Electronic Communications of the EASST
Volume 85 Year 2025

deRSE25 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany

Edited by: René Caspart, Florian Goth, Oliver Karras, Jan Linxweiler, Florian Thiery, Joachim Wuttke

Contributor Support for Maintaining Opensource Software. A Case Study of BIMserver

Zaqi Fathis, Helga Tauscher

DOI: 10.14279/eceasst.v85.2702

Contributor Support for Maintaining Opensource Software. A Case Study of BIMserver

Zaqi Fathis[1], **Helga Tauscher**[2]

[1] zaqi.fathis@htw-dresden.de
HTW Dresden - University of Applied Sciences, Dresden, Germany

[2] helga.tauscher@htw-berlin.de
HTW Berlin, Berlin, Germany

Abstract:

Maintaining software requires significant effort, as it goes well beyond the creation of new features, and includes updates to the existing code as well as communication with users. Cutting short on these activities can threaten a software project's existence. The consequences are even more drastic for opensource software with limited resources. In this paper, we present the Opensource BIMserver, a mature opensource server software that enables users to store and manage building information in a standardized format in a database, as a case study to illustrate the challenges of software maintenance and the structured approach, strategies and measures we employed to support new contributors during the onboarding process. The set of complementary measures includes community involvement, agile practices, pair-programming, guided issue processing, bite-sized tasks, targeted reading recommendations. While most of these methods as well as a structured onboarding process are common in commercial software development settings, they are rarely implemented in opensource or research context. We exemplify our approach with actions taken in four identified pivotal areas of software maintenance — namely documentation, issue processing, dependency management, and automated testing.

Keywords: opensource software; BIMserver; maintenance

1 Introduction

Free and Open Source Software (FOSS) plays a vital role in the modern software infrastructure. Generally speaking, such software is distributed under a license that allows users extensive freedom in using, studying, and modifying the software by accessing its unobfuscated source code which must be provided alongside the executable software product for this purpose.

In the architecture, engineering and construction (AEC) industry, however, there is historically a monopolistic culture with few software vendors dominating the market and most professionals still primarily relying on proprietary, closed-source software for design, planning and execution of construction projects. Yet, there is a growing interest in free and opensource AEC software. This shift is driven by factors such as the high cost of commercial software, slow development cycles, and its limitations in adapting to user needs. One such software is the Opensource BIMserver, currently maintained by the authors. We will outline the background in the AEC domain and the situation of FOSS in the domain in Section 2.1 and then describe the Opensource BIMserver in Section 2.2.

Sustainable development of a software, that is continuous improvement and addition of features while keeping the software maintainable is a well-known challenge, because technical debt (omitted or postponed technical work) can lead to the maintenance effort growing exponentially and eventually become disproportional in comparison to the perceived benefit of the software. With FOSS, due to the specifics of its production context, and in an academic environment, with limited resources, these challenges become even more severe. Maintaining software over the long term requires significant effort beyond the creation of new features along a roadmap. First of all the codebase needs to be continuously updated to improve existing features while avoiding regression of the existing functionality, performance and security. Second, there are communicative tasks such as documenting usage, supporting users, collecting, judging and processing feedback and requests. While maintaining the Opensource BIMserver, we identified four areas of maintenance activities: documentation, issue processing, dependency management and automated testing. These are acknowledged in the literature and studied within the context of FOSS. We highlight some existing work on these areas of maintenance activities and the broader challenges of FOSS development in Section 2.3.

We developed a strategy and measures to cope with these challenges while supporting a new contributor, which we are presenting in this paper. The remainder of this paper is structured as follows: After providing more context to our work and summarizing related work in the next Section 2, we then introduce the particular challenges identified during development of the Opensource BIMserver and propose strategies to cope with them with a focus on introducing and involving a new contributor into the maintenance and development routines in Section 3. In the main Section 4 we describe how the suggested strategies and measures have been employed in the four relevant key areas of software maintenance: documentation, issue tracking and resolution, dependency management, as well as automated testing. Finally we evaluate the effects of the application from our experience in Section 5 and discuss the results in Section 6.

2 Background and related work

2.1 FOSS in AEC and BIM

Building Information Modeling (BIM) has become a pivotal method in the AEC sectors due to its potential to improve information access, facilitate communication, increase productivity, and enhance process control across the building life cycle. With BIM, shared digital representations provide the foundation for consistent data exchange and software interoperability. Those shared representations are supported by the Industry Foundation Classes (IFC), an open and standardized data model [ISO18] developed and maintained by buildingSMART International[1] since 1995 [LW99]. IFC builds on and adapts the Standard for the Exchange of Product model data (STEP) from the mechanical engineering field, a series of ISO standards based on the EXPRESS data modeling language [ISO04].

The AEC industry has traditionally relied on proprietary commercial software and even the

[1] buildingSMART was formerly known as The International Alliance for Interoperability, IAI.

academic community often focuses on workflows with various Autodesk products applied to real world projects [BSV+25, e.g.], although FOSS offers greater opportunities for experimentation and innovation. Consequently, a small but steadily growing number of FOSS are gaining interest within the community. For example, Logothetis et al. developed an opensource spatial Database Management System (DBMS) integrated with FreeCAD, an opensource 3D parametric modeling application built on the OpenCascade geometry kernel, to extend its BIM capabilities [LVKS17]. Another opensource project is IfcOpenShell[2], a C++ library with Python bindings to parse and process IFC data, which has since evolved into an ecosystem of tools such as Bonsai[3] (formerly known as BlenderBIM) a Blender add-on for creating, viewing, and editing IFC data.

These two FOSS projects stand exemplarily for the community that we wish to see growing.

2.2 Opensource BIMserver

The Opensource BIMserver[4] has initially been developed over 15 years ago as an experiment initiated by TNO, the Dutch Organization for Applied Scientific Research. Its early incarnation has been described by its original authors, Beetz et al., in a seminal paper [BBLH10]. While still capturing the core intention and features of the software, many details in the paper have been outdated throughout the decade of development. There is a brief updated description in the most recent edition of the BIM handbook [SGBB25].

To allow for efficient storage, querying, and retrieval of the building data and generate different outputs, within BIMserver, data is stored using the key-value database BerkeleyDB, with one record per IFC entity instance. To this end, the IFC schema defined in the EXPRESS modelling language is translated into an EMF Ecore specification, to allow for model-driven software development methods to be applied. BIMserver is written in Java and runs in a JVM — either by deploying a WAR archive into any servlet containter such as Apache Tomcat or from a standalone JAR executable with an embedded Jetty web server and servlet container or by embedding the BIMserver itself in a Java application. It must be noted that as a server software, the Opensource BIMserver is meant to be used by developers to implement their own client applications and plugins using the provided API and extensions points.

For demonstration purposes, there is a web browser-based client called BIMvie.ws. Figure 1 illustrates some of the server functionalities via their appearance in the BIMvie.ws UI: IFC datasets are organized into projects and subprojects (Figure 1a). Every project contains multiple revisions, that is different versions of the same building model (Figure 1c). The 3D contents and spatial structures can be displayed in a viewer (Figure 1b). For every entity instance, its properties can be retrieved and displayed (Figure 1d) including the relationships between objects within a specific revision (Figure 1e). In addition, a subset of the model can be queried and downloaded (Figure 1f). Several serializer implementations are available for encoding the downloads, including GLTF, JSON, IFC2x3, IFC4, and a dedicated binary geometry for visualization purposes.

