

## Conceptual Architecture Description and Evaluation Report (T2)

Delivery Date: June 11, 2023

### Team Members:

Yunus Emre Terzi

Gamze Ergin

## 1. OSS PRODUCT OVERVIEW

**Table 1.** Description of the OSS product

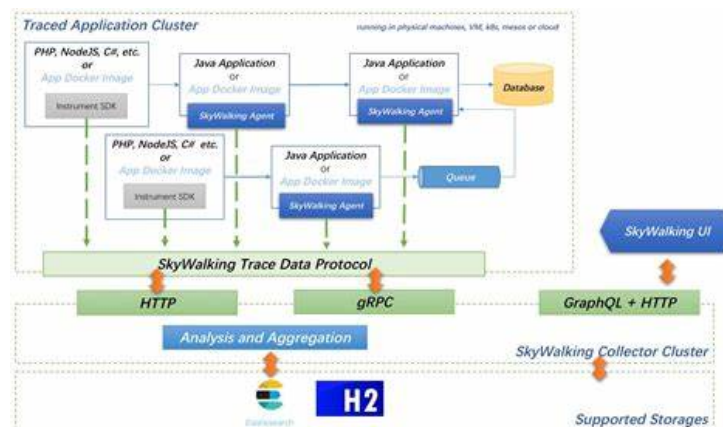
Name of OSS Product:	<b>Apache SkyWalking</b>
Description of OSS product:	This is an open-source APM( application performance monitor) system. It is designed for: <ul style="list-style-type: none"><li>● microservices,</li><li>● cloud-native,</li><li>● container-based architectures.</li></ul> It includes monitoring, tracing, and diagnosing capabilities for a distributed system in Cloud Native architecture.
URL of OSS product:	<a href="https://github.com/apache/skywalking">https://github.com/apache/skywalking</a>
Size of OSS product (KLOC):	204.133

## 2. CONTEXT DIAGRAM

### The projects web site

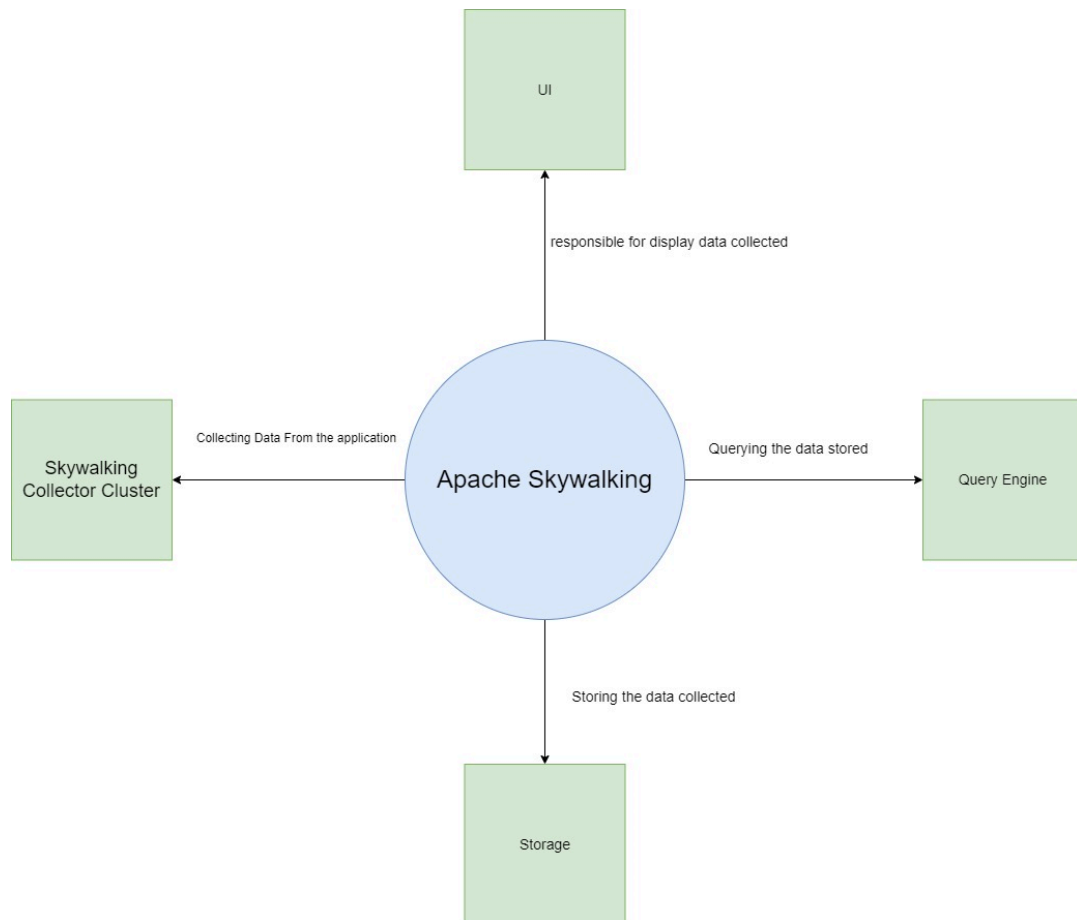
[Apache SkyWalking](https://skywalking.apache.org/)

The Apache Skywalking project is a distributed tracing system. It can be used to collect and analyze performance data from microservices applications. The project consists of a number of components, including a collector, a storage, a query engine, and a UI.



This is the shared document of the product

We can create a context diagram based on the product document details.



#### **UI:**

The IU is responsible for displaying the data stored in the system. UI can use these to show data:

- graphs,
- charts,
- tables

With a designed interface the users can filter and drill down into the data.

#### ***Skywalking Collector Cluster:***

The collector captures information about each request, such as the start time, the end time, and the number of calls made. The collector then sends this data to the storage.

#### ***Storage:***

This is used to store all the obtained data from the collector.

#### ***Query Engine:***

The query engine can be used to generate reports, dashboards, and alerts. The query engine can also be used to export the data to other systems.

### 3. CONCRETE ARCHITECTURE EVALUATION SUMMARY

We can make comments about architecture with the help of C&K metrics and the dependency graph.

We noticed that there are a few modules with complex relationships in the dependency graph. That means the project's architecture is not complex overall since complex ones are in the minority. On the other hand, we got good C&K metric values for the project. It proves that our project's implementation phase is well-organized.

We observed that our project has a well-designed concrete architecture by making inferences on the dependency graph and C&K metrics.

