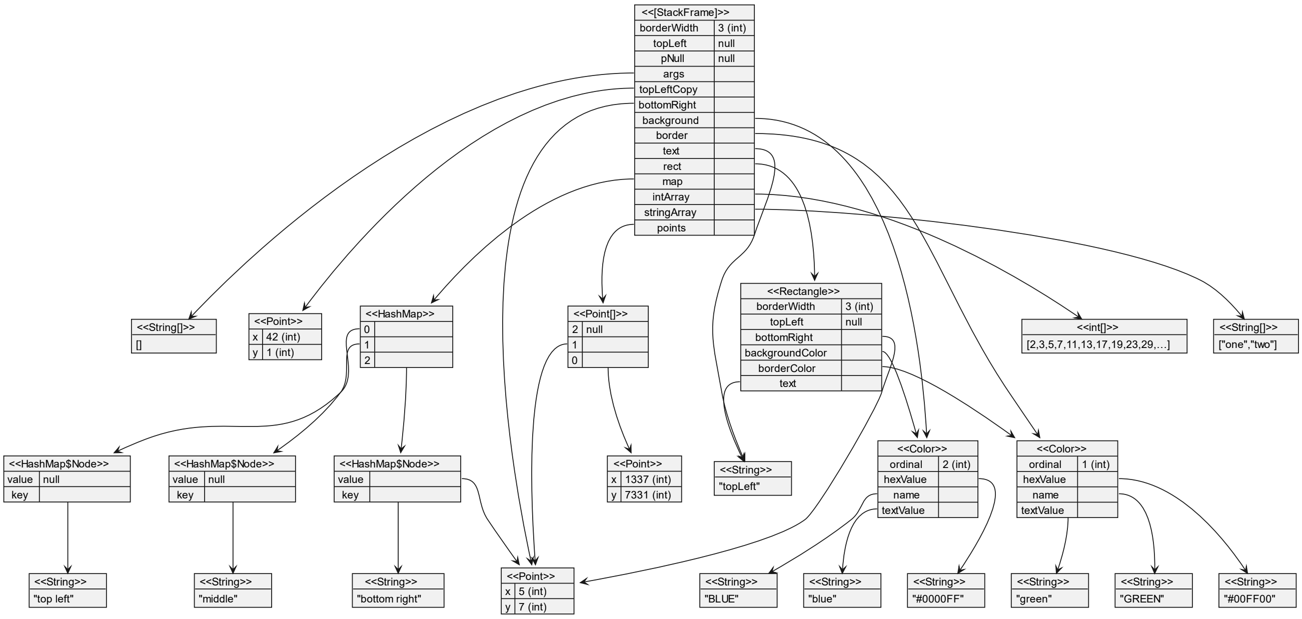


Figure 21 PlantUML export: Good symmetry, but inconsistent arrow directions

The GraphViz export offers similar symmetry and more consistent arrow orientations from the egress ports on the right but suffers even more from low angles between incoming edges and frequent edge crossings.

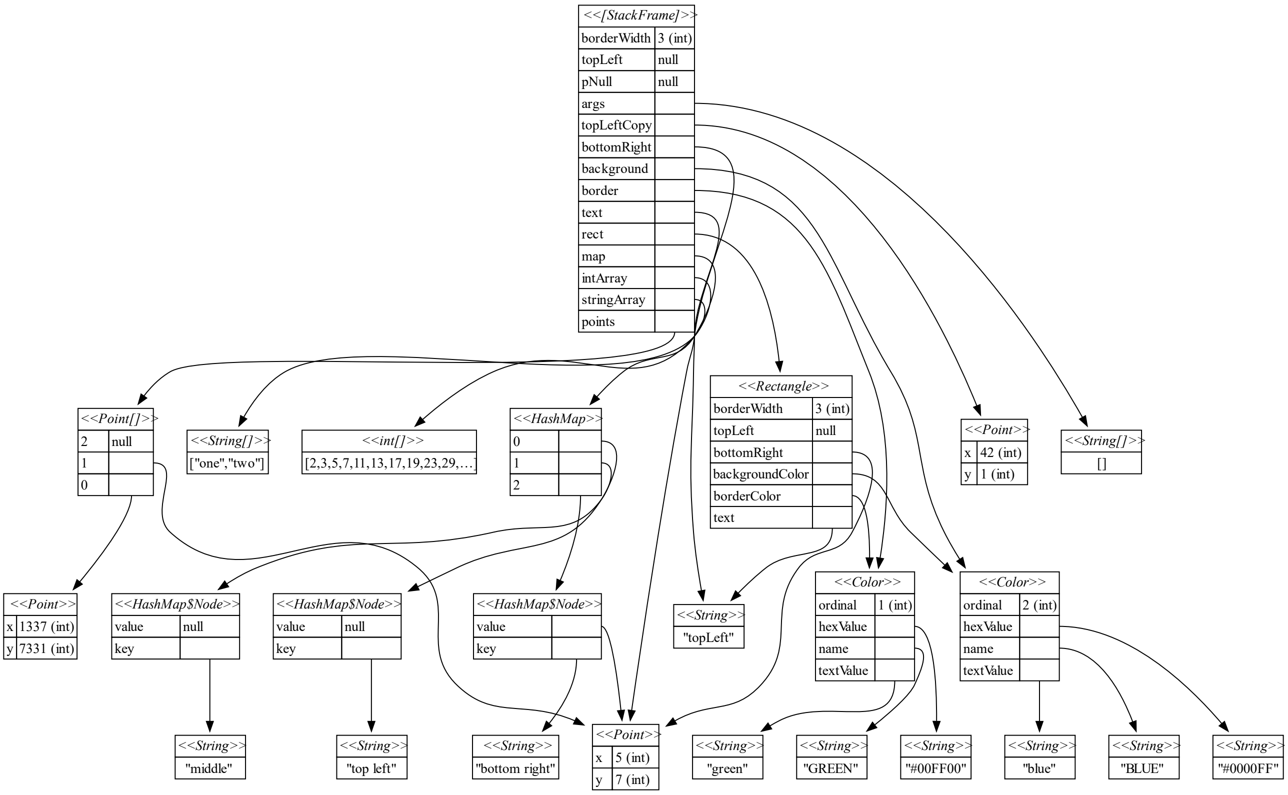


Figure 22 GraphViz export: Good symmetry, consistent arrow orientation, but more edge crossings

The UML class diagram example for JointJS[14] demonstrates pre-defined shapes that can be modified and re-used as building blocks for an object diagram.

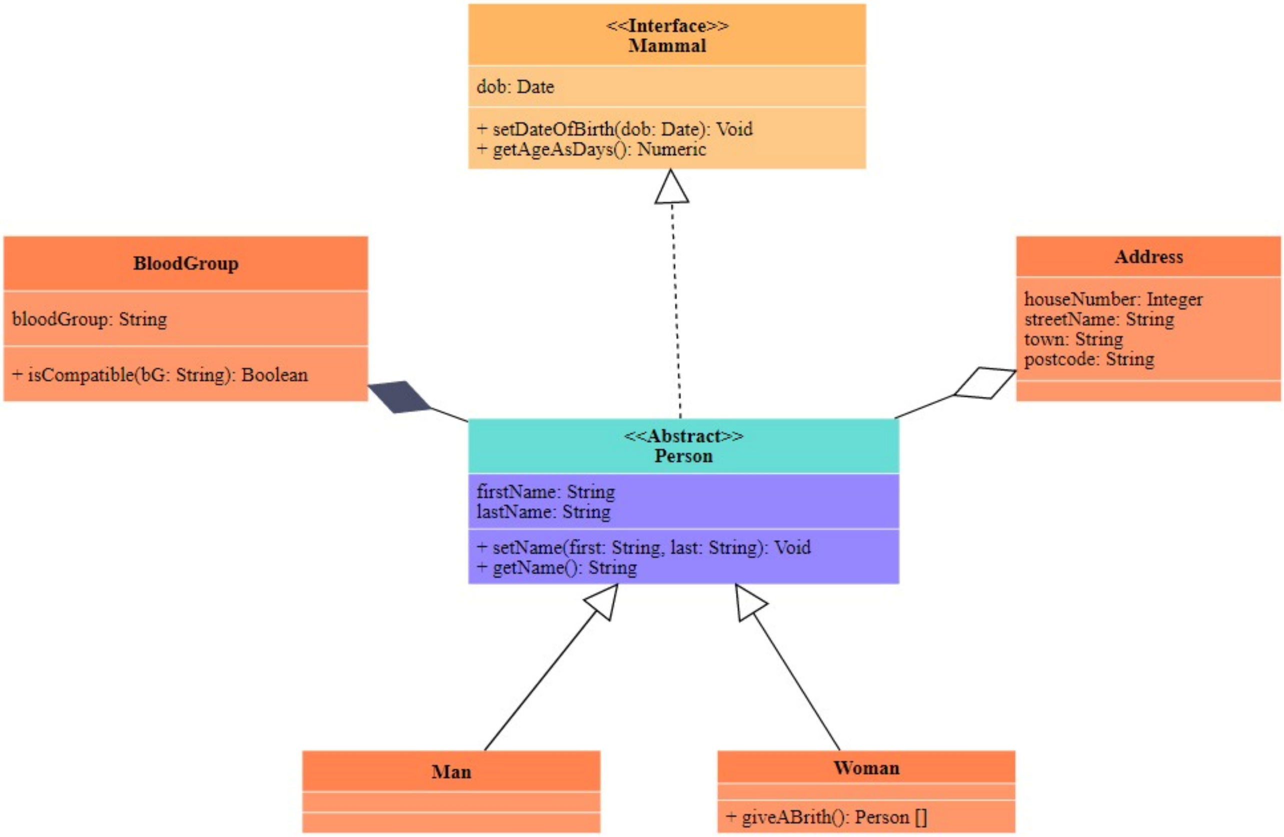


Figure 23 JointJS-based UML class diagram

The directed graph layout example for JointJS[15] demonstrates configurable layout optimization parameters. Such parameters could be useful to address layout flaws that are specific to certain kinds of topologies in the future.