Figure 2 shows the historical commit activity in the BIMserver Github repository. From 2010, when the source code was moved to Github, until 2019 there was a steady high activity carried out

[2] http://www.ifcopenshell.org
[3] https://bonsaibim.org/
[4] https://github.com/opensourcebim/bimserver

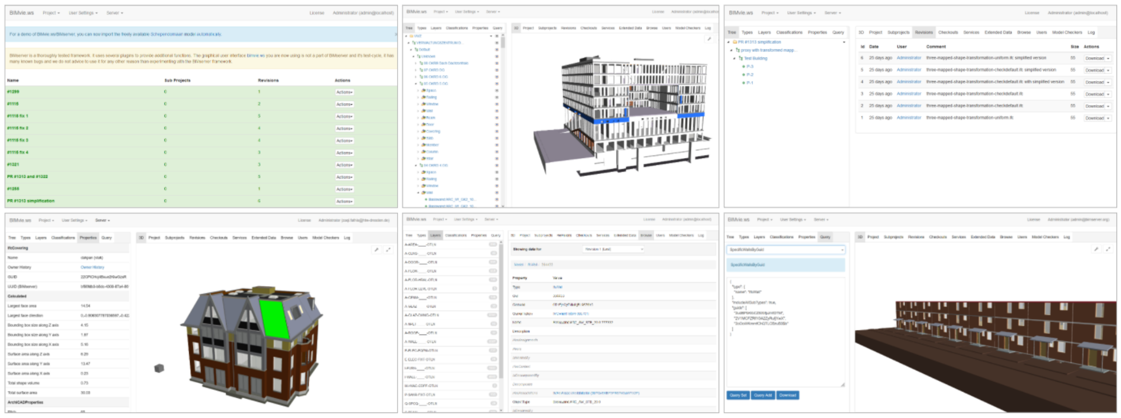


Figure 1: Screenshots of different features in BIMserver: a) create projects (upper left), b) 3D view and spatial structure (upper middle), c) project revision (upper right), d) object properties and e) browsers (lower left and middle), and f) query overview (lower right)

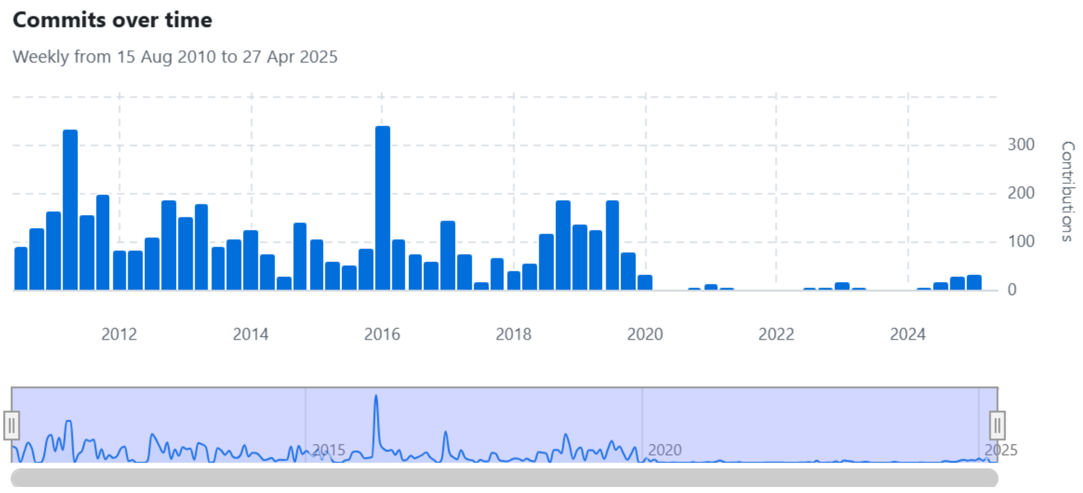


Figure 2: Commit activity in the BIMserver repository from August 2010 until April 2025

by a single paid full time developer within a small team affiliated to a large research organization. From 2020 onward, activity has been kept on a mere maintenance level as the financial and organizational situation changed abruptly. A wave-like pattern can be seen during both periods and possibly attributed to the project-driven development cycles in third-party-funded research.

BIMserver's codebase has grown significantly over time and now consists of approximately 6,000 Java classes, along with additional 4,000 generated classes. In total, the project contains around 2.7 million lines of code, including about 1.3 million lines of generated code.

2.3 Maintaining FOSS projects

With the growing role of opensource software, researchers have examined various aspects of FOSS projects, including their specifics compared to proprietary software projects, their potentials, challenges, and possible solutions. In this section, we summarize some of that work as it relates to the maintenance of FOSS and the four fields of maintenance activity identified. FOSS projects are often described as decentralized and meritocratic, relying on scattered, often voluntary, contributors and with structures evolving as they mature. Their organizational structures range from single-maintainer projects to informal gathering of individuals to large software foundations like Apache or Eclipse to domain-specific bodies like OSArch, each with distinct governance models. Izquierdo and Cabot provide an overview of these structures [IC20], while Forrest et al. show how these structures influence assignment of tasks (such as bug triaging, issue processing, and code contributions) within the community and backing organizations [FJMD12].

Geiger et al. study the various types of labour involved in software maintenance and how the execution of these works scales as the projects become larger and established. They point out that software projects need not only maintenance in the technical sense and working with the code, such as fixing bugs, patching security vulnerabilities, and managing dependencies, but also maintenance in a social sense, like enhancing the ability of a new contributors in the community to actively participate in the project, providing them with the necessary resources to contribute effectively [GHI21]. One of the key challenges that FOSS projects face, is to encourage more meaningful contributions, especially from individuals beyond the core development team, such as co-developers and users [AZRG05]. These groups represent a large portion of the community but typically contribute less frequently or not at all.

Accessible documentation plays a crucial role, serving as a primary medium of communication between core developers, the wider community, and new contributors. Yet, OSS developers often prioritize coding over documentation and consider in-code comments sufficient [SA10, DLT^{+}14]. Another important aspect of social maintenance is facilitating effective communication in issue trackers, which can become overwhelming when discussions grow lengthy and address complex problems that significantly impact the project. Arya et al. found that due to the diverse background of FOSS community members, extensive discussions are often required before reaching common ground. Moreover, they observed that FOSS contributors or users often ask questions that have been answered in earlier comments or threads. This phenomenon reflects the challenge of locating relevant information within documented issue discussions, particularly for those unfamiliar with the context and the project [AWGC19].

On the technical side, reliance on third-party software libraries introduces risks, as these libraries as well as the packages they depend on may contain severe security threats. Decan et al. demonstrate that while the majority of packages have only a few direct dependencies, they typically have a much higher number of transitive dependencies. This creates a higher risk of failure, which can arise from various causes such as discontinued development or backward-incompatible updates [DMC17]. Updating for security vulnerabilities should not be treated as as minor issue, as it is a direct threat to software quality, stability, and user trust. Software testing is a crucial part within the software development; however, conducting in-depth testing of a software system is often not feasible due to limited time and resources, especially with opensource software [YKB14, LLW+20].

