On the Road of Data-driven Discussion: a Comparison of Open-source Collaboration Platforms

Jerry Andriessen[1,†], Tiziano Citro[2,*,†], Thomas Schaberreiter[3,†] and Luigi Serra[4,†]

[1] *Wise & Munro Learning Research, Netherlands*
[2] *Department of Computer Science, Università degli Studi di Salerno, Italy*
[3] *CS-AWARE Corporation, Estonia*
[4] *Department of Computer Science, Università degli Studi di Salerno, Italy*

Abstract
Collaboration platforms empower teams to discuss, brainstorm and work together with immediate feedback regardless of location. The purpose of collaboration platforms is to integrate social networking capabilities into work processes, which can help raise productivity. With this study, we present a comparison of open-source modern collaboration platforms. It compares open-source platforms, publicly available on version control systems, such as GitHub or GitLab, regarding features, extensibility on the developer side, and end-user support regarding documentation. As a result, there is no perfect or better platform than the others in all cases. All of them are stable, with a community supporting them, adding new features, and keeping them updated with security patches. Besides discussing the platforms' assessment and comparison, the article contextualizes this theoretical analysis in a concrete use case in the context of the Horizon Europe project CS-AWARE-NEXT, which requires a collaboration platform to perform data-driven discussions, providing users with a continuous view of their data, and enabling them to effortlessly keep up with messages and reply in context, even on complex topics that involve significant amounts of data.

CCS CONCEPTS Evaluation, Surveys and overviews

Keywords
Open-source collaboration platforms, comparison, evaluation, assessment, data-driven collaboration

1. Introduction

A collaboration platform is an application that helps teams or groups of individuals accomplish specific goals collaboratively. Modern collaboration platforms enable contributors to engage in conversations or dialogues using text, video, audio, and imagery [1]. Contributors can make posts that are viewable by others, creating a dynamic and interactive environment. These conversations can be valuable for prompting reflection on topics of interest and sharing thoughts and information related to those topics. Collaboration platforms foster communication, knowledge-sharing, and collaboration among individuals and teams [2].

S3C'23: Sustainable, Secure, and Smart Collaboration Workshop, Hosted by CHITALY 2023, SEPTEMBER 20–22, 2023, TURIN, ITALY
*Corresponding author.
†These authors contributed equally.

jerry@wisemunro.eu (J. Andriessen); t.citro5@studenti.unisa.it (T. Citro); thomas.schaberreiter@cs-aware.com (T. Schaberreiter); lserra@unisa.it (L. Serra)

In today's digital landscape, the need for effective collaboration platforms has increased in various contexts. Whether it's in educational institutions [3], professional settings [2, 4], or open-source communities [5, 6, 7], the demand for platforms that facilitate the seamless sharing of materials and meaningful discussions is crucial. These platforms offer a range of features and capabilities that cover several user requirements and enable collaboration on a global scale. One critical aspect differentiating collaboration platforms is the level of support they provide to end users and the extent to which developers can customize and extend their functionalities. Factors such as the availability of comprehensive user manuals, the flexibility for developers to adapt and enhance the platform, and an open license can significantly influence the selection process. Additionally, the existence of an organization, foundation, or entity that supports or leads the projects is often a good sign of credibility and stability for the project, assuring the users to adopt the platform.

While existing related works assess collaboration platforms independently or mainly focusing on a single feature or are limited to enterprise context [8], the literature lacks a comparison among several platforms that consider both developers' requirements, such as the ability to start from an open-source project and customize it according to their specific needs, and end-users requirements, particularly in terms of support and documentation to facilitate platform adoption and learning.

This study explores and compares publicly available collaboration platforms hosted on version control systems, such as GitHub or GitLab. Through an examination of various aspects such as features architecture, customization, community, and provisioning, a comprehensive evaluation is performed to determine the strengths and weaknesses of each platform. Furthermore, this analysis contextualizes the findings within the specific context of the Horizon Europe project CS-AWARE-NEXT[1]. This study presents a practical use case of a real-world analysis implementation by identifying a platform that meets the project's requirements. In the context of CS-AWARE-NEXT, the objective was to identify the most suitable platform to serve as the building block for designing and developing a data-driven collaboration platform. Such a platform equalizes the importance of communication and the data it revolves around, providing users with a continuous view of their data and enabling them to effortlessly keep up with messages and reply in context, even on complex topics that involve significant amounts of data. Key considerations in this evaluation process of meeting the project's requirements were the platforms' capability for extensive customization to align with our specific needs and their viability for long-term maintenance. A crucial factor was the inclusion of a chat feature that would facilitate discussions among users, enabling collaboration around shared data.

2. Related Works

Rarely publicly available collaborative platforms are launched by academic contributions. However, it is pretty standard that their performances are assessed within the literature also contextualized in a specific domain. For instance, Burkert and Federrath [9] conducted a source code analysis of Mattermost to examine the usage of timestamps for privacy mining. Ćirković et al. [10] explored the use of Mattermost in learning environments, considering the benefits for

[1]https://cordis.europa.eu/project/id/101069543

both teachers and students. Barde and Raouf [11] presented a case study on Zulip to demonstrate the effectiveness of their accessibility evaluation approach. Ihle et al. [12] proposed a cryptographic challenge-response authentication mechanism for portable identities within Matrix. Long et al. [13] developed a web-based platform virtual world built on top of Matrix. Ermoshina et al. [14] provided an overview of messaging protocols focusing on end-to-end encryption, including Matrix as one of the presented protocols.

All the previously reported works focus on the assessment of a single platform. Focusing on related results that compare collaborative platforms, we can cite Schipper et al. [15], who conducted a forensic analysis of Matrix and Element (previously known as Riot). Schubert and Williams [8] conducted a review using a questionnaire to identify the enterprise collaboration platforms used in a set of organizations, analyzing the types of tools, level of functionality within tools, and degree of integration between tools.

However, to the best of our knowledge, there is no systematic review of publicly available, potentially open-source collaboration platforms that compares them against a defined set of evaluation criteria. While there is literature studying or comparing specific aspects of collaboration platforms or protocols, none has provided a comprehensive assessment. Hence, to cope with this limitation, our study compares publicly available and potentially open-source platforms via evaluation criteria, including architecture, customization, community, and provisioning, to assess these platforms comprehensively.

