A Heat Map for Software Visualizations based on the City Metaphor

Master’s Thesis

Tim-Niklas Reck

June 2, 2020

KIEL UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
SOFTWARE ENGINEERING GROUP

Advised by: Prof. Dr. Wilhelm Hasselbring
            Alexander Krause, M.Sc.
Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel, 2. Juni 2020
Abstract

The importance of modern and efficient software applications rises with the advancing digitalization and emergence of new fields of application for software systems. However, with a growing number of legacy software systems and the creation of new applications, their complexity and quantity of computed data increase. Thus, the maintainability, performance, and comprehensibility of said software systems suffer, and the needs for performance engineering and monitoring emerge. Dynamic software metrics provide a way of evaluating the quality and efficiency of certain processes and states within software systems during runtime. With these abstractions the comprehensibility of applications is increased, and thus the overall quality of software can be improved.

The monitoring tool ExplorViz provides the capabilities to monitor and visualize large software landscapes consisting of multiple software systems. However, in the current version the number of visualized metrics is restricted by the structure of the visualization, which applies the city metaphor. With our approach we add an additional layer to the visualization, which displays additional metrics in the form of heatmaps on top of the existing visualization. This feature enables the user to select different metrics, heatmap types and visualization styles to customize the monitoring experience. Furthermore, we developed a data model that allows an effortless addition of own metrics to the heatmap.

For the evaluation of this approach we conducted a formative usability study which consisted of a moderated usability experiment and the inquiry of questions which were handed to the test users in the form of a questionnaire. The results of this usability study provided positive feedback and suggestions for future improvements. With this validation of the usability of our implementation we can advise for the continuation and further development of this extension.
## Contents

1 Introduction  .................................................. 1  
   1.1 Motivation .................................................. 1  
   1.2 Goals ...................................................... 2  
   1.3 Document Structure ....................................... 3  

2 Foundations and Technologies  ....... 5  
   2.1 Heatmaps .................................................. 5  
   2.2 The Monitoring Application ExplorViz ................. 6  
      2.2.1 Architecture ......................................... 7  
      2.2.2 Extensions ........................................... 9  
   2.3 The 3D Web Rendering Library Three.js .......... 9  
   2.4 The Web Framework Ember.js ............................ 9  
   2.5 Software Usability ....................................... 11  
      2.5.1 Usability Testing .................................... 11  
      2.5.2 Questionnaires ..................................... 12  

3 Approach ................................................... 13  
   3.1 Backend Extension ...................................... 13  
      3.1.1 The Existing Microservice Architecture ............ 13  
      3.1.2 Integration into the Microservice Architecture ... 15  
      3.1.3 Computation of Metrics .............................. 16  
      3.1.4 Adaptation of the Data Model ...................... 16  
   3.2 Frontend Extension ...................................... 17  
      3.2.1 The Current Frontend Implementation ............... 17  
      3.2.2 Integration as an Ember Addon ..................... 18  
      3.2.3 Addition of Heatmaps to the Visualization ....... 19  
      3.2.4 Selection of Colors ................................. 19  
   3.3 Evaluation ................................................ 19  
      3.3.1 Sample Application .................................... 19  
      3.3.2 Usability Study ....................................... 20  

4 Selection and Implementation of Metrics ........ 21  
   4.1 Selection of Metrics ...................................... 21  
   4.2 Implementation of Metrics ............................... 22  
   4.3 Extended Metrics ......................................... 23  
      4.3.1 Aggregated Heatmap ................................... 23
## Contents

4.3.2 Windowed Heatmap ........................................... 24

5 Implementation of the Backend Extension ................................... 25
  5.1 Components .................................................................. 25
  5.2 Control and Data Flow .................................................. 26
    5.2.1 Extension Services .................................................. 27
    5.2.2 Metric Computation ................................................. 28
  5.3 Data Model .............................................................. 30

6 Implementation of the Frontend Extension .................................... 33
  6.1 Modification of the Rendering Component ................................. 34
  6.2 Heatmap Visualization ................................................... 35
    6.2.1 Array Heatmap ....................................................... 35
    6.2.2 Simpleheat Heatmap .................................................. 37
  6.3 Configuration ............................................................. 39
  6.4 Final Visualization ....................................................... 40

7 Evaluation ................................................................. 43
  7.1 Goals ........................................................................... 43
  7.2 Methodology ................................................................ 44
  7.3 Usability Experiment ..................................................... 44
    7.3.1 Questionnaire .......................................................... 44
    7.3.2 Experimental Setup ................................................... 49
    7.3.3 Execution of the Experiment ........................................ 50
    7.3.4 Results ................................................................... 51
    7.3.5 Threats to Validity ..................................................... 58
  7.4 Summary .................................................................. 59

8 Related Work ............................................................... 61
  8.1 City Metaphor .............................................................. 61
    8.1.1 Code City ............................................................... 61
    8.1.2 Displaying Metrics for Software Landscapes ...................... 62
    8.1.3 Applying Heatmaps to Software Landscapes ...................... 62
  8.2 Software Metrics .......................................................... 62
  8.3 Heatmaps ................................................................. 63
    8.3.1 Selection of Color Schemes ........................................... 63
    8.3.2 Harmful Color Schemes .............................................. 63

9 Conclusions and Future Work .................................................. 65
  9.1 Conclusions ................................................................ 65
  9.2 Future Work .................................................................. 66

Bibliography ................................................................. 69
# List of Figures

2.1 Eyetracking heatmap [Tschneidr 2017] ........................................... 5
2.2 Color gradient from blue to light red representing heat [Röthlisberger et al. 2009] ............................................................ 6
2.3 Color weaving applied to heatmaps by [Benomar et al. 2013] .............. 6
2.4 Visualizations in ExplorViz ............................................................ 8
2.5 The ExplorViz software stack of version 1.4.1¹ .................................. 8
2.6 Three.js node-map² ...................................................................... 10
3.1 Approach of the thesis in the style of ExplorViz ............................... 14
4.1 Metric class .................................................................................. 23
5.1 Major components of the backend extension .................................... 26
5.2 Control and data flow of the backend extension ............................... 27
5.3 Control and data flow for the creation of a **LandscapeMetric** .......... 29
5.4 Data model .................................................................................. 30
5.5 ClazzMetric class ......................................................................... 30
5.6 **ApplicationMetric** class ............................................................. 31
5.7 **ApplicationMetricCollection** class ........................................... 31
5.8 **LandscapeMetric** class .............................................................. 32
5.9 Heatmap class ............................................................................. 32
6.1 Major components of the frontend extension .................................... 33
6.2 Steps for creating the array heatmap visualization .......................... 36
6.3 Gradient used for the array heatmap implementation ........................ 36
6.4 Visualization of a sample application using the array heatmap implementation .................................................................................. 37
6.5 Gradient used for the simpleheat implementation .............................. 37
6.6 Visualization of a sample application using the simpleheat implementation .................................................................................. 38
6.7 Features of the final visualization ..................................................... 41
Chapter 1

Introduction

1.1 Motivation

With a growing number of software applications deployed in organizations the comprehension of their formed software landscapes is often impeded by different reasons like architectural erosion, personnel turnover or changing requirements. The ExplorViz approach offers hierarchical abstractions and multi-level visualization for these software landscapes, and thus improves their comprehensibility. One of the major benefits of ExplorViz over flat visualizations of other popular application performance management tools is the 3D visualization of monitoring data on the application level [Fittkau et al. 2017].

The visualized landscapes of ExplorViz are computed for timestamps on a set interval. However, there are currently no features to directly analyze differences between two or more landscapes from different timestamps. Thus, it is necessary to manually compare the values of regions which are representing parts of the monitored application for different timestamps. This requires the user to select the timestamps on the timeline and wait for the landscape to reload while remembering the previous landscape, which reduces the usability of the software. Moreover, the addition of further metrics to the view is complicated as the visualization does not provide enough layers to display them.

Inspired by the work of Benomar et al. [2013], who extended the visualization environment of VERSO [Langelier et al. 2005] with heatmaps for 3D visualizations, we introduce an approach to add heatmaps to the visualization of ExplorViz. Their work proves that the application of heatmaps to visualizations using the city metaphor is feasible and beneficial for software comprehension. For their approach they recorded data concerning the evolution and execution of software which they used to compute metrics for their heatmaps. Furthermore, they restructured the placement of particular elements according to the visualization needs. However, their implementation is unsuitable for dynamic monitoring approaches as the constant flow of information does not permit the use of computational intensive techniques, e.g. restructuring, as this would slow down the overall visualization.

With our approach we extend ExplorViz with heatmaps in order to provide improved comprehension of software dynamicities during the execution. As we have to generate heatmaps during the execution of the monitored application, we develop an algorithm that does not impede the speed of ExplorViz and can be computed independently from the core application. Furthermore, this additional layer in the visualization can be used to
1. Introduction

display different metrics, and thus opens up new and user friendly possibilities to present information of the monitored application to the user.

1.2 Goals

The main goal of this thesis is to provide a heatmap extension for ExplorViz to visualize the dynamicity of certain metrics during the runtime of an application. As ExplorViz already monitors the state of applications the main challenge is to extract useful information and prepare them for the visualization in the frontend component. This section introduces the envisioned subgoals we need to fulfill in order to achieve the main goal.

G1: Identification and Selection of Suitable Metrics in ExplorViz

ExplorViz monitors and visualizes many different types of information of the application under observation. It is not reasonable for all of them to be displayed in a heatmap additionally to the existing visualization. Furthermore there are dynamic software metrics which can not be implemented by ExplorViz without rewriting major components of ExplorViz. This goal covers an assessment of the usability of metrics in our context and finally choosing a subset of them for implementation. Furthermore we will specify strategies for displaying the dynamicity of these metrics within our extension.

G2: Implementation of the Heatmap as an Extension for ExplorViz

With the chosen metrics and their display strategies introduced in the previous goal we need an implementation that realizes these in form of an extension for ExplorViz. The given task requires a backend extension for collecting, preparing, broadcasting, and persisting the metrics as well as a frontend extension that is responsible for visualizing the metrics on top of the existing application visualization. Both extensions will be implemented following the microservice architecture given by existing components and extensions of ExplorViz.

Therefore, the backend extension will be implemented as a Java application for conformance to the existing backend services. To meet the requirements of a microservice, the extension has to be independently deployable of the existing services. Thus, we will implement it as a separate project and rely on own dependencies, e.g. the database. The communication with the backend will be handled using existing channels like TCP and the Kafka channels. The frontend extension will be implemented as an Ember addon in order to be able to extend the existing visualization. This choice contradicts the goal of implementing a completely independent microservice as the Ember framework requires a restart of the frontend in order to activate or deactivate the extension. Nevertheless it is necessary for comprehensible communication with the existing frontend and access to existing resources of ExplorViz. The communication between the backend and frontend extensions will be handled using the Jersey framework and HTTP requests.
The completion of this goal covers the implementation of the frontend extension and the backend extension as microservices as well as the connection to required software systems like databases.

G3: Evaluation of the Implementation Using a Sample Application

For testing and later on evaluating our implementation we need a program which simulates the kind of events we want to visualize with our heatmap feature. Therefore, we will instrument a version of the sample application Spring PetClinic\(^1\) with Kieker. Furthermore, we provide workloads that simulate the usage of the website for this instrumented version.

Using this sample application we will evaluate the usability of our implementation by conducting a formative usability study. Formative software usability studies require only small amounts of participants but still yield meaningful assessments of the tested application. For this study we will prepare a scenario and a questionnaire that all participants have to complete. We will provide the results of the survey and the completed questionnaires as well as a test suite for future repetitions of this evaluation or evaluations for comparable implementations.

1.3 Document Structure

Chapter 2 imparts prior knowledge about foundations and technologies which are utilized in our approach which we outline in Chapter 3. The detailed implementation of our approach is divided into three chapters. In Chapter 4 we describe the used metrics and why we decided to use them. Chapter 5 contains information about the architecture and functions of the backend component of our extension. Chapter 6 covers the implementation of the frontend extension including the architecture and the different visualization modes. With Chapter 7 we present the conducted evaluation of our approach in the form of a usability study. In Chapter 8 we introduce other projects and articles that inspired this thesis or are based on similar principles. In Chapter 9 we summarize the work and discoveries of our thesis and give the prospect of future improvements and uses for our approach.

\(^1\)https://github.com/spring-projects/spring-petclinic
In this chapter we introduce the foundations and technologies which are useful for the comprehension of our approach and the overall thesis.

2.1 Heatmaps

Heatmaps originated from the shading matrices in which colors are used for highlighting or directly representing the content of table or matrix fields [Wilkinson and Friendly 2009]. This technique has evolved into heatmaps which are data matrices that visualize values in their cells by the use of a color gradient to give an overview over the largest and smallest values in the matrix [Metsalu and Vilo 2015]. Heatmaps nowadays are used in many scopes of application and are not required to be of matrix shape as shown in Figure 2.1.

Röthlisberger et al. [2009] address the heat color metaphor for mapping values to a color gradient. Thus, it ranges from blue (cold) to red (hot) as presented in Figure 2.2.

Benomar et al. [2013] describe several methods to condense multi-variate data into one

![Figure 2.1. Eyetracking heatmap [Tschneidr 2017]](image)
2. Foundations and Technologies

![Color gradient from blue to light red representing heat](image1)

**Figure 2.2.** Color gradient from blue to light red representing heat [Röthlisberger et al. 2009]

![Heatmap](image2)

(a) Two different heatmaps

![Heatmap](image3)

(b) Combination of the two heatmaps

**Figure 2.3.** Color weaving applied to heatmaps by [Benomar et al. 2013]

The method they deemed most useful is *color weaving* which is performed by representing individual colors side by side as shown in Figure 2.3.

2.2 The Monitoring Application ExplorViz

*ExplorViz* [Fittkau et al. 2017] is a tool that improves the system comprehension of large software landscapes. In contrast to popular application performance management (APM) tools which often use flat graph-based representations of the system, *ExplorViz* provides live trace visualization of large software landscapes. This is achieved by introducing three hierarchical abstractions: the *systems* which consist of one or more server nodes, the *node groups*, designed for the representation of horizontal scalability in cloud environments, and the *amount of communication* between applications.

*ExplorViz* offers hierarchical visualization of the software on the landscape level and
multi-layer monitoring from landscape level to application level. The visualizations are interactive and allow for dynamic switches between the different perspectives.

**Landscapes** Figure 2.4a displays the visualization of a sample application on the landscape level. Systems present in the software landscape are represented by gray boxes (1). The green boxes in the systems represent node groups, or in this case a single node (2), that are contained in the system. The purple box in the node (3) represents a running application that is contained in the node. If multiple applications run on the same node, further purple boxes are added.