Community composition adds further challenges. Studies highlight the lack of diversity in FOSS communities, showing persistent underrepresentation of women and non-white developers, as well as unequal recognition of contributions. Trinkenreich et al. demonstrate that women's participation in FOSS development is even lower than non-OS commercial software production and industry and that a typical gender division of labour (e.g. frontend/backend) is reproduced in OS development [TWS+22]. Similarly, Nadi et al. report that only 16.56% of contributors were perceptibly non-white, and contributions from white developers have 6–10% higher odds of being accepted compared to those from non-white developers [NRN21]. We have not implemented specific strategies or measures to foster diversity, as the current maintenance team is exceptionally diverse. Yet, we find it important to keep these findings in mind, for example when we engage in discussions with the authors of issue reports.

3 Challenges and Strategies

Since taking over BIMserver maintenance around 2020, we try to establish a sustainable model of ongoing development, yet, most of the time can only do the bare minimum to keep it alive. As can be seen from the earlier Figure 2, there have been delimited periods with slightly increased activity due to third-party-funded research involving the usage of BIMserver. During the last research project, by now running for a period of roughly nine months, we were able to place a dedicated work package in the beginning of the project, where no new project-specific functionality had to be developed, but the existing documentation and code could be updated to a current state of the art and at the same time a new contributor could be educated and trained. In the following sections, we show the strategies and measures we developed based on the identified challenges.

3.1 Challenges

The challenges that BIMserver is facing have different provenience: Some are typical for all software projects, some are common in FOSS projects, others are specific to the research context and then there are those that can be attributed to the application domain.

First of all, like many FOSS projects, BIMserver operates with limited resources. Due to this, maintenance is hindered in a circular effect: With few resources, it becomes difficult to provide regular updates to the codebase, improve existing features, maintain documentation, and offer user support. These constraints can in turn create barriers for newcomers who wish to get involved, if onboarding materials or community guidance are lacking. However, simply adding more human resources for a limited period is not helpful either, since new contributors bind resources for their onboarding and the longer it takes until they have a solid understanding of the codebase, and the shorter the project runtime during which they contribute, the lower the benefit for the project if they don't stay affiliated for longer.

Project-based work with limited periods of development during funding periods is also unfortunate for another reason. The codebase is grown and extended with project-specific parts, one-shot experiments, discontinued development lines and becomes fragmented and harder to navigate. There is usually a tight deadline where compromises are to be made to achieve some project goals and no time to clean up or remove outdated parts. Sustainable development is not factored into research plans, because it is not considered worthy of funding.

After decades of development, the size of the project is intimidating for newcomers and they lack orientation. Looking at the number of classes and lines of code, it would take years to only read through the codebase fully once. For new contributors, navigating such a large and intricate codebase is impossible. Moreover, the complexity of the project itself adds another layer of difficulty. Projects of this size can only be realized with dedicated individuals who naturally become a single source of knowledge and cannot be replaced easily. Personnel rotation, the resulting lack of continuity and loss of knowledge is a serious issue and even more so with very small teams.

Finally, the context of BIMserver as an AEC software poses another challenge. The user group is heterogeneous with varying degree of expertise which makes communication difficult. Similarly potential contributors are diverse and require both domain knowledge as well as computer science and IT experience. On the downside they may also lack some of the expertise.

3.2 Strategies and measures

It must be noted that the goal of the onboarding is not to enable contributions of new functionality to the core — that would be out of reach — but rather it is designed to be focused and practice-oriented, establishing a general understanding of the functionality and architecture to be able to contribute to discussions, reproduce issues, verify solutions, implement custom plugins and clients. We suggest the following measures to foster the onboarding process:

- community involvement to ensure the ongoing development and maintenance,
- agile methods, frequent short meetings,
- pair-programming and shared code-reading,
- documentation as entry-point, dissemination as litmus test,
- issue processing, reproduction, guided solution,
- bite-sized tasks with potential for sensing achievement,
- reading recommendations and advanced training with focus on software development in research,
- recruiting of further (junior) contributors with a CS background.

The measures are embedded within a one-on-one mentoring, through which the experienced maintainer provides a guided plan for different areas of the onboarding process. This ensures that new contributors follow a tailored learning path in which they gradually build the competencies required for meaningful participation.

In the next section we will show how theses measures are applied to the four areas of software development and maintenance.

4 Implementation of the strategies

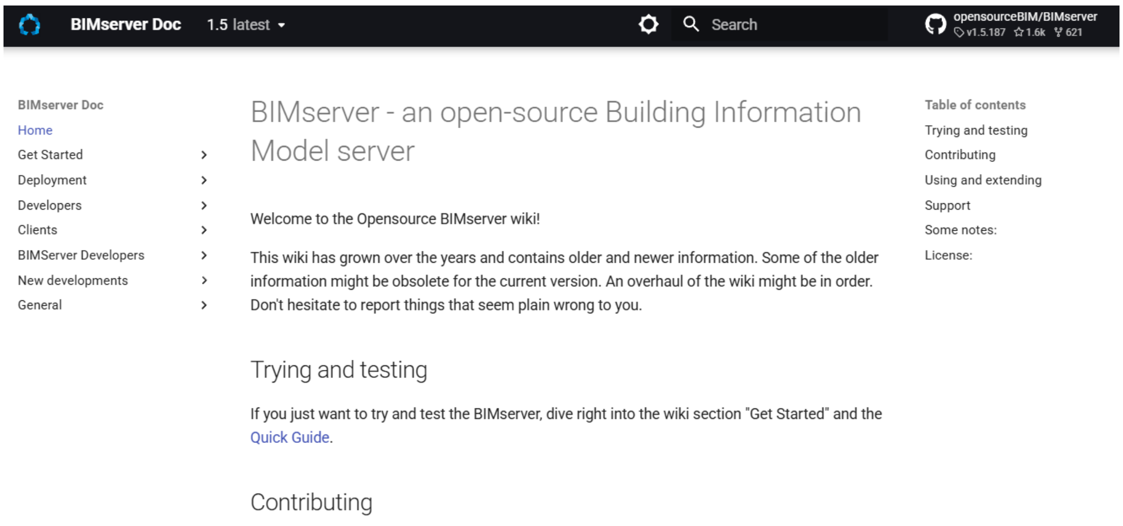
4.1 Documentation

Up-to-date documentation is essential for any opensource software, and thus an important part of BIMserver maintenance. This includes removing outdated documentation, updating and adding new sections to reflect the latest version, and incorporating more examples and tutorials. The documentation in BIMserver consists of seven parts, each covering a specific topic related to the technical details. Developers of applications using the BIMserver as well as of BIMserver extensions, use this structure to navigate the features and learn how to implement clients, plug-ins, and services. For the new contributor, this structure also served as guideline to the system architecture as well as both the capabilities and limitation of the software.