3. Platforms Overview

Collaboration platforms were identified through a combination of methods. Firstly, a comprehensive search was conducted on GitHub[2] using relevant keywords. The investigation was performed using the query "collaboration stars:>100" to identify projects on collaboration with a minimum of 100 stars. This criterion was chosen to indicate the community interest and facilitated our search process. Due to GitHub's limitations on result exports, the analysis of projects had to be done manually. Filtering for projects with at least 100 stars helped to narrow down the number of platforms to a manageable level for manual analysis. It's worth noting that while the minimum requirement was set to 100 stars, most platforms had thousands to tens of thousands of stars. Platforms with fewer stars were typically prototypes, personal projects, or not considered production-grade.

Additionally, a search of online forums and discussions was carried out to hand-search other publicly available collaboration platforms missed through the GitHub search or not available on GitHub but on other version control systems, such as GitLab[3].

Searched platforms were passed through a process of eliminating duplicated results. To check results were relevant to the study, all the remaining platforms were assessed against the inclusion and exclusion criteria presented in Figure 1. To be included, platforms had to be publicly available and offer a certain or potential degree of customization. Additionally, active community support and the availability of a chat feature were factors for inclusion. Platforms that did not meet these criteria were excluded from the analysis. Consequently, popular

[2]https://github.com
[3]https://about.gitlab.com

platforms like Discord[4] and Teams[5] were not included in this study due to their proprietary nature as non-open-source software.

As shown in Figure 1, of all the 309 projects (304 from GitHub search and five by hand-searching), 17 were discarded because they were considered duplicates, and the remaining 292 projects were checked against inclusion and exclusion criteria, resulting in 10 projects being included and considered for the study. Details related to each platform follow.

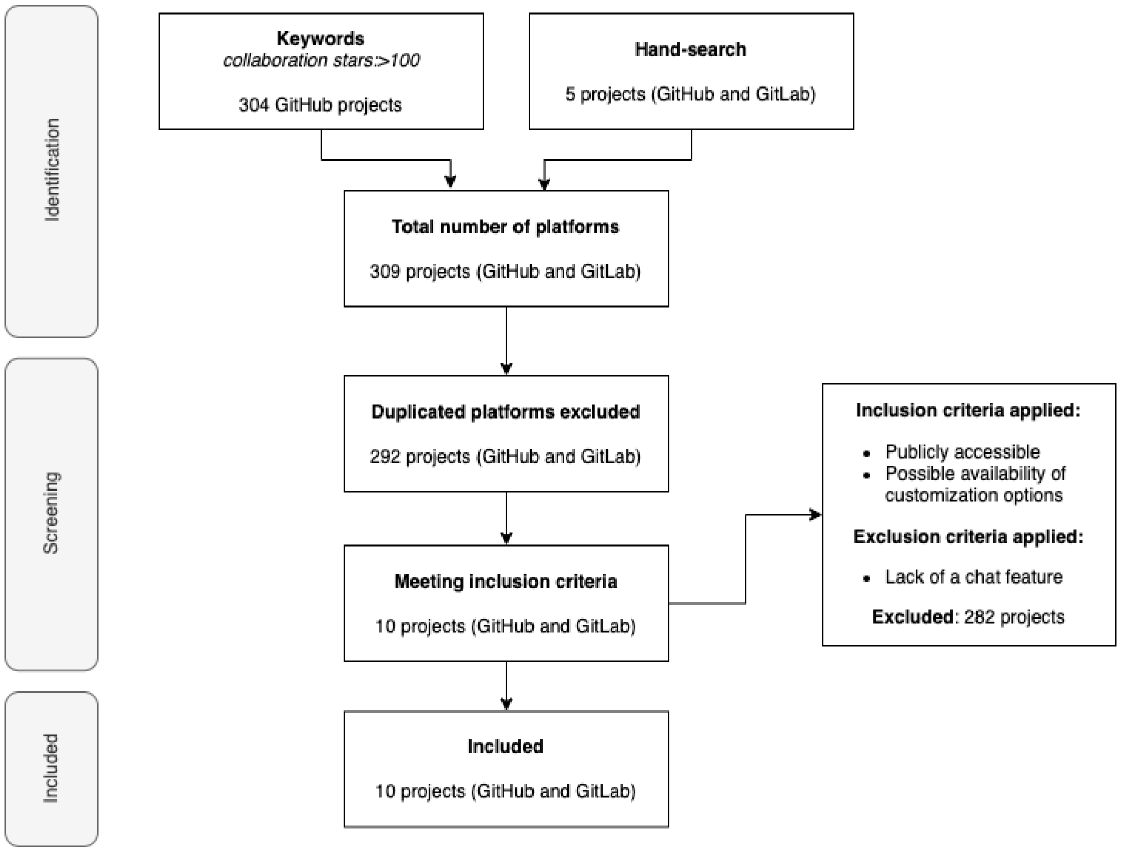
Identification
Screening
Included
Keywords
collaboration stars:>100
304 GitHub projects
Hand-search
5 projects (GitHub and GitLab)
Total number of platforms
309 projects (GitHub and GitLab)
Duplicated platforms excluded
292 projects (GitHub and GitLab)
Meeting inclusion criteria
10 projects (GitHub and GitLab)
Included
10 projects (GitHub and GitLab)
Inclusion criteria applied:
• Publicly accessible
• Possible availability of customization options
Exclusion criteria applied:
• Lack of a chat feature
Excluded: 282 projects

Figure 1: Diagram of the search process.

Mattermost Mattermost [16, 17] is a collaboration platform designed explicitly for digital operations. It offers a chat solution powered up by providing integrated productivity solutions matched to standard Research and Development processes: planning sprints, managing releases, resolving incidents, and more. It is open-source, extensible by design, and supported by an ecosystem of community-driven applications, an open Application Programming Interface (API), and a framework that allows you to customize and enhance the platform to meet your needs.

