

# Gropius-VSC: IDE Support for Cross-Component Issue Management

Sandro Speth<sup>1</sup>, Niklas Krieger<sup>1</sup>, Uwe Breitenbücher<sup>1</sup> and Steffen Becker<sup>1</sup>

<sup>1</sup>University of Stuttgart, Universitätsstraße 38, Stuttgart, 70569, Germany

## Abstract

Modern software systems are increasingly built as component-based architectures, e.g., as microservices. However, such architectural styles result in many challenges for development teams regarding issue management: Since the individual components are developed independently of each other, the teams manage the issues of the respective components often in separate issue management systems, making the tracking of cross-component issues much more difficult, i.e. issues that affect multiple components concurrently. To solve this problem, in previous work, we developed Gropius, which is a tool that acts as a wrapper for existing issue management systems enabling the management of issues across the different components managed in different issue management systems. While Gropius is particularly suitable for stakeholders who want to have an overall view of the architecture and the issues therein, a developer's daily work is primarily done in an IDE. Thus, frequent context switches between the issue management systems and the IDE reduce developer productivity, so IDE plugins for issue management have been developed in the past. However, these do not support cross-component issue management features as provided by Gropius. Therefore, in this work, we present *Gropius-VSC*, a Visual Studio Code extension that allows developers to manage cross-component issues directly in Visual Studio Code.

## Keywords

Issue Management, Component-based Architecture, Cross-Component Issues, IDE Extension

## 1. Introduction

Component-based architectural styles, such as microservices, are becoming increasingly common. Due to the independence in development and use of the individual components, these architectures bring many advantages. At the same time, however, new challenges arise, such as *cross-component issue management*: Due to the free choice of tools, the individual development teams typically manage the issues of the components in independent issue management systems, e.g. GitHub or Jira. As a result, tracking issues across component boundaries becomes much more difficult [1, 2, 3]. While an issue management system (IMS) helps individual teams to report, track, assign, and archive issues [4], no established IMS supports managing issues that affect components managed in other issue management systems. Thus, this requires manually synchronising issues between multiple issue management systems. To solve this problem, we introduced *Gropius* [2] in previous work, which is a tool that serves as a wrapper for existing IMSs and that can synchronise and link issues between different issue management systems.

---


ECSA'21: 15th European Conference on Software Architecture, Sep 13–17, 2021, Växjö, Sweden

✉ sandro.speth@iste.uni-stuttgart.de (S. Speth); niklas.krieger@studi.informatik.uni-stuttgart.de (N. Krieger); uwe.breitenbuecher@iaas.uni-stuttgart.de (U. Breitenbücher); steffen.becker@iste.uni-stuttgart.de (S. Becker)

ORCID 0000-0002-9790-3702 (S. Speth); 0000-0002-8816-5541 (U. Breitenbücher); 0000-0002-4532-1460 (S. Becker)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Gropius implements the *Cross-Component Issue Metamodel* [3] and represents issues using the graphical *Gropius Cross-Component Issue Modelling Language* [2], which enables modelling of issues of each component of the system in a typical architectural view. Thus, the *Gropius Web Frontend* [2] is particularly suitable for software architects, product owners, and other stakeholders who need an overall view of the system. The daily work of a developer, on the other hand, takes place primarily in an IDE. Therefore, frequent context switches between the IDE and issue management systems reduce the developer's productivity. For this reason, many IDEs offer plugins and extensions for IMSs [5, 6]. However, existing issue management IDE plugins and extensions do not support features for cross-component issue management in the way Gropius supports. Therefore, in this demonstrator paper, we introduce *Gropius-VSC*, a Visual Studio Code (VS Code) extension for the integrated management of issues in component-based architectures. Since Gropius-VSC uses the Gropius API, cross-component issues can be propagated directly from the IDE to the underlying issue management systems, such as GitHub.