To begin with, the first step involves updating all the outdated links, broken references and outdated resources, including information about the system requirements and links into the code-base, the structure of which might have been changed with files combined, moved, or renamed. In a second step, each page is reviewed to identify content that no longer reflects the latest version. Usually, pages that are no longer relevant would be either removed or rewritten, while new changes would be added in the related section. However while the latest version of BIMserver is 1.5, the previous version 1.4 might be still in use. Thus, part of the documentation might be outdated for the current version, but still valid and relevant for version 1.4 and worth keeping, but without cluttering the documentation with confusing content.

Current documentation uses GitHub's wiki feature. However, this wiki does not allow versioning of the documentation. This limitation becomes an issue when there are multiple versions of the software and a user or contributor needs to refer to a particular release. To address this limitation, documentation is migrated to MkDocs, an opensource static site generator that support versioning through its themes. Specifically, this project uses Material for MkDocs theme that provide built-in support for versioning. MkDocs has been chosen over Sphinx or other similar generators for its compatibility with Markdown, such that the existing documentation written in Markdown can be reused with minimal changes.

While working through the documentation provides a basic understanding of the software, deeper comprehension usually comes from practical involvement with the codebase. As a new collaborator, understanding features and the code samples from the documentation, particularly in sections about the development of plugins, clients and the core, can be challenging due to its complexity and initially limited background knowledge. Therefore, work on the documentation was complemented with a hands-on approach fixing issues related to the respective documentation sections in parallel.


BIMserver Doc 1.5 latest

Search

opensourceBIM/BIMserver v1.5.187 1.6k 621

BIMserver Doc
Home
Get Started
Deployment
Developers
Clients
BIMServer Developers
New developments
General

BIMserver - an open-source Building Information Model server

Welcome to the Opensource BIMserver wiki!

This wiki has grown over the years and contains older and newer information. Some of the older information might be obsolete for the current version. An overhaul of the wiki might be in order. Don't hesitate to report things that seem plain wrong to you.

Trying and testing

If you just want to try and test the BIMserver, dive right into the wiki section "Get Started" and the Quick Guide.

Contributing

Table of contents
Trying and testing
Contributing
Using and extending
Support
Some notes:
License:

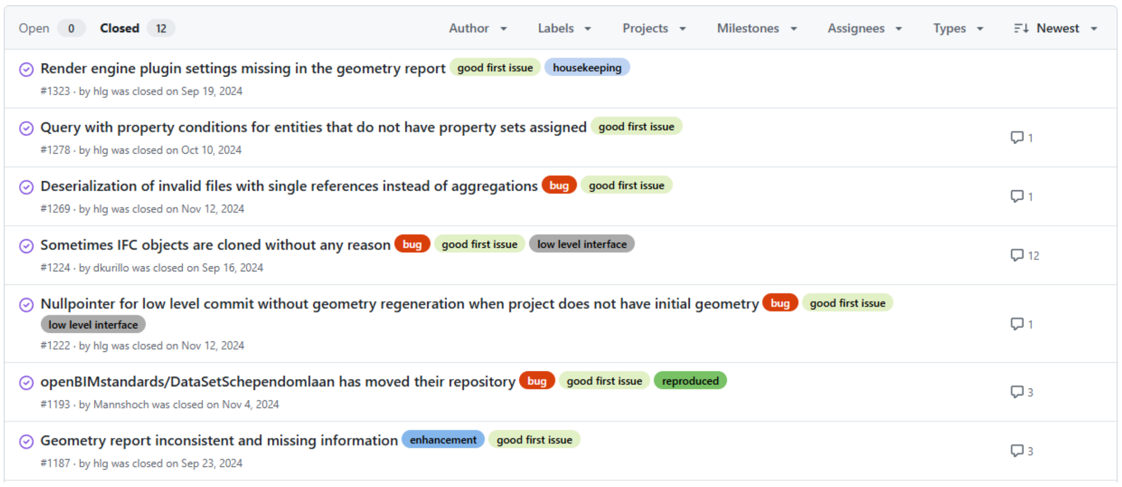
Figure 3: New documentation site of BIMserver that supports versioning

4.2 Issue tracking

Another area of maintenance work is to manage and address issues. This includes the classification and categorization of issues, prioritizing them to ensure critical problems are addressed first, and fixing small programming errors which improves the overall stability of the platform. For this step, the new contributor has been granted ”Triage” role for the BIMserver repository. The Triage role allows contributors to manage the issues, discussion, and pull request, even without write access. This way the new contributor does not have access to sensitive and potentially destructive actions, such as managing security or deleting repository.

At the start of the process, there were approximately 140 open issues in the BIMserver repository. However, through this maintenance, this number was reduced to around 100. Most of the issues are already organized into categories using labels. There are labels for 5 types of issues (*bug, enhancement, feature, housekeeping, howto*) as well as labels for 5 areas of software functionality or modules (*auth, java client library, low level interface, plugin installation, query language*). While the former labels devise how to treat a certain issue, the latter labels help to keep related issues connected and find the relevant part of the codebase more quickly. In addition, there are some labels to further mark certain types of issues in addition to the above categories: *FAQ* to mark recurring issues of type howto, *unsupported* to mark feature requests, *reproduced* for bug type issues and finally the well-known label *good first issue* to mark particular issues, mainly of type bug or feature. New contributors helped to improve the issue list by reviewing the unlabeled issues and categorized it accordingly to improve clarity and prioritization. Additionally, outdated issues that are no longer relevant to the latest version of BIMserver were closed.

The label *good first issue* (Figure 4) was used to mark simple tasks like fixes for minor programming errors and to provide additional information and instructions intended to assist new contributors to get started. These beginner-friendly issues serve as a entry points for contributors



Open 0 **Closed** 12 Author Labels Projects Milestones Assignees Types Newest

Render engine plugin settings missing in the geometry report good first issue housekeeping
#1323 · by hlg was closed on Sep 19, 2024

Query with property conditions for entities that do not have property sets assigned good first issue 1
#1278 · by hlg was closed on Oct 10, 2024

Deserialization of invalid files with single references instead of aggregations bug good first issue 1
#1269 · by hlg was closed on Nov 12, 2024

Sometimes IFC objects are cloned without any reason bug good first issue low level interface 12
#1224 · by dkurillo was closed on Sep 16, 2024

Nullpointer for low level commit without geometry regeneration when project does not have initial geometry bug good first issue low level interface 1
#1222 · by hlg was closed on Nov 12, 2024

openBIMstandards/DataSetSchependomlaan has moved their repository bug good first issue reproduced 3
#1193 · by Mannshoch was closed on Nov 4, 2024

Geometry report inconsistent and missing information enhancement good first issue 3
#1187 · by hlg was closed on Sep 23, 2024

Figure 4: Prioritizing fixes for issues labeled *good first issue*

to understand and build familiarity with the codebase. While some of these issues were related to bugs, others pertained to housekeeping or enhancements and various feature areas. This variety allowed the contributors to explore and gain insight in different parts of the codebase. For example, in (Figure 5), issue #1323 which falls under *good first issue* and *housekeeping* labels, involved fixing a geometry report which does not display all the properties from the render engine plugin. Although it was a minor issue, solving it required understanding when and where the report is generated, as well as locating the relevant code in the codebase. Further, through reproducing the issues, and verifying potential solutions, the new contributor got familiar with the functionality. From the experienced contributor side, solutions were prepared by appending hints and remarks about possible solutions including references to lines in the codebase to the respective issue.