[4]https://discord.com
[5]https://www.microsoft.com/en-us/microsoft-teams

Mattermost implements collaboration around channels that connect people, processes, and tools. They allow direct, private, and group messaging, meetings, calls, and file sharing within searchable channels. Kanban-style boards enable teams to manage projects and tasks that integrate with channel-based collaboration and playbooks: collaborative checklists with channel-based communication, automation, and reporting.

Matrix Matrix [18, 19] is an open standard for interoperable, decentralized, real-time communication. It is maintained by the non-profit Matrix.org Foundation, whose aim is to create an independent open platform for communication. It is interoperable because it is designed to interoperate with other communication systems, and it is an open standard because it is easy to understand how to interoperate with it. Decentralization is achieved by having no central point, meaning anyone can host their server and control their data. In Matrix, users connect to a server, which becomes their *homeserver*. Users can participate in rooms on any homeserver since each server federates with the others. Recently, the community released a new beta feature inspired by the Discord server model, introducing spaces: a way to group rooms together.

Element Element [20, 21] is an open-source platform built by the community behind the Matrix project. It is built on top of Matrix with the idea of being a light skin of customization on top of the underlying Matrix protocol. The platform allows for direct and group messaging, meetings, and calls.

Cinny Cinny [22, 23], just like Elements, is a platform built on top of Matrix, focusing on simplicity and a modern user interface designed to give a pleasant and calm feeling. The primary focus is on the web experience, which means no mobile application is available. Collaboration happens via direct messages and channels.

Gitter Gitter [24, 25] is an open-source collaboration and networking platform that helps to create, manage, grow, and connect communities through messaging and discovery. Communities are organized in channels that can also be accessed via any other Matrix client since Gitter has become part of the Matrix project.

Zulip Zulip [26, 27] is an open-source team collaboration platform with topic-based threading that combines features of email and chat to address the lack of organization and context in usual channels, which leads to searching hundreds of messages daily to find relevant content. Zulip channels are called streams and provide the benefits of real-time communication while being great at asynchronous communication. Zulip is inspired by the email's highly effective threading model: a stream's message has a topic, just like every message in an email has a subject line. Topics enable cohesive conversations and an efficient catch-up on messages and replying in context, even to old discussions.

Twake Twake [28, 29] is an open-source collaboration platform with all the essential features: team chats in channels, file storage, team calendar, and tasks manager. Twake channels allow team members to exchange messages in real-time, have video calls, share files, and collaborate on documents with a built-in file manager called Drive, which allows for standard operations: versioning, editing, etc.

Rocket.Chat Rocket.Chat [30, 31] is an open-source collaboration platform with integration with multiple conversation channels, such as emails, Facebook, Twitter, WhatsApp, and many more, to build a single view of communication that helps in reducing the time needed to switch between the tools.

In Rocket.Chat conversations are organized across different topics via rooms or channels because it is the most used room type. There are five different types of rooms:

1. Channels: public or private chat rooms designed to fasten communication.
2. Teams: digital workspaces where team members can collaborate and work together by sharing and managing multiple channels.
3. Discussions: a logical separation between conversations of larger topics in a team or channel.
4. Direct Messages: confidential one-on-one or one-to-many conversations;
5. Threads: conversations with a specific message as a starting point on a specific topic in a team or channel.

Corteza Corteza [32, 33] is a high-performance open-source collaboration platform designed for team productivity and to facilitate secure communication with other organizations or customers. In Corteza, teams can work together through public or private channels organized by topic and group messages into threads. Each channel can be managed with customized rules using a role-based access control system for better security and privacy.

Revolt Revolt [34, 35] is an open-source platform where collaboration happens in servers, similar to the more popular Discord. It allows teams to collaborate using topic-based channels, share files, and engage in video calls. It is not released for all operating systems because it misses an IOS application.

3.1. Features Summary

A comparison of the features offered by each platform is summarized in Table 1. We identified nine core features defined as follows.

Channels, direct messages, threads, calls, and file sharing. These are essential features expected from modern collaboration platforms to accommodate users' needs. It is important to note that terminology may differ across platforms, with variations such as rooms, groups, or other terms used to describe channels.

Teams. This feature refers to the capability of organizing discussions into separate and isolated namespaces. It is important to note that each platform uses different terminology for this feature. For example, they are called spaces in Matrix, while Revolt uses the term servers.

Task management. The platform offers features designed to assist teams in organizing and managing their tasks effectively, thereby enhancing overall productivity.

Table 1
Platforms' features overview.

	Teams	Channels	Direct messages	Threads	Calls	File sharing	Task management	Topic orientation	Discovery
Mattermost	✓	✓	✓	✓	✓	✓	✓	~	
Matrix	✓	✓	✓	✓	✓	✓		~	✓
Element	✓	✓	✓	✓	✓	✓		~	✓
Cinny	✓	✓	✓			✓		~	✓
Gitter	✓	✓	✓	✓	✓	✓		~	✓
Zulip		✓	✓	✓	✓	✓		✓	
Twake	✓	✓	✓	✓	✓	✓	✓	~	
Rocket.Chat	✓	✓	✓	✓	✓	✓		~	
Corteza		✓	✓	✓		✓		~	
Revolt	✓	✓	✓	✓	✓	✓		~	✓

Topic orientation. Most platforms enable users to organize their conversations based on topics of interest. However, in most platforms, there is no strict enforcement of a designated topic for each discussion or individual message, as indicated by the tilde (~) in Table 1. The enforcement of a topic is unique to Zulip, which draws inspiration from the email threading model and assigns a topic to every message, ensuring structured conversations.
Discovery. This refers to a feature that enables users to explore, join, and collaborate with online communities within the collaboration platform. Similar to the discovery functionality found in popular platforms like Discord, this feature allows users to search for communities based on specific interests, topics, or keywords. With the help of this feature, users can discover new communities, connect with like-minded individuals, and expand their network.

4. Platforms Evaluation

We evaluated the platforms against four predefined criteria: architecture, customization, community, and provisioning. By assessing each platform against these criteria, we could make an informed decision that aligns with our objectives and requirements.

4.1. Criteria

Each criterion has a score of 1 to 5, the lowest and the highest possible scores, respectively. The four criteria definitions are as follows.
Architecture. Understand the what and why of each component in the system.
Rating.