### 4. TOOLS USED FOR CONCEPTUAL ARCHITECTURE DESCRIPTION

**Table 2.** The list of tools used for architectural description

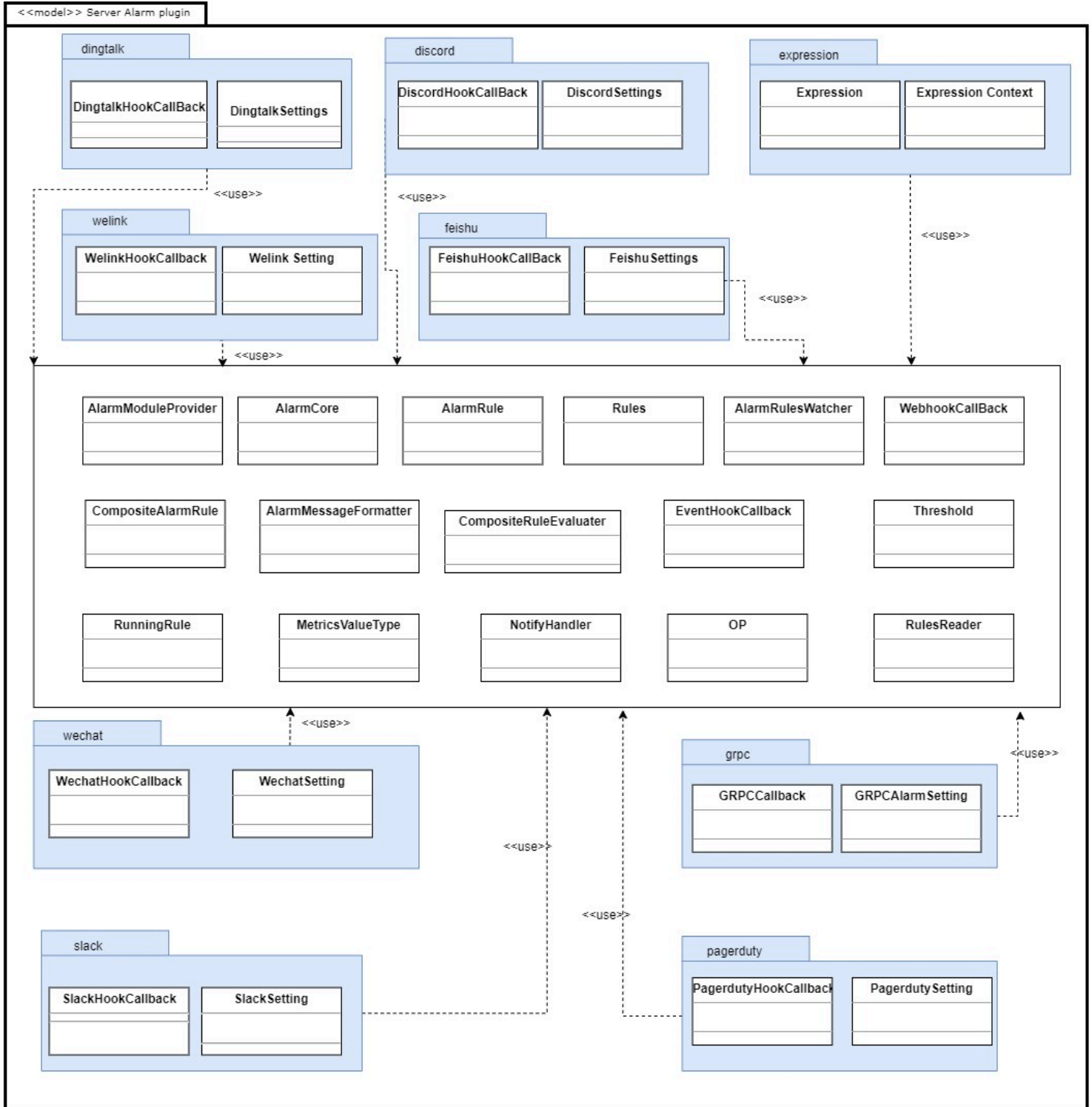
Tool Name	Purpose of Use	URL
Visual Paradigm	<ul style="list-style-type: none"><li>To scratch UML's</li></ul>	<a href="https://www.visual-paradigm.com/">https://www.visual-paradigm.com/</a>
Draw.io	<ul style="list-style-type: none"><li>To scratch UML's</li></ul>	<a href="https://app.diagrams.net/">https://app.diagrams.net/</a>
UML class diagrams plugin	<ul style="list-style-type: none"><li>To find to UML structure of the OSS product.</li></ul>	<a href="https://www.jetbrains.com/help/idea/class-diagram.html">https://www.jetbrains.com/help/idea/class-diagram.html</a>
Dependency Analysis plugin	<ul style="list-style-type: none"><li>To analyze code based on its dependencies.</li></ul>	<a href="https://www.jetbrains.com/help/idea/dependencies-analysis.html">https://www.jetbrains.com/help/idea/dependencies-analysis.html</a>
Creating Context Diagram	<ul style="list-style-type: none"><li>To create the context diagram.</li></ul>	<a href="https://app.creately.com/">https://app.creately.com/</a>

### 5. DESCRIPTION OF MODULE VIEW(S)

This OSS product is well-structured based on the code architecture. But when we choose to look at the more comprehensive modules it was hard to describe them with module views. They have more detailed architecture inside and if we choose the examine them with it, it was going to be superficial. So we choose 1 module with a large scale and 2 module views with a small scale.

## 1-Server Alarm Plugin module view for the Apache SkyWalking

This module represents the core logic for managing alarms based on configured rules in the Apache SkyWalking APM project. It periodically checks the rules, evaluates conditions, and triggers alarms if necessary while supporting composite rules and customizable alarm callbacks.



### ***The use design of each package and class in this module***

- ***DingTalk, discord, expression, welink, feishu, wechat, grpc, slack, pagerduty packages:***

These packages are specified for each platform. It involves two types of classes. The first class is for implementing the “webhook” method for each platform. Webhook is a method of communication used in web development and API integrations. It allows one application to send data to another application in real time whenever a specific event or trigger occurs. This package is used for *SkyWalking alarm platform webhook API*.

The second class type is used for keeping the set rules for each platform.



*(the examples from platforms)*

- ***AlarmCore:***

The alarm core includes metrics values in certain time windows based on alarm settings. By using its internal timer trigger and the alarm rules to decide whether send the alarm to the database and webhook(s).

- ***AlarmMessageFormatter:***

This is formatted especially for alarm messages. Format string in alarm-settings.yml, such as:

- Successful rate of endpoint {name} is lower than 75%

- ***AlarmRulesWatcher:***

Alarm rules' settings can be dynamically updated via configuration center(s), this class is responsible for monitoring the configuration and parsing them into {@link Rules} and {@link #runningContext}.