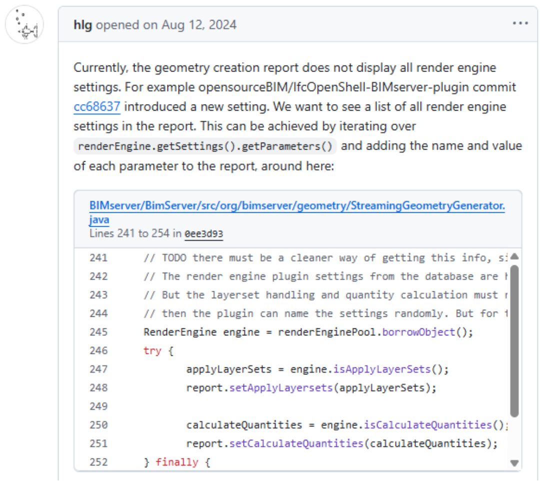
In addition to fixing issues, contributors also actively engaged in conversations within issue threads, where contributors follow up on some reported problems, answer questions, offer suggestions, clarify issues, or help other user troubleshoot their problem. This to make sure that the discussion are maintained well.

4.3 Dependency updates

At the time of writing this paper, BIMserver still supports Java 8 because it is used in some environments with no newer Java version available and the OpenJDK as well as downstream projects providing JDK binaries are keeping to publish Java 8 runtimes with long term support for the coming years. Thus, we first tried to update dependencies while maintaining compatibility with Java 8 to avoid forcing JVM updates to those users stuck with Java 8.

Towards the update contributors began by reviewing the existing dependencies, the so-called SBOM, Software Bill of Materials. BIMserver uses Apache Maven for dependency management and the explicit dependency declarations in POM[5] files make it easy to automatically check de-

[5] POM: Project Object Model, the file with definitions for the build and dependency management



hlg opened on Aug 12, 2024

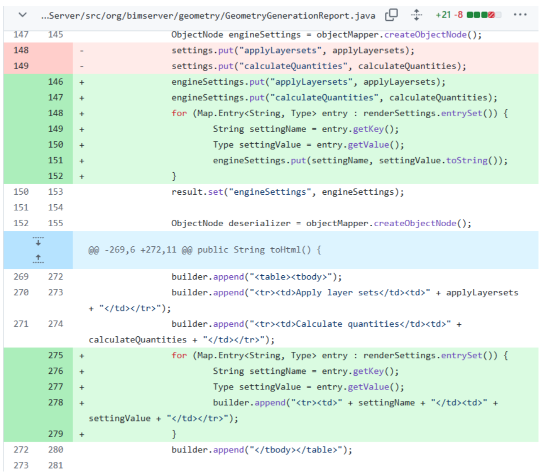
Currently, the geometry creation report does not display all render engine settings. For example opensourceBIM/IfcOpenShell-BIMserver-plugin commit cc68637 introduced a new setting. We want to see a list of all render engine settings in the report. This can be achieved by iterating over `renderEngine.getSettings().getParameters()` and adding the name and value of each parameter to the report, around here:

BIMserver/BimServer/src/org/bimserver/geometry/StreamingGeometryGenerator.java
Lines 241 to 254 in `0ee3d93`

```java
// TODO there must be a cleaner way of getting this info, si
// The render engine plugin settings from the database are
// But the layerset handling and quantity calculation must
// then the plugin can name the settings randomly. But for
RenderEngine engine = renderEnginePool.borrowObject();
try {
        applyLayerSets = engine.isApplyLayerSets();
        report.setApplyLayersets(applyLayerSets);

        calculateQuantities = engine.isCalculateQuantities();
        report.setCalculateQuantities(calculateQuantities);
} finally {
```

(a) Issue with description of solution



...Server/src/org/bimserver/geometry/GeometryGenerationReport.java +21 -8

```diff
 147 145                 ObjectNode engineSettings = objectMapper.createObjectNode();
 148     -               settings.put("applyLayersets", applyLayersets);
 149     -               settings.put("calculateQuantities", calculateQuantities);
     146 +               engineSettings.put("applyLayersets", applyLayersets);
     147 +               engineSettings.put("calculateQuantities", calculateQuantities);
     148 +               for (Map.Entry<String, Type> entry : renderSettings.entrySet()) {
     149 +                       String settingName = entry.getKey();
     150 +                       Type settingValue = entry.getValue();
     151 +                       engineSettings.put(settingName, settingValue.toString());
     152 +               }
 150 153                 result.set("engineSettings", engineSettings);
 151 154
 152 155                 ObjectNode deserializer = objectMapper.createObjectNode();
         @@ -269,6 +272,11 @@ public String toHtml() {
 269 272                 builder.append("<table><tbody>");
 270 273                 builder.append("<tr><td>Apply layer sets</td><td>" + applyLayersets
         + "</td></tr>");
 271 274                 builder.append("<tr><td>Calculate quantities</td><td>" +
         calculateQuantities + "</td></tr>");
     275 +               for (Map.Entry<String, Type> entry : renderSettings.entrySet()) {
     276 +                       String settingName = entry.getKey();
     277 +                       Type settingValue = entry.getValue();
     278 +                       builder.append("<tr><td>" + settingName + "</td><td>" +
         settingValue + "</td></tr>");
     279 +               }
 272 280                 builder.append("</tbody></table>");
 273 281
```

(b) Commit implementing the suggestion

Figure 5: Good first issue example from BIMserver repository

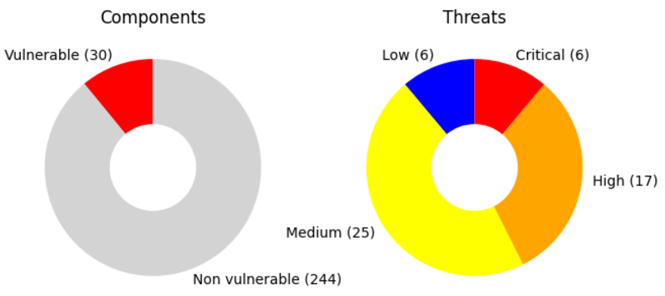


Figure 6: Dependencies analysis on BIMserver 1.5.186

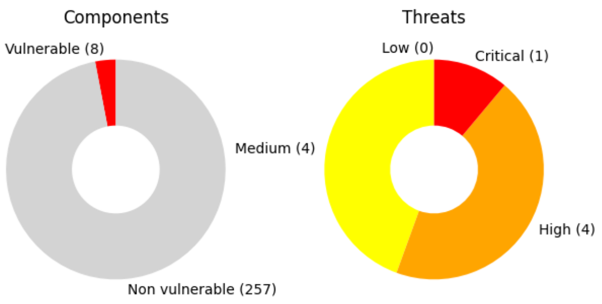
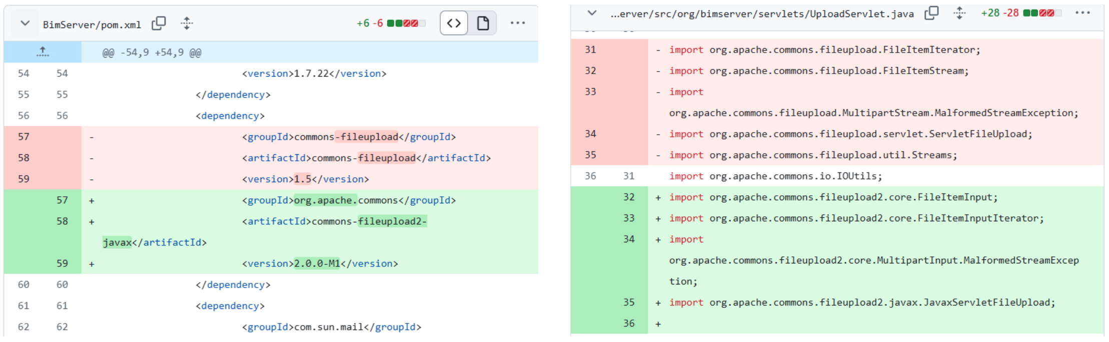