- **5**: *Excellent architecture.* The platform is exceptional in terms of structure and organization. Components are highly cohesive and modular, fostering flexibility and reusability. The architecture is great in scalability, maintainability, and extensibility.

- **4**: *Good architecture.* The platform has a well-defined structure and organization. Components are clearly defined and integrated, leading to efficiency and effectiveness. The architecture supports scalability, maintainability, and extensibility.
- **3**: *Adequate architecture.* The platform has a reasonable structure and organization. Components are well-defined and fulfill their intended purposes. The architecture allows for scalability, maintainability, and extensibility to a certain extent.
- **2**: *Below average architecture.* The platform has some structure, but it lacks clarity and cohesiveness. Components are somewhat defined, but there are inconsistencies. There are limitations in scalability, maintainability, and extensibility.
- **1**: *Limited architecture.* The platform lacks structure and organization. Components are poorly defined, leading to inefficiency. There is little consideration for scalability, maintainability, and extensibility.

Customization. Identify challenges in making changes to the platform to extend existing functionalities.
Rating.

- **5** *Excellent customization.* The platform has great customization capabilities to adapt and extend the system to specific needs. The platform supports comprehensive customization mechanisms, such as APIs, plugins, Software Development Kits (SDKs), and more, enabling extensive integration and customization possibilities.
- **4**: *Good customization.* The platform provides extensive customization options and flexibility. It is possible to change various aspects, and advanced customization options, such as APIs or plugins, are available to enable deep customization and integration with external systems.
- **3**: *Adequate customization.* The platform offers moderate customization capabilities. Certain aspects of the platform can be customized, but the scope of customization may still be limited, and more advanced modifications may require significant effort.
- **2**: *Below average customization.* The platform provides some basic customization options, limited in scope and flexibility. Advanced customization is not supported.
- **1**: *Limited customization.* The platform offers minimal or no options for customization.

Community. Analyzing documentation quality and community activity.
Rating.

- **5**: *Excellent community.* The platform has a strong and large community that actively drives the platform's development, growth, and support. Users can find a highly active community with extensive discussions and valuable resources.
- **4**: *Good community.* The platform has a large community with high engagement, participation, and support. Users can find a wealth of resources.
- **3** *Adequate community.* The platform has an active community. Users can find helpful resources, seek help and find relevant information quite easily.
- **2**: *Below average community.* The platform has a growing community. Users can find some support and resources, but the community may still be less active than other platforms.

- **1**: *Limited community.* The platform has a small community with minimal participation and support. There is a lack of resources available. Users may find it challenging to seek help or find relevant information.

Provisioning. Understand the challenges in provisioning and configuring the platform.
Rating.

- **5**: *Excellent provisioning process.* The platform provides a seamless provisioning process that makes setting up and configuring the platform easy. The process is well-documented, streamlined, and requires minimal intervention to get up and running without encountering significant obstacles.
- **4**: *Good provisioning process.* The platform offers a streamlined provisioning process, so it is quick to set up and configure it using comprehensive documentation.
- **3**: *Adequate provisioning process.* The platform has a good provisioning process that is relatively straightforward and well-documented. Clear instructions can be followed to set up and configure the platform.
- **2**: *Below average provisioning process.* The platform has a moderately challenging provisioning process. While some documentation is available, there may still be difficulties during the setup and configuration.
- **1**: *Limited provisioning process.* The platform has a complicated and challenging provisioning process. It may lack clear documentation, making setting up and configuring the platform difficult.

4.2. Results

The study results have been summarized in Table 2, which presents an overview of the platforms' performance based on our evaluation criteria. The table also includes a score column, which provides a summarized view of the results based on the average of all the scores given for each evaluation criterion to each platform.

Table 2
Result table based on evaluation criteria.
(*) *means the score is the same as Matrix because architecture is based on the Matrix's.*

	Architecture	Customization	Community	Provisioning	Score
Mattermost	5	4.5	5	5	4.9
Matrix	5	5	4.5	4	4.6
Element	5*	1	3	4	3.2
Cinny	5*	1	2	4	3
Gitter	4	2	3	3.5	3.1
Zulip	3	3	4	3	3.3
Twake	4	3.5	2	3.5	3.3
Rocket.Chat	4	4	4	4	4
Corteza	4	1	2	3.5	2.6
Revolt	5	2	2	4	3.3

Mattermost Mattermost architecture is designed to be highly operable and scalable and offers the flexibility to separate and manage some components separately. While customization is rated at 4.5, it can be seen as a 5 with the detraction because Matrix offers greater flexibility but at the cost of a significant effort to build everything from scratch. Mattermost stands out as one of the best choices for community and provisioning. It offers high-quality documentation, a reference architecture for local and fully managed clusters, a Kubernetes operator, and other features that make it easier to provide the platform.

Matrix One of the best advantages of Matrix architecture is its decentralized conversation store. Matrix is a building block with all the necessary features for a collaboration platform. It can be extended by developing other services to run alongside the homeserver to meet specific requirements. This gives the flexibility to design a user interface that best fits needs, but this advantage comes with a trade-off as it requires a significant effort. Matrix community and documentation receive a slightly lower rating than Mattermost, as the latter has better documentation. While provisioning Matrix is easy, it lacks the simplification provided by Mattermost, such as a reference architecture or a Kubernetes operator.

Element and Cinny Element is built on top of the Matrix architecture, with its architecture serving as a user interface. However, customization options for Element are limited, and changing the source code is the only way to customize the user interface. While Element has a large community, there is a lack of documentation, particularly on self-hosting and extending the code base without incurring additional costs. The provisioning process for Element is similar to Matrix but with a user interface to manage.

Cinny follows a similar pattern, with limited customization options and a lack of documentation. However, Cinny's disadvantage is its smaller community compared to Element.

Gitter Gitter architecture is robust, but customization options are limited. The platform offers some options we would build from scratch on other platforms. However, the Gitter community does not provide high-quality documentation.

Zulip Zulip architecture is similar to Gitter's. A single server handles most features, providing less decoupling than other platforms. Customization is limited to changing the source code, as with Element and Cinny. However, Zulip has well-written documentation that explains almost all the steps required to introduce and maintain a new feature. Zulip benefits from a large community and well-written documentation. Unfortunately, the Docker image provided by the community is still experimental, and there are rough edges to keep in mind when provisioning the platform.

