PodMerge: Democratizing Real-Time Collaboration in the Solid Ecosystem

Jonathan Gruss
jonathangruss@gmail.com
University of St.Gallen
St.Gallen, Switzerland

Andrei Ciortea*
University of St.Gallen
St.Gallen, Switzerland

Guido Salvaneschi†
University of St.Gallen
St.Gallen, Switzerland

Simon Mayer‡
University of St.Gallen
St.Gallen, Switzerland

ABSTRACT

Real-time collaboration has become commonplace in centralized Web applications, but decentralized Linked Data systems still lack readily accessible mechanisms. In this demo paper, we propose a novel approach that provides a viable solution to implement collaborative Linked Data in the Solid ecosystem using Conflict-free Replicated Data Types (CRDTs) and hypermedia-driven interaction: Specifically, we introduce a dedicated vocabulary for describing interactions with CRDT-based resources hosted in Solid Pods, empowering software clients to dynamically discover means for collaborative editing at run time. We demonstrate the practicality of our approach, we showcase a Solid-hosted website that utilizes the vocabulary to expose hypermedia controls, and a Chrome extension that effectively consumes these descriptions to enable real-time collaborative editing through CRDTs. By strategically shifting intelligence to the client side, our approach significantly reduces the entry barrier for publishing real-time collaborative resources on the (Semantic) Web.

KEYWORDS

Real-Time Collaboration, Linked Data, CRDT, Solid, Ontology, RDF

ACM Reference Format:
Jonathan Gruss, Andrei Ciortea, Guido Salvaneschi, and Simon Mayer. 2023. PodMerge: Democratizing Real-Time Collaboration in the Solid Ecosystem. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Real-time collaboration on the web represents a transformative shift in the way people work together, communicate, and share information in the digital age. Google Docs was one of the main drivers of this, revolutionizing real-time document collaboration. It allowed multiple users to simultaneously edit and comment on documents in a web browser, eliminating the need for email attachments and version control issues. A wide variety of web collaboration tools followed, such as Trello for task management, Miro for virtual whiteboards, or Overleaf for Latex document editing.

Although these tools provide valuable collaboration platforms, they often rely on server-centric architectures and siloed data that only lives within the platforms ecosystem. As a result, these systems can restrict access, hinder data interoperability, or even require users to give up control over their data. There is a growing need for open and decentralized real-time collaboration tools where the user is back in control over his data and where data can be discovered in a human and machine-readable way. A notable project in this area is the Solid protocol [1], a web decentralization project promoted by Sir Tim Berners-Lee [36]. Solid aims to give web users control over their personal data by enabling them to store and manage their data in social linked data personal online datastores (Solid Pods) hosted anywhere on the web, while maintaining data interoperability and privacy.

Still, while decentralized Linked Data systems like Solid have tremendous potential, they currently lack comprehensive solutions for real-time collaboration. Our proposal, *PodMerge*, aims to fill this gap by simplifying the development of real-time collaborative web content within the Solid framework.

1.1 PodMerge: Bridging the Real-time Gap

The majority of the modern co-editors, such as Google Docs or Overleaf, enable collaboration on Web resources in real time through Operational Transformation (OT) — a technique for transforming and applying concurrent operations on shared data without conflicts [40]. While distributed OT solutions theoretically exist, current co-editors often use architecture with centralized servers [**?**]. An emerging alternative are Conflict-Free Replicated Data Types (CRDTs), allowing seamless, uncoordinated concurrent updates to shared data [38]. The increasing adoption of CRDTs in co-editing systems is largely influenced by the principles of local-first software [21]. The principles such as offline operation and data privacy enabled privacy-centric, local-first applications like AFFiNE [2] and Anytype [3], platforms that prioritize decentralization and minimize

*Supervisor
†Co-Supervisor
‡Advisor

[1] https://solidproject.org/
[2] https://affine.pro/
[3] https://anytype.io/

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

their reliance on central servers. Such foundational principles also resonate with and inspired our PodMerge approach.