The question-answer dialog generator example for JointJS[16] demonstrates, among other aspects, the dynamic addition and removal of ports and text entries on the same level. This capability should be useful to render stack frame structures that grow and shrink over the course of navigation back and forth through debugging breakpoints.

[14] https://resources.jointjs.com/demos/umlcd
[15] https://resources.jointjs.com/demos/directed-graph
[16] https://resources.jointjs.com/demos/qad

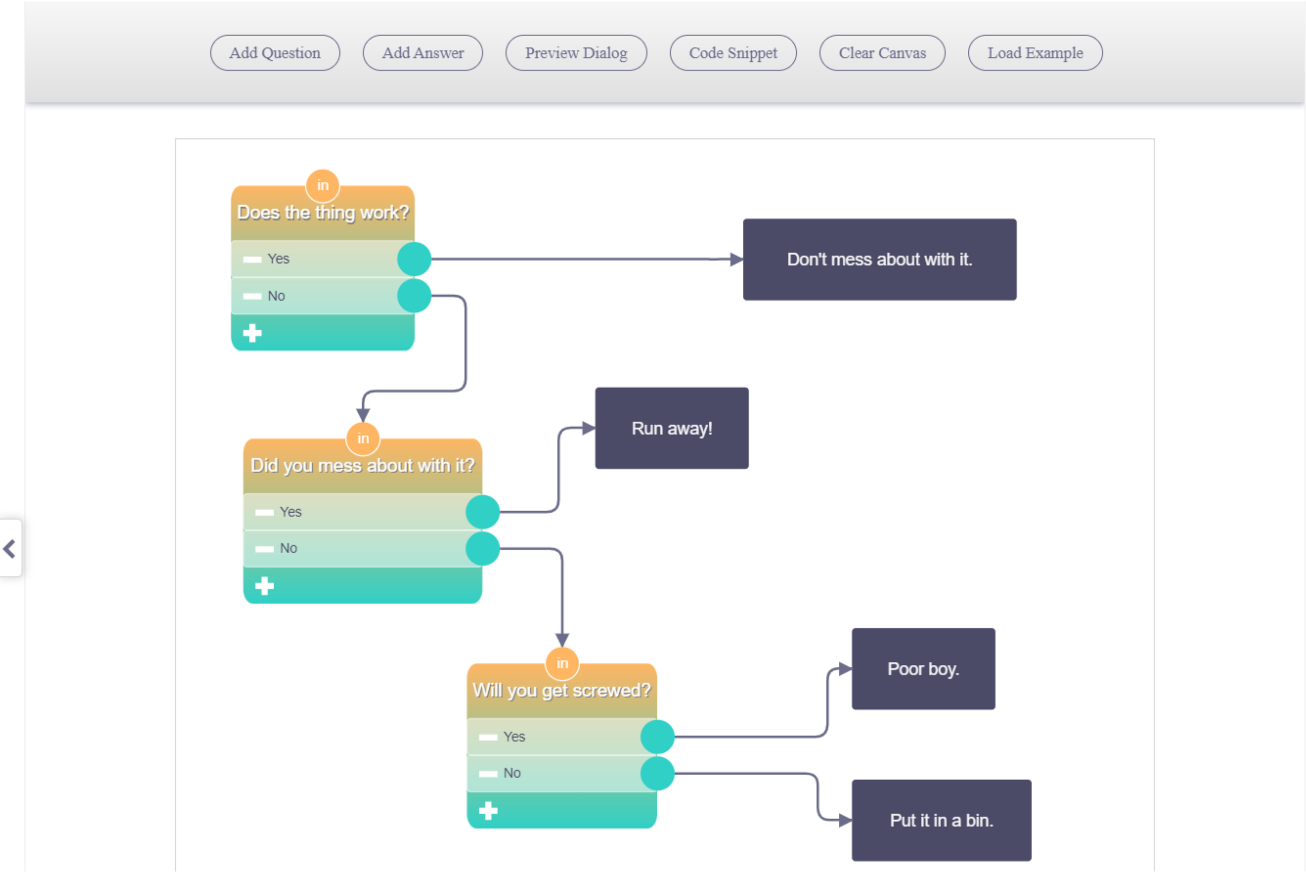
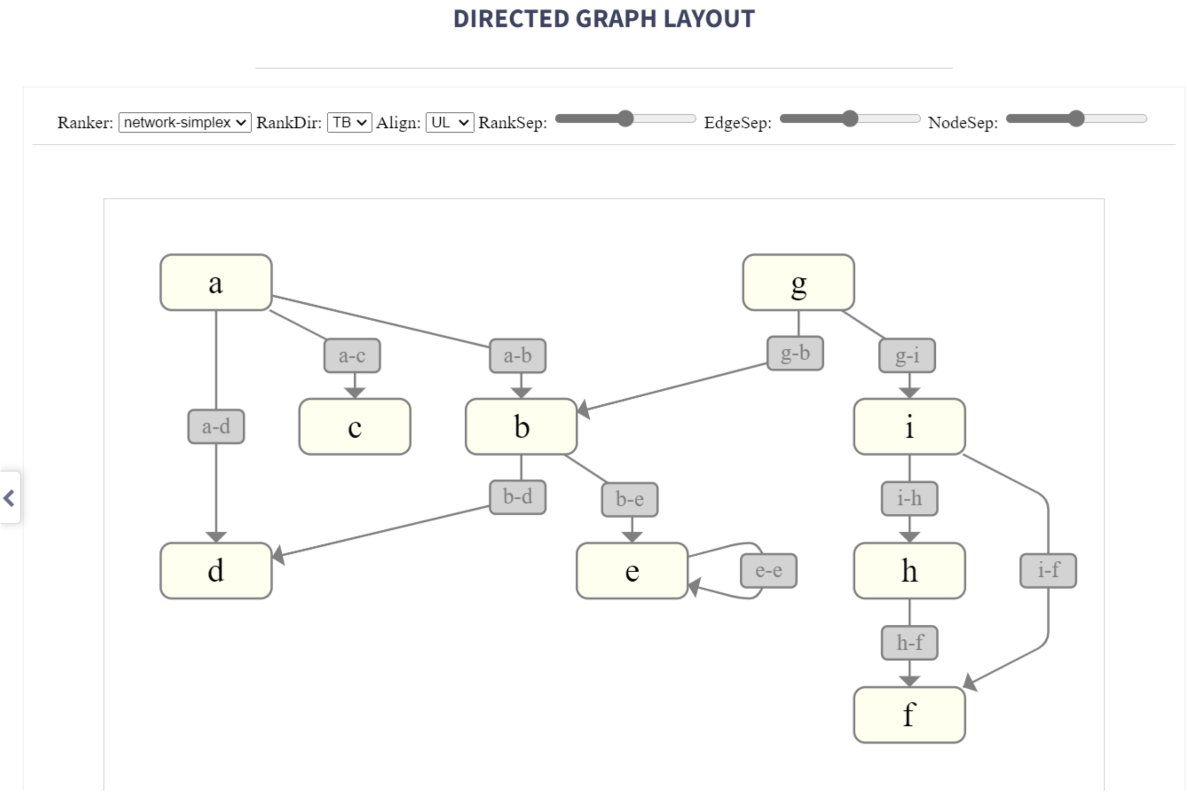


Figure 24 JointJS question-answer dialog generator example



Ranker: network-simplex RankDir: TB Align: UL RankSep: EdgeSep: NodeSep:

Figure 25 JointJS directed graph example with configurable layout optimization parameters

Among the features advertised on the official JointJS website the following were considered the most relevant and compelling by the project team:

Feature and Description	JointJS	JointJS+	Assessment
Custom Shapes Define custom and interactive shapes using a combination of SVG and JointJS/JointJS+ APIs that ease creation of these shapes (elements or links with optional labels).	✓	✓	We assume that some adjustments will be necessary or at least useful. For example, we might want to hide the methods section of the UML built-in class shape to show a stack frame or object structure.
Built-In Shapes Ready-to-use set of built-in shapes for most popular diagrams (rectangles, ovals, lines, ERD, state machines, logic, ORG, …). Use them as-is or customize to suit your need.	✓	✓	Some of the built-in shapes (such as the UML Class diagram class) could probably be used with minor modifications.
Support for Ports Easy API for adding, removing, updating, and connecting ports. Ports can be added to any shape.	✓	✓	We want to use a dedicated ingress port per structure and an egress port for each structure field to achieve a good level of readability (see Figure 22 and Figure 24).
Geometry Math Geometry API, providing a rich set of functions that deal with math in 2D space (rectangles, lines, curves, ovals, points, …).	✓	✓	Some more complex calculations are probably needed to replace features missing in the free JointJS version.
Interactive Diagrams Interactive shapes and links (moving, rotation, linking, …).	✓	✓	We need a high level of interactivity in order to enable manual post-processing by the users of our tool.
Rich Set of Events React on anything that happens in your diagrams (movement, property changes, structural changes, …) and run custom code.	✓	✓	A rich set of events is highly welcome as we would like to achieve a level of interaction like the vis.js-based debugger panel.
Automatic Layouts Automatically layout your diagrams in a tree, grid or any directed or undirected graph.	✗	✓	We can use the dagre[17] library to work around this problem.
Zoom and Pan Zoom and pan your diagrams using animation transitions.	✗	✓	Zooming and panning can be recreated manually.