**Applications** As shown in Figure 2.4b, the application-level perspective of our sample application is a 3D visualization that uses the city metaphor. The layout is a modified version of the layout used in CodeCity by Wettel [2010]. The flat gray box (1) at the bottom represents the application. The green boxes (2) atop are the packages and with their contained elements. The contents of all packages can be hidden (3) and shown through interaction. Classes are represented by purple boxes (4) and communication is displayed by orange lines (5). The thickness of each line corresponds to the frequency of communication between classes. The active instance count of each class is represented by the height of the respective purple box.

### 2.2.1 Architecture

*ExplorViz* is composed of a backend component and a frontend component which have been converted into a microservice structure with release version 1.3.0. Figure 2.5 displays the current structure of the software stack.

**Backend** The backend contains several microservices as shown in Figure 2.5. These microservices communicate via the streaming platform *Kafka* [Kreps 2011]. Apart from the analysis service all services are available to the reverse proxy *Træfik* [Containous 2016]. The services are implemented as (RESTful) webservices using *Jersey* [Oracle-Corporation and Eclipse-Foundation 2018]. The actual monitoring is done by instrumenting the application under observation with *Kieker* [Hasselbring and van Hoorn 2015] which provides dynamic analysis capabilities. The *Analysis* service uses the pipe and filter framework *TeeTime* [Wulf et al. 2014] to analyze the records provided by Kieker and provide traces for the *Landscape* service.

**Frontend** The frontend of *ExplorViz* is implemented as an *Node.js* application using the *Ember.js* [Katz 2011] framework. The frontend is responsible for handling the web clients and visualizing the data which is provided by the backend.

---

2. Foundations and Technologies

(a) Landscape level [Fittkau et al. 2017]

(b) Application level

Figure 2.4. Visualizations in ExplorViz

Figure 2.5. The ExplorViz software stack of version 1.4.1

8
2.2.2 Extensions

Extensions in the context of ExplorViz are services that add additional functions to the core application which are optional and situationally useful, e.g., the Virtual Reality Mode.

Extensions can be either pure frontend extensions, pure backend extensions, or both by composing separate frontend and backend extensions. The required type of extension is depending on the provided features.

Frontend extensions are implemented by taking advantage of the addon feature of Ember.js. Backend extensions are implemented as RESTful microservices that exchange resources via Jersey and communicate with other services by messaging through Kafka.

2.3 The 3D Web Rendering Library Three.js

Three.js [Cabello 2010] is a powerful JavaScript framework that allows the display of 3D and 2D content in a web browser. It uses the JavaScript API WebGL [Mozilla-Foundation 2011] which allows rendering of 2D and 3D graphics inside an HTML <canvas> element.

For displaying graphical elements with this library three components are mandatory. These are a scene, a camera, and a renderer. Scenes contain the objects, lights, and camera, which can be altered at runtime. Every object in the scene consists of a material and a geometry. The camera specifies the view angle, field of view (FOV) and the position of the camera in a scene. Finally the renderer is responsible for displaying the composed scene with the specific camera. Figure 2.6 shows the connection between those components.

Knowledge of this framework is crucial, as it is used for rendering the visualizations in the ExplorViz frontend.

2.4 The Web Framework Ember.js

Ember.js is an open-source JavaScript framework for building modern web applications [Katz 2011] that is based on the Model-view-viewmodel pattern. The frontend of ExplorViz is an Ember application, and therefore basic knowledge about the framework is necessary for the implementation of the extension as an ember addon. Ember applications consist of five essential concepts [Treacy 2019]:

- **Routes**: The router is responsible for parsing URLs and dispatching control to matching routes. These provide a model and initialize the controller/template for the requested route.

- **Models**: Models are representations of domain entities and every route has an associated model containing the current data of the application. Models can be represented in JSON format, and thus be stored on a remote host. The Ember Data library is a built-in addon

---

which provides a store that handles requests and either uses local, cached resources or fetches them from a given remote source.

- **Services**: Services are long-lived object singletons that provide functionalities for other ember objects. They are used to contain session-wide data, share data between components and perform actions that are not bound to a specific route. Once they are initialized services can be injected into all routes of the application.

- **Controllers/Templates**: Controllers are singleton objects that are initialized by the route. There is an implicit 1:1 relationship between routes and controllers. Templates contain the HTML representation of the controller’s route and share all properties with the associated controller. Ember uses the Handlebars\(^3\) templating library which allows the use of static HTML and dynamic content inside Handlebars expressions.

- **Components**: Components are a combination of a JavaScript source file and a corresponding Handlebars template. Components are reusable and can be called from any template. They are unaware of their surrounding context and receive data through attributes and actions. Thus, their state is not stored session-wide and lost during reloading of the route.

\(^3\)https://handlebarsjs.com/
2.5 Software Usability

The usability is by definition the "extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO 9241-11). Similarly to Zirkelbach [2015] we interpret it as the ease of use and learnability of our developed extension. Usability engineering is field of software engineering that is generally concerned with HCI (human-computer interaction) [Nielsen 1994], specifically with designing user interfaces that have a high usability and user friendliness.

2.5.1 Usability Testing

The usability of a software system can be evaluated by the usage of designated software engineering techniques. Usability testing is one of the most fundamental methods and to some point irreplaceable as it exposes the system directly to the users and provides information about their usage of computers and the problems which they have with the software [Nielsen 1994]. It can be understood as the process of learning about users from users while they are using the software under development to accomplish a specific goal of interest to them [Barnum 2010].

The experimental design of usability studies varies depending on the prerequisites and goals. The prerequisites include amongst other things the funding, time, test environment, and availability of participants. These factors influence the scope and methodology of the experiments. For instance a small budget and little time result in a smaller usability study with less participants. A dedicated laboratory with two rooms and space for the test subject and a moderator is desirable, however there are solutions for informal laboratories and remote tests. The availability of participants influences the scope and setup of the study as well. Goals include whether the study is supposed to support the developments process by diagnosing and fixing problems (formative testing) and thus supporting a user-centered design [Corry et al. 1997] or the evaluation of a finished product in terms of validating if it meets the requirements (summative testing). Quesenbery [2003] introduced the five dimensions of usability (the 5Es) which can be used as a baseline for identifying the testing goals. The dimensions are:

» **Effective**: How completely and accurately the work or experience is completed or goals reached

» **Efficient**: How quickly this work can be completed

» **Engaging**: How well the interface draws the user into the interaction and how pleasant and satisfying it is to use

» **Error Tolerant**: How well the product prevents errors and can help the user recover from mistakes that do occur
2. Foundations and Technologies

- **Easy to Learn:** How well the product supports both the initial orientation and continued learning throughout the complete lifetime of use

The selection of the test users is crucial for the significance of the study. Especially for "discount usability" approaches with few participants it is recommended to involve average users and exclude users from outlier groups [Nielsen 1994]. Even small studies with three to five test subjects detect most usability problems of a system, i.e. 80% [Virzi 1992; Lewis 1994] or 85% [Nielsen and Landauer 1993], as additional subjects are unlikely to reveal new information [Turner et al. 2006].

2.5.2 **Questionnaires**

Questionnaires are a way for the user to give structured information about their opinion and experience during the survey. There are different types of questionnaires that can be used during different stages of the test. Depending on the scope and goals of the study it is suitable to use either some or all of them. According to Barnum [2010] the types are:

- pre-test questionnaire
- post-task questionnaire
- post-test questionnaire

The **pre-test questionnaire** is used to gain additional relevant information about the test subjects. This is of importance as the screening of the chosen participants does not necessarily cover all relevant information as it is possible that there are differences among the participants that can be useful for the evaluation of their results. As the name implies the pre-test questionnaire is usually conducted before the actual test begins.

A **post-task questionnaire** is conducted to record the thoughts and answers to the tasks right after they are completed. This is useful as the participants can give their feedback immediately and do not have to recall their experience in the end, which might result in mixed up memories and fuzzy answers. Furthermore, given separate but similar scenarios the questions for each of them can be the same while is is still possible to provide questions targeted to specific scenarios.

**Post-test questionnaires** allow the user to rate the overall experience with the software. This can be done by creating a new questionnaire that is based on the testing goals or by using a standard post-test questionnaire. Two of the most popular questionnaires are the System Usability Scale (SUS) by Brooke [1986] which uses 10 questions based on a 5-point Likert-scale and the Computer System Usability Questionnaire (CSUQ) by Lewis [1995] which uses 19 different questions on a 7-point scale. Both questionnaires allow minor alterations in the wording in order to adapt the questionnaire to the tested application. Furthermore, both questionnaires yield reliable results in comparison with other questionnaires [Tullis and Stetson 2004] and have a high correlation in their outcome [Lewis 2019].
Chapter 3

Approach

In this chapter we introduce our overall approach for the heatmap extension and outline the required technologies for the different components of our implementation. Moreover, we point out how they are connected to the existing ExplorViz implementation and give an overview over their interaction. The detailed explanation for the implementation of the components can be found in the following chapters.

Figure 3.1 visualizes our approach as an extension to the current ExplorViz software stack as introduced in Section 2.2 with the additional systems, which we describe below in detail. The backend extension (●) is added to the current implementation as a microservice using the existing Java architecture of the ExplorViz backend services. Our frontend extension (―) is developed as an Ember addon, which provides an apprehensible integration into the main application. With this approach the extension can exist alongside the original visualization on the client side of a user (○). For the generation of test data, which is required for the development and the evaluation of our extension, we employ the open-source application Spring PetClinic (★). Said sample application is instrumented with Kieker to generate monitoring logs for the backend services.

3.1 Backend Extension

The current backend is implemented as a compilation of Java services that share the same landscape model and communication channels to create the landscapes and features which are required by the frontend application. In this section we describe our approach for the integration of our extension into the existing architecture of the backend and how we use and transform the provided landscape model. The detailed implementation of our approach can be found in Chapter 5.

3.1.1 The Existing Microservice Architecture

The existing services of the backend depicted in Figure 3.1 are implemented as independent microservices that rely mostly on the messaging framework Kafka for their internal communication. An exception is the analysis service that communicates directly via TCP by default. The purpose of this service is the analysis of the monitoring logs provided by the Kieker observer of a monitored application. For this task the service employs the pipe
Figure 3.1. Approach of the thesis in the style of ExplorViz
3.1. Backend Extension

and filter framework TeeTime [Wulf et al. 2014] and generates trace data which is published to and consumed by the landscape service. The landscape service accumulates the trace data into landscapes that use the shared data model and enriches them with additional information. Every ten seconds the current state of the internal model is copied and provided with the current time as a timestamp. Said landscapes are then serialized into a JSON:API compliant string and passed to the broadcast service and the history service through the corresponding Kafka channel. The former one broadcasts the landscapes to all registered frontend clients, and thus triggers a landscape update of the rendered model and timeline in the frontend visualization.

The history service has two purposes. Firstly, it persists the landscapes into the connected database and stores the string representation alongside the corresponding timestamp and database. The timestamps are also stored in a separate database to provide a faster lookup of timestamp entries. Secondly, the service provides an interface for external services to request the persisted entries either by timestamp or id. This feature is required by the reload feature of the frontend to reload older landscapes.

The remaining services are the user service, discovery service, and settings service, which all provide individual features for the authentication and configuration of ExplorViz features through the frontend. Thus, they are not directly concerned with the generation or modification of landscapes.

3.1.2 Integration into the Microservice Architecture

With the mentioned microservice architecture of the backend services it is most sensible to insert the backend component of our extension as another microservice. Therefore, we adapt the prevalent structure of the microservices and develop the extension as a Java application that relies on Kafka for the communication with other backend services. Although, with the goal of providing heatmaps as a supplementary source of information for the frontend application, it is not required to produce and publish data to other backend services. As outlined above, the only services that provide data concerning the monitored application are the analysis and the landscape service. However, as each heatmap is associated with exactly one landscape to allow unambiguous visualizations in the frontend, it is sufficient to subscribe to the updates of the landscape service for the generation of our heatmaps.

One principle of microservices is that they are loosely coupled to other services, and thus can be deployed independently and exchanged without completely restructuring the architecture. With the usage of Kafka as an asynchronous communication channel for our service, it is as loosely coupled to other services as possible with regards to communication. However, microservices are also required to come with their own auxiliary systems, for instance databases. For the generation and reload of our heatmaps we also require a persistent data model in our extension. Thus, we have to employ our own persistent data storage. In accordance with the example given by the history service we develop our own heatmap repository which has access to a separate instance of MongoDB and the means to serialize our heatmaps into a JSON:API compliant string. Similarly to the repository of the
3. Approach

history service we provide interfaces that allow for an effortless exchange of MongoDB with another database software. Due to the inherent usability we recommend the usage of Docker for the provision of the database and provide the necessary scripts for the initialization. However, it is still possible to utilize an independent database.

Without means of communication with the frontend however, we can not visualize the generated heatmaps alongside their associated landscape. Therefore, we have to provide communication channels with the frontend. With the history service and the broadcast service as examples we provide a RESTful interface with Jersey for the frontend clients to subscribe for heatmap updates and to request previous heatmaps. Similarly to the broadcast service we forward the serialized heatmaps to all registered clients. To simplify the communication of the whole software stack we integrate the provided API endpoints into the reverse proxy Traefik.

3.1.3 Computation of Metrics

The goal of our approach is the representation of metric dynamicity during the execution of applications. Therefore, we have to focus on a certain number of metrics and to develop means for their computation. The established data structure of ExplorViz makes the selection of dynamic class coupling metrics most sensible as classes and their communications hold most of the information that is gained from the chosen monitoring approach.

As the landscapes are processed and stored in the backend, the logic for the computation of our metrics has to be implemented there also. In order to visualize the dynamicities of the selected metrics we have to process the raw metrics further to gain information concerning their progression over time. The two strategies we pursue are an aggregating and a windowed approach. For the aggregating approach we add the metric values of consecutive landscapes up while decaying the older values over time. Thus, we achieve a new metric that represents an average value of the metric during the last timesteps. The windowed approach directly compares the metrics of two landscapes. The gap, respectively window, between these landscapes can be configured but defaults to one. This means that to gain the values for this windowed strategy we can take the difference between the metrics of two landscapes. In Chapter 4 we address the selection and implementation of specific metrics and the mentioned strategies in detail.

3.1.4 Adaptation of the Data Model

As mentioned in Section 2.2, the data model of ExplorViz is based upon landscapes which are structured objects that contain several abstraction levels of the monitored software system and supplementary information required for the individual abstractions. The abstractions from top to bottom are called systems, node groups, nodes, applications, components, and classes. Each of these model objects aggregates a list of lower level abstractions until the whole software system is represented. The first four abstractions are used for the visualization of the landscape perspective of a monitored software system, whereas the last three are
important for the application perspective of the visualization. Both visualizations utilize the application, which is used for the transition between the visualization levels.