In contrast to local-first software, PodMerge is based on web resources where discoverability is crucial for effective collaboration. In many co-editing platforms, documents remain siloed within their specific ecosystem, which hinders interoperability between various systems. Take for instance, Google Docs; it prevents users from seamlessly collaborating in real-time with alternate applications like Microsoft Word because of its ecosystem restrictions. In the PodMerge approach, we harness the power of the Semantic Web, integrating with the Resource Description Framework (RDF) to guarantee that documents are both accessible and interoperable across different platforms. By leveraging Solid, a Linked Data system, we can utilize Solid Pods as repositories for these collaborative web resources. These resources, represented in RDF formats and enriched with semantic annotations, improve machine readability and discoverability while having user controlled access permissions from Solid. As a result, we promote collaboration on content not just within a specific ecosystem, but across the (Semantic) Web, while providing a balance between openness and security.

Moreover, we introduce the Collaborative Resource Description Ontology (CRDO), an ontology for the PodMerge approach to describe interactions with collaborative web resources in Solid Pods. CRDO offers a vocabulary for hypermedia controls similar to ontologies like Hydra [24] and HCTL [4], but specifically for collaborative web resources. Using the CRDO to describe a collaborative web resource in a Solid Pod allows software clients to discover at runtime the required means for collaboratively editing it. In contrast to relying on central coordination servers, our approach shifts the intelligence to the client — and lowers the entry barrier for making real-time collaborative Web resources.

In essence, PodMerge combines the decentralized storage and access control capabilities of Solid, the dynamic real-time collaboration of CRDTs, and the expansive discoverability of Linked Data Principles into a comprehensive system for seamless web collaboration.

1.2 Showcasing PodMerge in Action

To demonstrate our approach, we showcase the addition of real-time co-editing on a static website using the PodMerge approach. The website is hosted in a Solid Pod that uses the CRDO to expose a description of the web resource with hypermedia controls that acts as a collaboration interface. Additionally, we enable smart clients through the implementation of a browser extension for real-time collaboration based on CRDTs. This browser extension consumes the CRDO descriptions and transforms the websites content into collaboratively editable elements. Moreover, it integrates a Linked Data editor into the website that provides a user interface to enrich the web resource with semantic annotations.

To provide a clear path for the PodMerge approach we first introduce our main design goals. The PodMerge approach combines concepts from multiple research and industry fields. Hence, we continue by exploring the underlying technologies and related works. Moving forward, we introduce the PodMerge approach and its architecture with the specific building blocks that are required. Then, we look at the implementation of the various building blocks, including the collaboration technique, the CRDO vocabulary, and the Solid integration. In the demonstration, we showcase how the PodMerge approach can simplify the addition of real-time co-editing on a static website. Finally, we conclude with an overview of the approach by highlighting potential advancements and areas of further research.

[4] https://www.w3.org/2019/wot/hypermedia#

2 DESIGN GOALS OF PODMERGE

To ensure the effective implementation and wide adoption of the PodMerge approach, we set clear design goals. Below, we detail the core design goals that underpin PodMerge:

(1) **Compliant with the Solid Specification:** PodMerge should adhere to the Solid specification and use Solid Pods as an online repository. This compliance ensures privacy, interoperability, and user control, and makes the approach more robust and reliable.
(2) **Heterogeneous Client Services:** At the heart of the PodMerge approach is its adaptability across different client services. Users should be able to interact with various client services, including Progressive Web Apps (PWAs), native apps, and browser extensions, all within the same co-editing session. This guarantees seamless collaboration regardless of the specific client platform chosen by individual users.
(3) **Semantic Web Integration:** An integral part of PodMerge is its alignment with the Semantic Web principles. By relying on RDF formats combined with discoverable and linked data, PodMerge ensures that data is not only stored but is also meaningful and interconnected in the broader web landscape.
(4) **Simplified Creation of Collaborative Resources:** PodMerge should enable users to easily convert existing web resources into collaborative spaces or create new ones with minimal effort. The creation or transformation process should be simplified with a straightforward description of the collaborative resource that is not dependent on complex configurations or services.
(5) **Generality of Supporting Techniques:** The approach should be able to be used in combination with a wide range of co-editing techniques. This means that the users are not tied to a specific co-editing framework or type but can choose one that applies best to their specific use case.
(6) **Built on Established Open Source Technologies:** PodMerge should mainly be built on popular and available open source co-editing technologies. By integrating with and extending existing, proven technologies, the approach can benefit from established ecosystems and avoid reinventing the wheel. Such a foundation also aids in user trust and lowers the entry barrier, as they are likely familiar with the technologies in use.
(7) **User Data Sovereignty:** A fundamental design goal is to ensure that users retain full control and ownership of their data. The approach should never compromise on this principle, ensuring that users can decide where their data is stored, who has access, and for what purpose.