## 2. Research Design

Before we describe the concept of Gropius-VSC, we explain our research design. First, we developed the concept by asking various stakeholders for the required features of such an IDE extension. Based on these requirements and the Cross-Component Issue Metamodel [3], the concept for the IDE extension was developed. For an initial evaluation of whether the concept makes sense, a prototype was developed with the Eclipse Modelling Framework (EMF), which we gave to industry experts for evaluation. The API connection to the Gropius backend server [2] and test data were mocked up. Questions for the evaluation were created using a Goal-Question-Metric [7] approach. Due to the positive evaluation, we decided to build this Gropius IDE extension integrated with the Gropius server instead of a mocked API. Based on the limitations in generated views of EMF Parsley, the evaluation of the industry developers and their feature requests, we decided to switch to VS Code as IDE.

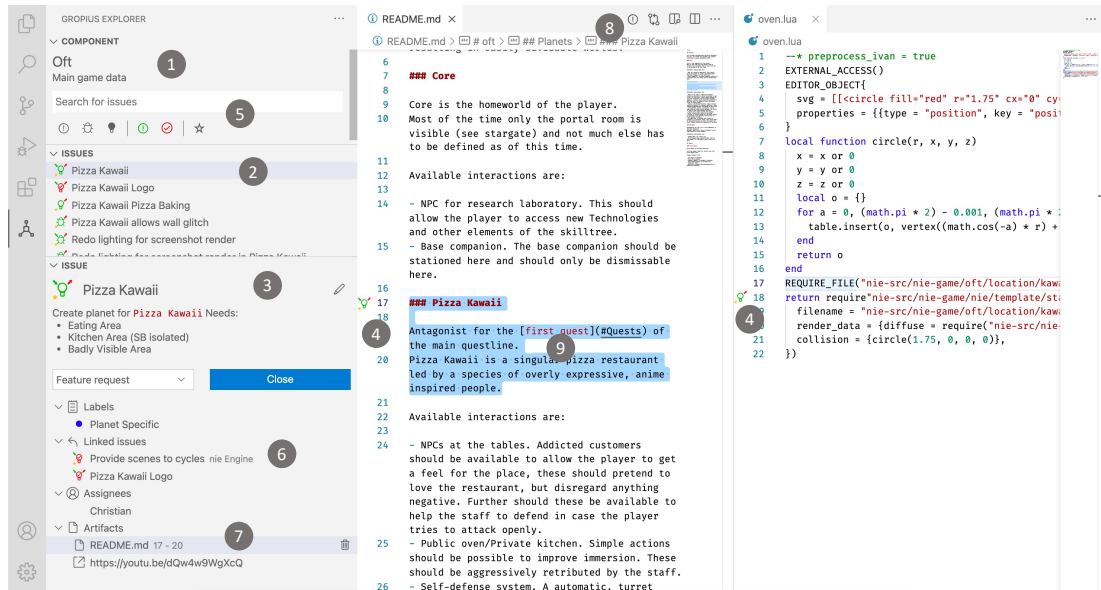
## 3. The Concept of Gropius-VSC

The presented IDE extension Gropius-VSC is a client for the Gropius backend that supports the Cross-Component Issue Metamodel [3] and, thus, the cross-component features of Gropius [2]. However, in contrast to the original *Gropius Web Frontend* [2], Gropius-VSC aims to provide a focused component-specific view of cross-component issues while keeping the cross-component issue management features, e.g., linking issues to other components' issues and traversing these links. A demonstrator video presenting our Gropius-VSC concept is available on YouTube<sup>1</sup>.