Figure 7: Dependencies analysis on BIMserver 1.5.187



```
@@ -54,9 +54,9 @@
                        <version>1.7.22</version>
                </dependency>
                <dependency>
-                       <groupId>commons-fileupload</groupId>
-                       <artifactId>commons-fileupload</artifactId>
-                       <version>1.5</version>
+                       <groupId>org.apache.commons</groupId>
+                       <artifactId>commons-fileupload2-
javax</artifactId>
+                       <version>2.0.0-M1</version>
                </dependency>
                <dependency>
                        <groupId>com.sun.mail</groupId>
```

```
- import org.apache.commons.fileupload.FileItemIterator;
- import org.apache.commons.fileupload.FileItemStream;
- import
  org.apache.commons.fileupload.MultipartStream.MalformedStreamException;
- import org.apache.commons.fileupload.servlet.ServletFileUpload;
- import org.apache.commons.fileupload.util.Streams;
  import org.apache.commons.io.IOUtils;
+ import org.apache.commons.fileupload2.core.FileItemInput;
+ import org.apache.commons.fileupload2.core.FileItemInputIterator;
+ import
  org.apache.commons.fileupload2.core.MultipartInput.MalformedStreamExcep
  tion;
+ import org.apache.commons.fileupload2.javax.JavaxServletFileUpload;
+
```

Figure 8: Code refactoring in BIMserver to support migration from commons-fileupload to commons-fileupload2

pendencies against vulnerability databases. For example, the Jetbrains Java-IDE IntelliJ Idea has a bundled "Package checker" plugin that consults mend.io and OS.dev[6]. These databases aggregate publicly available information on software vulnerabilities assigned to Common Vulnerabilities and Exposures (CVE) numbers. A CVE provides a standardized method to identifying security vulnerabilities and exposures in software. Another such aggregator database is Sonatype OSS Index [7], which provides scanning tools besides the actual vulnerability data.

In addition to the Jetbrains IDE plugin, we employed the OSS Index Maven Plugin to assess the dependencies. The tool examines the project's dependencies and generates a detailed vulnerability report for each dependency, including CVE identifiers, description, severity score, and references. This report can be exported to JSON, XML, or TXT file format for further processing if necessary. The diagram shown in Figure 6 is based on an analysis performed on January 15, 2025, showing that BIMserver 1.5.186 contains 30 vulnerable components, with a total of 54 identified vulnerabilities with different severity: 6 critical, 17 high, 25 medium, and 6 low severity. After the update, as shown in Figure 7, the number of vulnerable components was reduced from 30 to 8, with a total of 9 identified vulnerabilites consisting of 1 critical, 4 high, and 4 medium severity.

During the Java 8 compatible update it turned out that some dependencies could not be updated either because of discontinued development (end-of-life and no longer receiving security updates) or because the security updates break Java 8 compatibility. For example, Jetty 9 will soon reach end-of-life and stop receiving security updates; Apache Oltu has been retired and still contains vulnerabilities; and the latest Java 8 - compatible versions of some libraries, such as Eclipse JDT, JavaX JSON, Commons Fileupload, still contain unresolved security issues. Hence the above diagrams still show remaining vulnerabilities. Those, in a second step Java 8 compatibility has been dropped and the remaining dependencies were updated where possible. For the unmaintained dependencies, various options have to be explored, either finding other active replacement projects or helping to fix them ourselves. This process has not yet been completed. Also, it is not clear whether it is worth to keep a Java 8 compatible branch with known vulnera-

[6] Jetbrains package checker: https://plugins.jetbrains.com/plugin/18337-package-checker
[7] Sonatype OSS Index: https://sonatype.github.io/ossindex-maven/maven-plugin/

bilities.

During this updating process, contributors gained a deeper understanding of when these libraries are used and where they are called from in the codebase. Yet, without lots of experience it remains challenging to assess the impact of a dependency update and to infer what to test after an update, particularly given the low test coverage (see Section 4.4). Another challenge is that some dependency updates with larger version jumps require not only updating to a newer version, but also modifying the related classes and adapting the codebase to accommodate the changes. For instance, when migrating from org.apache.commons.fileupload to org.apache.commons.fileupload2 (Figure 8), class references such as FileItemIterator, FileItemStream, had to be replaced with their updated counterparts in commons.fileupload2 namespace, FileItemInput and FileItemInputIterator, respectively.

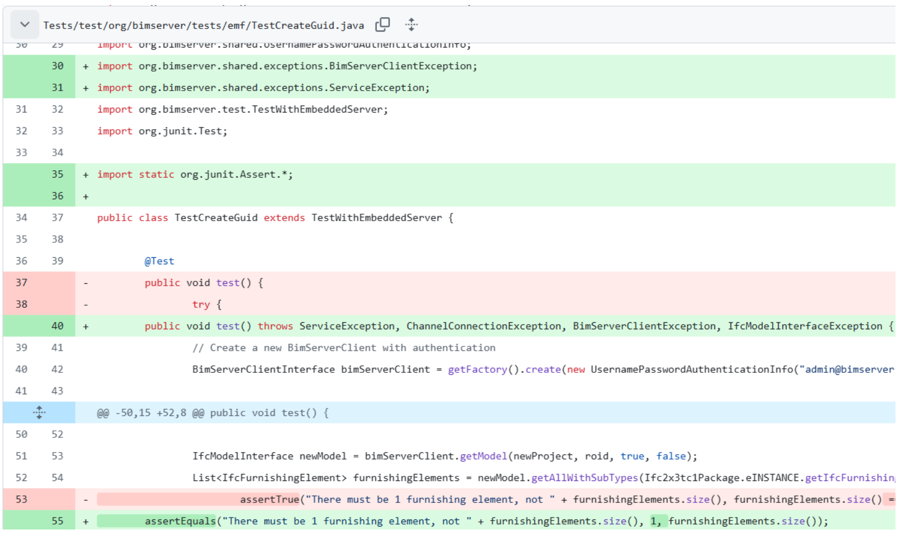
Through this process, contributors not only gained more confidence in working with the codebase, but also made a meaningful contribution to the overall quality of BIMserver. The first step in updating the dependencies already resulted in a more secure system by significantly reducing vulnerabilities.