3 BACKGROUND AND RELATED WORK

The Web has undergone a significant transformation, beginning as a platform primarily for accessing and viewing content to becoming a dynamic space for collaboration and interaction. In recent years, collaborative web resources have emerged as an essential component of this interactive Web, providing platforms for multiple users to simultaneously contribute and edit content in real-time. While these platforms offer a great experience for their specific use cases, they still come with limitations. Many of them operate as closed ecosystems, leading to data silos. Users often give up control over their data, and there is a noticeable gap in integration with the Semantic Web. There exists a growing need for an open approach that offers seamless collaboration on user-controlled, semantically enriched web resources. We aim to fill this gap with out PodMerge approach.

This chapter delves into the foundational concepts and state-of-the-art technologies that influence and guide our research, highlighting the present constraints. We start with an exploration of the historical advances in collaboration on web resources and transition into modern real-time co-editing methodologies. Our analysis encompasses popular applications, focusing on the OT and CRDT collaboration techniques. We further discuss semantic annotations and delve into the contemporary methodologies of collaborative linked data. Finally, we conclude with a comprehensive overview of hypermedia controls vocabulary and its applications and significance for the PodMerge approach.

3.1 Beginnings of Web Collaboration

Since its inception, the Web has been widely perceived as a space for reading rather than writing. However, the landscape began to change with the introduction of WebDAV (Web Distributed Authoring and Versioning). WebDAV laid the groundwork for users to collaboratively edit and author content directly onto an HTTP web server by enhancing the capabilities of the Hypertext Transfer Protocol (HTTP). With facilities for concurrency control and namespace operations, WebDAV revolutionized the way we perceive the Web, converting it from a passive platform for consumption into an active space for creation and collaboration [47].

This transformation aligned with Tim Berners-Lee's vision of a read-write Web, where content creation and collaboration are as fundamental as content consumption. In the read-write Web, users are not merely passive consumers, but active participants, who contribute to the vast repository of information and shape the digital landscape [6, 8]. PodMerge draws inspiration from this vision by emphasizing on user agency and focusing on seamless collaboration in the Semantic Web. The approach aims to bridge the gaps in current collaborative web resources and realize the full potential of a truly interactive, democratic Web.

3.2 Modern Real-time Co-Editors

Nowadays, real-time collaborative editing platforms with Google Docs as a major pioneer, have transformed the way people collaborate online. Central to Google Docs' functionality is the Operational Transformation (OT) technique. OT offers robust methods to transform and apply concurrent operations on shared data that ensure that no conflicts arise [40]. While decentralized OT systems exist [34, 39, 41, 48], most current frameworks and co-editors using OT are centralized [42].

Centralized OT solutions have distinct drawbacks when considering the design goals of the PodMerge approach. Firstly, PodMerge aims to follow a local-first approach where collaboration is still possible when offline; with OT, achieving this is still problematic because it typically requires constant communication with a central server to resolve conflicts and maintain document consistency. By using such a central server, end-users are also required to give up control over their data and rely on the availability of the server, introducing a single point of failure. Moreover, in the PodMerge approach we leverage static Solid pods as the repository for collaborative web resources. Using centralized OT systems, we would be dependent on complex server services and setups while by using a distributed approach we could shift the intelligence to the client. Thus, lowering the entry barrier for creating collaborative web resources.

An example of a centralized OT based system that underlines these issues is WikiDocs [12]. Similarly to our demonstrative use case, WikiDocs enabled real-time collaborative editing of websites and specific resources in websites by integrating its library on the client side. Although it offered simple integration and a user-friendly collaboration experience, WikiDocs was centralized and relied on proprietary servers. Consequently, when WikiDocs ceased operations after its acquisition [25], all applications that used its collaboration features were left unsupported. This emphasizes the need for an approach that is open source and can run on distributed client services without a central coordinator.

In recent years, CRDTs have gained traction as an alternative technology to OT [38]. Unlike OT, CRDTs allow replicas to update independently without immediate coordination and merge later, while ensuring that no conflicts arise. Automerge and Yjs are noteworthy frameworks that employ CRDTs in co-editors [20, 30]. Although OT requires central authority for conflict resolution, CRDTs are distributed by design and ensure conflict-free merges. This distributed nature of CRDTs align better with the design goals of the PodMerge approach. Which is further emphasized in Section 4.2 where we compare the characteristics of popular OT and CRDT frameworks.