For our approach we chose to provide heatmaps for the application perspective of the visualization, as most of the available dynamic metrics are based upon the coupling within applications. Therefore, we are only interested in the applications and their contained components and classes. Thus, a first step for the creation of our heatmaps is to flatten the landscapes to application level and then to extract and process the acquired information.

To store the raw metrics for every individual landscape we develop an associated data model that contains the metric information for every application of the software system. In order to store multiple metrics for one application we collect them in their own model. Focusing on class coupling metrics we can omit the component level for our representation and aggregate the class metrics directly. For the representation of the current metrics of one landscape this leaves us with the following abstraction levels:

- landscape level
- application metric collection
- application metric
- class metric

To create the different heatmap types, which we mentioned before, we can compare the metric models of two or more landscapes and store the new values as a new landscape level model and use these as an adequate model.

3.2 Frontend Extension

The frontend of ExplorViz is implemented as an Ember application as described in Section 2.4. As we want our extension to fit into the existing implementation, we choose to implement it as an addon as made possible by the Ember framework. This choice allows users to add the frontend extension to the application with small effort and provides us with the means to utilize existing components and services. This approach is not completely independent from the main application as we rely on some of the provided services and features to imitate the existing rendering implementation of landscapes and applications. However, a completely independent microservice is not sensible as we would have to introduce many redundant features into the frontend which would impact the performance of the web service. A detailed description of our implementation can be found in Chapter 6.

3.2.1 The Current Frontend Implementation

The frontend application consists of several services and components which handle the communication, data processing and visualization of the landscapes and applications.
3. Approach

One of the base components is the visualization controller which controls the associated visualization template on the visualization route. The controller stays active during the whole lifetime of the executed web service. It processes the performed actions on the visualization route and starts the landscape listener service which subscribes to the landscape broadcast service of the backend and listens to server sent events (SSE) that contain new landscapes. These landscapes are parsed into a JSON:API compliant representation due to the effort made in the backend. Furthermore, the controller updates the flags to show different components on the site, whenever relevant changes in the data structure happen. This allows for the usage of different functions on the same page without requiring the user to reload the page repeatedly.

One of these components is the rendering component which extends the rendering core and contains the logic to render selected applications as a 3D model. The rendering core contains core functions that are necessary for setting up the Three.js scene and initializing the WebGL renderer. The rendering component then populates this scene with meshes based upon the model which is stored in the landscape repository service.

When selecting an entry on the timeline or using one of the functions of the toolbar the controller fires the corresponding actions and executes functions, like triggering the reload of a landscape from the backend.

3.2.2 Integration as an Ember Addon

As already mentioned we integrate our frontend extension into the main application as an addon. This allows us to reuse many of the components of the existing implementation. The extension is hosted on its own route, and thus requires a corresponding controller. This heatmap controller works like the described visualization controller, however instead of starting the landscape listener we start a heatmap listener service which is responsible for receiving and forwarding new heatmaps from the backend.

The heatmap template that defines the style and content of our extension route is the same as the one from the visualization with the exception that we replace some guards and the rendering component with our own ones. Thus, we can reuse most of the existing components like the timeline and the toolbar.

For our rendering component we also extend the rendering core and copy the population logic from the main rendering component. To add our heatmaps we modify and extend the existing functions as necessary. Unfortunately it is not possible to just extend the existing landscape rendering component as we can not override all required functions and methods.

Apart from the broadcast events we need to provide means to communicate with the backend for reloading previous heatmaps. Therefore, we employ the Ember data store by implementing our data model and providing the required configurations.

To improve the usability for individual users we add customization and accessibility features by preparing a configuration route which can be used to change the color gradients of our heatmap and switch between provided heatmap and visualization types.
3.3. Evaluation

3.2.3 Addition of Heatmaps to the Visualization

The current visualization of applications in ExplorViz is accomplished by creating a scene in Three.js and adding meshes for the application, every component, respectively package, and each class to this scene. The foundation mesh is rendered as a gray box and works as a base for the other components. We introduce two approaches to adding our heatmap to the foundation mesh. The first one is to directly manipulate the colors of the foundation mesh for the corresponding metric value of a class during its creation. The second approach is to generate the heatmap separately as a canvas element by using the lightweight simpleheat.js tool and projecting it onto the existing foundation mesh afterwards.

3.2.4 Selection of Colors

An important feature of our approach is the coloring of the heatmaps. Heatmaps usually come with the rainbow color scheme to express the heat color metaphor [Röthlisberger et al. 2009]. Borland and Taylor li [2007] advise against the usage of the rainbow color scheme as it confuses the user due to an ambiguous perceptual ordering of the colors. Nonetheless, we decide to apply the standard color scheme for heatmaps as they are useful for the detection of extreme values and distinction of different values. To improve the usability we provide a legend that supports the user with ordering the colors and matching certain colors to their corresponding metric value. MacDonald [1999] provides some guidelines for selecting colors which we take into account for the final choice of our color schemes.

3.3 Evaluation

One of our goals is to achieve a high usability for our implementation. In this section we address our approach the evaluation of the final product which requires the preparation of a sample application and the conduct of a survey.

3.3.1 Sample Application

For the evaluation of our implementation we instrument a version of the Java based sample application Spring PetClinic with Kieker. This enables us to test the functionality of our implementation by applying different workloads, and thus generating landscapes for specific scenarios. For the usability study we prepare a set of landscapes in order to provide comparability of our distinct testing sessions, rather than utilizing a live version of Spring PetClinic. We generate the landscapes by applying different work loads to the sample application and recording the landscapes in their JSON representation. Based upon these landscapes, which replace the ones generated by the landscape service through minor modification, we can generate heatmaps during the conduction of our experiments.
3. Approach

3.3.2 Usability Study

The evaluation itself is executed as a formative usability study. The individual testing sessions with different test users are conducted as moderated tests, which allow us to supervise the user and assess the comprehensibility and usability of our implementation. During the experiments the users have to complete different tasks under the usage of the extended version of ExplorViz and aforementioned landscapes. The results of these tasks are recorded by filling out questionnaires. Rather than using a laboratory for the testing sessions we apply a remote approach over the internet, and thus allow the test user to participate from home.
Selection and Implementation of Metrics

ExplorViz processes monitoring logs generated by Kieker within the analysis service component. The processed logs are sent to the landscape service which creates a landscape model from them. The landscape models are then broadcast to all backend services that are subscribed to the corresponding Kafka channel. The landscape models contain structured information about the systems, node groups, nodes, applications, and components which is enriched with data of the class communications. Every Landscape object can be identified and fetched by the unique id or timestamp which are saved during its creation.

4.1 Selection of Metrics

Based upon the structure of the current ExplorViz visualization we decided to focus on dynamic class coupling metrics. Kumar Chhabra and Gupta [2010] and Tahir and MacDonell [2012] collected dynamic metrics and discussed their benefit of improving the comprehensibility and maintainability of software as a counterpart to static metrics. As show by Arisholm et al. [2004] and Gupta [2011], coupling metrics can be validated using a validation framework [Briand et al. 1996] and thus fulfill criteria such as non-negativity and monotonicity. Schnoor and Hasselbring [2019] researched and proved the correlation between dynamic weighted coupling metrics and their static counterparts.

The visualization of an application in ExplorViz shows the package and class structure of all active and monitored classes during runtime. Furthermore, the class communication between classes for certain timestamps is aggregated and depicted by lines between the classes (cf. Section 2.2). Based on this visualization we focus on metrics with class and object granularity and a focus on class communication. The metrics of our choice are listed in Table 4.1 with a short description as given by [Geetika and Singh 2014].

1. While not being one of the metrics listed in [Geetika and Singh 2014] the instance count indicates the usage of classes. Therefore, this metric enables to monitor the activity of the specific class overall and detect malfunctions of classes, e.g. if a class gets instantiated more or less than anticipated during certain phases of the runtime cycle of the monitored application.

2. The import coupling metric for dynamic messages of classes $IC_{CD}$ was introduced by Arisholm et al. [2004]. It accumulates the the outgoing communication of all objects of
4. Selection and Implementation of Metrics

**Table 4.1. Implemented metrics of our approach**

<table>
<thead>
<tr>
<th>#</th>
<th>Metric</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Instance Count</td>
<td>The number of instantiated objects of a given class.</td>
<td>-</td>
</tr>
<tr>
<td>(2)</td>
<td>Import Coupling</td>
<td>Total number of messages sent by all methods in all objects of a class.</td>
<td>[Arisholm et al. 2004]</td>
</tr>
<tr>
<td>(3)</td>
<td>Export Coupling</td>
<td>Number of messages received by all methods of all object of a class.</td>
<td>[Arisholm et al. 2004]</td>
</tr>
<tr>
<td>(4)</td>
<td>Dynamic Key</td>
<td>The percentage of calls sent and received by a class compared to the total number of calls.</td>
<td>[Singh and Singh 2010]</td>
</tr>
</tbody>
</table>

A class at a given time and thus indicates how much the class impacts other classes due to communication. This can help with identifying performance issues of a program as a class that sends too many messages may overstrain other classes and thus slow down the whole application.

(3) The export coupling metric for dynamic messages of classes EC_CD is the counterpart to IC_CD and also introduced by Arisholm et al. [2004]. It accumulates the incoming messages for all objects of a class at a given time. Thus, it can indicate how important the class is for other classes and how much they rely on methods of this specific class. This can also help identifying classes that are completely overwhelmed by the requests of other classes and thus slow down the execution of the application.

(4) The dynamic key class metric DKC was introduced by Singh and Singh [2010] along three other class-level dynamic coupling metrics. The focus of these metrics was the identification of the key classes, i.e. the most active classes at runtime. Therefore, they utilize both import and export coupling measures [Geetika and Singh 2014]. The DKC metric combines the import and export coupling metrics and displays them in relation to the total communication activity. Thus, it can be used to identify and highlight the most active classes which are prone to being bottlenecks for the entire application.

4.2 Implementation of Metrics

The metrics are implemented in the backend as subclasses of the abstract class Metric, which is depicted in Figure 4.1. The subclasses contain the fields typeName, name, and description and the computeMetric used method to compute them from the given Clazz.
4.3 Extended Metrics

The metrics introduced in Section 4.1 only show the state of classes of the monitored application at a certain time. One of the goals of our approach is to visualize the dynamicity of metrics during the runtime of an application. To achieve this goal we implemented two strategies of comparing the metrics of different timestamps to be visualized in the frontend.

4.3.1 Aggregated Heatmap

The first strategy is aggregating the values over time and thus keeping a piece of the information of previous timestamps at each consecutive timestamp. Whenever a new Landscape is consumed, and thus the creation of a new LandscapeMetric is triggered, 50% of the previous value is added to the newly computed metric value of all classes. If there is no previous value, as it is the case with the first measured metric for each class, the raw value is taken. Over time this strategy flattens the amplitude of the change curve for the metrics and allows only for positive values. Furthermore, the values which can be reached for each class in each time step are bounded by twice the maximum value measured for
4. Selection and Implementation of Metrics

that specific metric so far. Therefore, we avoid uncontrolled growth of this extended metric.
This can be proven with the convergence of the geometric series as follows:

Let \( V(c) \) be the set of aggregated values of the metric of class \( c \) and \( M(c) \) the set of newly measured, raw metrics of the same class \( c \) for each timestep \( t \in \mathbb{N} \). Per definition of our aggregated metric \( v_0 = m_0 \) holds as \( v_0 \) does not have a previous value. For \( v_n \in V(c) \), \( m_m \in M(c) \) and \( i \in \mathbb{N} \) holds that \( v_i = \sum_{k=0}^{i} m_{i-k} \cdot \frac{1}{2^k} \). As we only add values of \( M(c) \) in each iteration of our metric which can, by definition, not be negative it holds that \( v_i \geq 0 \) for all \( v_i \in V(c) \). Let \( m_{\text{max}} = \max(M(c)) \) be the maximum value of \( M(c) \). Then the values in \( V(c) \) are bounded by \( 2 \cdot m_{\text{max}} \) as \( v_i = \sum_{k=0}^{i} m_{i-k} \cdot \frac{1}{2^k} \leq \sum_{k=0}^{i} m_{\text{max}} \cdot \frac{1}{2^k} \leq \sum_{k=0}^{\infty} m_{\text{max}} \cdot \frac{1}{2^k} = 2 \cdot m_{\text{max}} \) holds for all \( i \in \mathbb{N} \).

4.3.2 Windowed Heatmap

For the second strategy we compare the raw metrics of two landscapes for each class and use their difference as an indicator for changes in the behavior of the monitored application. The distance in time of these landscapes is defined by a time window, which gives this strategy the name. The values of this metric range from negative to positive, with negative values representing a decline in the currently observed metric for a class and a positive value representing a growth with respect to their previous value. For example if a class had an instance count of 43 during one timestamp and an instance count of 35 in the next, this extended metric would yield a value of \(-8\), indicating the decline of the raw metric in the observed time. The strategy is realized by comparing the current LandscapeMetric with the one from \( n \) time steps before where \( n \) is the window size. This implementation is valid, as the sets of values of two consecutive windows will only differ by the values of the first time step of the first window and the values of the most recent time step of the second window. The window size can be configured in the backend configuration properties. Moreover, the values of this metric have an upper bound of the absolute difference of the maximum value and the minimum value of the raw metrics and a lower bound of the negative absolute difference. Thus, an uncontrollable growth of this metric is prevented.
Chapter 5

Implementation of the Backend Extension

The backend is implemented as a Java microservice using the ExplorViz dummy extension\(^1\) as a starting point. The dummy extension provides a framework for setting up the application and subscribing to the landscape-update Kafka channel. Furthermore, it brings a dependency injection binder and links the project to shared ExplorViz resources like the Landscape model due to the included libraries and build scripts.

5.1 Components

The backend extension consists of six major components which are represented in Figure 5.1 and described below:

- The MongoHeatmapRepository and MongoLandscapeMetricRepository\(^2\) provide methods to store heatmaps and landscape metrics in a local MongoDB instance. Furthermore, they allow the request of their respective objects from the database either by id or by timestamp. The objects are serialized and deserialized using the JSONAPI-Converter\(^3\) in order to provide a JSON:API compliant data representation.

- The KafkaLandscapeExchangeService\(^4\) is one of the classes originating from the dummy application we used as a starting point for this extension. The service handles the connection to the Kafka channel and consumes the landscapes that are produced by the landscape service of the main application. With our modification of this exchange service, the landscapes are used to create landscape metrics and heatmaps, which are then stored in a repository which is a required interface for this component. Another requirement is a component that handles the broadcasting of the heatmaps.

- The reload service\(^5\) is implemented as a Jersey resource and allows authenticated clients to fetch heatmaps from the backend extension. The service only provides access to the heatmaps as the landscape metrics are only meant for internal use during the computation of the heatmaps and not to be used in the frontend.