Table 25 The most relevant and compelling JointJS features and their coverage by JointJS and JointJS+

[17] (dagre GitHub, 2022)

10.2.1 Limitations

- Not all operations can be executed by the JointJS debugger panel. GIF and WebM recordings usually involve a canvas element that can be recorded. JointJS uses SVG elements for rendering.
- While the features offered by JointJS+ are very attractive, the licensing fees are too high for this open-source project. The lowest price is € 2,963.85 for 3 years.

10.2.2 Decision

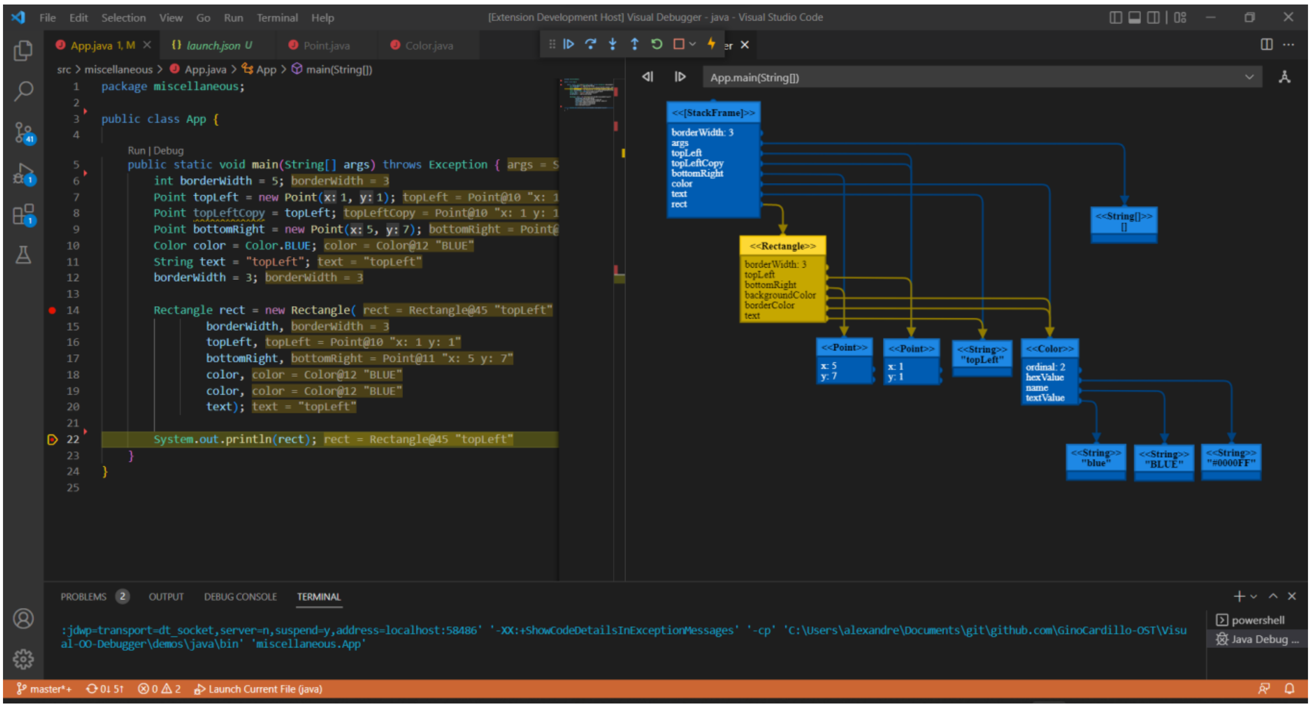
- Get the most out of JointJS.
- Use dagre to arrange structures properly. After some trial and error, we decided to use the following parameters (among others) for maximum readability and minimum edge crossings:

Parameter	Value	Description	Interpretation
`layout.DirectedGraph.LayoutOptions.align`	`'UL'`	Alignment	Upper-left
`layout.DirectedGraph.LayoutOptions.rankDir`	`'TB'`	Rank direction	Top-to-bottom
`layout.DirectedGraph.LayoutOptions.ranker`	`'network-simplex'`	Ranker type	Network-simplex ranking algorithm
`dia.Link.GenericAttributes.router.name`	`'manhattan'`	Router type	Manhattan-style routing algorithm
`routers.ManhattanRouterArguments.excludeEnds`	`['source', 'target']`	Ends to be excluded from routing	Preserves start and end directions
`routers.ManhattanRouterArguments.startDirections`	`['right']`	Egress connection direction	Egress connections go to the right
`routers.ManhattanRouterArguments.endDirections`	`['top']`	Ingress connection direction	Ingress connections come from the top
`dia.Link.GenericAttributes.connector.name`	`'rounded'`	Connector type	Rounded (but orthogonal) connectors

Table 26 Layout, router and connector parameters

10.2.3 Implementation

- The built-in UML class shape is extended and used to visualize objects and stack frames.
- The dagre library is used to arrange structures properly. The layout, router and connector parameters were used according to our decision.



```
package miscellaneous;

public class App {

    public static void main(String[] args) throws Exception { args = S
        int borderWidth = 5; borderWidth = 3
        Point topLeft = new Point(x: 1, y: 1); topLeft = Point@10 "x: 1
        Point topLeftCopy = topLeft; topLeftCopy = Point@10 "x: 1 y: 1
        Point bottomRight = new Point(x: 5, y: 7); bottomRight = Point@
        Color color = Color.BLUE; color = Color@12 "BLUE"
        String text = "topLeft"; text = "topLeft"
        borderWidth = 3; borderWidth = 3

        Rectangle rect = new Rectangle( rect = Rectangle@45 "topLeft"
                borderWidth, borderWidth = 3
                topLeft, topLeft = Point@10 "x: 1 y: 1"
                bottomRight, bottomRight = Point@11 "x: 5 y: 7"
                color, color = Color@12 "BLUE"
                color, color = Color@12 "BLUE"
                text); text = "topLeft"

        System.out.println(rect); rect = Rectangle@45 "topLeft"
    }
}
```

```
:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:58486' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\alexandre\Documents\git\github.com\GinoCardillo-OST\Visu
al-OO-Debugger\demos\java\bin' 'miscellaneous.App'
```

Figure 26 JointJS visualization

10.3 REQ-BA:7 Show objects that are no longer referenced.

In the SA version of VOOD, objects are not rendered when they are no longer referenced. A feature should be implemented so these objects stay visible.

10.3.1 Limitations

The Debug Adapter Protocol provides us with the necessary data for the diagrams but does not deliver objects without a reference. It would be possible to save an object without references locally and render them in the diagram. But without access to the garbage collector, it would be impossible to know how long this object should be shown in the diagram.

10.3.2 Decision

This feature will not be implemented.

10.4 REQ-BA:8 Connection with the editor: Click on the variable in the visualization, the variable is highlighted in the source code.

To help students mentally link the rendered diagram with the corresponding source code, a feature was requested where a user could click on variables in the diagram which is then highlighted in the code.

10.4.1 Limitations

This feature depends on values which need to be provided by the underlying Java environment.