The principles of local first software (LFS) are a major driver of this paradigm shift [21]. LFS promotes a client-centric model, emphasizing privacy, data ownership, control, performance, and longevity. LFS uses CRDTs for consistency and employs peer-to-peer mechanisms for online data synchronization. However, while LFS prioritizes local data access, web data accessibility and discoverability can be compromised. For instance, accessing offline user data remains a challenge. PodMerge addresses this by enabling data accessibility and discoverability regardless of the application or library.

Jupyter Notebook provides open standards for interactive computing based on an Interactive Computing Protocol, Notebook Document Format, and Kernel interfaces that exemplify this. Its development environment recently transformed its co-editing library from a OT-based approach to using CRDTs [17, 27]. This allows the creation of web based and LFS applications for processing and co-editing that integrate seamlessly with one another through the open

standards. Drawing inspiration, PodMerge employs linked data descriptions for collaboration on semantic web resources, enriched with hypermedia. Unlike Jupyter's fixed interfaces, our approach is more adaptable, catering to varying CRDT frameworks and types of web resources.

3.3 Semantic Annotations and LOD Integration

The advent of Web 2.0 was marked by the transition from static web pages to dynamic, interactive platforms. The crux of this transformation lies in rendering web resources comprehensible to machines. To achieve this, semantic annotations, commonly known as linked data tags, have been introduced. Semantic annotations enable machines to interpret the meaning and context of web resources, enriching user experience and fostering data interoperability [19, 31]. The integration of semantic annotations into the PodMerge system will increase the discoverability and contextual understanding of shared resources. Furthermore, in the demonstrative use case we showcase a streamlined process for adding semantic annotations to web resources, drawing parallels with existing works [4, 13, 22, 33]. Additionally, we introduce real-time co-editing capabilities complemented by a user-friendly interface.

The Linked Open Data (LOD) initiative further highlights the significance of semantic annotations by promoting the use, sharing, and interlinking of open data sets on the web [5]. LOD extends the capabilities of web resources by connecting them to a large interconnected semantic graph. This does not only add semantic enrichment to the data but also increases its discoverability and interoperability [9, 37]. The Solid protocol, introduced by Sir Tim Berners-Lee, aligns with and enhances the objectives of the LOD initiative. It offers users control over their own data that lives within the semantic graph and promotes data decentralization, interoperability, and a user-centric web experience [26, 36]. PodMerge integrates with the Solid protocol to offer a novel approach that adds semantic annotations to collaborative web resources and thus enriches the LOD graph with real-time collaborative data.

3.4 Fine-grained Collaboration on Linked Data

Real-time collaboration on Linked Data has seen the use of CRDTs to ensure strong eventual consistency for RDF graphs in distributed systems [3, 14, 15, 28, 49]. These techniques focus predominantly on the formulation of shared RDF data types. However, aspects like the discoverability of a collaboration interface, seamless integration with widely-used data formats, and the governance and access control over linked data are often omitted.

Delving deeper, Ibáñez et al. have introduced a novel method named Col-Graph [15], that combines collaborative editing with scalable query processing. This allows multiple users to edit the Linked Open Data (LOD) cloud simultaneously, maintaining data integrity and ensuring it remains queryable. Drawing parallels with PodMerge, Col-Graph also employs a replication and conflict resolution strategy, supporting local and offline modifications. A stark contrast, however, is the granularity of edits. Col-Graph uses RDF-triples as the smallest directly manageable piece of knowledge while our approach allows for diverse CRDT implementations and types. This makes it possible to have more fine-grained collaboration, such as single-character text updates on an RDF node. Additionally, the paper does not thoroughly consider access control and storage challenges that may arise in a writable and collaborative LOD cloud environment, which we address with the Solid Protocol integration.

Another notable platform is LinkZoo [29]. LinkZoo is a Linked Data platform designed to facilitate collaborative management of heterogeneous resources, from documents to multimedia, while ensuring data consistency and interlinking. Similar to the demonstrative use-case of PodMerge, it offers tools for resource creation, editing, and semantic annotation. Yet, it falls short when it comes to real-time co-editing capabilities, including conflict resolution and offline synchronization. Additionally, LinkZoo is its own encapsulated platform which complicates an integration with other systems such as the Solid Protocol

To summarize, while progress has been made in real-time collaboration within linked data systems, a comprehensive solution is still missing. Existing implementations often regard triples, or individual RDF nodes like the subject, predicate, or object as the smallest possible item that can be updated. This limits concurrent modifications to finder scopes. Additionally, there is currently no real-time collaborative solution for Solid pods to offer a user-controlled co-editing experience in the Semantic Web. With PodMerge we seek to bridge these gaps, prioritizing diverse design goals to enable seamless co-editing of RDF data in the Solid ecosystem.