- ***CompositeAlarmRuleEvaluator:***

Evaluate composite rule using expression eval.

- ***EventHookCallback:***

When an alert is present, an event is generated for each alert message. These events are then sent to the internal event analyzer.

- ***RulesReader:***

Rule Reader parses the given `alarm-settings.yml` config file, to the target {@link Rules}.

- ***RunningRule:***

RunningRule represents each rule in running status. Based on the {@link AlarmRule} definition.

- ***WebhookCallBack:***

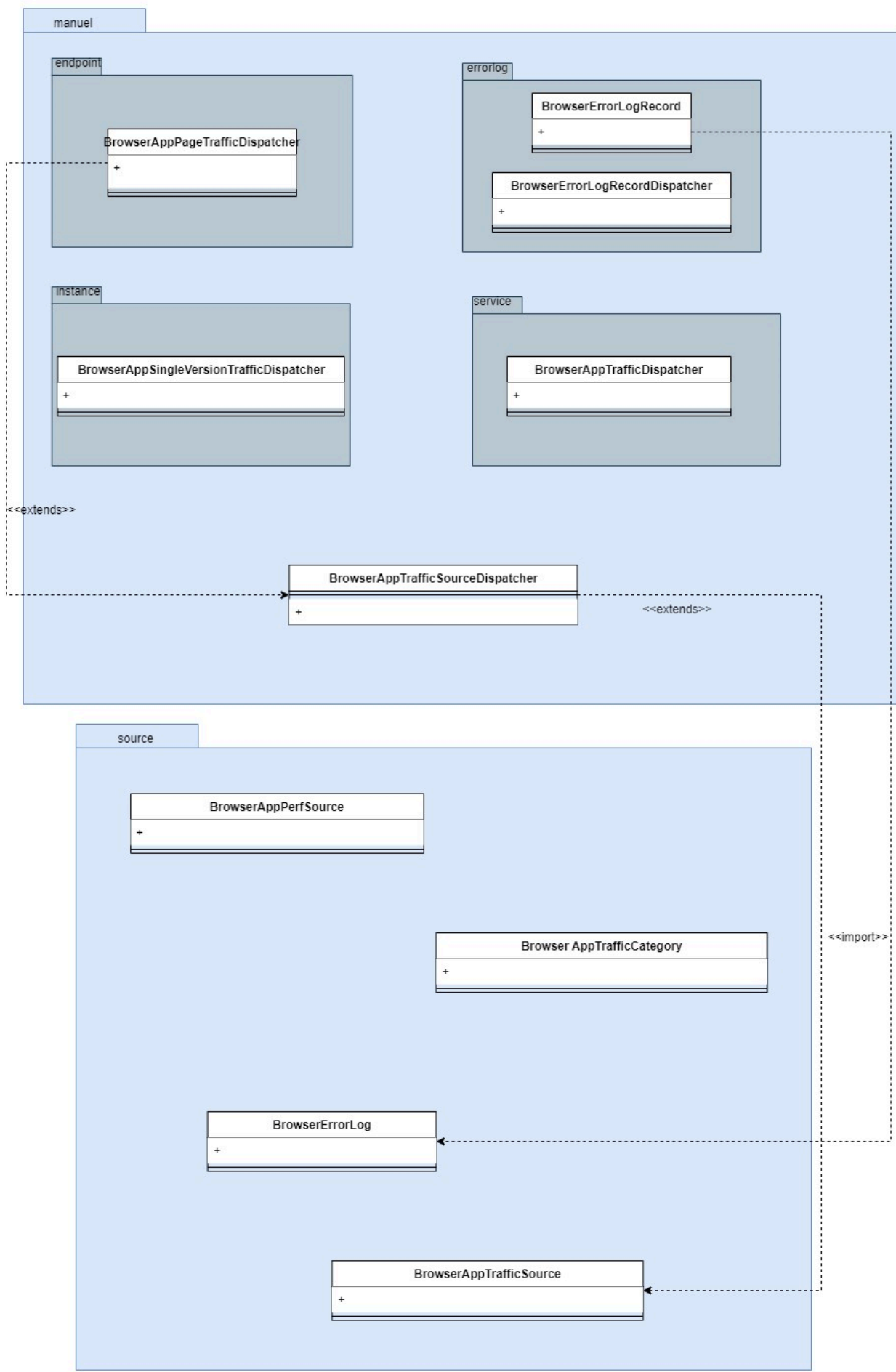
Use SkyWalking alarm webhook API to call a remote endpoint.

- ***Additional classes AlarmModuleProvider, AlarmRule, CompositeAlarmRule, MetricsValueType, NotifyHandler, OP, Rules, Threshold***

## **2-Server-core browser module view for the Apache SkyWalking**

The OAP (Observability Analysis Platform) server is the core module of Apache SkyWalking. In this module, we wanted to focus on the submodule called browsers. It allows for the collection, analysis, and visualization of performance metrics, errors, and other relevant data from browser-based applications. We can visualize that the module has a very well-defined code architecture. It is divided into packages based on the need.

<<server-core browser module >>



## ***The use design of each package and class in this module***

### ***Manual package:***

- ***endpoint/ BrowserAppPageTrafficDispatcher:***

This code demonstrates the dispatch and transformation of browser page traffic events into endpoint traffic instances within the Apache SkyWalking project's browser traffic analysis module.

- ***errorlog/ BrowserErrorLogRecord:***

This code represents a record class for storing and processing browser error log data within the Apache SkyWalking project.

- ***errorlog/BrowserErrorLogRecordDispatcher:***

This code defines a dispatcher class that receives browser error log events.

- ***instance/BrowserAppSingleVersionTrafficDispatcher:***

This code defines a dispatcher class that receives browser application traffic events.

- ***service/BrowserAppTrafficDispatcher:***

This code defines a dispatcher class that receives browser application traffic events.

- ***BrowserAppTrafficSourceDispatcher:***

This code provides a template for implementing dispatchers for browser application traffic events.

### ***Source package:***

- ***BrowserAppPerfSource:***

Browser performance details.

- ***BrowserAppTrafficCategory:***

From the BrowserPerfData.

NORMAL,

\* From the BrowserErrorLog and BrowserErrorLog#firstReportedError = true.

FIRST\_ERROR,

\* From the BrowserErrorLog and BrowserErrorLog#firstReportedError = false.

ERROR



- **BrowserErrorLog:**

Browser error logs raw data.