Twake Twake architecture stands out for its emphasis on security, a top priority for the platform's community. While Twake's features can be extended without modifying the source code by using some built-in options, these are limited, and more complex functionalities may require coding changes. Unfortunately, the Twake community is relatively small, and the available documentation is not always helpful. Provisioning Twake is straightforward but may be challenging for some users due to the lack of detailed documentation.

Rocket.Chat Rocket.Chat architecture is well-designed and allows disabling features for improved performance. Rocket.Chat offers several features for customization, but some are still experimental or limited in their capabilities. One notable feature of Rocket.Chat is its ability to handle bulk loads particularly well. While provisioning the platform is generally straightforward, it's worth mentioning that users may experience a greater complexity when compared to other platforms.

Corteza Corteza architecture is designed to use its monitoring system, which can be both an advantage and a drawback. On the one hand, it simplifies provisioning the platform. On the other hand, it can limit observability. The community has deprecated the plugin API for customization. Due to its small community, finding support or well-written documentation during the provisioning process can be challenging.

Revolt Revolt architecture is unique in that it is designed to be fragmented, with components deployed separately and communicating with each other. This means changes can be made to specific parts without restarting the entire platform. While customization options are limited and the documentation is not comprehensive, provisioning Revolt is not significantly more complex than other platforms. Additionally, the community is not as large as some others.

Architecture All platforms are built upon a distributed architecture, clearly distinguishing components with real-time responsibilities from those without. Matrix sets itself apart from other platforms by aiming to provide a decentralized infrastructure, allowing anyone to run their server instead of relying on resources provided by the platform provider. Most platforms rely on APIs for synchronous and asynchronous communication and web sockets for real-time communication. Rocket.Chat and Matrix differentiate themselves by relying on event-oriented communication. Notably, there is no specific preference towards SQL or NoSQL databases among these platforms, as both options can be utilized to achieve persistence for collaboration data.

Customization In terms of customization, there is a significant disparity among platforms. Mattermost stands out by offering multiple approaches to extend existing features and incorporate new ones, providing a wide range of options unmatched by other platforms. Matrix supports customization by design as it serves as a building block for any possible collaboration platform, making it easy to implement desired features. Opposed to them, Element lacks support for customization, while Corteza has deprecated its plugin API.

Community Most communities are very active and have a large user base. However, younger projects like Corteza and Revolt may have comparatively lower activity levels and smaller user bases. There is a noticeable disparity among platforms in terms of documentation. Whereas platforms like Mattermost and Matrix stand out by offering detailed and exhaustive documentation, others like Cinny lack high-quality documentation. Among the commonly used programming languages across platforms, JavaScript is the most used. Notably, it is interesting the adoption of Golang, which is predominant in some platforms such as Mattermost. Additionally, the Matrix community is developing new core components in Golang, gradually replacing the existing Python-based ones.

Table 3
Platforms' community metrics.
(*) *means it is partially under the underlined license.*
(~) *means there is partial support.*

	License	First commit	Last commit	Commits	Contributors	Organization	Support
Mattermost	Apache-2.0	2015	2023	16,925	782	✓	[36]
Matrix	Apache-2.0	2014	2023	22,756 + 2,750	463 + 163	✓	[37]
Element	Apache-2.0	2015	2023	12,616	525	✓	[38]
Cinny	AGPL-3.0	2021	2023	1,064	38	✓	
Gitter	MIT	2012	2023	48,308	100+	✓	
Twake	AGPL-3.0	2020	2023	1,240	42	✓	[39]
Zulip	Apache-2.0	2012	2023	51,971	843	✓	[40]
Rocket.Chat	MIT*	2015	2023	23,741	826	✓	[41]
Corteza	Apache-2.0	2018	2023	4,775	26	✓	[42]
Revolt	AGPL-3.0	2020	2023	1,593 + 887	56 + 19		[43]~

The data presented in Table 3 was gathered from the GitHub or GitLab pages of the respective projects. It is important to note that most platforms have code bases organized into multiple repositories. Therefore, we had to collect information from all repositories or select a relevant subset for our analysis. We opted for the latter approach to ensure efficiency in our manual analysis process. In particular, we focused on core projects, where core means repositories required to set up the platform with the bare minimum in a way that works. For instance, Mattermost can be tailored to the users' needs thanks to many community-developed plugins made available via dedicated repositories. However, the built-in basic collaborative features of the server are enough to make it work. Conversely, Matrix has two primary servers, released via two different repositories, that we need to consider to have an overall view of what users can use to set up the platform.

To find the license, we searched for the LICENSE.txt file in the project's root directory on both GitHub and GitLab. We utilized the GitHub insights feature to identify the first and last commits, which provides the dates of these commits. However, for Gitter, we had to search on GitLab and manually find these commits. The number of commits was calculated by summing all from the selected repositories. Similarly, the number of contributors was determined, except in GitLab, where we faced a limitation of the latest 6,000 commits in tracking contributors.

The organization aspect refers to the presence of an organization, foundation, or any other entity that supports or leads the projects. The involvement of an organized entity often adds credibility and stability to the project, assuring the users. On the other hand, the support aspect relates to the community's provision of user guides and documentation. These resources serve as valuable references for users with different levels of technical proficiency, aiding them in effectively using the platform. In the support column, the tilde (~) symbol means partial support provided by the community to users. This indicates that some platforms primarily focus on the administrative/development aspects without offering guidance for users in using them.

Provisioning Our primary focus was understanding the challenges in provisioning and configuring the platforms, particularly in containerized environments using Docker and container orchestration tools like Kubernetes or cloud providers. While all platforms offer a public Docker image for containerized solutions, it is worth noting that Zulip's Docker image is still in an experimental phase, and the Zulip installer is considered a more reliable option. Some platforms provide dedicated documentation and tools, such as Kubernetes operators and Helm charts facilitating deployment on Kubernetes, while others require manual configuration from scratch. Similarly, deploying to cloud providers varies among platforms, with some offering specific documentation and guidance, particularly for Amazon Web Services.