\(^1\)https://github.com/ExplorViz/explorviz-backend-extension-dummy

\(^2\)https://github.com/jasminb/jsonapi-converter

\(^3\)https://github.com/ExplorViz/explorviz-backend-extension-dummy

\(^4\)https://github.com/jasminb/jsonapi-converter
5. Implementation of the Backend Extension

The broadcasting of heatmaps is handled by the HeatmapService (white object), which contains methods to register listening clients as server sent event sinks and broadcast serialized heatmaps to all registered clients. The subscription requests are handled by a Jersey resource that validates the HTTP requests and forwards the registration requests to this component.

Finally, the ModelFactory (gray object) is a helper class that handles the transformation of all heatmap relevant data objects. This includes the creation of objects with new identifiers, computing Metric and LandscapeMetric objects and processing the different LandscapeMetrics into the windowed and aggregated versions which are used for the representation of the different heatmap types in a Heatmap object.

5.2 Control and Data Flow

In this section we introduce the different control and data flows which are active in the backend extension. In Section 5.2.1 we present the flows of the services which are active in the backend extension after start up. The control and data flows for the generation of single LandscapeMetric objects are further described in Section 5.2.2.
5.2. Control and Data Flow

5.2.1 Extension Services

Figure 5.2 shows the control and data flow of the separate extension components. The initialization of the extension deals with registering classes and resources to Jersey, initializing the dependency injections, and finally starting the KafkaLandscapeExchangeService. Afterwards four separate services are executed and running concurrently to process landscapes and expose interfaces for external components:

- Whenever a new Landscape is available on the Kafka channel, it is consumed by the KafkaLandscapeExchangeService. After deserialization the Landscape is passed to the model factory which contains the generator methods for all of our model classes. The factory instantiates a new LandscapeMetric object and triggers the computation of all metrics for the given Landscape as described in Section 5.2.2. Based on these raw landscape metrics a new Heatmap is created by computing and adding the aggregated heatmap and the windowed heatmap. Afterwards, the Heatmap and LandscapeMetric are serialized into a JSON:API compliant string. If no error occurs during that process the serialized Heatmap and LandscapeMetric are inserted into the database. Thereafter the serialized Heatmap is broadcast to all registered event sinks.

- The HeatmapService provides the feature of broadcasting serialized heatmaps to all external clients that are subscribed to the internal broadcaster. The heatmaps are
5. Implementation of the Backend Extension

published as server sent events, which are one-way connections between the server and the clients.

- The HeatmapBroadcastResource (explored) is a Jersey resource that provides a RESTful interface for external clients to register for server sent events. The incoming HTTP requests are processed and validated. If they are accepted, they are forwarded to the HeatmapService where the actual registration and storage of the SseEventSink is handled.

- The reload feature of the backend extension is handled by the HeatmapReloadResource (explored) which is a Jersey resource that provides a RESTful endpoint for reload requests. The requests must contain a timestamp which is associated with a Heatmap in the database. If the request is valid, the corresponding Heatmap is fetched from the HeatmapRepository and send to the client as an HTTP response.

5.2.2 Metric Computation

Based upon the received Landscape, the creation of a new LandscapeMetric is handled by the ModelFactory. In Figure 5.3 we present the control and data flow of this creation process.

- The first step taken is to find and collect all applications that are contained in the Landscape, as we are developing our heatmaps for the application level of ExplorViz. Therefore, the findLandscapeApplications (explored) method scans all Systems, NodeGroups and Nodes for nested Applications and stores them in a list.

- Afterwards, we create an ApplicationMetricCollection for every found Application (explored) which is used as a storage class, and thus contains the ApplicationMetrics for all metrics of a specific Application. In the frontend this collection can be matched with the associated Application by using the name and the id of the source Application.

- To compute the individual ApplicationMetrics (explored) we iterate over the set of Metrics. An Application containsClazzes which are nested in Components, i.e. packages, more often than not. Therefore, for every distinct Metric we use the helper method getChildrenComponentClazzes of the ModelHelper, which comes with the ExplorViz data model, to acquire a list of the nested Clazzes.

- With this list of Clazzes we can finally compute the individualClazzMetrics (explored). In order to do this, we need the Metric at hand, the respectiveClazz, and the Application it belongs to. All of these objects are passed down by the calling instances. The Metric provides a method to compute the value of the specificClazzMetric under the use of the providedClazz and Application.

- After their creation the individual metrics for each level are aggregated in lists and stored in their parent classes, using the corresponding setter methods (explored). The complete LandscapeMetric is then passed back to the KafkaLandscapeExchangeService, where it is used for the creation of Heatmaps and stored in the database after serialization.
5.2. Control and Data Flow

Figure 5.3. Control and data flow for the creation of a LandscapeMetric
5. Implementation of the Backend Extension

Figure 5.4. Data model

5.3 Data Model

Figure 5.4 shows an overview of the data model we developed for the heatmaps. All classes are extensions of the class BaseEntity which provides the id field for the serialization and deserialization. The classes are described in detail below.

ClazzMetric Figure 5.5 depicts theClazzMetric class, which is the lowest level class of our data model. EveryClazzMetric contains the fully-qualified name of the corresponding class of the monitored application and the computed metric value in order to be able to match them in the frontend. The methods addValue and subtractValue enable us to modify the value field by adding and subtracting a given parameter. These methods are primarily used during the computation of the windowed and aggregated metrics.
5.3. Data Model

<table>
<thead>
<tr>
<th>ApplicationMetric</th>
</tr>
</thead>
<tbody>
<tr>
<td>-metricType : String</td>
</tr>
<tr>
<td>-largestValue : long = 0</td>
</tr>
<tr>
<td>-classMetricValues : List&lt;ClazzMetric&gt;</td>
</tr>
<tr>
<td>+computeLargestValue(scalar : double) : void</td>
</tr>
<tr>
<td>+getClazzMetricByName(clazzName : String) : ClazzMetric</td>
</tr>
</tbody>
</table>

Figure 5.6. ApplicationMetric class

<table>
<thead>
<tr>
<th>ApplicationMetricCollection</th>
</tr>
</thead>
<tbody>
<tr>
<td>-appName : String</td>
</tr>
<tr>
<td>-appId : String</td>
</tr>
<tr>
<td>-metricValues : List&lt;ApplicationMetric&gt;</td>
</tr>
<tr>
<td>+getByMetricType(metricType : String) : ApplicationMetric</td>
</tr>
</tbody>
</table>

Figure 5.7. ApplicationMetricCollection class

ApplicationMetric The ApplicationMetric depicted in Figure 5.6 holds the values of all metric values of one specific metric of a monitored application. The class name of the metric is held in the metricType field. The list classMetricValues contains one ClazzMetric object for every class existing in the Application received from the landscape-service. The largest absolute metric value is held in largestValue for easier access in the frontend. The method computeLargestValue updates the largestValue for the current list of metrics using a scalar which is given as a parameter. The method is called by the model factory during the creation of the specialized landscape metrics which are contained in the heatmap. For easier access to the single ClazzMetrics they can be requested with the getClazzMetricByName method.

ApplicationMetricCollection The class ApplicationMetricCollection, as shown in the class diagram in Figure 5.7, contains the application metrics of all implemented metrics for one single Application in its metricValues list. The name and id of the corresponding Application are held in the appName and appId field respectively. To get the application metrics for one specific metric the method getByMetricType may be used.

LandscapeMetric The LandscapeMetric depicted in Figure 5.8 is a container class for all ApplicationMetricCollection objects of a Landscape. For its creation the landscapes are flattened to application level by omitting the systems, node groups and nodes and creating
5. Implementation of the Backend Extension

<table>
<thead>
<tr>
<th>LandscapeMetric</th>
</tr>
</thead>
<tbody>
<tr>
<td>-timestamp : long</td>
</tr>
<tr>
<td>-landscapeId : String</td>
</tr>
<tr>
<td>-metrics : List&lt;Metric&gt;</td>
</tr>
<tr>
<td>-applicationMetricCollections : List&lt;ApplicationMetricCollection&gt;</td>
</tr>
<tr>
<td>+getApplicationMetricCollectionByName(applicationName : String) : ApplicationMetricCollection</td>
</tr>
</tbody>
</table>

Figure 5.8. LandscapeMetric class

<table>
<thead>
<tr>
<th>Heatmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>-windowsize : long</td>
</tr>
<tr>
<td>-timestamp : long</td>
</tr>
<tr>
<td>-metricTypes : List&lt;String&gt;</td>
</tr>
<tr>
<td>-landscapeId : String</td>
</tr>
<tr>
<td>-aggregatedHeatmap : LandscapeMetric</td>
</tr>
<tr>
<td>-windowedHeatmap : LandscapeMetric</td>
</tr>
</tbody>
</table>

Figure 5.9. Heatmap class

an ApplicationMetricCollection for each application that is found in a landscape. The landscape metrics contain the metric collections for all applications in addition to the timestamp and id as landscapeId of the originating landscape. Furthermore, they contain a list of the metrics that are computed for every class, which is responsible for carrying the metric data to the frontend. Single applications can be requested with the getApplicationMetricCollectionByName method using the fully-qualified name of the application.

This class has two use cases. The first one is containing the base metrics for every landscape after their computation and persisting them within the database. The second one is to be used as a container for the further specialized aggregated and windowed heatmap metrics within the Heatmap as described in Section 4.3.

Heatmap Finally, the Heatmap as shown in Figure 5.9 is the top level container of our data model. It contains meta information about the contained heatmap metrics. These are the windowsize used to compute the windowed heatmap, the timestamp of the latest landscape at the moment of this objects creation, a list of the metric types used for the creation of the heatmap, and the landscape id that matches with the id of the landscape this heatmap originates from. Furthermore, it contains the aggregatedHeatmap and windowedHeatmap as LandscapeMetrics.
The frontend extension is implemented as an *Ember* addon as the frontend of *ExplorViz* is an *Ember* application. This structure allows the usage of our own routes, controllers and data models while still being able to access the services and components of the main application. The extension consists of six major extension components which perform different tasks. These components are shown in Figure 6.1.

- The first component is the `heatmap-rendering` component (1) which contains the rendering logic for our heatmaps. Based upon the application visualization of the main application we modify the existing *Three.js* methods to add our heatmap to the foundation mesh of the implementation. This extension is explained in Section 6.1 and can be reached using the `/heatmap` route or by using the navigation bar on the left side of the visualization. For the visualization we developed two different visualization types which we introduce in Section 6.2.

![Figure 6.1. Major components of the frontend extension](image-url)
6. Implementation of the Frontend Extension

- The second component is the heatmap-listener service which handles the connection to the broadcast updates from the backend extension. Whenever a new heatmap is received while the visualization is not paused, the service deserializes the received object and passes it to the Ember data store and notifies the heatmap-repository service to trigger an update of the intermediate data. Furthermore the heatmap-listener initializes the metric list with the first received heatmap, and thus enables the metric selector for the visualization.

- The heatmap-repository service contains the intermediate data and configurations of the extension instance. Ember services are available through injection after their creation and are not bound to a specific route. Therefore, they can amongst other things be used for data sharing between components. Furthermore, the evented character of this service allows the triggering of events which can be observed by other components.

- The heatmap-reload-handler service handles reload requests which are triggered by the user when clicking on a previous timestamp of the timeline. At first it pauses the visualization by notifying the landscape-listener service of the main application. Afterwards it requests the landscape and the heatmap belonging to the requested timestamp from the backend.

- The landscapes and heatmaps are provided by the backend in a JSON:API compliant format. Therefore, by implementing the data model as an Ember data model the communication part of these requests can be automatically handled by the Ember data store which provides local data handling capacities and automated fetching of missing entries from a connected database. After handing the requested objects over to their respective repositories a re-rendering of the visualization is triggered by the heatmap-repository.

- To offer an individual experience and some accessibility features we added a configuration tab to the ExplorViz configuration route, which is further described in Section 6.3. The chosen configuration values are stored in the heatmap-repository service to make them available for the other components.

6.1 Modification of the Rendering Component

In order to use existing components and event triggers, and to preserve the current look of the visualization provided by the main application, we used the application-rendering component of the ExplorViz frontend as a starting point. This implies, that our rendering component is an extension of the rendering-core of the main application. We accomplished the addition of our heatmap feature by modifying and extending the existing functions of the rendering component.

At first we replaced the landscape update trigger from the component with a heatmap update trigger. The existing component redraws the entire scene whenever a new landscape is broadcast to the linked controller. As our heatmaps are broadcast separately from their
6.2. Heatmap Visualization

For the visualization of our heatmaps provided by the backend we implemented two different ways of adding them to the foundation mesh of the existing visualization. The implementation of these two approaches is described in the following sections.

6.2.1 Array Heatmap

The array heatmap visualization is a naive approach to coloring the rendered foundation box which represents the monitored application. The name originates from the array which is used to accumulate and store the metric values for each class. For this approach we utilize the face mesh material of Three.js and segment the surface of the foundation mesh into squares consisting of two triangles, respectively faces, that can be colored separately as shown with a sample application in Figure 6.2a. Every square on this foundation is linked to a field of our array. Using a raycaster and the coordinates of each class we determine the square on which the corresponding metric value is projected. This process is depicted by the blue line shown in Figure 6.2b. In order to color our heatmap we first prepare the corresponding landscape and generally arrive after the landscape we have to update the scene whenever a new heatmap is registered at the frontend.

The second major modification is done in the createBox function which is responsible for creating the box meshes for all model components and adding them to the scene upon redraw. For the addition of the heatmap we require to manipulate the material and geometry of the foundation mesh, i.e. the mesh of the highest order which is used as a base for the other meshes. Based on the chosen visualization type we either set the used material of the foundation mesh to a multi-material consisting of six separate materials or add width and depth segmentation to the geometry of the mesh. These changes are required by the visualization types described in Section 6.2. For the subsequent identification of the meshes created with this method we set the name field to the fully-qualified name of the model object they are based upon. Furthermore, we reduce the opacity of all package boxes by using the value set in the configuration as they obstruct the view on the heatmap.

The final modification is the computation of the list of all classes used in the current application and the application of the heatmap using the applyHeatmap method in the existing populateScene function, which handles the addition of all meshes that are rendered in the final view. Depending on the chosen heatmap type the method prepares either the canvas element used for creation of a simpleheat object or the color array for the array heatmap. Afterwards we compute the position on the heatmap of every class of the monitored application by using a raycaster and register the corresponding value from the heatmap model to the heatmap. The final step is the rendering of the heatmap on the top plane of the foundation mesh by either setting the relevant face colors of the geometry or adding an emissive color map to the material as described in Section 6.2.
6. Implementation of the Frontend Extension

Figure 6.2. Steps for creating the array heatmap visualization

Figure 6.3. Gradient used for the array heatmap implementation

heatmap array by adding the metric value of a class obtained from our data model to the linked field in the array. As the aggregated heatmap only contains positive values we subtract 0.5 times the maximum value beforehand to transform all values into the range \([-0.5 \times \text{max}, 0.5 \times \text{max}\)]. This transformation is necessary, as our gradient expects values in this range to obtain color values.