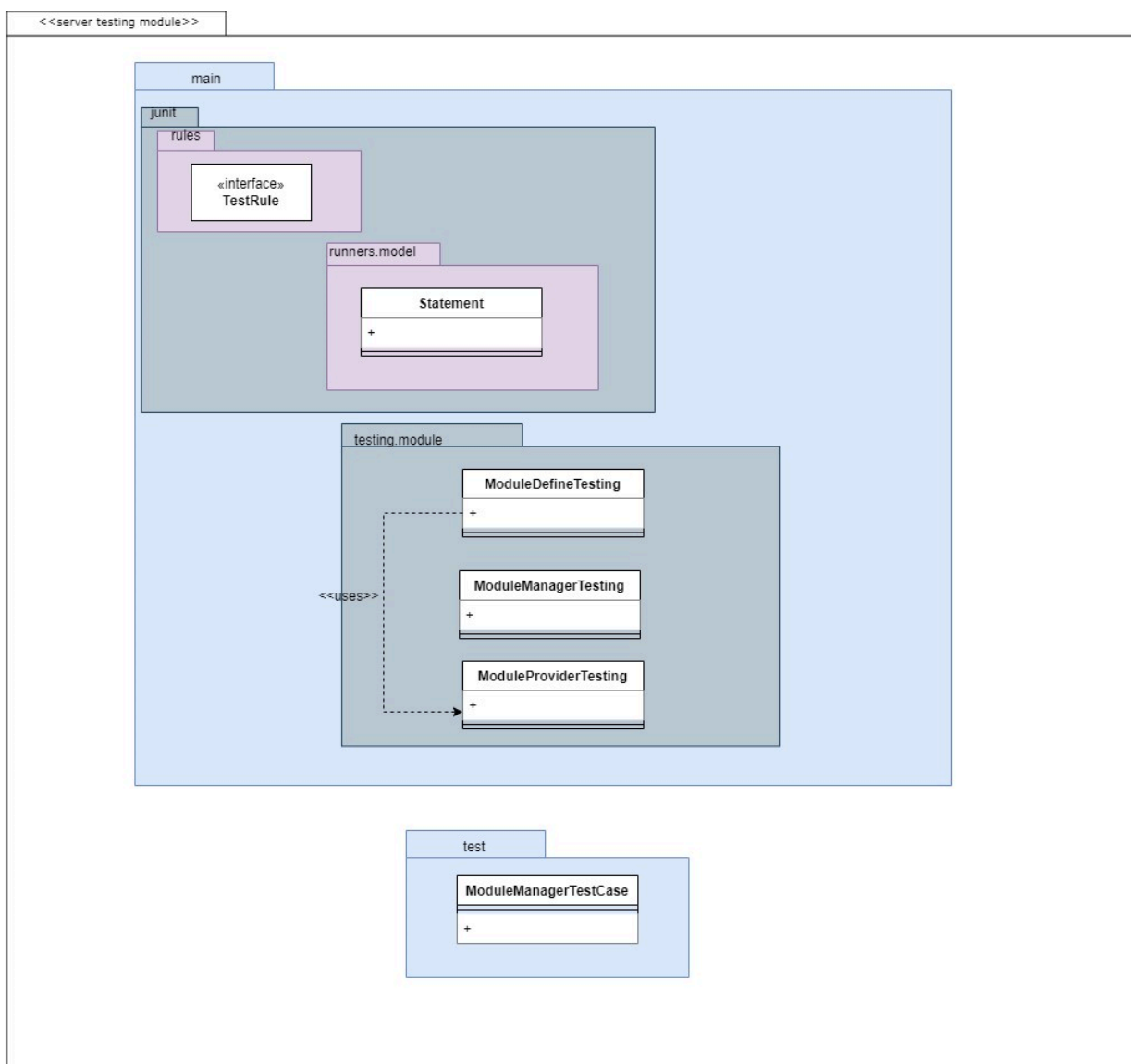
- **BrowserAppTrafficSource:**

From js, the client reported access traffic.

- **Additional classes** *BrowserAppPagePerf, BrowserAppPageTraffic, Browser AppPerf, Browser AppSingleVersionPerf, BrowserAppSingleVersionTraffic, BrowserAppTraffic, BrowserErrorCategory*

### 3-Server testing module view for the Apache SkyWalking

This module is created to see the correctness, performance, and integration aspects of the Apache SkyWalking server. It's one of the most important modules in this project.



### ***The use design of each package and class in this module***

#### ***Test package:***

- ***ModuleManagerTestCase:***

This code adds a test model to the module manager.

#### ***Main package:***

- ***junit package:***
  - ***rules/TestRule interface***
  - ***runners.model/Statement***
- ***testing.module:***
  - ***ModuleDefineTesting:***

This code defines a testing module implementation in the Apache SkyWalking server testing module.

- ***ModuleManagerTesting:***

This code represents a testing module manager in the Apache SkyWalking server testing module.

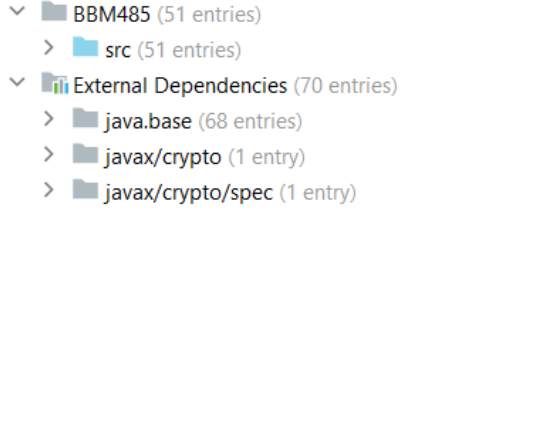
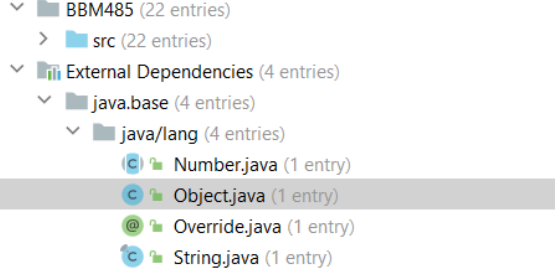
- ***ModuleProviderTesting:***

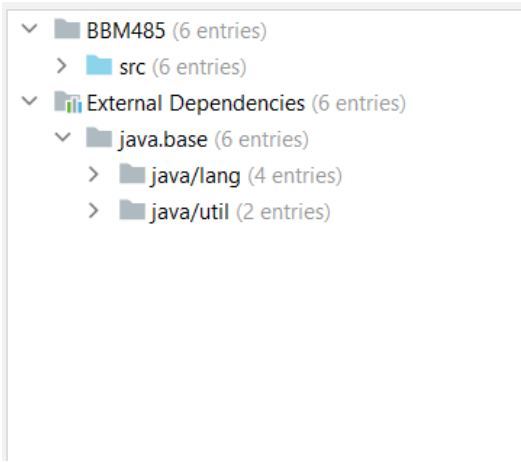
This code represents a testing module provider in the Apache SkyWalking server testing module.