4.3. Discussion

The discussion focuses on three main aspects that emerged from the analysis: the open-source nature of the platforms, their stability, and the level and quality of support they offer. These factors are crucial for users and organizations to decide on a collaboration platform that aligns with their needs and requirements.

Open-source nature Customization is crucial in tailoring collaboration platforms to specific needs and integrating them with existing systems. The evaluated platforms are open-source, as indicated by the licenses, which inherently allows users and developers to extend and customize their functionalities, even if no built-in feature is offered to customize the platform, as indicated in Table 2 where different levels of customization capabilities are highlighted. While this study focuses on open-source platforms, it's important to underline that proprietary platforms like Discord or Teams also provide features to build open-source customization. Therefore, users should assess their customization needs and other requirements and evaluate both open-source and non-open-source platforms to choose the one that best aligns with their needs.

Stability The stability of the projects can be assessed by considering several factors based on Tables 2 and 3. The duration of the projects can be inferred from the first and last commit dates in Table 3, revealing a considerable span of commits for all projects, starting from years in the past and continuing until the present year (2023). This longevity indicates a certain level of stability and ongoing development. Additionally, the number of commits and contributors offers further insights because projects with many commits and contributors typically have an active development community putting a noticeable effort into improving and maintaining the platform. While the study included projects with at least 100 stars, most had thousands to tens of thousands of stars, a measure of popularity and stability in the open-source community. As said, an organization backing the projects is another good indicator because it often implies a structured approach, long-term commitment, and resources dedicated to supporting the project. Most platforms have an organization that either leads or funds platform development, as stated in Table 3. The results in Table 2 highlight how platforms have adequate or above provisioning processes, meaning that provisioning them is relatively straightforward and well-documented at least.

Support Since most platforms have their code bases organized into multiple repositories, our analysis focused on a relevant subset of repositories. As a result, the reported number of commits and contributors in Table 3 may be higher than presented, indicating a potentially more substantial activity level. More contributors can be beneficial when seeking support for a platform, especially for projects with fewer contributors than others. However, if your requirements include active and consistent support, it is advisable to consider platforms with at least an adequate community, as indicated in Table 2. Furthermore, Table 3 reveals that not all platforms provide comprehensive support for administrative and usage aspects. Some platforms focus on administrative functionalities while offering limited guidance for regular use. This aspect should be carefully considered based on the specific usage scenarios envisioned for the platform. Therefore, users should be aware of the varying levels of support.

5. Use case

A concrete use case for data-driven collaboration lies in the context of the Horizon Europe project CS-AWARE-NEXT which requires a collaboration platform to provide users with a continuous view of their data, enabling them to effortlessly keep up with discussions, even on complex topics involving significant amounts of data. As a result, we needed to identify the most suitable platform to work as the foundation for a suitable data-driven collaboration platform. To do so, we used this study to choose a platform among the analyzed ones.

CS-AWARE-NEXT is a follow-up to the H2020 project CS-AWARE[6]. CS-AWARE focuses on a novel approach to socio-technical cybersecurity management in organizations. It follows a data-driven approach to create awareness by working with people who can understand how an organization works: the people working in the organization and with organizational IT systems and networks daily. CS-AWARE has developed a novel socio-technical system and dependency analysis methodology based on the tried and proven soft systems methodology (SSM) [44] to generate an organizational knowledge repository for cybersecurity-relevant assets.

In CS-AWARE-NEXT, the focus shifts to awareness and collaboration in inter-organizational settings. The issue of understanding and data availability in individual organizations was addressed through CS-AWARE. However, we have identified a gap in awareness, the ability to collaborate effectively, and ultimately the availability of data and information when dealing with cybersecurity incidents that are not contained within one organization but span across an organization's supply ecosystem. We have identified the need to improve collaboration in this context. To that end, we have found a solution in a data-driven collaboration platform

[6]https://cordis.europa.eu/project/id/740723

that connects individual organizations and utilizes all the available data to foster and facilitate collaboration on cybersecurity.

The role of such a platform is to support cybersecurity collaboration between parties in the ecosystem. Collaboration cannot be a goal as such (similarly: we do not tell people how to read) but serves a particular purpose. In CS-AWARE-NEXT, collaboration strengthens cybersecurity in the ecosystem through increased awareness about the primary needs, issues, and achievable solutions for the ecosystem. Platform support for collaboration involves the agile design of coordinated affordances and data-driven information exchange in collaborative scenarios. Scenarios describe a designed series of actions to achieve some objective in context. Scenarios are abstractions that specify the necessary parameters for successful operation, not the actions themselves. Table 4 briefly describes our data-driven collaboration platform's three main scenario types.

From the description of the scenarios, it can be seen that the foreseen collaborative activities that the platform should be able to support are quite different. For example, exchanging and discussing sensitive threat information is essential in collaborative incident-handling scenarios. In consultancy scenarios, an exchange of documents and an explanation of specific policies and practices is expected. For knowledge management scenarios, access to and adding to repositories seems crucial. Consequently, the platform chosen as the building block for our collaboration platform must be highly customizable and open-source to accommodate these varying requirements. In the worst-case scenario, if the platform's built-in customization capabilities are insufficient, we may need to modify the underlying code. Since CS-AWARE-NEXT is a long-term project spanning multiple years of development, maintenance, and operation, the selected platform must offer exceptional stability and continuous community support, including patches, security fixes, and general improvements. Additionally, we require the platform to be provisioned in containerized environments. Therefore, support for delivering the platform in such environments and guidance throughout the process are critical factors in our decision-making.

Mattermost emerged as an excellent fit as the foundation for our platform, especially considering our requirements for extensive customization and long-term maintenance. Mattermost offers a wide range of powerful customization options that give us the flexibility to customize and extend its features, making it an ideal platform to tailor to our specific needs. It is also open-source, allowing us to work on the source code should we ever need it. One of the crucial advantages of Mattermost is its large and active community. With a large user base and dedicated contributors, we can rely on the community for ongoing support, updates, and continuous improvements to the platform. The community ensures access to the latest developments, bug fixes, and security patches, providing a robust and reliable platform that can be maintained and enhanced over time. Additionally, Mattermost's documentation and resources further contribute to its suitability for our project. The platform provides comprehensive documentation and guidelines for customization, configuration, and provisioning, making it a solid foundation to build and maintain our platform successfully.