3.5 From Fixed Contracts to Dynamic Discoverability with Hypermedia

Traditional co-editing systems primarily rely on fixed contracts — a set of predefined interactions and operations that clients are bound to. These contracts, although effective in controlled environments, pose limitations in flexible and dynamic collaborative contexts. Fixed contract are unable to adapt or evolve without modifying every participating client, which limits scalability and interoperability. Moreover, any changes to the contracts require updates across all integrated systems, leading to maintenance overheads and reduced agility.

Hypermedia-driven APIs and ontologies offer a compelling alternative. Rather than adhering to inflexible contracts, they enable dynamic discoverability of interactions and operations at runtime. The essence of the hypermedia ontology lies in embedding rich metadata in Web resources to provide the dual benefit of cataloged, structural, and semantic enrichment while promoting dynamic discoverability [11]. Hydra illustrates this by developing an ontology that provides hypermedia controls tailored to APIs that provide both read and write capabilities and are customizable at runtime [24]. However, the design is focused on generic web operations and does not address the requirements of specialized frameworks such as CRDTs.

The Hypermedia Controls Ontology (HCTL) [5] is a light-weight ontology that provides a foundational layer for hypermedia controls. This includes the core concepts of links and forms that can be referred to as hypermedia controls. HCTL attemps to standardize the hypermedia controls for the Thing Description (TD) model and the Constrained RESTful Application Language (CoRAL). Hence, HCTL offers a more generic approach in comparison to Hydra, which is tailored to Web APIs. Due to its generic approach and

[5] https://www.w3.org/2019/wot/hypermedia#

popularity in other systems, the *form* concept in HCTL can be utilized as a versatile foundation for operation descriptions. Hence, we decided to integrate HCTL into PodMerge's CRDO ontology to create more specific operation descriptions for collaborative web resources.

In conclusion, by embracing hypermedia-driven ontologies like HCTL, PodMerge can transition from fixed contract to dynamic hypermedia-driven operations that allow the establishing of web collaboration at runtime.

4 CONCEPTUAL FRAMEWORK

The PodMerge approach is inspired by the principles of local-first software [21] to enable user-controlled and privacy-preserving collaboration using CRDTs and intelligent clients. Additionally, we use the Solid protocol to provide decentralized and portable data hosting with access control, and introduce a vocabulary that complements the Hypermedia Controls Ontology (HCTL) [6] to create hypermedia controls for real-time collaboration operations required for CRDT-based resources. In this chapter, we start in 4.1 by giving an introduction to the architecture of PodMerge. Specifically, we look at the different components, how these components interact with one another, and variability and extensibility of these components and the whole architecture. In 4.2, we will explain the different components of the system in more detail. Starting with requirements of the distributed concurrency control using CRDTs, the use of popular CRDT libraries in an RDF compatible format, and the proposition and methodology of using a textual representation of the CRDT document state. Next, in **??** we introduce a new ontology that defines hypermedia controls for collaborative Web resources. We explain the classes, properties and rules of the ontology and how it complements with HCTL. Finally, in **??** we delve deeper into the integration with the Solid specification that allows users to preserve data control and privacy in the Web and provide fine grained access to the collaborative resources.

4.1 Architecture

One of the main ideas of the PodMerge architecture is that it should not be dependant on centralized data in a cloud service where the collaborators have little control over their data. We enable this by shifting intelligence from the servers back to the client and by connecting user-owned Solid Pods that host a static version of the data in RDF to provide discoverability, accessability, and collaboration operations. Our goal of the architecture is to enable collaboration with a co-editor like experience for Web resources that is open, interoperable with existing systems and technologies, and offers high modularity and extensibility.