4.4 Automated testing

BIMserver has two test types; automated tests using JUnit and manual tests using executable classes. Most of the latter seem to have been written to try something out before implementation rather than testing implemented functionality. Yet, some of those tests could be turned into useful tests. In total, there are 123 test files, consisting of 63 JUnit test files and 60 executable test classes— distributed across seven test packages. These tests can be categorized further into three types based on how they interact with BIMserver during execution. The first category includes tests that do not require BIMserver to be running; these are mostly automated tests. The second category includes tests in which BIMserver is started as an embedded server during execution. The third category includes tests that require an external BIMserver instance to be running before the test can begin.

As part of the maintenance process, contributors began by refactoring the existing test files to improve code clarity, for example by removing unnecessary try/catch blocks (Figure 9). After this initial cleanup, contributors examined the current status of unit and integration tests in BIMserver. 13 out of 64 JUnit tests are failing, either due to assertion errors or errors originating from BIMserver, and these require further attention. In parallel with the automated tests, contributors also performed manual testing via BIMvie.ws and the console web application, using IFC sample data in two versions of the schema: IFC 2X3TC1 and IFC 4.

As part of good development practice, test-driven development suggests to write tests before implementation, e.g. to reproduce a bug-type issue with a test before fixing it. Even though this would be great way to increase test coverage, we only applied it in few cases due to the order in which we initially went through the four maintenance areas with automated testing last.



```
Tests/test/org/bimserver/tests/emf/TestCreateGuid.java
    import org.bimserver.shared.UsernamePasswordAuthenticationInfo;
  + import org.bimserver.shared.exceptions.BimServerClientException;
  + import org.bimserver.shared.exceptions.ServiceException;
    import org.bimserver.test.TestWithEmbeddedServer;
    import org.junit.Test;

  + import static org.junit.Assert.*;
  +
    public class TestCreateGuid extends TestWithEmbeddedServer {

            @Test
  -         public void test() {
  -                 try {
  +         public void test() throws ServiceException, ChannelConnectionException, BimServerClientException, IfcModelInterfaceException {
                    // Create a new BimServerClient with authentication
                    BimServerClientInterface bimServerClient = getFactory().create(new UsernamePasswordAuthenticationInfo("admin@bimserver

    @@ -50,15 +52,8 @@ public void test() {

                    IfcModelInterface newModel = bimServerClient.getModel(newProject, roid, true, false);
                    List<IfcFurnishingElement> furnishingElements = newModel.getAllWithSubTypes(Ifc2x3tc1Package.eINSTANCE.getIfcFurnishin
  -                         assertTrue("There must be 1 furnishing element, not " + furnishingElements.size(), furnishingElements.size() =
  +         assertEquals("There must be 1 furnishing element, not " + furnishingElements.size(), 1, furnishingElements.size());
```

Figure 9: Refactoring existing test files

5 Results, evaluation of the measures

The effectiveness of the approach is substantiated by the measurable improvements the project has received in several aspects: On the documentation side, a total of 29 pages were updated and enhanced. In addition to that, the documentation was experimentally migrated to a new platform that supports multiple versions.

From the technical side, several improvements further demonstrate the impact of the approach. A total of 24 bug-related issues were resolved, while 10 obsolete issues were identified and closed. Moreover, more than 40 active discussions were followed up, with around 11 question-related issues being answered and closed. Security and maintainability were further improved through comprehensive dependency updates, which eliminated known vulnerabilities and unused packages—reducing the total from 54 identified threats (6 critical, 17 high, 25 medium, and 6 low severity) to 8 threats (1 critical, 4 high, and 3 medium). Additional enhancements were achieved by refactoring test files to strengthen the automated test suite.

While these improvements clearly display the positive impact of the approach, the limited number of contributors restricts the extent to which the measures can be generally evaluated. However, they still provide meaningful outcomes that can serve as an input for future work. The following evaluation is therefore mainly from the new contributor's view. A more systematic assessment would also be required from the maintainer's point of view and that of other contributors with minor contributions. Furthermore, measures should also be judged in terms of whether they are successful or less successful ones, possibly only good for a certain application field, feasible on the long term or impractical.

Through active involvement in different parts of BIMserver, contributors gradually gained a deeper understanding of the codebase, which in turn enabled them to contribute confidently. This progress was inseparable from the support and guidance from the main developer, who actively engaged in conversations and provided continuous feedback. In particular, setting up frequent discussions during the first months, as well as establishing short daily stand-up meetings, proved crucial for new contributors to clarify uncertainties, align priorities, and establish a shared understanding of the development goals. While daily stand-up meeting were effective for tracking progress, longer meetings of 1–2 hours twice a week were necessary to address technical issues that required deeper analysis such as when working on complex issues and dependency updates. By contrast, updating the documentation required less intensive coordination, and a longer meeting once a week was sufficient. As meetings became more intensive, it was increasingly necessary to ensure the key information was not lost and systematically documented. GitLab was used for both documenting the discussion and managing progress and facilitating the identification of bottlenecks, with the wiki and the kanban board serving these purposes, respectively.

Working on the documentation as a preliminary step proved to be an effective entry point, as it familiarized contributors with the software and lowered the entry barriers. However, the goal in the initial stage should be to obtain an overview of the system architecture rather than detailed knowledge of individual parts, since exhaustive documentation can be overwhelming. Continuous updates throughout the work at later stages might be more effective than one-off revisions. For example, updating pages to reflect the current status of BIMserver was initially difficult or impossible due to limited understanding of the overall software structure, but it became achievable once contributors gained hands-on experience with specific tasks such as issue fixing, updating vulnerable dependencies, and maintaining automated tests. However, the longer contributors focused solely on documentation, the less engaging it became, as the work offered fewer challenges compared to other development tasks. Therefore, balancing documentation with other technical tasks is essential to maintain both contributor engagement and steady project progress. This became particularly evident when contributors worked on migrating the documentation to a new platform. Although still part of the documentation process, this task involved several small technical challenges that not only kept the work engaging but, more importantly, gave contributors a sense of meaningful contribution to both the project and the community.

As new contributors gradually transitioned from documentation to more technical code-related tasks, the guided support from the experienced maintainer played a vital role during the onboarding process. Particularly, the list of small and actionable tasks provided by the experienced maintainer, for example, those issues labelled as *good first issue*, showed highly effective entry points. Their effectiveness was further enhanced when they were accompanied by partial ”hints” indicating where an issues are caused from within the codebase, reducing initial barrier of navigating a complex system. By following these guided solutions, contributors were able to track their progress and experience a sense of accomplishment with each completed tasks. Once the new contributors gained more confidence, the number of hints could gradually be reduced, encouraging them to explore the codebase independently. On the other hand, not only this approach worked for the technical dimension but also for the social aspect, as guided solutions also included advice on how to engage in community discussions, ask for feedback effectively, and collaborate with other contributors.