Table 4
Scenarios types for the employment of data-driven collaboration in CS-AWARE-NEXT.

Type	Description
Collaborative incident handling	This is a group of scenarios focusing on the collaborative handling of cybersecurity incidents. The collaboration is between a group of organizations (more than 2) that have agreed about a common cybersecurity interest or interdependencies (e.g., a shared service on which they all rely) and who collaborate in understanding incidents for improving their (ecosystem) cybersecurity. The collaboration can be about activities after (at some location) a cybersecurity issue showed up in their systems. The collaboration in this scenario can call for relatively quick decision-making and action and reflections on these actions and decisions afterward. Such types of collaboration improve when participants are well-prepared and backed up by effective policies and task descriptions.
Consultancy	In these scenarios, organizations in the ecosystem have agreed to support each other in realizing improved cybersecurity by contributing to discussions, reviewing, or supporting understanding of cybersecurity in an organization. Topics can be brought up by one of the partners, expecting support from experienced others, or by all, from a longer-term perspective of collaboration and community building. Examples: reviewing data collection and analysis, critical services, vulnerabilities, defining adapted incident handling procedures, implementing and testing developed sharing procedures, risk assessment, business impact analysis, etc. Activity coordination, information sharing, and increased situational awareness will characterize successful consultancy scenarios. Evidence can be found by looking at the number of contributions, the balance of participation (about the same number of contributions per participant), and user satisfaction, as indicated by survey outcomes. Such characteristics can be indicators for monitoring and evaluation.
Knowledge management	Knowledge management is the process of organizing, creating, using, and sharing collective knowledge within an organization. The ultimate goal of knowledge management is for an organization to share its knowledge assets and to be able to retrieve useful information. For CS-AWARE-NEXT, this implies cybersecurity knowledge, particularly knowledge for the benefit of cybersecurity of the ecosystem. In CS-AWARE-NEXT, we focus on situational awareness of local/regional cybersecurity, and knowledge management can be taken as one of the (few) tools to realize sustained awareness. In this scenario, participants share experiences and trajectories and learn how to build on experiences within the context of their trajectories of cybersecurity awareness development.

6. Conclusions and Future Works

In today's modern era, collaboration platforms have become an essential part of our daily lives, changing how we work, connect, and collaborate. With many options available, the abundance of these platforms is a challenge in choosing the most appropriate one to meet specific needs. To make well-informed decisions, it is crucial to consider various factors such as open-source nature, stability, support, customization capabilities, and alignment with requirements. This study aims to help navigate this landscape by offering an overview of open-source collaboration platforms and conducting a comparison based on four key evaluation criteria (architecture, customization, community, and provisioning).

This analysis points out that no perfect platform exists, but users must choose the right one based on their specific requirements. For users seeking customization capabilities, Mattermost and Matrix are the top choices. Mattermost, in particular, stands out for its readily available platform that enables rapid provisioning. On the other hand, Matrix offers the flexibility for users to choose their preferred user interface from options such as Element, Cinny, and Gitter. Mattermost, Rocket.Chat and Zulip have large and active communities that make them recommended for users prioritizing ongoing development and the quick release of fixes or security patches. Rocket.Chat is an excellent choice for users with multiple existing communication channels seeking to consolidate them. For users emphasizing topic-oriented communication, Zulip's enforced logic of assigning topics distinguishes it from other platforms that provide optional topic assignments. Mattermost and Twake provide integrated tools and solutions for effective task management. Cinny, with its simple user interface and focused feature set, suits scenarios where extensive functionalities are not required. However, it's worth noting that Cinny lacks the calls feature, which may be necessary for some use cases. Corteza is a suitable option for users seeking fine-grained access control on discussions. However, if the focus is on finding a platform with a particular emphasis on security, Twake can be a fitting choice, given the community's focus on security. Lastly, for users who require the ability to discover and connect with other communities in a Discord-like fashion, Matrix, Element, Cinny, and Gitter provide suitable options within the Matrix ecosystem. Meanwhile, Revolt, which shares a similar structure to Discord, offers a choice outside the Matrix ecosystem.

A potential extension to this research involves relaxing the boundaries enforced by the search query to encompass a broader range of platforms, thereby enhancing the scope of the study. The current query excludes popular platforms like Discourse[7], but by adopting a less restrictive approach, we could include these platforms to gain further insights.

In conclusion, this study presents a practical use case for a real-world analysis implementation by identifying a platform that meets the requirements of the CS-AWARE-NEXT project. In this context, the objective was to identify the most suitable platform to serve as the foundation for designing and developing a data-driven collaboration platform to improve collaboration in cybersecurity. Given our requirements for extensive customization, open-source nature, long-term maintenance, stability, and continuous support, Mattermost emerged as the platform that best met our needs and ultimately became our final choice.

Acknowledgments

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement No 101069543. This communication reflects only the author's view, and the Commission is not responsible for any use that may be made of the information it contains.

[7]https://www.discourse.org

References

[1] P. Schubert, J. H. Glitsch, Use cases and collaboration scenarios: how employees use socially-enabled enterprise collaboration systems (ecs), International Journal of Information Systems and Project Management (2022). doi:htpps://doi.org/10.12821/ijispm040203.

[2] J. E. Greer, G. Mccalla, J. A. Collins, V. S. Kumar, P. Meagher, J. Vassileva, Supporting Peer Help and Collaboration in Distributed Workplace Environments, International Journal of Artificial Intelligence in Education 9 (1998) 159–177. URL: https://hal.science/hal-00588744, special Issue on Computer Supported Collaborative Learning.

[3] C. Moore, The future of work: What google shows us about the present and future of online collaboration, TechTrends 60 (2016) 233–244. doi:https://doi.org/10.1007/s11528-016-0044-5.