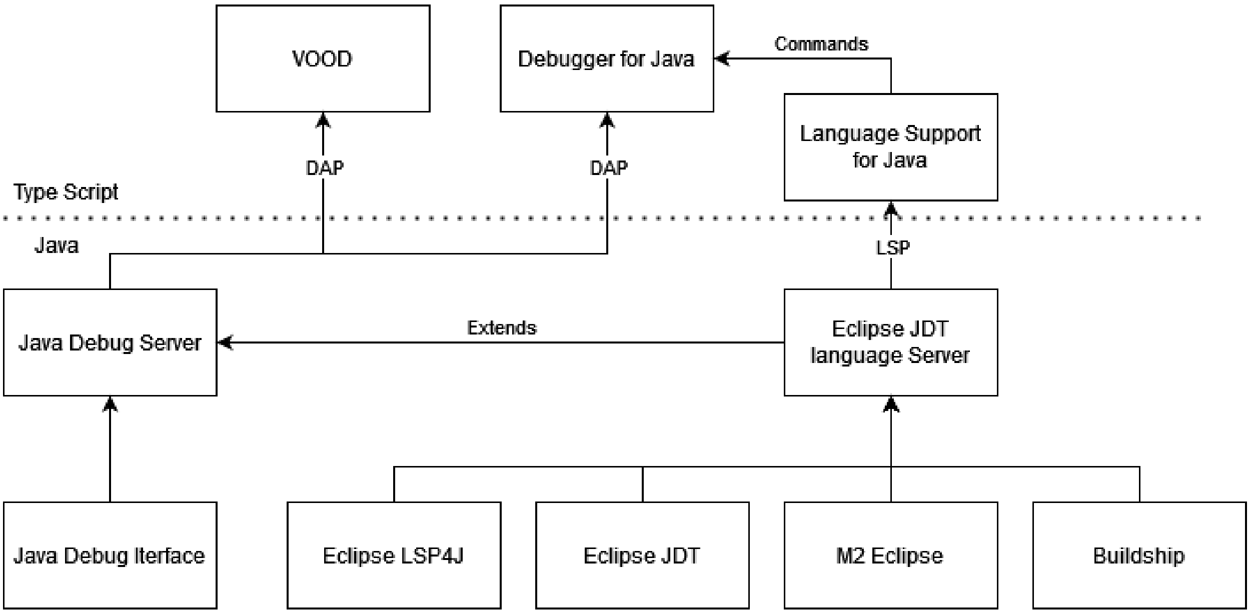


Figure 27 Underlying Java environment

The system consists of multiple Java components that run the source code. The “Eclipse JDT language Server” provides the LSP protocol, which is used by the “Language Support for Java” VS Code extension to offer the use of Java in VS Code. An extension of the “Eclipse JDT language Server” is the “Java Debug Server”. It implements the DAP, which is used by VOOD and the “Debugger for Java” extension.

To implement the requested feature, a reference from a variable to the source code is needed. The DAP has an optional field that could contain such a value, but it is left undefined by the Java Debug Server.

The LSP, on the other hand, was developed to offer support for new languages. Microsoft advises against the use for minor features.

“…In general, it is advised that LSP language server extensions be used for providing new language experiences, not extending existing ones.”[18]

10.4.2 Decision

Because of the limitation of the provided protocols, it was decided not to implement this feature.

[18] (Microsoft, 2022)

10.5 REQ-SA:3.5 Usability [of the system].

In the SA version of VOOD, the vis.js visualization was rather static. The user could change the position of the rendered nodes and step back to older states, but it was not possible, for example, to hide certain nodes.

During testing of the vis.js visualization it became clear that some additional features, to customize the vis.js graph, would increase the usability of the extension.

This chapter also includes the requirements REQ-BA:9.1 and REQ-BA:9.2 both of which are also concerned with improving the usability of the vis.js solution. These requests are rather minor and are therefore bundled here together under "Change visualization for variables"

10.5.1 Limitations

The features are mostly concerned about the visualization. Therefore, they depend on features provided by the vis.js library.

10.5.2 Decision

The following features were implemented to give the user more control over the visualization.

Customizable colours

To make it easier to use an exported graph in other documents, the colours used in the graph should be customizable. The following settings should therefore be added:

- Default node colour
- Default variable colour
- Changed node colour
- Changed variable colour

The other colours used, for example the font colour or the border colour, will be calculated using the user selected colours.

Make PNG and GIF exports accessible via editor menu

To make it easier to access the GIF export and the PNG export, they were added to the editor menu.

Expand/Collapse nodes

The possibility to group certain nodes in the diagram was already discussed during the SA.

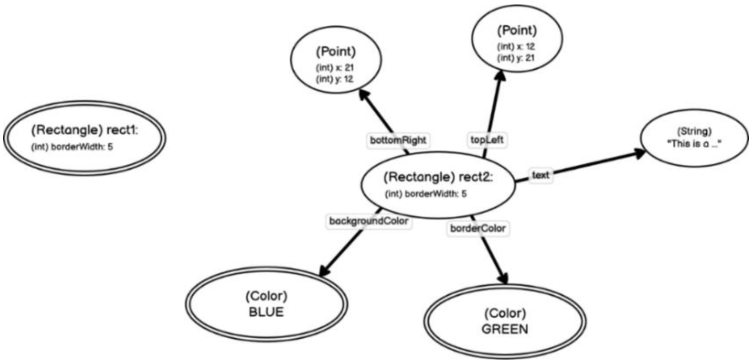


Figure 28 Expand/collapse nodes wireframe

When a node is clicked, all child nodes should be hidden and the node itself is visually marked with a bold border. If the node is clicked again, all nodes become visible again.

Hide Nodes/Edges for export

A feature was requested to remove nodes from the visualization.

Change visualization for variables

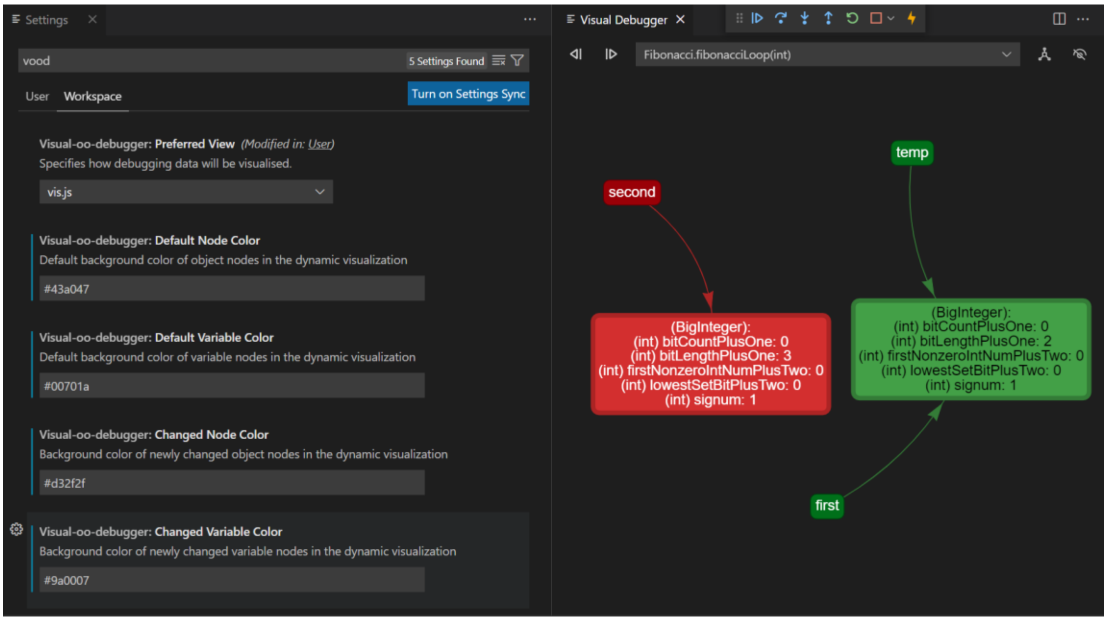
Variables and objects are rendered similarly in the SA version of VOOD. To better highlight the difference of these two elements, it was requested that variables have a different colour then objects.

10.5.3 Implementation

All those features were implemented.

Customizable colours

Options to set the required colours were added to the settings of the extension. The PanelView proxy interface was extended to use these new colours. New functions were implemented to incorporate the rendering in the vis.js visualization.


Settings
vood
5 Settings Found
User
Workspace
Turn on Settings Sync
Visual Debugger
Fibonacci.fibonacciLoop(int)
second
temp
first
(BigInteger):
(int) bitCountPlusOne: 0
(int) bitLengthPlusOne: 3
(int) firstNonzeroIntNumPlusTwo: 0
(int) lowestSetBitPlusTwo: 0
(int) signum: 1
(BigInteger):
(int) bitCountPlusOne: 0
(int) bitLengthPlusOne: 2
(int) firstNonzeroIntNumPlusTwo: 0
(int) lowestSetBitPlusTwo: 0
(int) signum: 1

Visual-oo-debugger: Preferred View *(Modified in: User)*
Specifies how debugging data will be visualised.
vis.js

Visual-oo-debugger: Default Node Color
Default background color of object nodes in the dynamic visualization
#43a047

Visual-oo-debugger: Default Variable Color
Default background color of variable nodes in the dynamic visualization
#00701a

Visual-oo-debugger: Changed Node Color
Background color of newly changed object nodes in the dynamic visualization
#d32f2f

Visual-oo-debugger: Changed Variable Color
Background color of newly changed variable nodes in the dynamic visualization
#9a0007

Figure 29 Customizable colors

Make PNG and GIF exports accessible via editor menu

The new commands where registered in the package.json of the extension to instruct VS Code to offer these options in the editor menu.

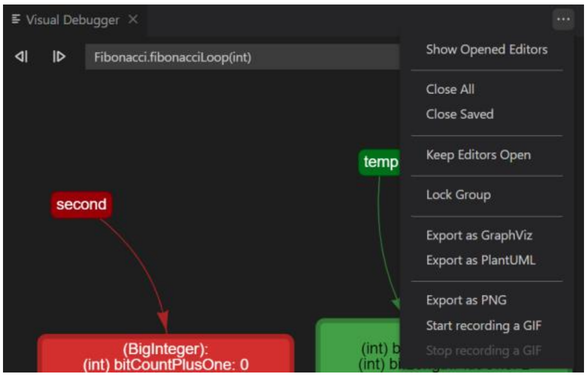


Figure 30 Editor menu with export options

Expand/Collapse nodes

The feature was implemented according to the wireframe. An additional icon was added to give the user the option to open all clusters at once.