Gropius-VSC consists of four components which are depicted in fig. 1: ① a *Component Detail View*, ② an *Issue List*, ③ a *Issue Detail View*, which is a detailed view of a single issue, and ④ *Issue Bookmarks*, which display the areas affected by an issue in a file. The sidebar next to the line numbers, which is also used for breakpoints, shows these issue bookmarks. After the IDE extension is configured for a Gropius project's component, the Component Detail View

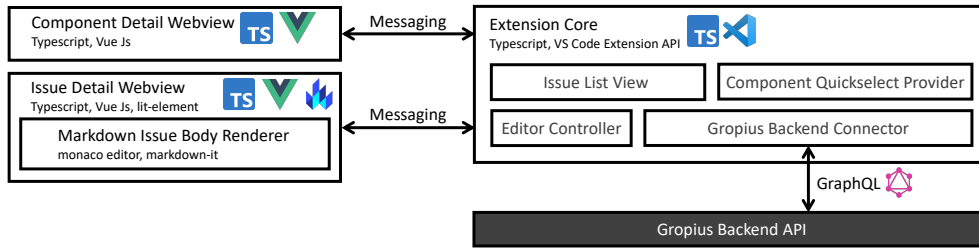
---

<sup>1</sup><https://youtu.be/1sfY18-R-YE>



**Figure 1:** Example project in Gropius-VSC showing all primary components of the concept.

① provides the current component's name and description and issue filter functionality ⑤. Additionally, the Issue List ② shows all issues of the current component. This list can be filtered in the Component Detail View ⑤ by the status of the issue (open or closed), type of issue, e.g. bug report, and issues assigned to the developer using the extension. Issues assigned to the developer show a star in the issue's icon. Furthermore, the Component Detail View provides a search bar to filter according to the issues' titles and body matching to the filter text. The individual Issue Detail View ③ shows the title, body, issue type, labels and assigned users of the issue. Additionally, the semantical links to other issues ⑥ and artefact links ⑦ are displayed. Ingoing or outgoing issue links to another issue are hinted with an ingoing or outgoing arrow in the issue's icon. The issue links can be traversed by clicking on the linked issue ⑥, whereby the detailed view ③ always opens the linked issue. Furthermore, the linked issues and artefacts do not have to belong to the same component. If a linked issue concerns another component or the issue concerns multiple components, the detail view shows all affected components' names next to this issue to clarify that this issue's location is another component. A click on an artefact link opens the artefact, e.g. a source code file, in the text editor and jumps, if specified, to the area affected by the issue ⑨. All artefacts are specified as URIs. An URI is opened in the web browser if it could not be transformed to a relative path to a file. In addition to displaying an issue in the detailed view, issues can be edited, e.g., changing the body text or adding new issue links and artefact links. Issue Bookmarks ④ in the sidebar show for a linked artefact file which areas are affected by an issue. However, the Issue Bookmarks contain only open issues to keep it more concise. Additionally, bookmarks cover a range of lines as shown in fig. 1. In the concept, clicking on an issue bookmark opens the issue which is linking to this artefact. However, since clickable custom sidebar elements are an open issue for VS Code, our extension contains a filter button ⑧ for issues linking to this artefact until this issue is resolved.



**Figure 2:** Architecture of the Visual Studio Code extension Gropius-VSC.

## 4. Architecture and Implementation

The architecture for Gropius-VSC consists of three main components: (1) the *Extension Core*, (2) the *Issue Detail Webview*, and (3) the *Component Detail Webview*. The components communicate with each other via messaging, which is handled by VS Code. Figure 2 shows the overview of the architecture. The extension is written in Vue JS and is open source available in GitHub<sup>2</sup>.

The Extension Core uses the Gropius' GraphQL API in the *Gropius Backend Connector* component to communicate with the Gropius backend. The *Component Quickselect Provider* component is a *QuickInput* which sets the component's id of the current workspace. This allows the user to search the name and description of all available components instead of setting the id in the settings manually. The *Editor Controller* decorates the *TextEditor* with Issue Bookmark icons. The *Issue List View* is a *TreeView* showing all issues of the current component. If the user clicks on an issue, the Issue Detail Webview opens this issue to show the details of the selected issue as stated in section 3 and enables editing it. The issue's body is rendered with the *Markdown Issue Body Renderer* which consists of two parts: (1) a *Monaco Editor* (the same editor VS Code uses) for editing the body and (2) *markdown-it* with *emoji plugin* to render markdown. The Component Detail Webview displays information about the selected component and provides search and filter functionality for the Issue List View. *VS Code Commands* offers some internal commands to the user, e.g. create a new issue, reload issue view, and check API connection.