As we are aiming for a heatmap styled look, we simulate the style by letting each metric value also impact the surrounding squares. This is done by adding a fraction of the value to the array elements belonging to the surrounding squares. For the squares directly left, top, right, and below of the square we add 0.4 times the value of the metric. The diagonally neighboring squares are impacted with 0.2 times the metric value. As the metric values can also be negative in case this might increase or decrease the current value of the array element. The color is determined by iterating over the array and setting each square to the color provided by the array gradient shown in Figure 6.3.

The gradient contains color values for numbers in the range \([-0.5, 0.5]\]. Numbers that do not fit into this range use the color associated with the closest number. The colors of the gradient are chosen by adapting the typical heat color metaphor. To increase the usability of this approach we focused on colors that have a high contrast and are easily distinguished. Thus, the colors representing negative values range from a blue over green
6.2. Heatmap Visualization

![Figure 6.4](image1.png)

**Figure 6.4.** Visualization of a sample application using the array heatmap implementation

![Figure 6.5](image2.png)

**Figure 6.5.** Gradient used for the simpleheat implementation

to gray. The colors for positive values range from gray over yellow and orange towards red. Gray was chosen as the color representing the zero value, such that classes without a change for a specific timestep are not highlighted in the corresponding visualization. In order to transform the metric values from the color array into this range we divide them by the maximum value of all values of this specific heatmap for the application. This maximum value is the absolute maximum value computed in the backend and contained in the data model. Finally the color value for all squares is set to the color obtained from the gradient by using the corresponding value of the color map. An example of the final product is shown in Figure 6.4.

### 6.2.2 Simpleheat Heatmap

The simpleheat heatmap implementation uses the external `simpleheat`\(^1\) library to create a heatmap canvas which is then projected onto the foundation mesh.

\(^1\)https://github.com/mourner/simpleheat
The size of this canvas is derived from the size of the foundation mesh. Data points can be added by providing their position in form of an x- and a z-coordinate in addition with a value for that position. The position of each class is determined by using the center point of the class mesh and a raycaster from an imaginary observer position and registering coordinates of the first intersection with the foundation mesh. However, as the coordinates on the canvas element are not centered like the foundation mesh we have to add this center point to get the final coordinates for the heatmap. The required data values are extracted from our data model. Similar to the transformation of the aggregated heatmap in the array heatmap implementation we have to transform the data values of the windowed heatmap in this case as our gradient only supports positive values. Therefore we add the maximum value to the data values to transform them into the positive range $[0, 2 \times max]$.

The `simpleheat` library uses a maximum value which divides all data points in order to scale them to the range $[0, 1]$. We use the precomputed value provided by the backend during the creation of the data model for this division. To choose the color values representing the different data values, we set color stops to be used by the gradient. These color stops are values between 0 and 1 and determine the color gradient which is displayed for values between these color stops. An example of our default gradient is shown in Figure 6.5. As
6.3 Configuration

During the implementation of the frontend we had to make some design decisions like the choice of the color gradients and the look of the heatmap. This resulted in the development and implementation of different features which can not be activated at the same time. Using the configuration panel of ExplorViz we implemented a configuration route for the extension to let the user choose which features are the most suited for the task at hand.

The first available settings are the heatmap mode and the visualization type. These settings allow the user to choose between aggregated and windowed heatmap data and if the array heatmap or the simple heat implementation should be used for visualization. If the simple heat implementation is chosen the user gets the option to alter the point radius and the blur radius which impact the size of each data point and the range at which the colors start blurring. The default radii are chosen such that the centers of the data points do not interfere with each other for classes that are rendered closely. To generate a stronger effect or if the visualization allows it this setting can be chosen freely.

As the heatmap is rendered on top of the foundation, the class boxes might be rendered far apart from their corresponding heatmap data point as the height of each package the class belongs to is added to the distance. This makes it hard to connect them visually from certain angles and viewpoints. Furthermore, the rendered boxes of the packages may obstruct the view of the heatmap. We addressed these issues by adding the option to add helper lines to the view which connect the classes with their corresponding data points. These lines follow the raycaster which is used to determine the coordinates of each class and can be observed in Figure 6.2b. Furthermore, the user can choose the opacity of the package boxes in order to make the heatmap more visible. This setting can be used to make the boxes completely transparent or opaque.

The last part of the settings is targeted at users with color blindness and users that do not like the chosen colors of the gradients. We are aware that the chosen colors for the heat metaphor might be difficult to differentiate for people with red-green color blindness. The choice of other colors might however impair people with other types of color blindness or just not meet the aesthetic standards of the user. In order to improve the accessibility and

with the array heat implementation these colors use the heat metaphor and range from blue to red and are chosen to have a high contrast.

The Three.js library provides the CanvasTexture class which allows the automated creation of a texture from an existing canvas element. Adding this texture to the foundation mesh without further changes, however, will project it onto all sides of the cube. We avoid this by interchanging the original foundation material with a multi material which consists of 6 different material objects. Of these we select the material which belongs to the plane at the top of the cube and set the canvas texture as the emissive map. Thus, the renderer will produce a light source on top of the foundation which produces the colors given by the heatmap canvas. The final projection is shown in Figure 6.6.
6. Implementation of the Frontend Extension

usability of the heatmap feature we added the option to set the colors for each color stop of the gradients using css styled color strings. The final gradient is shown as a legend in the visualization view of our extension.

6.4 Final Visualization

In Figure 6.7 we present the final visualization view of our extension. Shown in the center of the page is the modified application rendering model with the heatmap, which is projected onto the gray foundation mesh. The navigation bar on the left is extended with the route to our alternative heatmap visualization (1). The settings can be reached by following the configuration route (2) and then selecting the heatmap tab on the next page. The gradient for the currently selected visualization style and heatmap type is shown on the right as a legend (3) and can be toggled on and off by using the button below the ‘Back to Landscape’ button (4). The related metric score for the colors of the gradient can be deduced from the numbers scale next to the gradient, which shows the minimum, maximum, and mean score. Apart from the modifications required for the addition of our features, the visualization retains the functionalities of the main application. This includes the timeline (5) with the possibility to click on a timestamp to pause the visualization and reload the landscape with that specific timestamp which in this view also loads the corresponding heatmap entry. The underlying metric of the visualization can be selected dynamically using the metric selector which is added to the toolbar as a drop down menu (6).

The classes that do not have an associated heatmap color (7) were only present during the creation of the first landscape, and thus they do not have any metric scores that can be aggregated. The depicted combination of aggregated heatmap and the simple heat visualization therefore omits these data points. The classes of the owner package (8) can be determined as the key classes, in terms of the instance count, during this execution as their aggregated metric score shows the highest value. This means that for the last execution timestamps the classes either had a constantly high number of instances or increased the amount of active instances for this timestamp by a number large enough to overshadow the scores of other classes.
6.4. Final Visualization

Figure 6.7. Features of the final visualization
This chapter covers the conduct of the usability experiment we prepared for the evaluation of our approach and the implementation. Our main goal for the experiment was the evaluation of the usability and functionality of our extension. The implementation consists of a frontend and a backend extension. The former one does not provide a component which is directly accessible by a user. Therefore, we evaluated the functionality implicitly with the evaluation of the frontend extension.

The frontend extension provides different metrics, heatmap modes and visualization styles. A software engineer should be able to identify the different features and extract the information from the presented visualization as intended. Furthermore, the selection of different configurations is designed to be simple to understand and easy to use.

With our survey we identify the user interaction of participants and the overall usability of our software. As mentioned in Section 2.5 a group of approximately 5 test users can already identify 80%-85% of all usability problems of a software system. According to Nielsen [1994] an iterative approach with a small group is preferable to a single study with a larger group. As we do not have the time for an exhaustive iterative usability study we perform the first iteration with 5-7 test users, implement suggested improvements as far as possible, and thus facilitate subsequent iterations for future work.

Section 7.1 outlines the specific goals of our usability study. In Section 7.2 we describe our concrete methodology, which is used for the experiment presented in Section 7.3 in detail. Section 7.4 summarizes the conducted experiment and the results as well as our conclusions.

### 7.1 Goals

For the identification of our testing goals we use the five dimensions of usability (the 5Es) by Quesenbery [2003], which are described in Section 2.5, as a base line. This leaves us with the following questions we want to answer as testing goals:

- For each of the implemented metrics and visualization types: Can the user identify the meaning of the given information and use it to solve predefined tasks?
- General usage: Can users find and use the given tooltips and customization options?
- Does the heatmap extension help the users with finding a solution for each task?
7. Evaluation

- Is the chosen visualization satisfying and do the users enjoy the chosen color scheme and features?
- If the users encounter errors: Do they understand what do to recover? Do they understand how to reset the visualization?
- Is the application easy to understand? Do they need longer than expected for tasks or have to ask questions to solve problems?

7.2 Methodology

For our experiment we choose a remotely moderated usability experiment with task based scenarios and a survey in the form of a questionnaire. As described in Section 2.5 we use a pre-test questionnaire to gain relevant data about the test user before the start of the survey.

Instead of exposing the test users directly to the scenarios, we first introduce them to the software and show them the basics of ExplorViz and information about the features added with our extension. In order to verify the usability of our software the users then complete a guided scenario that provides scenarios in the form of landscapes and tasks they have to complete by giving an answer on the post-task questionnaire. Due to the vast number of different possible outcomes during the use of the ExplorViz monitoring software we prepare landscapes in order to confront all test users with the same problems.

Afterwards the users have to fill out a post-test questionnaire based on the System Usability Scale (SUS) questionnaire. During the test we measure the time spent on the tasks by each user to gain insight about the learnability of our software. At last we conduct a short closing interview with the participant to learn their opinion and thoughts of the experienced software.

7.3 Usability Experiment

In this section we describe the execution of our planned usability service. Therefore we firstly introduce the questionnaire and used software and hardware configurations before presenting and discussing the results.

7.3.1 Questionnaire

The questionnaire is separated into three parts that we introduce in detail below. The pre-test questionnaire contains questions about the personal information and experience of the test user. During the experiment the user has to complete certain tasks and record the results in the post-task questionnaire. Afterwards the test user evaluates the general usability of our approach in the post-test questionnaire. The scores displayed in the questionnaire give us a number to work with for the evaluation. For all scale based questions we translate
the given answers to a 1-5 scale and use the number for comparison. Most of the post-task questions require the user to give questions which can be wrong or correct. For these questions we use the score as a point value that can be achieved and compared.

**Pre-test questionnaire**

For the pre-test questionnaire we inquire the test subject about their current occupation, experience, and qualification. As we target a mixed group of scientists, students and graduates, we differentiate for instance between years of experience and number of semesters at the university. Furthermore, the experience of the users with ExplorViz as the base application for our extension and their experience with certain concepts of programming are relevant for the results of this survey. Therefore, we inquire them in this first debriefing before we start with actual questions for the scenarios. The starting time is taken at the end of the questionnaire to evaluate the time the test user requires for the tasks which are directly related to our extension.

**Post-task questionnaire**

The post-task questionnaire allows the test user to give feedback during the scenario, for which we assume the participant to be a software engineer that conducts an analysis of a web application. The goal of the analysis is to gain more insight about the dynamicity of certain metrics and their development during the execution of the monitored application. In this subsection we introduce the questions for the different tasks, respectively subscenarios. These subscenarios require the user to analyze a prepared landscape and use the tools the extension offers to solve the tasks. Even though the questions for the different tasks from 4-7 have the same structure, we request answers for different metrics to test the users understanding of this feature. Apart from testing the understanding of the user during each task, we also ask them for thoughts and improvements for the extension.

**Visualization**  The first set of questions regards the visualization in general and the identification of the implemented metrics and features. The questions shown in Table 7.1 require the user to write down certain observations. The given answers can be either correct or incorrect. The goal of this first set of questions is to test if the provided features are detected and understood with ease.

**Configuration**  This task is related to the configuration options we provide with the heatmap tab in the extension route. The questions shown in Table 7.2 help us to introduce the configurations to the test user and to perceive if a participant could grasp the reason for each option. Furthermore we inquire if there are any missing options or suggestions for further improvements.
7. Evaluation

Table 7.1. Questions regarding the overall visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a</td>
<td>How many different metrics can be selected for the visualization?</td>
<td>1</td>
</tr>
<tr>
<td>2b</td>
<td>Please name the metrics.</td>
<td>4</td>
</tr>
<tr>
<td>2c</td>
<td>Give a short summary of the information each of the metrics provides.</td>
<td>4</td>
</tr>
<tr>
<td>2d</td>
<td>Please name the type of the visualized heatmap</td>
<td>1</td>
</tr>
<tr>
<td>2e</td>
<td>Are there classes with a neutral or non-existent metric score in the visualization?</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7.2. Questions regarding the configuration route

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3a</td>
<td>What is the difference between the heatmap types?</td>
<td>2</td>
</tr>
<tr>
<td>3b</td>
<td>Which visualization styles can be chosen?</td>
<td>2</td>
</tr>
<tr>
<td>3c</td>
<td>Please give any thoughts and comments about the provided configuration options.</td>
<td>-</td>
</tr>
<tr>
<td>3d</td>
<td>Are there any options missing in your opinion?</td>
<td>-</td>
</tr>
</tbody>
</table>

**Aggregated heatmap with simple heat visualization** These questions regard the combination of the aggregated heatmap with the simple heat visualization. The first questions in Table 7.3 require the user to understand the given visualization and give the value or choose the correct class for the specific metric. As this combination does not necessarily provide color values for each metric, i.e. for 0 values, we ask questions to inquire if this is clear to the user. With the last question the test user can rate the combination of heatmap type and visualization mode to give us an indication on how useful the combination is for a user.