When a user clicks a node, a cluster node is created with the same content as the original node. The id of the cluster is prefixed to differentiate normal nodes from clusters. The cluster contains all referenced nodes of the original nodes as well as their referenced nodes and so on. All nodes, that are contained in the cluster, are removed.

To visually differentiate the clusters from normal nodes, clusters have an increased border width and references to a cluster have a dashed arrow.

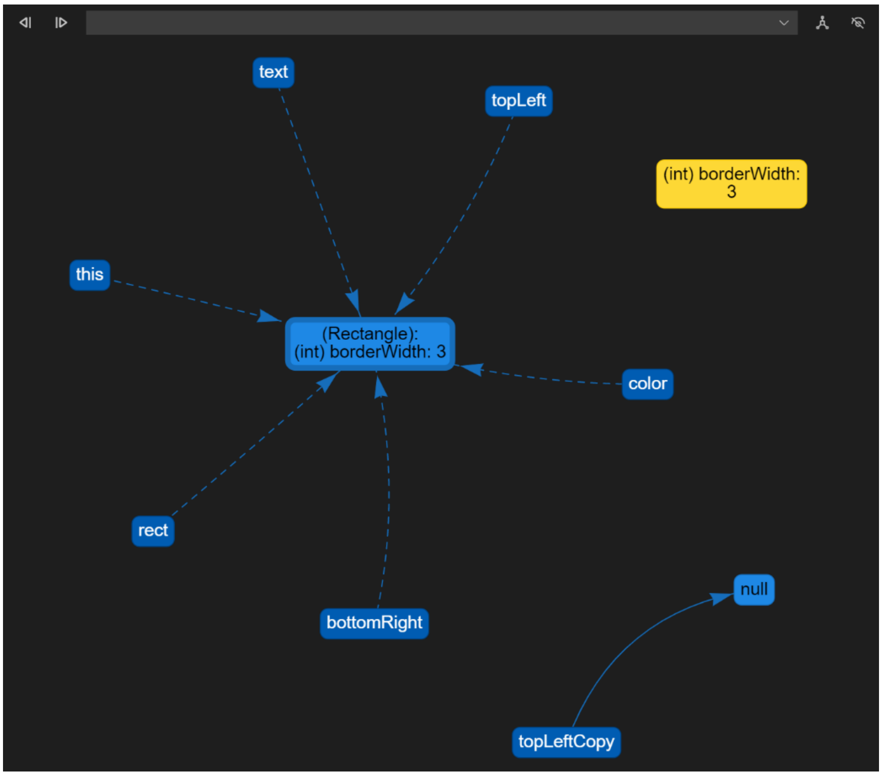


Figure 31 Collapsed node “(Rectangle)”

Hide Nodes/Edges for export

The hide function is controlled by the new eye icon in the toolbar. If a node or cluster is dragged onto the icon, it will be removed from the visualization. When the eye icon is clicked, it will reveal all the hidden nodes and clusters.

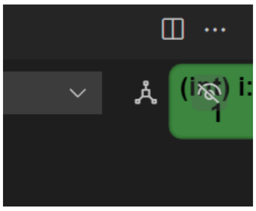


Figure 32 Hide nodes

Hidden clusters will not be opened by the button to open all clusters.

Change visualization for variables

A new colour was introduced to better distinguish between variables and objects. Furthermore, when an existing variable is assigned to a new object the variable node will be marked as changed. Previously only the corresponding edge was marked.

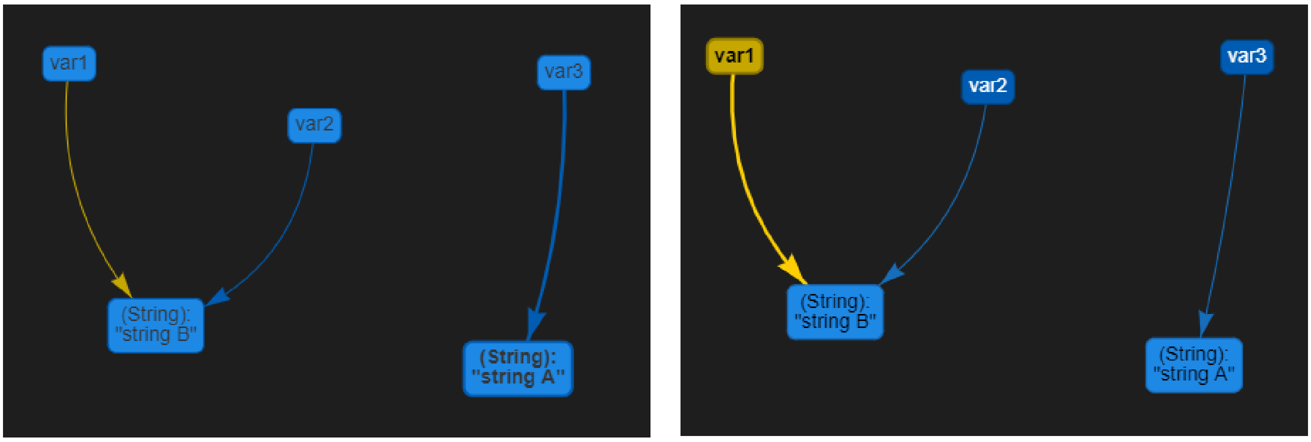


Figure 33 Visualization of changed references for old and new version of VOOD

10.6 REQ-SA:4.3 Source code that is open to extensions.

To make it possible to extend VOOD, certain changes to the code were necessary.

Additionally, since VOOD will be an open-source project, information and guidance for new contributors were added.

10.6.1 Limitations

There were no limitations for these changes.

10.6.2 Decision

The following decisions were made to make the code more open to extensions.

Add Contribution Information

Information on how to contribute to the project and guidelines need to be in place and documented.

Extract Java specific parts

The current version of VOOD only supports Java, and Java specific code is intertwined with the rest of the debugger. These specific parts include:

- Information on which are the primitive types
- What the name of the string class is (in Java it is “String”, in other languages it is “string”) since strings receive special treatment
- How to convert the debugger information into our intermediate data structure

This change is necessary to support more languages in the future.

10.6.3 Implementation

This section describes how those decisions were implemented.

Add Contribution Information

A contributing.md was added to the repository, as well as a code-of-conduct.md.

The contributing.md contains all the necessary information to contribute to the project. This includes information on how to ...

- create an issue,
- open a pull request,
- set up the IDE,
- add and run tests,
- ensure a high code quality by using our code quality tools,
- document changes.

We also added templates for bug and feature request issues and templates for pull requests to further assist a contributor.

As the code of conduct, we used the well-known Contributor Covenant Code of Conduct[19].

[19] (Contributor Covenant Code of Conduct, 2022)

Extract Java specific parts

Implementing the strategy design pattern, we created the AbstractDataExtractor and its specification for Java, the JavaDataExtractor.