[4] Q.-H. Vuong, N. K. Napier, T. M. Ho, V. H. Nguyen, T.-T. Vuong, H. H. Pham, H. K. T. Nguyen, Effects of work environment and collaboration on research productivity in vietnamese social sciences: evidence from 2008 to 2017 scopus data, Studies in Higher Education 44 (2019) 2132–2147. doi:https://doi.org/10.1080/03075079.2018.1479845.

[5] A. Hemetsberger, C. Reinhardt, Collective development in open-source communities: An activity theoretical perspective on successful online collaboration, Organization Studies 30 (2009) 987–1008. doi:https://doi.org/10.1177/0170840609339241.

[6] J. P. Johnson, Collaboration, peer review and open source software, Information Economics and Policy 18 (2006) 477–497. doi:https://doi.org/10.1016/j.infoecopol.2006.07.001.

[7] J. Teixeira, T. Lin, Collaboration in the open-source arena: The webkit case, in: Proceedings of the 52nd ACM Conference on Computers and People Research, Association for Computing Machinery, 2014, p. 121–129. doi:https://doi.org/10.1145/2599990.2600009.

[8] P. Schubert, S. P. Williams, Enterprise collaboration platforms: An empirical study of technology support for collaborative work, Procedia Computer Science 196 (2022) 305–313. doi:https://doi.org/10.1016/j.procs.2021.12.018, international Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies 2021.

[9] C. Burkert, H. Federrath, Towards minimising timestamp usage in application software: A case study of the mattermost application, in: Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2019 International Workshops, DPM 2019 and CBT 2019, Luxembourg, September 26–27, 2019, Proceedings, Springer-Verlag, 2019, p. 138–155. doi:https://doi.org/10.1007/978-3-030-31500-9_9.

[10] A. Ćirković, Z. Bogdanović, B. Radenković, Project-based learning with mattermost in higher education, E-business technologies conference proceedings 3 (2023) 260–264. URL: https://ebt.rs/journals/index.php/conf-proc/article/view/158.

[11] R. Barde, A. A. Raouf, Evaluating and improving accessibility of web applications: Zulip chat case study, in: 2021 17th International Computer Engineering Conference (ICENCO), 2021, pp. 112–117. doi:https://doi.org/10.1109/ICENCO49852.2021.9698944.

[12] C. Ihle, F. Deifuß, M. Schubotz, B. Gipp, Towards portable identities in the matrix protocol,

in: 2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW), 2022, pp. 88–89. doi:https://doi.org/10.1109/ICDCSW56584.2022.00025.

[13] R. Long, N. Martin, A. Bura, Third room: Decentralized virtual worlds on matrix, in: Proceedings of the SIGGRAPH Asia 2022 Real-Time Live!, SA '22, 2023. doi:https://doi.org/10.1145/3550453.3586015.

[14] K. Ermoshina, F. Musiani, H. Halpin, End-to-end encrypted messaging protocols: An overview, in: Internet Science, Springer International Publishing, 2016. doi:https://doi.org/10.1007/978-3-319-45982-0_22.

[15] G. C. Schipper, R. Seelt, N.-A. Le-Khac, Forensic analysis of matrix protocol and riot.im application, Forensic Science International: Digital Investigation 36 (2021) 301118. doi:https://doi.org/10.1016/j.fsidi.2021.301118.

[16] mattermostGitHub, Mattermost, 2015. Https://github.com/mattermost.

[17] mattermostSite, Mattermost | secure collaboration for technical teams, 2015. Https://mattermost.com.

[18] matrixGitHub, matrix.org, 2014. Https://github.com/matrix-org.

[19] matrixSite, Matrix.org, 2014. Https://matrix.org.

[20] elementGitHub, Element, 2015. Https://github.com/vector-im.

[21] elementSite, Element | secure collaboration and messaging, 2016. Https://element.io.

[22] cinnyGitHub, Cinny, 2021. Https://github.com/cinnyapp.

[23] cinnySite, Cinny | site, 2021. Https://cinny.in.

[24] gitterGitLab, gitter · gitlab, 2012. Https://gitlab.com/gitterHQ.

[25] gitterSite, Gitter — where developers come to talk., 2015. Https://gitter.im.

[26] zulipGitHub, Zulip, 2012. Https://github.com/zulip.

[27] zulipSite, Zulip: Open-source team chat with topic-based threading, 2012. Https://zulip.com.

[28] twakeGitHub, Linagora, 2020. Https://github.com/linagora.

[29] twakeSite, The open digital workplace | twake, 2020. Https://twake.app.

[30] rocketChatGitHub, Rocket.chat, 2015. Https://github.com/RocketChat.

[31] rocketChatSite, Rocket.chat: Communications platform you can fully trust, 2016. Https://www.rocket.chat.

[32] cortezaGitHub, Corteza project, 2018. Https://github.com/cortezaproject.

[33] cortezaSite, Corteza - corteza, 2019. Https://cortezaproject.org.

[34] revoltGitHub, Revolt, 2020. Https://github.com/revoltchat.

[35] revoltSite, Revolt - find your community, 2021. Https://revolt.chat.

[36] mattermostDocumentation, Mattermost documentation, 2015. Https://docs.mattermost.com.

[37] matrixDocumentation, Matrix.org - matrix for instant messaging, 2014. Https://matrix.org/-docs/chat_basics/matrix-for-im.

[38] elementDocumentation, User guide | get started in element, 2016. Https://element.io/user-guide.

[39] twakeDocumentation, Twake - twake, 2020. Https://doc.twake.app.

[40] zulipDocumentation, Overview — zulip 8.0-dev+git documentation, 2012. Https://zulip.readthedocs.io/en/latest/overview.

[41] rocketChatDocumentation, Rocket.chat - rocket.chat docs, 2016. Https://docs.rocket.chat.

[42] cortezaDocumentation, Corteza :: Corteza docs, 2019. Https://docs.cortezaproject.org/corteza-docs/2023.3.

[43] revoltDocumentation, Introduction | revolt, 2021. Https://developers.revolt.chat.

[44] V. Kupfersberger, T. Schaberreiter, C. Wills, G. Quirchmayr, J. Röning, Applying soft systems methodology to complex problem situations in critical infrastructures: The cs-aware case study, International Journal on Advances in Security 11 (2018) 191–200. URL: http://eprints.cs.univie.ac.at/5904/.