**Aggregated heatmap with array heat visualization** These questions regard the combination of the aggregated heatmap with the array heat visualization. The first questions in Table 7.4 require the user to understand the given visualization and give the value or choose the correct class for the specific metric. In contrast to the question block above, this combination shows medium values in the same color as the foundation mesh due to the chosen gradient to highlight low and high values. The goal of Question 5d is to
7.3. Usability Experiment

Table 7.3. Questions regarding the combination of aggregated heatmap and simple heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>4a</td>
<td>What are approximately the highest and lowest represented metric scores for the export coupling metric?</td>
<td>2</td>
</tr>
<tr>
<td>4b</td>
<td>Name at least one class with a high metric score for the number of instances</td>
<td>1</td>
</tr>
<tr>
<td>4c</td>
<td>Name at least one class with a low but existing export coupling score</td>
<td>1</td>
</tr>
<tr>
<td>4d</td>
<td>Name at least one class with a neutral or non-existent import coupling score</td>
<td>1</td>
</tr>
<tr>
<td>4e</td>
<td>Please rate this combination of heatmap type and visualization style</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 7.4. Questions regarding the combination of aggregated heatmap and array heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>5a</td>
<td>What are approximately the highest and lowest represented metric scores for the dynamic key metric?</td>
<td>2</td>
</tr>
<tr>
<td>5b</td>
<td>Name at least one class with a high import coupling score</td>
<td>1</td>
</tr>
<tr>
<td>5c</td>
<td>Name at least one class with a low export coupling score</td>
<td>1</td>
</tr>
<tr>
<td>5d</td>
<td>Name at least one class with a medium instance count score</td>
<td>1</td>
</tr>
<tr>
<td>5e</td>
<td>Please rate this combination of heatmap type and visualization style</td>
<td>5</td>
</tr>
</tbody>
</table>

test if this is apprehensible for the user. With the last question the test user can rate the combination of heatmap type and visualization mode to give us an indication on how useful the combination is for a user.

Windowed heatmap with simple heat visualization These questions regard the combination of the windowed heatmap with the simple heat visualization. The first questions in Table 7.5 require the user to understand the given visualization and give the value or choose the correct class for the specific metric. Unlike the aggregated heatmap the windowed heatmap provides negative, neutral and positive values. These may be difficult to get used to, thus we inquire if the test user can grasp the meaning of the color values. With the last question the test user can rate the combination of heatmap type and visualization mode to
7. Evaluation

Table 7.5. Questions regarding the combination of windowed heatmap and simple heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>6a</td>
<td>What are approximately the highest and lowest represented metric scores for the import coupling metric?</td>
<td>2</td>
</tr>
<tr>
<td>6b</td>
<td>Name at least one class with a high, positive export coupling score</td>
<td>1</td>
</tr>
<tr>
<td>6c</td>
<td>Name at least one class with a low, negative score for the number of instances</td>
<td>1</td>
</tr>
<tr>
<td>6d</td>
<td>Name at least one class with a neutral or non-existent dynamic key score</td>
<td>1</td>
</tr>
<tr>
<td>6e</td>
<td>Please rate this combination of heatmap type and visualization style</td>
<td>5</td>
</tr>
</tbody>
</table>

give us an indication on how useful the combination is for a user.

Windowed heatmap with array heat visualization These questions regard the combination of the windowed heatmap with the array heat visualization. The first questions in Table 7.6 require the user to understand the given visualization and give the value or choose the correct class for the specific metric. The combination of windowed heatmap and array heatmap visualization style were designed for each other and are therefore promising candidates for an apprehensible visualization. The questions are designed to test the comprehensibility of this combination. With the last question the test user can rate the combination of heatmap type and visualization mode to give us an indication on how useful the combination is for a user.

Post-test questionnaire

The design of the post-test questionnaire is based upon the SUS questionnaire but further adapted to the context of our thesis. We use the questions to inquire the user about the learnability of the features, i.e. the metrics and gradients. Furthermore, we want them to evaluate the ease of use of our extension by rating the difficulty of the provided features and configuration options. We also want to know how successful and appealing the integration into the existing visualization is. The usability of a program is coupled with the general impression it leaves on the user. Thus, we ask the user to rate our overall extension and at last to give suggestions for improvements and to name ambiguities.
Table 7.6. Questions regarding the combination of windowed heatmap and array heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>7a</td>
<td>What are approximately the highest and lowest represented metric scores for the export coupling metric?</td>
<td>2</td>
</tr>
<tr>
<td>7b</td>
<td>Name at least one class with a high, positive metric score for the number of instances</td>
<td>1</td>
</tr>
<tr>
<td>7c</td>
<td>Name at least one class with a low, negative dynamic key score</td>
<td>1</td>
</tr>
<tr>
<td>7d</td>
<td>Name at least one class with a neutral score for the number of instances</td>
<td>1</td>
</tr>
<tr>
<td>7e</td>
<td>Please rate this combination of heatmap type and visualization style</td>
<td>5</td>
</tr>
</tbody>
</table>

7.3.2 Experimental Setup

As we conducted the study remotely we had to provide some way for the test users to test the extension. Therefore we deployed the extended ExplorViz frontend and backend on a web server on the local machine we used for development. The local server was then exposed to the internet using port forwarding. As the users are required to have a comparable experience we omitted the monitored application. Instead we modified the landscape service of the backend to send prepared landscapes circularly and let the user select the designated landscape for each task. The landscapes were created using the aforementioned Spring PetClinic by instrumenting the application with Kieker and adding different loads to simulate the use case of an application on a web server under real use.

The users were handed a PDF questionnaire to be filled out in tandem with the tasks which had to be completed in a web browser. To instruct the user we established a voice connection using Discord\(^1\) or Zoom\(^2\) which also provide the functionality of sharing the screen of the test subject. Therefore we could supervise the activities of the test user during the tasks and aid them if they were stuck, which we also recorded as an indicator for possible improvements.

Hardware Configuration As aforementioned we deployed the application on our development machine which uses the hardware setup listed in Table 7.7. We also used the machine for the supervision of the experiments and the inspection of the given answers which requires our machine to have an adequate display set-up.

As for the hardware configuration of the test users, which was required for the client side of the application we do not have much information as every test user used a different

---

\(^1\)https://discordapp.com/
\(^2\)https://zoom.us/
7. Evaluation

Table 7.7. Hardware configuration

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Core i5-8600K 6x 3.6GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>16.0 GB</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA GeForce GTX 1070 8 GB</td>
</tr>
<tr>
<td>Display Size</td>
<td>24 inch (widescreen)</td>
</tr>
<tr>
<td>Display Resolution</td>
<td>1920x1080 px</td>
</tr>
</tbody>
</table>

configuration. However, we inquired about their display setup which we list along the raw test data, which can be found in Appendix B.

Software Configuration  The software configuration of our machine is based upon Microsoft Windows 10 Pro N with version 10.0.18363 as the operating system. For the execution of the backend and the corresponding extension we applied the Java Development Kit (JDK) in version 11. The necessary auxiliary services, e.g. MongoDB for the backend were executed with Docker in version 2.2.0.5 and the docker-compose file which is contained in the release version of the ExplorViz 1.5.0 backend.

The frontend application and extension were build with the ember-cli addon of Ember which was executed with Node.js v10.13.0. The frontend was then deployed on an Apache web server provided by XAMMP v7.4.5. Due to connection issues caused by the IPv6 protocols we had to fall back on the Docker version of our evaluation setup in some cases, which required the user to execute the software stack on their machine. The corresponding docker-compose file is contained in the source code of our backend extension.

7.3.3 Execution of the Experiment

Before starting the actual usability study we conducted a small pilot study with one participant to identify weaknesses and ambiguities of our chosen approach for this evaluation. With the received feedback we were able to correct the issues by giving additional hints and rephrasing or replacing whole questions. For the study the users were given a short introduction to ExplorViz to help them understand and navigate through the software. Afterwards they received a questionnaire as described in Section 7.3.1, of which a printable version can be found in Appendix A. The questionnaire required the users to fill in some personal information and then wait for instructions to progress to the next step. Supervised by the moderator the users had to perform different tasks with our extension and answer the corresponding questions on the questionnaire. After completing all tasks the users were told to fill out the post-test questionnaire which contained debriefing questions about the general usability of our extension. The experiment was ended by a short closing
interview in which we inquired the user about possible improvements and problems they encountered while solving certain tasks.

7.3.4 Results

In this section we present and discuss the results of our conducted survey. We start with the pre-test questionnaire and continue with the post-task questionnaire, which is separated into individual sections. In the end we address the given ratings and suggestions for the overall approach of the post-test questionnaire.

Pre-test questionnaire

Our test users consisted of a group of six computer scientists, which was composed as follows: one bachelor student, two graduates with bachelor degree, one master student, and two researchers with master degree of whom the latter are targeting a PhD. We asked our test subjects to rate their experience levels for certain programming skills on a scale of 1-5. The standard deviation and mean of their answers are shown in Table 7.8.

The results for programming in general, object oriented programming and java programming show a mean of 3.83 and all range between intermediate (3) and expert (5). This leads to the conclusion that all subjects are experienced enough to understand the presented concepts of our experiment as our chosen sample application is a Java application. The experience level for development of web applications, which could impact the overall understanding of our implementation, averaged at 3.17 with one participant indicating that he has no experience. All test subjects have some experience with monitoring software with an average of 2.83. The students indicated their level as beginner (2) and the graduates and researchers were intermediate (3) or above. For the experience level with ExplorViz only the researchers were experts (5), while the rest of the participants had none (1) or only beginner (2) level of experience. This resulted in an average level of 2.50 and a standard deviation of 1.97. We measured the time each test user needed for completing the post-task and the post-test questionnaire which averaged in 37m 20s with a standard deviation of 11m 3s. Combining the results, we conclude that the experience level with ExplorViz primarily impacted the time the test subjects required to get used to the software. Thus, the experts in this category needed noticeably less time to complete the questionnaire.

Introduction

The introductory section of our post-task questionnaire was targeted at getting the users to learn the features of our extension and how to use the information tooltips. During our observation we noticed that the identification of some features could take some time but afterwards the questions could be answered correctly. As mentioned before an advanced experience level with ExplorViz was beneficial with being able to distinguish established
7. Evaluation

Table 7.8. Results of the pre-test questionnaire experience ratings

<table>
<thead>
<tr>
<th>ID</th>
<th>Field</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1d</td>
<td>Programming in general</td>
<td>3.83</td>
<td>0.75</td>
<td>5</td>
</tr>
<tr>
<td>1e</td>
<td>Object oriented programming</td>
<td>3.83</td>
<td>0.75</td>
<td>5</td>
</tr>
<tr>
<td>1f</td>
<td>Java programming</td>
<td>3.83</td>
<td>0.41</td>
<td>5</td>
</tr>
<tr>
<td>1g</td>
<td>Development of web applications</td>
<td>3.17</td>
<td>1.17</td>
<td>5</td>
</tr>
<tr>
<td>1h</td>
<td>Monitoring software</td>
<td>2.83</td>
<td>0.75</td>
<td>5</td>
</tr>
<tr>
<td>1i</td>
<td>ExplorViz</td>
<td>2.50</td>
<td>1.97</td>
<td>5</td>
</tr>
</tbody>
</table>

features from extended ones. The results are shown in Table 7.9 and clearly indicate that all test users were able to answer these simple questions correctly.

Table 7.9. Results of the introduction section

<table>
<thead>
<tr>
<th>ID</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2b</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2c</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2d</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2e</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Configuration

The first two questions of this section introduced the test users to the selection of heatmap types and visualization styles, which were also necessary for the following questions. As the results in Table 7.10 show, the questions were simple enough to be answered correctly by all participants. With the other questions we inquired the users if there are options missing or any suggestions for improvements, which we could not rate with a score system. The received feedback was positive, although one of the participants criticized that a graphical example for some of the options might be better for understanding their functionality over the existing text based tooltip. Another point of criticism was that disabling the helper lines is an unnecessary feature as they are important for connecting the classes with their corresponding point on the heatmap. However, there might be situations in which they are obstructing the view, e.g. for the creation of an image or the creation of a model. Therefore,
7.3. Usability Experiment

Table 7.10. Results of the configuration section

<table>
<thead>
<tr>
<th>ID</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3a</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3b</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

we left the option and turned it on by default. A major issue for the usability that came up was that for many test users the sliders for the color configuration of the gradients were not visible enough to be found. This issue was recorded and addressed in a later update.

The answer for the last question was left blank by all users, as they did not yet have enough experience with our extension to give an answer. However, in the post-test questionnaire the users could again mention desired features and options. Nonetheless, this leads to the conclusion that we did not miss obvious and important options.

Aggregated heatmap with simple heat visualization

This part of the post-task questionnaire addressed the combination of the aggregated heatmap type and the simple heat visualization. The first questions concerned the ability of test users to match the classes with their corresponding color values according to the tasks. With the last question we asked the user to rate the combination in order to get an impression of the usability.

As shown in Table 7.11 the tasks could be completed by the test users without major complications and all color values could be matched to their associated classes. The deviation from the expected value in question 4a was caused by the color scale of the used gradient, which lead some participants to guess the metric score inaccurately. The other questions of this section were answered correctly by all test users. None of the test subjects gave a negative rating to this combination with half of the users giving it a clearly positive rating. This results in a mean of 3.67 and a standard deviation of 0.82. In combination with the correctness of the results we conclude that the usability of this combination is good but still has room for improvement like an improved color gradient and a better scale to prevent users from guessing values inaccurately.

Aggregated heatmap with array heat visualization

The combination of the aggregated heatmap with the array heat implementation required the test subjects to get used to a different color gradient and the pixel-like visualization of the array heat map. Similarly to the previous section the users had to identify color values of the heatmap and the classes that were associated with them.

In Table 7.12 we present the results of this section. While being able to identify the maximum and minimum scores for classes, the results indicate that the test users had
7. Evaluation

Table 7.11. Results for the combination of aggregated heatmap and simple heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>4a</td>
<td>1.83</td>
<td>0.26</td>
<td>2</td>
</tr>
<tr>
<td>4b</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4c</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4d</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4e</td>
<td>3.67</td>
<td>0.82</td>
<td>5</td>
</tr>
</tbody>
</table>

difficulty identifying the medium values with the underlying visualization style. However, they often chose a class that had a value close to the optimal value, which results in means of 1.5 and 0.5 for questions 5a and 5d respectively. One of the factors could be a different perception of the rendered colors on the heatmap and the shown legend due to the different technologies that were used for their creation. The rating for this combination is, with a mean of 3.5, slightly worse than for the previous combination, although none of the test users gave a negative rating. In combination we can conclude that this combination is useful for presenting extreme values, but can be improved for medium values to increase the usability.

Table 7.12. Results for the combination of aggregated heatmap and array heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>5a</td>
<td>1.5</td>
<td>0.32</td>
<td>2</td>
</tr>
<tr>
<td>5b</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5c</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5d</td>
<td>0.5</td>
<td>0.32</td>
<td>1</td>
</tr>
<tr>
<td>5e</td>
<td>3.5</td>
<td>0.84</td>
<td>5</td>
</tr>
</tbody>
</table>

Windowed heatmap with simple heat visualization

With the combination of the windowed heatmap type and the simple heat visualization style the test subjects were required to adapt to negative values which represent a decrease of the metric in comparison to the previous heatmap.