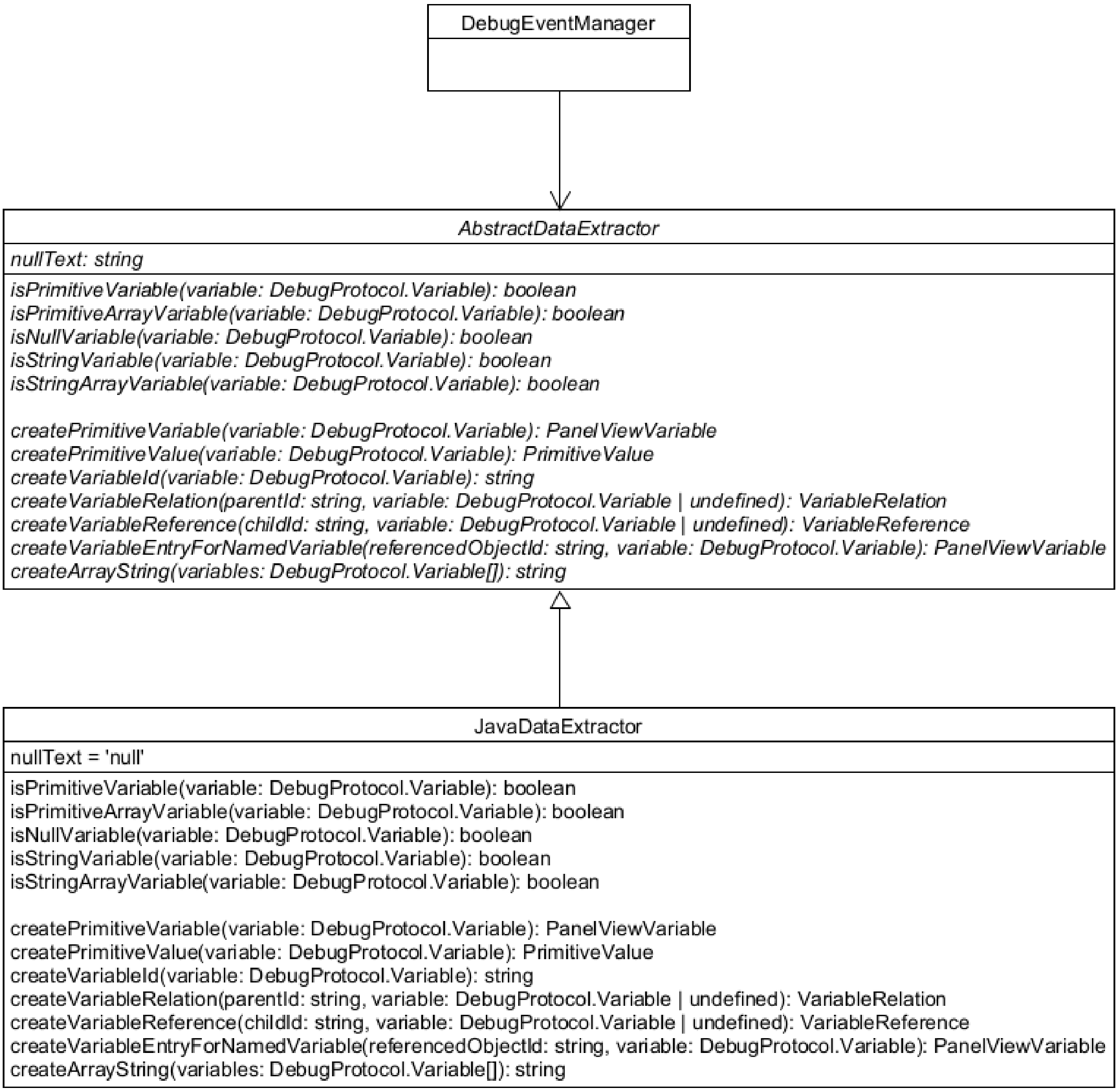

DebugEventManager
AbstractDataExtractor
nullText: string
isPrimitiveVariable(variable: DebugProtocol.Variable): boolean
isPrimitiveArrayVariable(variable: DebugProtocol.Variable): boolean
isNullVariable(variable: DebugProtocol.Variable): boolean
isStringVariable(variable: DebugProtocol.Variable): boolean
isStringArrayVariable(variable: DebugProtocol.Variable): boolean
createPrimitiveVariable(variable: DebugProtocol.Variable): PanelViewVariable
createPrimitiveValue(variable: DebugProtocol.Variable): PrimitiveValue
createVariableId(variable: DebugProtocol.Variable): string
createVariableRelation(parentId: string, variable: DebugProtocol.Variable | undefined): VariableRelation
createVariableReference(childId: string, variable: DebugProtocol.Variable | undefined): VariableReference
createVariableEntryForNamedVariable(referencedObjectId: string, variable: DebugProtocol.Variable): PanelViewVariable
createArrayString(variables: DebugProtocol.Variable[]): string
JavaDataExtractor
nullText = 'null'
isPrimitiveVariable(variable: DebugProtocol.Variable): boolean
isPrimitiveArrayVariable(variable: DebugProtocol.Variable): boolean
isNullVariable(variable: DebugProtocol.Variable): boolean
isStringVariable(variable: DebugProtocol.Variable): boolean
isStringArrayVariable(variable: DebugProtocol.Variable): boolean
createPrimitiveVariable(variable: DebugProtocol.Variable): PanelViewVariable
createPrimitiveValue(variable: DebugProtocol.Variable): PrimitiveValue
createVariableId(variable: DebugProtocol.Variable): string
createVariableRelation(parentId: string, variable: DebugProtocol.Variable | undefined): VariableRelation
createVariableReference(childId: string, variable: DebugProtocol.Variable | undefined): VariableReference
createVariableEntryForNamedVariable(referencedObjectId: string, variable: DebugProtocol.Variable): PanelViewVariable
createArrayString(variables: DebugProtocol.Variable[]): string

Figure 34 Data extractor class diagram

At the start of a debugging session, the DebugEventManager identifies the language of the application and uses the corresponding data extractor. If the language is anything other than Java, the extension will throw an error.

10.7 R:6 API Deprecations by VS Code

As stated in the risk analysis, VOOD is susceptible to changes in VS Code.

During the BA this risk occurred once. VS Code changed the web views so that SharedArrayBuffer is no longer supported. This change was addressed in the issue 116715 of the VS Code GitHub page[20]. The FFmpeg[21] library, used for the GIF export, relies on the SharedArrayBuffer and is therefore no longer working.

10.7.1 Decision

To fix the GIF export it was decided to use a different library. The original GIF export records a WebM file and converts it to a GIF after the recording.

10.7.2 Implementation

The reimplementation of this feature was time consuming because of the limitation of the webview environment and limitations of the potential replacement libraries. Many libraries were evaluated. The export was reimplemented using the gif-encoder-2[22] library.

An additional option for the WebM export was added.

10.8 R:7 Incompatible 3rd party extensions

Changes to the Debugger for Java extension led to an error in the visualization of objects. Primitive values were not included in the object but rendered as separate nodes in the graph.

10.8.1 Implementation

The error could be fixed with some minor changes in the DebugEventManager.

[20] (VS Code community, 2022)
[21] (FFmpeg, 2022)
[22] (gif-encoder-2, 2022)

11 Conclusion

This chapter contains an evaluation of the project as well as an outlook for further work on the Visual OO Debugger.

11.1 Overview of Changes for the BA

The overall structure of the BA as depicted in the Chapters 9.1 and 9.2 hasn't changed. But there were substantial changes inside each building block to facilitate the new functionalities and improve the overall code quality.

11.1.1 White Box DebugEventManager

The biggest change since the SA is the separation of Java specific code, which led to the addition of the abstract class "AbstractDataExtractor" and the corresponding implementation for Java in "JavaDataExtractor". Another notable change is the new function getStackFrameData in the DebugEventManager class to load data for specific stack frames, as required by REQ-BA:5.