## **5.2. Traceability of the Module View(s) to the Dependency Graph(s)**

The dependency graph of the product was not capable enough to tell details about the dependency view so I added all dependency analysis results in the references section.

**Table 3.** Traceability of the module view(s) to the dependency graph(s)

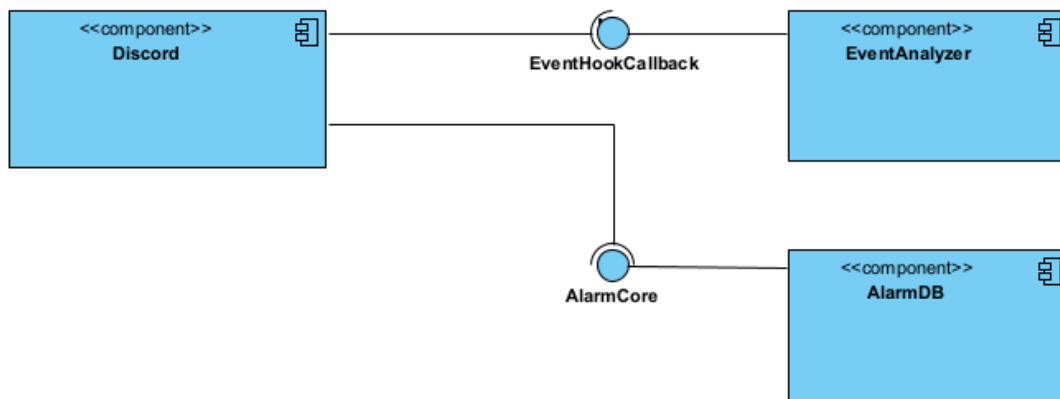
Module View (number or name)	Dependency Graph (number or name)
Server Alarm Plugin module view	Server Alarm Plugin dependency view
<p>Modules:</p> <ul style="list-style-type: none"> <li>• AlarmCore,</li> <li>• AlarmMessageFormatter,</li> <li>• AlarmRulesWatcher,</li> <li>• CompositeAlarmRuleEvaluator,</li> <li>• EventHookCallback,</li> <li>• RulesReader,</li> <li>• RunningRule,</li> <li>• WebhookCallBack,</li> <li>• AlarmModuleProvider,</li> <li>• AlarmRule,</li> <li>• CompositeAlarmRule,</li> <li>• MetricsValueType,</li> <li>• NotifyHandler,</li> <li>• OP,</li> <li>• Rules,</li> <li>• Threshold</li> <li>• DingTalk,</li> <li>• Discord,</li> <li>• Expression,</li> <li>• WeLink,</li> <li>• Feishu,</li> <li>• WeChat,</li> <li>• gRPC,</li> <li>• Slack,</li> <li>• PagerDuty</li> </ul>	 <p>121 usages</p>
Server-core browser module view	Server-core browser dependency view
<p><b>Manual package:</b></p> <ul style="list-style-type: none"> <li>• <i>endpoint/BrowserAppPageTrafficDispatcher</i></li> <li>• <i>errorlog/ BrowserErrorLogRecord</i></li> <li>• <i>errorlog/BrowserErrorLogRecordDispatcher</i></li> <li>• <i>instance/BrowserAppSingleVersionTrafficDispatcher</i></li> <li>• <i>service/BrowserAppTrafficDispatcher</i></li> <li>• <i>BrowserAppTrafficSourceDispatcher</i></li> </ul> <p><b>Source package:</b></p> <ul style="list-style-type: none"> <li>• BrowserAppPerfSource:</li> <li>• BrowserAppTrafficCategory:</li> <li>• BrowserErrorLog:</li> </ul>	 <p>26 usages</p>

<ul style="list-style-type: none"> <li>• <i>BrowserAppTrafficSource:</i></li> <li>• <i>Additional</i> <i>classes</i>  <i>BrowserAppPagePerf,</i>  <i>BrowserAppPageTraffic,</i> <i>Browser</i>  <i>AppPerf,</i> <i>Browser</i>  <i>AppSingleVersionPerf,</i>  <i>BrowserAppSingleVersionTraffic,</i>  <i>BrowserAppTraffic,</i>  <i>BrowserErrorCategory</i></li> </ul>	
<b>Server testing module view</b>	<b>Server testing module view</b>
<p><b>Test package:</b></p> <ul style="list-style-type: none"> <li>• <i>ModuleManagerTestCase</i></li> </ul> <p><b>Main package:</b></p> <ul style="list-style-type: none"> <li>• <i>junit package:</i> <ul style="list-style-type: none"> <li>○ <i>rules/TestRule interface</i></li> <li>○ <i>runners.model/Statement</i></li> </ul> </li> <li>• <i>testing.module:</i> <ul style="list-style-type: none"> <li>○ <i>ModuleDefineTesting</i></li> <li>○ <i>ModuleManagerTesting</i></li> <li>○ <i>ModuleProviderTesting</i></li> </ul> </li> </ul>	 <p><b>12 usages</b></p>

## 6. DESCRIPTION OF COMPONENT AND CONNECTOR (C&C) VIEW(S)

A C&C view describes a runtime structure of the system—what components exist when the system is executing and how they interact during the execution.

### C&C View 1.



#### Components

- **Dicord**

Discord is a popular communication platform that allows users to chat, voice chat and connect with other in various communities. In our project, usage of discord can trigger other events of the system or create an alarm for users.

- **EventAnalyzer**

This component is responsible for analyzing the events from different parts of the system. In this view, it is responsible for analyzing signals from discord.

- **AlarmDB**

AlarmDB basically responsible for keeping Alarm data in the storage. It is responsible for storing Alarm objects that are caused by discord.