The results are shown in Table 7.13. For question 6a the scores deviate from the expected values with a mean of 1.5 and a standard deviation of 0.5, as half of the observed test
7.3. Usability Experiment

Subjects did not verify if the possible values shown on the legend were present on the displayed heatmap. This could be caused by ambiguous formulation of the question or misinterpretation of the visualization. Due to the bright color which is used for positive values of the simple heat visualization all participants identified classes with a high color value correctly. However, due to the underlying functionality of the simple heat visualization negative values are harder to spot and were thus not correctly identified in many cases. This leads to a mean value of 0.67 for question 6d. The medium values for question 6e were easier to spot, and thus were correctly identified in most cases. Overall the test users gave a rating of 3.5 to this combination, which is still on the positive side of the scale. Therefore, we conclude that the usability of this combination is decent but could be increased by improving the detection of lower values and the distinguishability of different colors.

Table 7.13. Results for the combination of windowed heatmap and simple heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>6a</td>
<td>1.5</td>
<td>0.55</td>
<td>2</td>
</tr>
<tr>
<td>6b</td>
<td>1</td>
<td>0.41</td>
<td>1</td>
</tr>
<tr>
<td>6c</td>
<td>0.67</td>
<td>0.41</td>
<td>1</td>
</tr>
<tr>
<td>6d</td>
<td>0.83</td>
<td>0.41</td>
<td>1</td>
</tr>
<tr>
<td>6e</td>
<td>3.5</td>
<td>0.55</td>
<td>5</td>
</tr>
</tbody>
</table>

Windowed heatmap with array heat visualization

The last combination we tested was the windowed heatmap with the array heat visualization. Neither the heatmap type nor the visualization style were completely new to the test subjects, although they did not encounter this combination before.

Although the results shown in Table 7.14 for this section contain the low average result of 1.33 for question 7a they are overall satisfying. The bad results for this specific question might again be caused by the ambiguities discussed in the previous section, as most of the test subjects did not correctly identify the requested lower metric score. Nonetheless, the values for questions 7b and 7c are correct in almost all cases. The deviation from the expected value in could be caused by the different color values of the heatmap color and the gradient of the legend we discussed before. The overall rating for this combination is the best of the conducted experiments with an average of 4.17. Therefore, taking into account the good results of this section, we conclude that the usability of this implementation is at a satisfying level and only requires minor tweaks and improvements.
7. Evaluation

Table 7.14. Results for the combination of windowed heatmap and array heat visualization

<table>
<thead>
<tr>
<th>ID</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>7a</td>
<td>1.33</td>
<td>0.52</td>
<td>2</td>
</tr>
<tr>
<td>7b</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7c</td>
<td>0.92</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>7d</td>
<td>0.75</td>
<td>0.27</td>
<td>1</td>
</tr>
<tr>
<td>7e</td>
<td>4.17</td>
<td>0.75</td>
<td>5</td>
</tr>
</tbody>
</table>

Post-test questionnaire

The debriefing questions of the post-test questionnaire required the test subjects to rate the overall usability and learnability of our implementation and give suggestions for further improvements and optimizations. The results shown in Table 7.15 indicate positive, above average results for all values. However, the difficulty rating for understanding the metrics contains mixed results, and thus has a high standard deviation of 1.21, from which we infer a low comprehensibility for the metrics. The difficulty for the configuration and use of the extension features was rated between medium (3) and very good (4) by all test subjects which results in a mean of 4.33 and a standard deviation of 0.82. Our observations showed that the test subjects learned how to use the provided features, as their required time for the different tasks decreased during the experiment. Furthermore, the integration into the existing visualization and the overall impression of the extension were rated in average with scores of 4 and 3.83. Thus, we conclude that apart from the introduction of the used metrics the overall usability of our extension was perceived as satisfying, although there is still room for improvement and optimization.

The given answers for question 8e contained suggestions for improvements and comments concerning the implementation, which we summarize below. The entirety of the answers can be found in Table 3 of Appendix B.

1. A mouseover feature that highlights the color value of the class at the position of the mouse pointer on the color gradient or show the value as a pop-up tooltip could prevent guessing of values.
2. Better examples/tutorials for metrics would help with their comprehensibility.
3. Color values of 0 that were either not rendered or had the same color as the ground were confusing and hard to understand.
4. The values of the array heat visualization were easier to distinguish.
5. The hierarchy of the application is lost due to the transparency of packages.
6. The helper lines are not centered and not visible enough for larger applications.

The suggested features (1) and (2) would improve the usability of our approach. Therefore, we suggest their implementation in future releases of the extension.

The comments (3) and (4) address the distinguishability and identifiability of the color values. A possible solution is achieved by improving and optimizing the color gradients or even implement a variety of gradients that can be chosen for the task at hand. However, this would require further experiments and usability studies. Therefore, we have to leave this issue for future work on this extension.

The transparency of the packages mentioned in (5) is required for our heatmap to be visible on the foundation mesh. By increasing the opacity of the package boxes the color values of the heatmap are affected due to the transparent green layer of the boxes, which is rendered between heatmap and the point of view. This could be circumvented by changing the color of the packages to a neutral, transparent color, e.g. gray, or using a wire frame texture for the boxes which only displays the edges of each box.

The orientation of the helper helper lines criticized in (6) is caused by the raycaster (cf. Figure 6.2b) we utilized to identify the corresponding point on the foundation for each class. As this raycaster only identifies the intersected segment face of the foundation, which is a triangle, the computation of the center point for the corresponding square is not trivial and might impede the performance of the visualization. However, if work on this extension is continued in the future, it might be possible to find and test an efficient implementation. To improve the visibility and distinguishability of the helper lines it might be useful to allow the configuration of the thickness and color of helper lines.

<table>
<thead>
<tr>
<th>ID</th>
<th>Mean</th>
<th>SD</th>
<th>Max Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>8a</td>
<td>3.33</td>
<td>1.21</td>
<td>5</td>
</tr>
<tr>
<td>8b</td>
<td>4.33</td>
<td>0.82</td>
<td>5</td>
</tr>
<tr>
<td>8c</td>
<td>4</td>
<td>0.89</td>
<td>5</td>
</tr>
<tr>
<td>8d</td>
<td>3.83</td>
<td>0.41</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 7.15. Results of the post-test questionnaire

Conclusion

Overall we are satisfied with the usability of the implementation for our heatmap extension. Nonetheless, we detected some usability problems concerning the selection of the color gradients with our survey, which leave room for improvement. The main issues were caused by the ambiguous perception of colors in the medium color range which lead the users to estimate associated values and a general preference of the array heatmap. However,
7. Evaluation

in most cases the displayed extreme values, which are most interesting for the detection of performance issues, were identified correctly. With the mouseover feature we identified a possible improvement for the association of the color values with their specific scores.

In the closing interview we asked for the display setups of the test users which consisted of 4 different combinations of size and resolution as shown in Table 3 in Appendix B. Moreover, the displays were different models which results in different screen settings for color values, brightness and contrast. Therefore, it is hardly feasible to deliver a configured gradient that looks exactly the same on all display settings. As a solution for this issue requires further iterations of the development cycle, we suggest the implementation of different color schemes that can be chosen at will, while leaving the option for users to configure their own gradients, for future work on this expansion.

Another issue that was mentioned in closing interview was that the correctness of the displayed metrics was difficult to comprehend without further knowledge. With our tooltips we provided a text based description for each metric which enabled the users to understand their meaning, but a detailed tutorial with graphical examples could improve the overall comprehensibility of the displayed metrics. Furthermore, graphical examples could also improve the descriptions for the different settings in the configuration tab.

7.3.5 Threats to Validity

The validity of our results is threatened by several factors on which we focus in this section. First of all, the remote nature of our usability study involved the usage of different systems on the test user side. We had little control over the used software and hardware configurations of these systems. Our results show that the choice of the display already might introduce deviations from the expected results in our experiments. We can not assess how much the different system specifications of the test users influence the overall results. To handle this issue we would either require a larger group of test users with different display setups or require the same hardware for all test users.

Another threat is the number and selection of our test users. We discussed in Section 2.5 that a small group of test users will suffice for the execution of a usability study. However, it is recommended to conduct multiple iterations of the study to show improvements and guarantee for all usability problems to be identified. We did not have the capacities for further iterations, and thus might have missed some usability problems with our survey. Furthermore, we did not create a separate group of color blind people to test our adaptations. However, one of our test users was colorblind and did not perform worse than the other users. This result is not representative due to the small sample size but could indicate that our chosen color scheme is colorblind friendly.

The third issue we want to mention is the choice of our prepared example landscapes and the choice of Spring PetClinic as our sample application. Although we tried to simulate real traffic on a web page by applying different loads to the running web server, we did not necessarily generate landscapes that are representative for all similar activities. Furthermore, Spring PetClinic itself is a sample application which was created for the
demonstration of the Spring framework. Different applications might produce divergent interactions and usability problems under monitoring, if compared to our chosen sample application in the same setting.

7.4 Summary

In summary, our conducted usability survey yielded positive results regarding the usability and learnability of our implementation for this small group of test users. Our observations and the correctness rate of the answers for the post-task questionnaire match our expectations, especially for the combinations aggregated heatmap with simple heat visualization and windowed heatmap with array heat visualization. The ratings of the post-test questionnaire indicate that the tool was well received and has the potential to support software engineers with monitoring applications from a usability perspective after further optimizations.

However, as this study was focused on the usability of the implementation, we did not directly evaluate the sensibility of the selected metrics and the utility the heatmaps provide during performance analysis of a monitored application. Therefore, the resulting impact on the development and quality of software have not been clarified. These research questions have to be answered with further evaluations and applications to real world examples.
Chapter 8

Related Work

In this chapter we focus on the presentation of related work from other researchers and explain how they are related to the context of our thesis. Due to the different domains of research that are relevant for our approach we introduce them in the sections below. In Section 8.1 we introduce further articles related to software design under the aspect the city metaphor and their use of colors. Further research in this area might impact the overall structure of the *ExplorViz* visualization and therefore open up new opportunities for the application of metrics and heatmaps. Section 8.2 contains additional work on dynamic coupling metrics and their importance for designing efficient software systems. In Section 8.3 we present further uses of heatmaps and color schemes in the context of software design. Although most of them are not directly used for highlighting software metrics in a monitoring approach they still reveal useful insights about the perception and practicality of heatmaps and color theory.

8.1 City Metaphor

8.1.1 Code City

Wettel and Lanza [2007] present their approach of a 3D visualization of object-oriented software systems gravitating around the city metaphor, which resulted in further research and a PhD thesis [Wettel 2010]. Their approach produces an interactive city map with the goal to give the viewer a sense of locality to ease program comprehension. They focused on the investigation of several concepts that contribute to the urban feeling of the city such as appropriate layouts, topology and facilities to ease navigation and interaction.

With their approach they experimented with color and transparency tagging of class representations based upon specific filter criteria. Thus, they created city maps that are comparable to our heat map approach. However, based solely on static information extracted from source code, their approach does not have the capabilities of providing dynamic monitoring information and can especially not visualize software metric dynamics.

Nonetheless, their approach to coloring software landscapes by coloring the generated class visualizations directly provides a comprehensible presentation of certain metrics. An investigation of this technique might prove worthwhile for certain metrics in the future.
8. Related Work

8.1.2 Displaying Metrics for Software Landscapes

An earlier implementation of the city metaphor for object-oriented software systems was presented by Langelier et al. [2005]. Their approach already applied a color scale from blue to red to the classes to show the corresponding CBO (Coupling Between Objects) value. Furthermore they emphasize the importance of software evolution and thus present a visualization that is growing for subsequent versions of the source code.

In a later publication Langelier et al. [2008] classify their used metrics into static structural metrics and version control metrics, which can be displayed with a context switch of the visualization. Thus, they found a way to comprehensively present different types of metrics in one application. This technique is similar to the way our heatmaps are implemented into the existing ExplorViz structure. Furthermore it could prove helpful to add additional metrics and data to the visualization.

8.1.3 Applying Heatmaps to Software Landscapes

As aforementioned, Benomar et al. [2013] extended the approach of Langelier et al. [2005] by adding heatmaps to the city visualization. They provide heatmaps to visualize the progression of software evolution and software execution over time. In both cases the time dimension was measured by using the timestamps of changes in the source code and the execution traces. This data was measured over time and evaluated later on. This allowed for complex and time consuming computations and restructuring of the city landscape to place entities with similar values close to each other.

Unlike CodeCity and ExplorViz however this approach works with a 2D package representation, which makes it impossible for the former ones to use this restructuring without completely breaking the existing visualization. Furthermore the dynamic approach of ExplorViz does not allow for complex computations as the heatmaps have to be broadcast in the same interval as the landscapes in order to keep the existing update cycle. However, Benomar et al. [2013] present several techniques for heatmaps, e.g. color weaving, that can be applied to future versions of this extension to add additional data to a heatmap.

8.2 Software Metrics

In the development of software applications object-oriented programming plays a vital role [Ponnala and Reddy 2019] and with it rises the need for measurements to determine the quality and efficiency of object-oriented programs. Generally software metrics are divided into static metrics which are based upon the source code and dynamic metrics that are collected from the runtime behavior of software. ExplorViz currently solely relies on dynamic monitoring data acquired from Kieker. Amongst others Arisholm et al. [2004] and Singh and Singh [2010] presented metrics that proved useful for the creation of our heatmap feature but for a wholesome application new technologies are also of interest. The
work of Geetika and Singh [2014] and Ponnala and Reddy [2019] shows that research on dynamic software metrics is still ongoing, therefore it is most reasonable to keep an eye out for the emergence of new metrics that can be used for this approach.

Furthermore Schnoor and Hasselbring [2019] researched the correlation between static coupling metrics, dynamic unweighted coupling metrics and dynamic weighted coupling metrics. They showed that dynamic software coupling metrics are statistically significant correlated to their static counterparts while complementing each other. Thus, it might prove useful to benefit from this correlation and find a way to integrate static coupling metrics in future iterations of this extension.

8.3 Heatmaps

8.3.1 Selection of Color Schemes

MacDonald [1999] presents in his work how the human color vision and perception work in detail and which side effects come along with the choice of certain colors and color schemes. He explains how different displays ad display types create the colors for the user and which characteristics are inherent for certain technologies. However, since the article 20 years old it is slightly outdated and does not cover present technologies, such as AMOLED (Active Matrix Organic Light Emitting Diode) displays. Nonetheless, it imparts knowledge about the basics of color theory and the best application practices in different fields. The presented color selection guidelines and design principles can be used as a starting point for the development of additional and improved gradients for our extension and color based implementations in the context of ExplorViz.

8.3.2 Harmful Color Schemes

In their article Borland and Taylor Ii [2007] discuss the prevalence of rainbow color maps in scientific publications and their disadvantages compared to other color choices. They state that the typical color map is confusing to the viewer as the colors are not perceptually ordered, and thus are not ordered in the same way when given to different people. Furthermore, the common rainbow color map is obscuring compared to a gray-scale color map as the visual system perceives high spatial frequencies through changes in luminance.