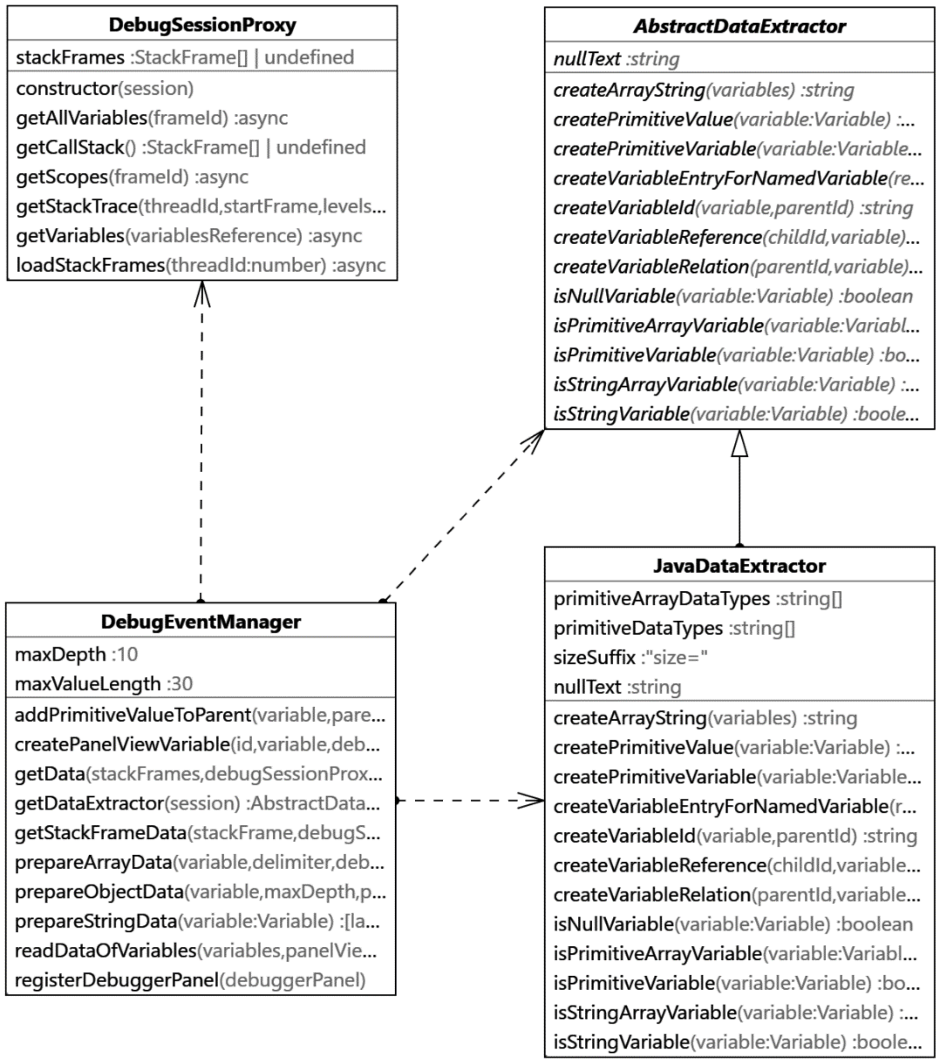

DebugSessionProxy
stackFrames :StackFrame[] | undefined
constructor(session)
getAllVariables(frameId) :async
getCallStack() :StackFrame[] | undefined
getScopes(frameId) :async
getStackTrace(threadId,startFrame,levels...
getVariables(variablesReference) :async
loadStackFrames(threadId:number) :async
AbstractDataExtractor
nullText :string
createArrayString(variables) :string
createPrimitiveValue(variable:Variable) :...
createPrimitiveVariable(variable:Variable...
createVariableEntryForNamedVariable(re...
createVariableId(variable,parentId) :string
createVariableReference(childId,variable)...
createVariableRelation(parentId,variable)...
isNullVariable(variable:Variable) :boolean
isPrimitiveArrayVariable(variable:Variabl...
isPrimitiveVariable(variable:Variable) :bo...
isStringArrayVariable(variable:Variable) :...
isStringVariable(variable:Variable) :boole...
JavaDataExtractor
primitiveArrayDataTypes :string[]
primitiveDataTypes :string[]
sizeSuffix :"size="
nullText :string
createArrayString(variables) :string
createPrimitiveValue(variable:Variable) :...
createPrimitiveVariable(variable:Variable...
createVariableEntryForNamedVariable(r...
createVariableId(variable,parentId) :string
createVariableReference(childId,variable...
createVariableRelation(parentId,variable...
isNullVariable(variable:Variable) :boolean
isPrimitiveArrayVariable(variable:Variabl...
isPrimitiveVariable(variable:Variable) :bo...
isStringArrayVariable(variable:Variable) :...
isStringVariable(variable:Variable) :boole...
DebugEventManager
maxDepth :10
maxValueLength :30
addPrimitiveValueToParent(variable,pare...
createPanelViewVariable(id,variable,deb...
getData(stackFrames,debugSessionProx...
getDataExtractor(session) :AbstractData...
getStackFrameData(stackFrame,debugS...
prepareArrayData(variable,delimiter,deb...
prepareObjectData(variable,maxDepth,p...
prepareStringData(variable:Variable) :[la...
readDataOfVariables(variables,panelVie...
registerDebuggerPanel(debuggerPanel)

Figure 35 Class diagram of DebugEventManager after BA

11.1.2 White Box PanelView

A new abstract class was created for the PanelViewProxy. This was done to separate the vis.js specific code from more general implementations. It is noticeable that all classes received additional functions to provide the new functionalities added during the BA.

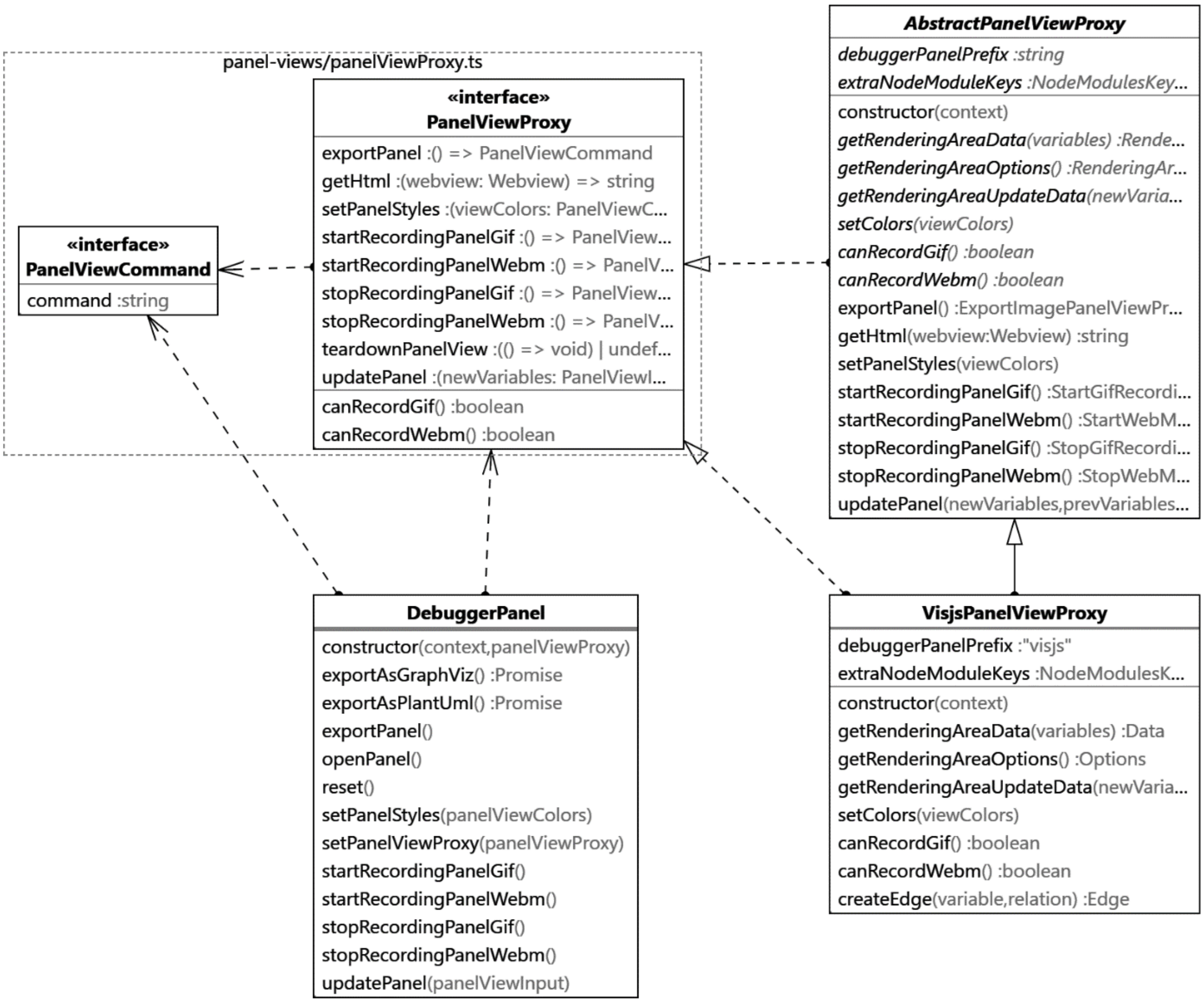

panel-views/panelViewProxy.ts
«interface»
PanelViewProxy
exportPanel :() => PanelViewCommand
getHtml :(webview: Webview) => string
setPanelStyles :(viewColors: PanelViewC...
startRecordingPanelGif :() => PanelView...
startRecordingPanelWebm :() => PanelV...
stopRecordingPanelGif :() => PanelView...
stopRecordingPanelWebm :() => PanelV...
teardownPanelView :(() => void) | undef...
updatePanel :(newVariables: PanelViewI...
canRecordGif() :boolean
canRecordWebm() :boolean
«interface»
PanelViewCommand
command :string
AbstractPanelViewProxy
debuggerPanelPrefix :string
extraNodeModuleKeys :NodeModulesKey...
constructor(context)
getRenderingAreaData(variables) :Rende...
getRenderingAreaOptions() :RenderingAr...
getRenderingAreaUpdateData(newVaria...
setColors(viewColors)
canRecordGif() :boolean
canRecordWebm() :boolean
exportPanel() :ExportImagePanelViewPr...
getHtml(webview:Webview) :string
setPanelStyles(viewColors)
startRecordingPanelGif() :StartGifRecordi...
startRecordingPanelWebm() :StartWebM...
stopRecordingPanelGif() :StopGifRecordi...
stopRecordingPanelWebm() :StopWebM...
updatePanel(newVariables,prevVariables...
DebuggerPanel
constructor(context,panelViewProxy)
exportAsGraphViz() :Promise
exportAsPlantUml() :Promise
exportPanel()
openPanel()
reset()
setPanelStyles(panelViewColors)
setPanelViewProxy(panelViewProxy)
startRecordingPanelGif()
startRecordingPanelWebm()
stopRecordingPanelGif()
stopRecordingPanelWebm()
updatePanel(panelViewInput)
VisjsPanelViewProxy
debuggerPanelPrefix :"visjs"
extraNodeModuleKeys :NodeModulesK...
constructor(context)
getRenderingAreaData(variables) :Data
getRenderingAreaOptions() :Options
getRenderingAreaUpdateData(newVaria...
setColors(viewColors)
canRecordGif() :boolean
canRecordWebm() :boolean
createEdge(variable,relation) :Edge

Figure 36 Class diagram of PanelViewProxy after BA

11.2 Usability Test

Some quality goals for VOOD are dependent on the subjective perception of a regular user. Especially the following requirements can only be judged by an average user.

- REQ-SA 1.1.2 The visual debugger is intended to support students in object-oriented programming.
- REQ SA 2.2.1 The visual debugger should make it possible to understand how objects change over the course of the program
- REQ SA 2.2.2 The VOOD should make it possible to understand how variables change over the course of the program.
- REQ SA 3.5 Another goal of the project is usability [of the system].

To confirm that these quality goals are met, a usability test was conducted.

The test consisted of one user that was tasked to solve exercises from OOP1 and a custom-made exercise which involved analysing an existing project. An examiner observed how VOOD is used for debugging and understanding the given project during the exercise.

The key findings of the test were:

- VOOD is helpful for debugging and understanding existing code.
- VOOD is easy to use and quickly adopted by a new user
- Some minor hurdles in the UI were found. It was noticed that, especially during the start-up of VOOD, more guidance was needed.
- The user wished for more connectiveness between the diagram and the code. For example, it was wished that variables could be changed in the diagram and then automatically updated in the code.

These findings led to the inclusion of additional help texts in VOOD. The wish for more connectivity between the diagram and the code was already discussed for requirement REQ-BA:8.

11.3 Target Achievement

The target achievement is evaluated by looking at each of the requirements defined in section 1.1. The evaluation states if the requirement was achieved and on how the fulfilment of the requirement was verified.