#### Connectors

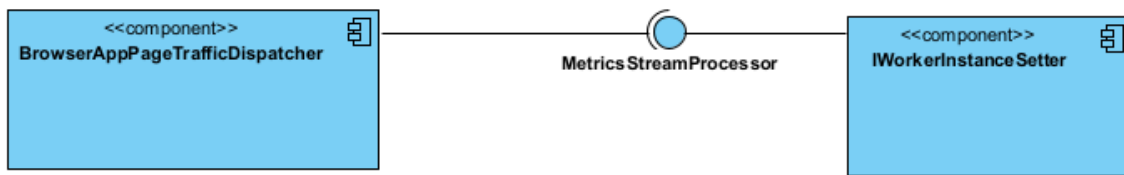
- **EventHookCallback**

This object connects Discord and EventAnalyzer components. It is responsible for gathering event triggers from Discord and converting them to events. After that it transfers them to EventAnalyzer.

- **AlarmCore**

This object connects Discord and AlarmDB. It is responsible for gathering alarm signals from Discord and converting them to alarms. After that it transfers them to AlarmDB.

## C&C View 2.



### Components

- **BrowserAppPageTrafficDispatcher**

This component defines and creates objects from inputs of the user. It is responsible for sending them to different parts of the system.

- **IWorkerInstanceSetter**

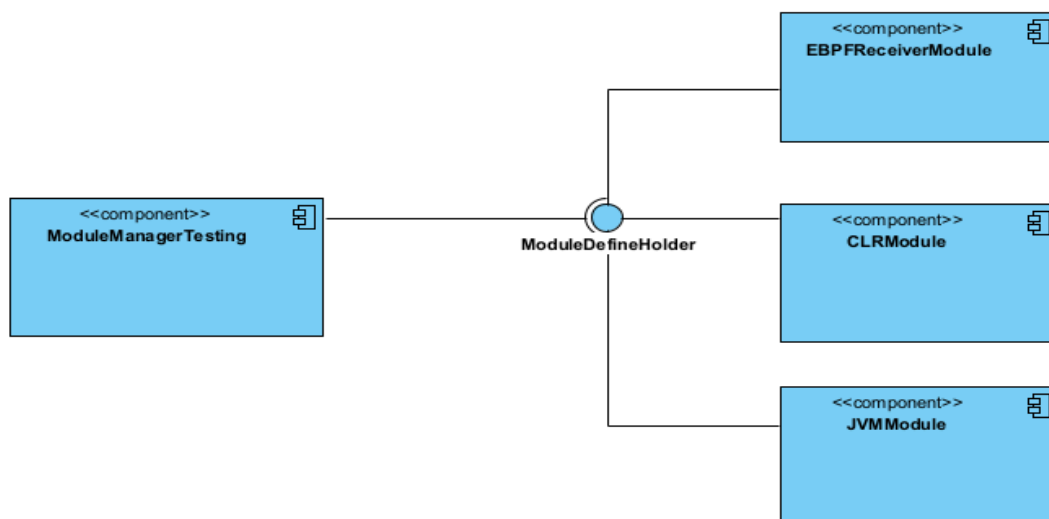
This component is for creating worker instances. It needs worker's info in order to create it.

### Connectors

- **MetricsStreamProcessor**

This object connects **BrowserAppPageTrafficDispatcher** and **IWorkerInstanceSetter** components. It is responsible for collecting user inputs from **BrowserAppPageTrafficDispatcher** and converting them to Stream objects. It passes stream objects to **IWorkerInstanceSetter**.

## C&C View 3.



## Components

- **ModuleManagerTesting**

This component manages tests for modules. It is responsible for managing generic tests for different modules.

- **EBPFRecieverModule**

eBPFR is used for safely and efficiently extend the capabilities of the kernel without requiring to change kernel source code or load kernel modules.

EBPFRecieverModule is responsible for receive eBPFR module.

- **CLRModule**

Common Language Runtime is programming that manages the execution of programs written in any of several supported languages. CLRModule is responsible for receive CLR module.

- **JVMModule**

A Java virtual machine (JVM) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode. JVMModule is responsible for receive JVM module.

## Connectors

- **ModuleDefineHolder**

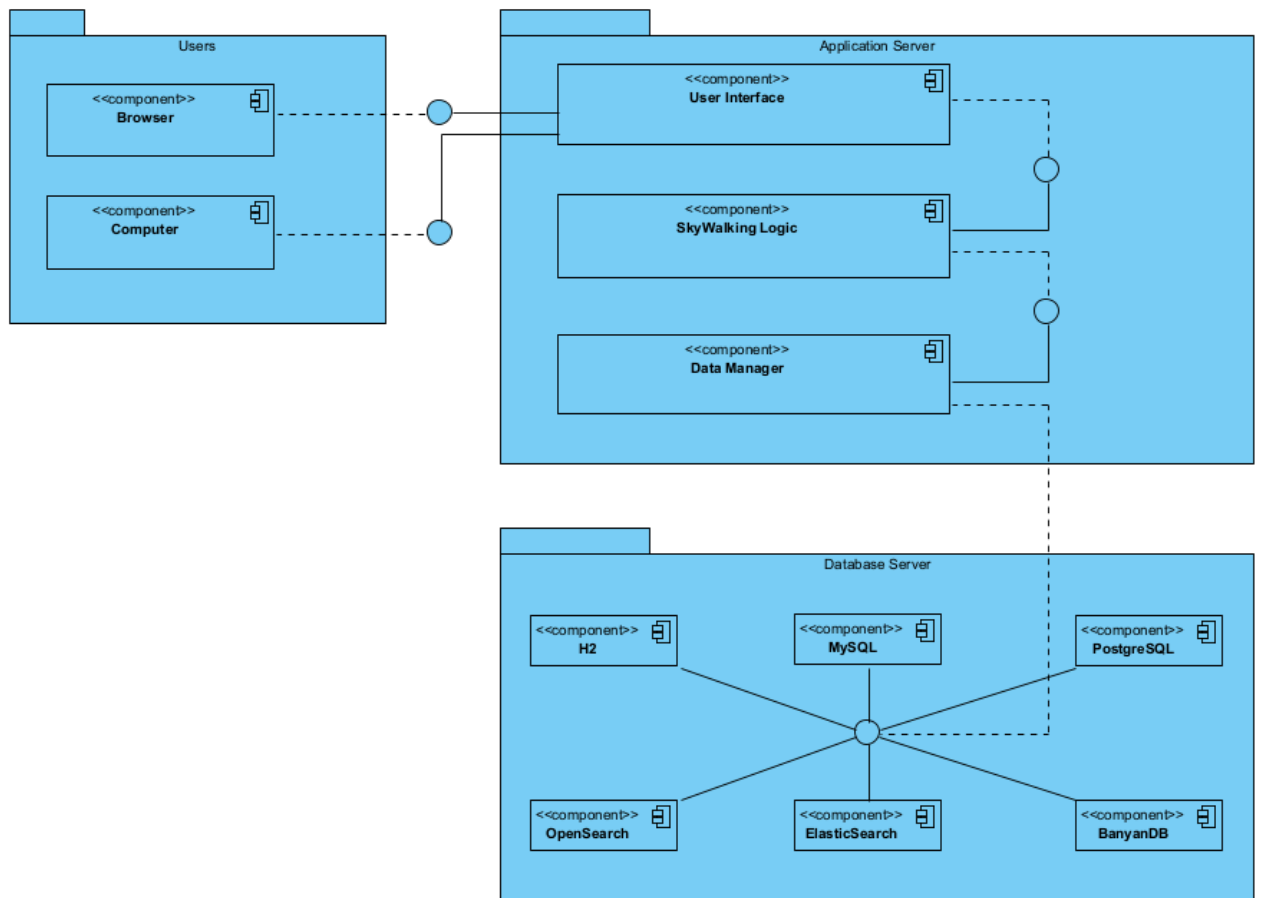
This object connects ModuleManagerTesting and other components. It is responsible for defining a module and passing tests that are arranged by ModuleManagerTesting to them.