We utilized the rainbow color scheme to illustrate the heat color metaphor and to be able to depict negative and positive values that can be easily distinguished at certain thresholds. To help with ordering the colors we provided a legend which contains the current color gradient. Furthermore, we implemented settings for the user to select own colors and thus customize the heatmaps. Nonetheless, this article can be used as a base for the implementation of completely different color schemes and the development of new gradients.
In this chapter we summarize our work and introduce ideas for future work and further improvements of our extension.

9.1 Conclusions

In this thesis we developed an extension for the monitoring tool ExplorViz [Fittkau et al. 2017] which displays dynamic data of applications in visualizations based upon the city metaphor. With our approach we add an additional layer to the current visualization on which we visualize the dynamicity of software coupling metrics during the execution of the monitored application. We described the selection and implementation of the metrics and the extension. For the evaluation of the extension we conducted a formative usability study which uncovered potential improvements and optimizations for the future development.

Our implementation consists of a backend extension and a frontend extension. The former one is implemented as a microservice which is responsible for the computation of four different metrics and the creation of two heatmap types for every landscape provided by the core application. Moreover, the generated metrics and heatmaps are persisted into a database after being broadcast to the frontend and can be requested by frontend services. The latter is an Ember addon which provides an extended visualization for landscapes on the application level. The adaptations include the display of heatmaps on the foundation mesh in two different styles for every metric. To improve the comprehensibility and usability of our tool we implemented a configuration route to allow the users to customize their experience and to be able to select the features which are required for specific tasks.

The evaluation in form of a usability study with a small group of six participants verified the general usability of our tool while uncovering potential improvements and optimizations for future iterations of the tool. We evaluated the different implemented combinations of heatmap types and visualization styles and came to the conclusion that the combinations of aggregated heatmap with simple heat visualization and the windowed heatmap with array heat visualization are preferred by the test users. The ratings are supported by the high average correctness of the results regarding these combinations. Overall the array heat visualization received the highest ratings due to the increased distinguishability of the pixel-like squares it is composed of. In Section 7.3.4 we discussed the uncovered usability problems and approaches to fix them.
9. Conclusions and Future Work

However, we did not evaluate the actual usefulness of our tool with regards to the individual metrics when applied to real world applications thoroughly. Moreover, the new heatmap layer of the visualization could be used in different ways we did not consider during the development of the different heatmap types. Therefore, we advise the continuation and further development and evaluation of this extension, as overall our tool provides helpful insights about the dynamicity of metrics and new ways of displaying information for software engineers.

9.2 Future Work

With the implementation of our extension we introduced a new way of displaying metrics and information within the ExplorViz application visualization. However, with this being the first approach to adding heatmaps to the existing application, our implementation has the potential for further improvement and optimization.

First of all we encountered the issue of large metric values which could completely overshadow other metrics, for example a class with a metric score that has the magnitude of 1000 would impact the scale of the gradient so much that a class with a score of 10 would completely vanish. On the other hand if the scale of the gradient was capped at some point the information about the magnitude of certain metrics would be lost. In the future it might be worthwhile to introduce a smart normalization technique for the metrics. This could, for instance, be implemented by introducing a new heatmap type.

Moreover, our evaluation produced some suggestions for improvements and additional features to increase the usability and quality of our software. The implementation of these suggestions might prove worthwhile in future versions of this extension but requires further development and evaluation. One of the requested features was a mouse over function for the classes and the gradient to show the current value of a class and highlight all classes belonging to a certain color value. The addition of this feature would prevent users from estimating metric scores and allow them to receive more precise information about the state of the monitored application. Improving the given descriptions of features and options with graphical examples and tutorials would increase the usability and comprehensibility of the entire extension. Furthermore, there is the potential for optimizing the changes we applied to the visualization to make the heatmaps more comprehensible.

As mentioned in Chapter 8 the addition of further metrics to the representation might prove worthwhile although we already chose adequate metrics as described in Chapter 4. Furthermore, the currently implemented visualization styles and chosen colors for the gradients could be optimized and extended with different tools. Although we invested a lot of time into their development there might be some tools and techniques that we missed.

Finally, we want to address the scalability of our approach. Although we did not perform tests with applications other than our sample applications, we can estimate that the computation time of our heatmaps is worse for larger applications as it is impacted by multiple factors. Many of the computations which are required for the generation of
9.2. Future Work

Our metrics are handled with nested loops for every class and every application. Thus, monitored landscapes with many large applications might slow down the generation of heatmaps too much to keep up with the landscape service. Furthermore, while it is fairly easy to implement and add new metrics to our approach, every new metric also adds up to the required time for each computed heatmap. Therefore, further exploitation of concurrency for the computations could improve the required time, and thus reduce the delay between the broadcast of a landscape and the associated heatmap.
Bibliography


Bibliography


Bibliography


Appendices
Usability Study - Heat Map Extension for ExplorViz

Welcome to this usability test for the evaluation of our heat map extension for the monitoring application ExplorViz. In this questionnaire we will begin with the introduction of our setting and the inquiry of some personal information which will help us to evaluate the following questions regarding the program. These questions require you to interact with the application to find the solutions and to answer the given questions. In the end we ask you some general questions regarding your overall experience with the software.

Introduction

ExplorViz  ExplorViz is a monitoring application used for visualizing dynamically gathered monitoring data from an inspected application. The tool shows software applications using the 3D city metaphor, i.e. programs are visualized in a city like shape to improve their comprehensibility. Programs are depicted as a gray foundation box and their packages and classes are rendered atop as green and blue boxes. The communication of classes is depicted with orange communication lines between the class packages.

Interaction is possible by using the mouse as follows: By holding the right mouse button you can turn the application. The mouse wheel is used to zoom in and out. By clicking on the application you can open and select packages to see their content. Alternatively, you can use the 'Open All Components' button in the toolbar above. By hovering over certain elements, e.g. classes, you get additional data as a pop-up window. The timeline below can be used to select previous landscapes and observe the overall communication on form of requests.

Extension  Our extension adds heatmaps as an additional source of information to the visualization. Therefore, certain metrics, which can be selected in the toolbar, are projected onto the foundation box below each class as a color. Each color represents a specific color value as shown on the corresponding legend. We do not directly show the values of the metrics but their progression over time, which is computed using different techniques. These techniques will be introduced to you during this survey. Furthermore, we implemented different visualization styles for the heatmaps for you to evaluate.
Please read the questions carefully. If you do not know how to proceed at any point please notify the moderator and wait for him to help you.

Personal information

1a. What is your occupation? □ student □ researcher □ other: _____________

1b. What is your qualification/target degree? □ Bachelor □ Master □ PhD

1c. Please name your overall years of experience / semester: _____________

Please rate your level of experience with:

1d. Programming in general: none □ □ □ □ □ expert

1e. Object oriented programming: none □ □ □ □ □ expert

1f. Java programming: none □ □ □ □ □ expert

1g. Development of web applications: none □ □ □ □ □ expert

1h. Monitoring software: none □ □ □ □ □ expert

1i. ExplorViz: none □ □ □ □ □ expert

1j. Please note the time: _____________
Overview

2a. How many different metrics can be selected for the visualization? __________

2b. Please name the metrics:

__________________________________________________________________________

__________________________________________________________________________

2c. Please give a short summary of the information each of the metrics provides:

__________________________________________________________________________

__________________________________________________________________________

2d. Please name the type of the visualized heatmap: ________________

2e. Are there classes with a neutral or non-existent metric score in the visualization?
   □ Yes  □ No

Configuration

3a. What is the difference between the heatmap types?

__________________________________________________________________________

__________________________________________________________________________

3b. Which visualization styles can be chosen?

__________________________________________________________________________

3c. Please give any thoughts and comments about the provided configuration options.

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________

3d. Are there any options missing in your opinion?

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________
Aggregated Heat Map and Simple Heat Visualization

If not already selected, please select the aggregated heat map type and the simple visualization. Then return to the heat map route and then select one of the timestamps with 114 total requests on the timeline below.

4a. What are approximately the highest and lowest represented metric scores for the export coupling metric?
   highest: 
   lowest: 

4b. Name at least one class with a high metric score for the instance count:

4c. Name at least one class with a low but existing export coupling score:

4d. Name at least one class with a neutral or non-existent import coupling score:

4e. Please rate this combination of heatmap type and visualization style
   very bad □ □ □ □ □ very good

Aggregated Heat Map with Array Heat Visualization

Please select the array heat visualization in the configuration tab and return to the heat map view. Then select the timestamp with 8 total requests int the timeline below.

5a. What are approximately the highest and lowest represented metric scores for the dynamic key metric?
   highest: 
   lowest: 

5b. Name at least one class with a high import coupling score:

5c. Name at least one class with a low export coupling score:

5d. Name at least one class with a medium score for the instance count:

5e. Please rate this combination of heatmap type and visualization style
   very bad □ □ □ □ □ very good
**Windowed Heat Map and Simple Heat Visualization**

*Please visit the configuration route to select the windowed heat map type and the simple heat visualization. Then return to the heat map view and select the timeline entry with 116 requests.*

6a. What are approximately the highest and lowest represented metric scores for the import coupling metric?

- highest: _______
- lowest: _______

6b. Name at least one class with a high, positive export coupling score:

6c. Name at least one class with a low, negative score for the instance count metric:

6d. Name at least one class with a medium dynamic key score:

6e. Please rate this combination of heatmap type and visualization style

very bad ☐ ☐ ☐ ☐ ☐ very good

**Windowed Heat Map and Array Heat Visualization**

*Please visit the configuration route to select the windowed heat map type and the array heat visualization. Then return to the heat map view and select the timeline entry with 124 requests.*

7a. What are approximately the highest and lowest represented metric scores for the export coupling metric?

- highest: _______
- lowest: _______

7b. Name at least one class with a high, positive metric score for the instance count metric:

7c. Name at least one class with a low, negative dynamic key score:

7d. Name at least one class with a neutral score for the instance count metric:

7e. Please rate this combination of heatmap type and visualization style

very bad ☐ ☐ ☐ ☐ ☐ very good
Debriefing Questions

8a. How difficult was it to understand the different metrics and gradients?
   very difficult □ □ □ □ □ very easy

8b. How difficult was the use and configuration of the extension?
   very difficult □ □ □ □ □ very easy

8c. How would you rate the integration of our extension into the existing software?
   very bad □ □ □ □ □ very good

8d. How was your overall impression of the extension?
   very bad □ □ □ □ □ very good

8e. Do you have any comments/suggestions concerning the usage of the program? Was something ambiguous or not clear?

________________________________________________________________________

________________________________________________________________________

________________________________________________________________________

8f. Please note the time: __________
Table 1. Results of the pre-test and post-task questionnaire

| ID | 1a | 1b | 1c | 1d | 1e | 1f | 1g | 1h | 1i | 1j | 2a | 2b | 2c | 2d | 2e | 3a | 3b | 3c | 3d |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1  | 1  | 1  | 8  | 3  | 3  | 4  | 3  | 2  | 1  | 18:33 | 1  | 4  | 4  | 1  | 1  | 2  | 2  | -  | -  |
| 2  | 3  | 1  | 20 | 5  | 5  | 4  | 3  | 3  | 1  | 19:48 | 1  | 4  | 4  | 1  | 1  | 2  | 2  | -  | -  |
| 3  | 1  | 2  | 12 | 3  | 3  | 3  | 1  | 2  | 2  | 15:07 | 1  | 4  | 4  | 1  | 1  | 2  | 2  | Disabling helper lines is an unnecessary feature. |
| 4  | 3  | 1  | 18 | 4  | 4  | 4  | 4  | 3  | 1  | 14:23 | 1  | 4  | 4  | 1  | 1  | 2  | 2  | The configuration options good, i was able to get an better overview using the opacity value and the visualization style |
| 5  | 2  | 3  | -  | 4  | 4  | 4  | 4  | 4  | 5  | 14:05 | 1  | 4  | 4  | 1  | 1  | 2  | 2  | Many configuration options for the heatmap, but without further knowledge difficult to consider if changing them is useful or not. A graphical example might be useful. Option Buttons for simpleheat gradient configuration and array heatmap gradient configuration are transparent and are visible for the user. |
| 6  | 2  | 3  | 16 | 4  | 4  | 4  | 4  | 3  | 5  | 15:30 | 1  | 4  | 4  | 1  | 1  | 2  | 2  | Looks great, has detailed information and is easy to use |

- Indicates no comment or preference.
Table 2: Results of the post-task questionnaire

<table>
<thead>
<tr>
<th>ID</th>
<th>4</th>
<th>4a</th>
<th>4b</th>
<th>4c</th>
<th>4d</th>
<th>4e</th>
<th>5</th>
<th>5a</th>
<th>5b</th>
<th>5c</th>
<th>5d</th>
<th>5e</th>
<th>6</th>
<th>6a</th>
<th>6b</th>
<th>6c</th>
<th>6d</th>
<th>6e</th>
<th>6f</th>
<th>7</th>
<th>7a</th>
<th>7b</th>
<th>7c</th>
<th>7d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 3. Results of the post-test questionnaire

<table>
<thead>
<tr>
<th>ID</th>
<th>8a</th>
<th>8b</th>
<th>8c</th>
<th>8d</th>
<th>8e</th>
<th>8f</th>
<th>Display setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>19:19</td>
<td>24 inch, 1920x1080p</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>20:24</td>
<td>27 inch, 1920x1080p</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>15:38</td>
<td>24 inch, 2650x1440p</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>15:18</td>
<td>24 inch, 1920x1080p</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>14:31</td>
<td>27 inch, 2650x1440p</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>16:00</td>
<td>27 inch, 2650x1440p</td>
</tr>
</tbody>
</table>

- Mouseover to get the exact color value.
- The default color value for the 0 value of the heatmap with style "array" had the same color value as the packages/the ground; color value should be visible in configuration.
- Landscape does not automatically reload after switching from config; confusion under which condition 0 is depicted by "no color".
- Everything was comprehensible; mouseover/hover effect for the heatmap legend is desirable to avoid estimating values.
- A tutorial which explains the usage of the extension and employed metrics would be very useful. Colors are sometimes hard to distinguish. Transparency of packages led to losing the hierarchy of the application. Angles of Instances/Classes seem to differ in the visualization - is this a bug or design? The pixel-like color-usage in some of the heatmaps are way better recognizable than the gradient.
- The understanding of the metrics is probably not so difficult, but a small example in source code would be nice. The simple Heat visualization is harder to distinguish for me. I like the array one. The help lines are not centered in the array mode, I would fix that. I think a bigger application with many more classes would be more difficult to analyze, since the help lines are not easy to spot in this case.