Requirement	Achieved	Verified by
REQ-SA:1.1.1	✓	Mirko Stocker as Product Owner and OO-lecturer tested the application during the development and provided feedback.
REQ-SA:1.1.2	✓	The Usability test conducted during the BA confirmed the fulfilment of this requirement.
REQ-SA:1.2	✓	As a functional requirement is the fulfilment given when it is implemented.
REQ-SA:2.1.1.1	✓	As a functional requirement is the fulfilment given when it is implemented.
REQ-SA:2.1.1.2	✓	As a functional requirement is the fulfilment given when it is implemented.
REQ-SA:2.1.2	✓	As a functional requirement is the fulfilment given when it is implemented.
REQ-SA:2.2.1	✓	The Usability test conducted during the BA confirmed the fulfilment of this requirement.
REQ-SA:2.2.2	✓	The Usability test conducted during the BA confirmed the fulfilment of this requirement
REQ-SA:3.1	✓	The Visual OO Debugger is a VS Code extension which requires no further setup other than installing it. Since GitPod also uses VS Code extensions, it is also possible to use VOOD with GitPod.
REQ-SA:3.2	✓	The import of a program depends on the IDE in use. In the case of VS Code, the code needs to be on the file system and can be easily imported using VS Code. In the case of GitPod, a program can be imported by creating a connection to a git repository.
REQ-SA:3.3	✓	As a functional requirement is the fulfilment given when it is implemented.
REQ-SA:3.4	✓	Since GitPod uses the same extensions as VS Code, the Visual OO Debugger can be use with GitPod as well.
REQ-SA:3.5	✓	The usability test conducted during the BA confirmed the fulfilment of this requirement
REQ-SA:4.1	✓	The source code is kept as simple as possible, and comments are added where necessary.
REQ-SA:4.2	✓	The source code is well structured.
REQ-SA:4.3	✓	The source code is modular, which allows for easier extensions.
REQ-BA:5	✓	As a functional requirement is the fulfilment given when it is implemented.

REQ-BA:6	✓	As a functional requirement is the fulfilment given when it is implemented.
REQ-BA:7	×	This requirement was dropped after it was deemed impossible to implement.
REQ-BA:8	×	This requirement was dropped after it was deemed impossible to implement.
REQ-BA:9.1	✓	As a functional requirement, the fulfilment is given when it is implemented.
REQ-BA:9.2	✓	As a functional requirement, the fulfilment is given when it is implemented.

Table 27 Target Achievement

11.4 Outlook

The development of the Visual OO Debugger is far from over. It will be continued as an open-source project. Soon after the bachelor thesis, the GitHub repository will be transferred to an organization managed by OST. This is necessary because the repository is currently owned by a team member, who will not have access to the account anymore after his studies.

As part of the bachelor thesis, preparations were made to support other languages, in addition to Java. Support for additional languages is planned soon.

Indices

Indices over the tables and images are contained within this part.

12 Glossary

Term	Definition
AD	Algorithms and Data structures (a course at OST)
API	Application Programming Interface
BA	Bachelorarbeit (bachelor thesis)
CI/CD	Continuous Integration/Continuous Deployment
FOSS	Free and Open-Source Software
IDE	Integrated Development Environment
MVP	Minimum Viable Product
OO	Object-Oriented
OOP1	Object-Oriented Programming 1 (a course at OST)
OST	Ostschweizer Fachhochschule
PF	Patterns and Frameworks (a course at OST)
SA	Studienarbeit (term project)
VOOD	Visual OO Debugger
VS Code	Visual Studio Code

Table 28 Glossary

13 List of Figures

Figure 1 JointJS visualization 1
Figure 2 Mind map of the SA requirements 7
Figure 3 Mind map of the BA requirements 9
Figure 4 Quality tree 19
Figure 5 Risk matrix 29
Figure 6 Wireframe of VS Code integration 30
Figure 7 SonarCloud summary 31
Figure 8 GitHub workflow overview 32
Figure 9 Context diagram 33
Figure 10 Class diagram of debug adapter protocol models 35
Figure 11 Class diagram of debug adapter protocol models 36
Figure 12 Component diagram of debug backend 38
Figure 13 Component diagram of webview 39
Figure 14 Class diagram of DebugEventManager 40
Figure 15 Class diagram of PanelView 41
Figure 16 Class diagram of PanelView variables 42
Figure 17 Visualization of the call stack over time 44
Figure 18 Wireframe for stack frame dropdown 45
Figure 19 Implemented stack frame dropdown 46
Figure 20 A vis.js-based graph 47
Figure 21 PlantUML export: Good symmetry, but inconsistent arrow directions 48
Figure 22 GraphViz export: Good symmetry, consistent arrow orientation, but more edge crossings 48
Figure 23 JointJS-based UML class diagram 49
Figure 24 JointJS question-answer dialog generator example 50
Figure 25 JointJS directed graph example with configurable layout optimization parameters 50
Figure 26 JointJS visualization 53
Figure 27 Underlying Java environment 55
Figure 28 Expand/collapse nodes wireframe 56
Figure 29 Customizable colors 57
Figure 30 Editor menu with export options 58
Figure 31 Collapsed node “(Rectangle)” 58
Figure 32 Hide nodes 59
Figure 33 Visualization of changed references for old and new version of VOOD 59
Figure 34 Data extractor class diagram 61
Figure 35 Class diagram of DebugEventManager after BA 63
Figure 36 Class diagram of PanelViewProxy after BA 64

14 List of Tables

Table 1 SA requirements ... 6
Table 2 BA requirements ... 8
Table 3 List of Stakeholders ... 11
Table 4 Stakeholder analysis ... 12
Table 5 Relation map of the stakeholders ... 13
Table 6 Existing comparable products ... 15
Table 7 Constraints ... 16
Table 8 Quality goals derived from quality requirements ... 18
Table 9 Quality scenario: OO lecturer prepares a course for a semester ... 20
Table 10 Quality scenario: Students analysing a sample code for the OO course ... 20
Table 11 Quality scenario: Student uses VOOD during an OO exercise ... 21
Table 12 Quality scenario: Student uses VOOD during an OO exercise ... 21
Table 13 Quality scenario: Student participates in their first exercise of the OO course and has no IDE installed for developing Java ... 22
Table 14 Quality scenario: User installs VOOD ... 22
Table 15 Quality scenario: User uses system to debug a simple solution ... 23
Table 16 Quality scenario: New developers want to contribute to VOOD extension ... 23
Table 17 Quality scenario: After the BA is finished, a feature of VS Code used by VOOD is no longer supported. A new developer is tasked with fixing the issue ... 24
Table 18 Quality scenario: After the bachelor thesis is finished, the need for a new visualization option arises ... 24
Table 19 Solution strategy ... 26
Table 20 Risk identification and management plan ... 27
Table 21 Simplified risk list ... 28
Table 22 Description of the business context ... 33
Table 23 Required API's and protocols ... 34
Table 24 Used VS Code API features ... 34
Table 25 The most relevant and compelling JointJS features and their coverage by JointJS and JointJS+ ... 51
Table 26 Layout, router and connector parameters ... 52
Table 27 Target Achievement ... 67
Table 28 Glossary ... 68

15 Bibliography

arc42. (2022). Retrieved from https://www.arc42.de/

Balsamiq. (2022). Retrieved from https://balsamiq.com/

BlueJ. (2022). Retrieved from https://www.bluej.org/

C4 model. (2021). Retrieved from https://c4model.com/

Cardillo, G., Lagadec, A., & Schürmann, P. (2022). *VOOD Term Project Documentation at OST.*

Contributor Covenant Code of Conduct. (2022). Retrieved from https://www.contributor-covenant.org/version/1/4/code-of-conduct/

dagre GitHub. (2022). Retrieved from https://github.com/dagrejs/dagre

Debug Adapter Protocol. (2022). Retrieved from https://microsoft.github.io/debug-adapter-protocol/specification

ESLint. (2022). Retrieved from https://eslint.org/

FFmpeg. (2022). Retrieved from https://ffmpeg.org/

gif-encoder-2. (2022). Retrieved from https://www.npmjs.com/package/gif-encoder-2

GitHub. (2022). Retrieved from https://github.com/

GitPod. (2022). Retrieved from https://www.gitpod.io/

ISO 25000 Portal. (2022). Retrieved from https://iso25000.com

JointJS. (2022). Retrieved from https://www.jointjs.com/

Microsoft. (2022). *Add a Language Server Protocol extension*. Retrieved from https://docs.microsoft.com/en-us/visualstudio/extensibility/adding-an-lsp-extension?view=vs-2022

OST - Ostschweizer Fachhochschule. (2022). Retrieved from https://www.ost.ch/

SonarCloud. (2022). Retrieved from https://sonarcloud.io/

Sophist GmbH. (2022). *sophist.de*. Retrieved from https://www.sophist.de/

vis.js. (2022). Retrieved from https://almende.github.io/vis/

Visual Studio Code. (2022). Retrieved from https://code.visualstudio.com/

VS Code API. (2022). Retrieved from https://code.visualstudio.com/api/references/vscode-api

VS Code community. (2022). *github.com*. Retrieved from https://github.com/microsoft/vscode/issues/116715