**Table 4.** Traceability of the components in the C&C view(s) to the elements in the module view(s).

C&C view	Component	Module view	Element
C&C view 1	Discord	Server Alarm Plugin module	Discord
C&C view 1	EventHookCallback	Server Alarm Plugin module	EventHookCallback
C&C view 1	AlarmCore	Server Alarm Plugin module	AlarmCore
C&C view 2	BrowserAppPageTrafficDispatcher	Server-core browser module view	BrowserAppPageTraffic Dispatcher
C&C view 3	ModuleManagerTesting	Server testing module view	ModuleManagerTesting

## 7. DESCRIPTION OF ALLOCATION VIEW

Allocation view contains views of elements related to the deployment of the software onto hardware.



### Users

- **Browser**

We can define it as a device. It is responsible for providing online usage of SkyWalking.

- **Computer**

We can define it as a device. It is responsible for providing usage of SkyWalking through machine.

### Application Server

- **User Interface**

It is a interface for users of SkyWalking. It is responsible for presenting easy usage of SkyWalking to the users. Also it makes interaction between application and users.



- **SkyWalkingLogic**

Application itself. It is responsible for executing system functionality.

- **Data Manager**

It is responsible for making data transfers between application and database servers.

## **Database Server**

- **H2**

H2 is an open source lightweight Java database. It is responsible for data storing.

- **MySQL**

MySQL is a relational database management system developed by Oracle that is based on structured query language. It is responsible for data storing.

- **PostgreSQL**

PostgreSQL is a powerful, open source, object-reational database system. It is responsible for data storing.

- **OpenSearch**

OpenSearch is a fully open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and clickstream analysis. It is responsible for data storing.

- **ElasticSearch**

Elasticsearch is a distributed, free and open search and analytics engine for all types of data. It is responsible for data storing.

- **BanyanDB**

Banyan DB is an observability database, aims to ingest, analyze and store Metrics, Tracing and Logging data. It is responsible for data storing.

## 8. DESCRIPTION OF SPECIFIC QUALITY ATTRIBUTE (QA) SCENARIO

### QA-1) Customer complains about the performance

*Quality attributes: Reliability, usability*

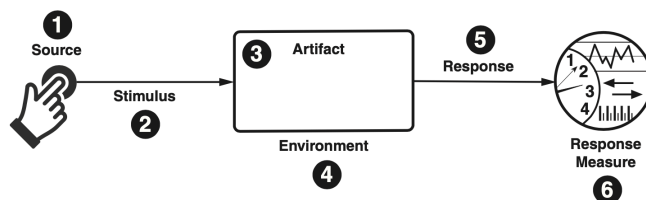
*Reference to the question(s) from T1: Is this product stable enough? (Q5), Is this product testable enough? (Q4)*

We can check the feedback from the customers about the OSS product. The customer provides logs from the production environment. The problems determined by users can be found such as;

- slow requests,
- high CPU usage,
- Database bottlenecks,
- memory leaks,
- other potential problems.

By using Apache Skywalking to analyze logs, developers can improve the maintainability of OSS products by:

- Identifying performance bottlenecks,
- Fixing performance bottlenecks,
- Providing developers with insights into how their code changes affect performance.



#### 1)Source: Customer

The source is the customer who has complaints about the software.

#### 2)Source of stimulus: Performance complaints

This is the event that triggers the problem.

#### 3)Environment: Runtime

This portion specifies when the problem occur. It is at the runtime in our case.

#### 4)Artifacts: Logs

Artifacts are a collection of systems, the whole system, or some piece or pieces of it. In this case, it's the data used to investigate the problem.

#### 5)Response: Analyze logs

The activity undertaken as the result of the arrival of the stimulus.

**6)Response Measure: Performance bottleneck is identified and fixed**

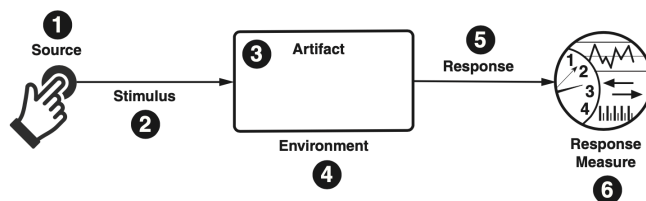
The outcome of the response.

**QA-2) Developer wants to change the user interface. This change will be made to the code at design time. It takes more time than it needs.**

*Quality attributes: Maintainability, Modifiability*

*Reference to the question(s) from T1: Are the complexity of classes high in general? (Q2), Is this product's code readable? (Q6)*

It takes more time than it needs, that means making changes on software is hard and time consuming. The reason behind that could be the complex classes and code parts that are hard to read. Developers can refactor old classes and functions in order to make changes on software easy and take less time. It also improves the product maintainability.



**1)Source: Developer**

The source is the developer who wants to change the user interface.

**2)Source of stimulus: Request to make a modification**

This is the event that triggers the problem.

**3)Environment: Design time**

This portion specifies when the change is done. It is at the design time in our case.

**4)Artifacts: User interface**

Artifacts are a collection of systems, the whole system, or some piece or pieces of it. In this case, it's the user interface.

**5)Response: Deploys modification**

The change deployed by the developer with no side effects.

**6)Response measure: Modification should take less time**

The changes on software should be done quickly and harmless.