VS Code's flexible extension API allows web view extension panels which developers can develop using modern web development tools and frameworks. As a current limitation, custom sidebar elements like our Issue Bookmarks are not clickable.

## 5. Related Work

In his PhD thesis [8], Janák introduces two categories for IMS IDE extensions: (1) universal extensions, which allows integrating different IMSs while not supporting IMS-specific features, and (2) extensions limited to specific issue management systems which provide support of IMS-specific features. We classify Gropius-VSC as a universal IMS IDE extension since it integrates through Gropius several IMSs while focuses on the features provided by Gropius. Other universal extensions, e.g. *Mylin*<sup>3</sup>, do not support cross-component issue management

<sup>2</sup><https://github.com/ccims/ccims-vsc/tree/gropiusify>

<sup>3</sup><https://marketplace.eclipse.org/content/mylyn>

features as Gropius-VSC does. Therefore, such IMS IDE extensions are not suited for integrated cross-component issue management. Examples for specific IMS IDE extensions are *Atlassian IntelliJ IDEA Connector*<sup>4</sup> and *JiraBuddy - Eclipse Plugin for JIRA*<sup>5</sup>. However, they do not provide cross-component issue features. Furthermore, there is the *Teamscale Integration for Eclipse*<sup>6</sup> which allows users to browse defects found by the Teamscale Software Quality Analysis Server. However, all these IDE extensions are limited to the boundaries of a single IMS. Therefore, issues cannot be managed beyond the boundaries of an IMS as supported by our Gropius approach.

## 6. Conclusion

The presented VS Code extension Gropius-VSC can improve issue management in component-based architectures in a less error-prone and time-consuming way by allowing developers to manage cross-component issues directly in their IDE without context switches. In particular, traversing the links between issues of different components and opening the affected resources enables efficient issue management. Especially in systems with many components, this could significantly increase the focus on the issues that actually affect the developer's components.

## References

- [1] S. Mahmood, M. Niazi, A. Hussain, Identifying the challenges for managing component-based development in global software development: Preliminary results, in: 2015 Science and Information Conference (SAI), IEEE, 2015, pp. 933–938.
- [2] S. Speth, U. Breitenbücher, S. Becker, Gropius—a tool for managing cross-component issues, in: Communications in Computer and Information Science, volume 1269, Springer, Springer, 2020, pp. 82–94.
- [3] S. Speth, S. Becker, U. Breitenbücher, Cross-component issue metamodel and modelling language, in: Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER 2021), INSTICC, SciTePress, 2021, pp. 304–311.
- [4] D. Bertram, A. Volda, S. Greenberg, R. Walker, Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams, in: Proceedings of the 2010 ACM conference on Computer supported cooperative work, 2010, pp. 291–300.
- [5] S. H.-H. Chang, X. Chen, R. A. Priest, B. Plimmer, Issues of extending the user interface of integrated development environments, in: Proceedings of the 9th ACM SIGCHI New Zealand Chapter's International Conference on Human-Computer Interaction: Design Centered HCI, 2008, pp. 23–30.
- [6] A. I. Wasserman, Tool integration in software engineering environments, in: Software Engineering Environments, Springer, 1990, pp. 137–149.
- [7] V. R. B. G. Caldiera, H. D. Rombach, The goal question metric approach, Encyclopedia of software engineering (1994) 528–532.
- [8] J. Janák, Issue tracking systems, Ph.D. thesis, Masarykova univerzita, 2009.

---

<sup>4</sup><https://plugins.jetbrains.com/plugin/2190-atlassian-connector-for-intellij-ide>

<sup>5</sup><https://marketplace.eclipse.org/content/jirabuddy-eclipse-plugin-jira>

<sup>6</sup><https://marketplace.eclipse.org/content/teamscale-integration-eclipse>