In some cases, issues required deeper analysis. This was when pair programming became particularly beneficial, as the experienced maintainer could review and provide suggetions in real time. Pair programming was also helpful during the process of updating dependencies where contributors faced several challenges that required thoughtful decision-making to balance compatibility with security improvements. Besides working with maintainer, pair programming also showed valuable when new contributors collaborated with one another, as it not only facilitated knowledge sharing but also made the learning process more enjoyable and engaging. However, this strategy demonstrated helpful largely because collaborators were working within the context of a research project. Further analysis would be necessary to assess its applicability in the context of opensource communities, where work location and dedicated time slots can vary significantly among contributors.

Testing, in particular, offered contributors the opportunity to identify the neglected areas and improve test coverage. Refactoring tests not only helped cleaning up redundant try/catch blocks, but also taught contributors the importance of maintaining synchronization between the code and its associated tests.

Some additional measures have emerged during the study and suggested for implementation, but need further discussion: Documentation could be extended beyond the target group of plugin and client implementers to core developers by adding sections on debugging, contribution, and code of conduct. Tasks could be prioritized via issues by introducing a new issue type such as *high-priority* can be introduced to organize tasks more efficiently.

6 Discussion and future work

In this paper, we presented a structured and strategic approach for maintaining the Opensource BIMserver project, focusing on improving new contributor involvement through an effective onboarding process. The approach has been employed in four key maintenance areas: updating documentation, handling and resolving issues, managing dependency updates, and reviewing and improving automated tests. Overall, the selected strategies have been proven effective when maintaining a complex opensource project with limited resources and time.

The approach was only implemented on a small scale and no structured approach to evaluation has been employed. The strategies and measures have only been evaluated in the context of a research project with three persons working on the BIMserver: the two authors — one as the experienced maintainer and the other as a new contributor receiving full support and feedback from the main developer — as well as two interns, each employed for a period of few months. As a result, the findings may not fully reflect the collaborative process typically present in opensource communities. Additionally, the scope of the work was limited by the objective and time constraints of the project, which might have affected the depth of implementation.

Future work is needed to validate the proposed approach in broader community setting, involving more contributors with varying level of experience. This could include the implementation of onboarding guideline and the introduction of the structured evaluation method such as contributor surveys, contribution tracking, and code quality metrics.

7 Acknowledgements

This work is funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the central innovation programme for small and medium-sized enterprises (ZIM-program), with funding provided under grant number 16KN106902.

Bibliography

[AWGC19] D. Arya, W. Wang, J. L. Guo, J. Cheng. Analysis and Detection of Information Types of Open Source Software Issue Discussions. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. Pp. 454–464. 2019. doi:10.1109/ICSE.2019.00058

[AZRG05] M. AlMarzouq, L. Zheng, G. Rong, V. Grover. Open source: Concepts, benefits, and challenges. *Communications of the Association for Information Systems* 16(1):37, 2005. doi:10.17705/1CAIS.01637

[BBLH10] J. Beetz, L. van Berlo, R. de Laat, P. van den Helm. BIMserver.org: An open source IFC model server. In *Proceedings of the CIB W78 2010: 27th International Conference - Applications of IT in the AEC Industry*. Cairo, Egypt, November 2010.

[BSV$^+$25] R. M. A. Baracho, L. G. da Silva Santiago, M. J. M. Vidigal, M. F. Porto et al. Building information modeling BIM for planning and construction. *Architecture, Structures and Construction* 5(1):1–9, 2025. doi:10.1007/s44150-024-00119-x

[DLT$^+$14] W. Ding, P. Liang, A. Tang, H. Van Vliet, M. Shahin. How do open source communities document software architecture: An exploratory survey. In *2014 19th International conference on engineering of complex computer systems*. Pp. 136–145. 2014. doi:10.1109/ICECCS.2014.26

[DMC17] A. Decan, T. Mens, M. Claes. An empirical comparison of dependency issues in OSS packaging ecosystems. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. Pp. 2–12. 2017. doi:10.1109/SANER.2017.7884604

[FJMD12] D. Forrest, C. Jensen, N. Mohan, J. Davidson. Exploring the role of outside organizations in Free/Open Source Software projects. In *Open Source Systems: Long-Term Sustainability: 8th IFIP WG 2.13 International Conference, OSS 2012, Hammamet, Tunisia, September 10-13, 2012. Proceedings 8*. Pp. 201–215. 2012. doi:10.1007/978-3-642-33442-9_13

[GHI21] R. S. Geiger, D. Howard, L. Irani. The Labor of Maintaining and Scaling Free and Open-Source Software Projects. *Proc. ACM Hum.-Comput. Interact.* 5(CSCW1), Apr. 2021. doi:10.1145/3449249

[IC20] J. L. C. Izquierdo, J. Cabot. A Survey of Software Foundations in Open Source. 2020. arXiv preprint. doi:10.48550/ARXIV.2005.10063

[ISO04] ISO 10303-11. Industrial automation systems and integration. Product data representation and exchange. Part 11: Description methods: The EXPRESS language reference manual. Technical report 10303-11, International Organization for Standardization, Geneva, Switzerland, 2004.

[ISO18] ISO 16739-1:2018. Industry Foundation Classes IFC for data sharing in the construction and facility management industries. Part 1: Data schema. Technical report 16739-1:2018, International Organization for Standardization, Geneva, Switzerland, 2018. IFC 4.0.2.1 (IFC4 ADD2 TC1).

[LLW$^+$20] Y. Lu, C. Li, S. Wang, Y. Liu, Y. Lu, J. Dai. A Research on Testing Strategies of OSS Used by Equipment Software. In *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*. Pp. 121–128. 2020.
doi:10.1109/ICSESS49938.2020.9237726

[LVKS17] S. Logothetis, E. Valari, E. Karachaliou, E. Stylianidis. Spatial DMBS architecture for a free and open source BIM. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42:467–473, 2017.
doi:10.5194/isprs-archives-XLII-2-W5-467-2017

[LW99] T. Liebich, J. Wix. Highlights of the development process of industry foundation classes. In *Proceedings of the 1999 CIB W78 Conference*. Volume 18, pp. 2758–2775. 1999.

[NRN21] R. Nadri, G. Rodríguez-Pérez, M. Nagappan. On the relationship between the developer's perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering* 48(8):2955–2968, 2021.
doi:10.48550/arXiv.2104.06143

[SA10] K.-J. Stol, M. Ali Babar. Challenges in using open source software in product development: a review of the literature. In *Proceedings of the 3rd international workshop on emerging trends in free/libre/open source software research and development*. Pp. 17–22. 2010.
doi:10.1145/1833272.1833276

[SGBB25] R. Sacks, L. Ghang, L. Burdi, M. Bolpagni. *BIM Handbook: A Guide to Building Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers*. Wiley, Hoboken, NJ, 4th edition edition, 2025.

[TWS$^+$22] B. Trinkenreich, I. Wiese, A. Sarma, M. Gerosa, I. Steinmacher. Women's Participation in Open Source Software: A Survey of the Literature. *ACM Trans. Softw. Eng. Methodol.* 31(4), August 2022.
doi:10.1145/3510460

[YKB14] I. Yahav, R. S. Kenett, X. Bai. Risk based testing of open source software OSS. In *2014 IEEE 38th International Computer Software and Applications Conference Workshops*. Pp. 638–643. 2014.
doi:10.1109/COMPSACW.2014.107