**Table 5.** Traceability of the specific quality attribute scenario(s) to the question(s) and answer(s) in your GQM tree

Specific QA scenario	Question in GQM tree	Explanation (or rationale)
QA-1	Is this product stable enough? (Q5)	We can detect the problem in the software with the help of the customer's logs. For example, it can be in the database if the software is experiencing performance issues. The logs can detect which queries are taking the longest time to execute. We can improve the maintainability of the OSS product.
QA-1	Is this product testable enough? (Q4)	If our code is not testable enough when we release the version to our customers there will be an unexpected number of bugs and errors we had to face. To prevent that we can create a code structure more testable.
QA-2	Are the complexity of classes high in general? (Q2)	Software complexity is a natural byproduct of the functional complexity that the code is attempting to enable. When a system becomes more complex, it becomes harder to understand how different parts of software interact with each other. Thus it becomes more difficult to make changes on software. We can prevent that by reducing complexity of classes with refactoring old complex classes.
QA-2	Is this product's code readable? (Q6)	Readability is a measurement of how easy it is to understand code. It refers to the clarity and organization of the code, as well as the use of clear and understandable names for variables, functions, and classes. Readability makes it easier to understand the software system and make changes to the software. So readable code make developers to make changes easily and quickly.

## 9. SUGGESTIONS FOR IMPROVEMENT OF THE PRODUCT ARCHITECTURE

### Suggestion-1: The code should be more testable

In our first T1 report we were looking answer to the “**Is this product testable enough?**” (Q4 in the GQM tree) question. Our metrics for the question were:

- NOC: 0,30 **(1,3)**
- RFC: 7.98 **(6,36)**

The low NOC gives limited opportunities. If the base class is not properly abstracted, it may require more testing and lead to increased testing effort. A low NOC can make it more difficult to test and maintain the codebase.

The code's testability was something we found insufficient in the first report too.

In this report, we examined the server testing module view and for the QA scenario we were creating a scenario that supports the **"Is this product testable enough?"** question in (QA-1).

Our suggestions for this OSS product are:

- ***Use a code coverage tool:***

A code coverage tool can help you to measure how much of your code is being tested by your unit tests. This can help you to identify areas of your code that are not being tested.

- ***Use a unit testing framework:***

This was already involved in the project but combining it with a code coverage tool and increasing the number of tests can help.

- ***Use a design review process:***

We are not sure about the design schedule of this project but if the design review process was not involved in the schedule using it can help. We can identify potential problems in the early stages and prevent them.

## **Suggestion-2: The code should be more readable**

In our first T1 report we were looking answer to the **"Is this product readable enough?"** (Q6 in the GQM tree) question. Our metrics for the question were:

- NOC: 0,30 (1,3)
- LCOM: 1,54 (<3)
- DIT: 1,29 (<5)

We mentioned that product is readable enough but it can be better. The low NOC value is the reason why.

When a module has a low NOC value, it may still have multiple levels of nested logic or intricate control flow. Understanding the execution flow and the relationships between different parts of the code can become more challenging. This increased cognitive load can make it harder for developers to make changes on codebase.

In this report, we created scenario that argue the **"Is this product readable enough?"** question in (QA-2).

Our suggestions for this OSS product are:

- ***Refactor Large Modules***

Identify large modules with low NOC values and assess if they can be broken down into smaller, more focused units. Look for cohesive subfunctionalities

within the module that can be extracted and encapsulated in their own child modules. It will decrease high NOC value and increase readability of code.

- **More Usage Of Encapsulation and Abstraction**

Encourage developers to design modules with clear responsibilities and interfaces. Promote encapsulation and abstraction by ensuring that each module encapsulates a specific functionality or component and defining clear interfaces. It helps developers to understand code easier.

- **Review Code Architecture**

Conduct regular code reviews and architectural assessments of identify opportunities for improving the NOC value. It results in a way decreasing in NOC value and increasing in readability.

### **Suggestion-3: The code should consist of less complex classes**

In our first T1 report we were looking answer to the **“Are the complexity of classes high in general?”** (Q2 in the GQM tree) question. Our metrics for the question were:

- WMC: 5,76 (There should be a maximum of 10% of classes that have more than 24 methods.)
- CBO: 7,85 (<14)
- LCOM: 1,54 (<3)

We mentioned that product is well designed with respect to complex classes but it can be better. The less complex classes means less time to modify.

Complex classes are harder to comprehend due to their intricate logic and extensive functionality. When developers need to modify or extend such classes, the complexity can make it difficult to grasp the existing codebase and its implications. This lack of understanding can lead to inefficient modifications and make them consume more time.

In this report, we created scenario that argue the **“Are the complexity of classes high in general?”** question in (QA-2).

Our suggestions for this OSS product are:

- **Single Responsibility Principle**

Encourage adhering to the SRP, which states that a class should have a single responsibility. Complex classes often arise when they try to handle multiple responsibilities. By refactoring complex classes into smaller, focused classes, each with a clear responsibility, you can improve modifiability.

- **Extract and Delegate**

Analyze the complex class and identify cohesive sections of code that can be extracted into separate classes or methods. This approach helps reduce the complexity of the original class and promotes better code organization. And that leads to better modifiability and readability.

## 10. REFERENCES

- <http://www.faadooengineers.com/online-study/post/cse/software-engineering/205/component-and-connector-view#:~:text=A%20C%26C%20view%20describes%20a,node s%20and%20connectors%20as%20edges.>
- <https://www.georgefairbanks.com/modeling/viewtype/>
- <https://docs.aws.amazon.com/opensearch-service/latest/developerguide/what-is.html>
- <https://www.drupal.org/docs/8/core/modules/views>

## 11. ALLOCATION OF RESPONSIBILITIES WITHIN TEAM MEMBERS

<b><i>Name of Team Member</i></b>	<b><i>Description of Responsibility</i></b>	<b><i>Allocation Unit</i></b>
Gamze Ergin	Filling section 1	section 1
Gamze Ergin	Filling section 2	section 2
Yunus Emre Terzi	Filling section 3	section 3
Yunus Emre Terzi	Filling section 4	section 4
Gamze Ergin	Filling section 4	section 4
Gamze Ergin	Filling section 5	section 5
Yunus Emre Terzi	Filling section 6	section 6
Yunus Emre Terzi	Filling section 7	section 7
Gamze Ergin	Filling QA-1	section 8
Yunus Emre Terzi	Filling QA-2	section 8
Gamze Ergin	Filling Suggestion-1	section 9
Yunus Emre Terzi	Filling Suggestion-2	section 9
Yunus Emre Terzi	Filling Suggestion-3	section 9