Figure 1 outlines a possible architecture of PodMerge, the exact architecture and the components involved can variate and be updated according to the specific use case. The real-time co-editing of the resources is done client side, on the collaborators local device, and updates are propagated to other collaborators directly with the use of peer-to-peer technology. The Solid Pod will act as the Web based hub to store relevant resources similar to what GitHub or GitLab provide for the Git version control. There are three main components that are hosted within the Solid Pod; a

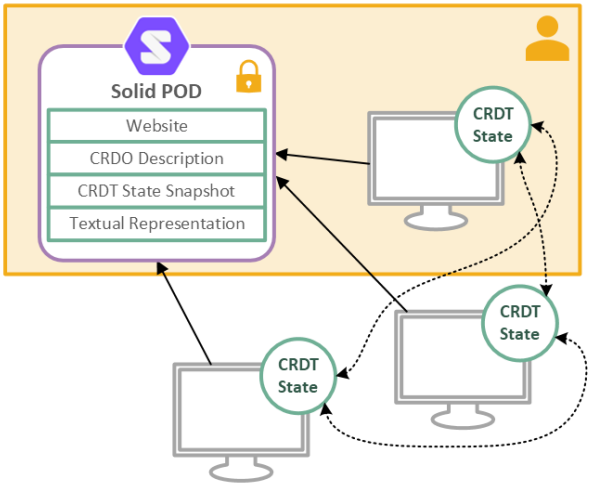

Figure 1: Architecture overview ...

CRDO description of the collaborative resource, the last pushed state of the library specific CRDT state, and a textual representation of that library specific CRDT state. The CRDO description uses our proposed CRDO ontology that is described in more detail in 4.3 to provide hypermedia controls in the form of operations that are necessary to collaborate on the described Web resource. Additionally, it adds linked data metadata to define requirements for collaborating on the resource, such as the CRDT library and other dependencies, as well as enhancing the resources discoverability and interoperability in the Web. Taking the analogy from GitHub, the CRDT document containing the last pushed library-specific CRDT state in PodMerge represents the codebase in GitHub. The state stores the document with the last significant updates that were ready to publish and its entire change history. However, as we will discuss in 4.2, the library specific CRDT state is typically binary data. Meaning, it is not human-readable, which limits its accessibility and makes it difficult to integrate with other systems. Hence, we also store the textual representation of that state in the Solid Pod. It is a snapshot of the latest updated document state and does not store any change history. Additionally, there can be more resources that are specific to the use case, such as a static website, which we make collaboratively editable and provide an editor to add linked data annotations in our demonstrator.

In contrast to other real-time collaborative systems, PodMerge utilizes smart clients that enable us to reduce the server complexity. This is required to make personal data in Solid Pods collaborative because the Solid Pods primarily act as the storage and authentication provider of the personal data. The intelligence in the Solid system is provided by apps that consume the data from a users Solid Pod. Therefore, a user would have previously been required to build his own application to enable collaboration on his Web resource. However, with the PodMerge approach, the user only has to provide a CRDO description and use an intelligent client or application that can read this description and contains all its requirements. These intelligent clients and applications will be further explained in the following chapter.

[6] https://www.w3.org/2019/wot/hypermedia#

4.1.1 Intelligent Clients and Applications. Intelligent clients are applications capable of executing complex tasks, processing data, and rendering user interfaces independently, minimizing the reliance on server-side logic. PodMerge works best when intelligent clients and applications are based on the principles of local-first software by Kleppmann. One key aspect is that the main truth of data for the clients is on the local machine. For that, we use CRDTs to enable seamless real-time collaboration in such a distributed system. The intelligent clients and applications maintain a local copy of the CRDT document and use this as the directly manageable piece of data. Updates from and to other peers are handled concurrently in the background. Examples of intelligent clients are applications that can be used within the PodMerge approach: progressive web applications (PWAs), browser extensions, and desktop or mobile applications. We provide a set of requirements to build such intelligent clients and applications. The requirements are as follows:

- They must be based on a CRDT library or implementation that is open source and thus can be used by other application.
- They must be able to read and interpret the CRDO Description and its Web and CRDT library specific operations.
- They must comply with the requirements of the Solid Project Specification [7] such as WebID-OIDC authentication, explicit user consent, or rules for CRUD operations on resources.
- They must be able to run on the local machine without an essential server dependency. [OPTIONAL]
- They must be able to communicate with other applications using open and standardized communication protocols and preferably peer-to-peer technology.

These requirements are provided by the PodMerge approach to offer an open, interoperable, and modular ecosystem for seamless real-time collaboration. However, the requirements are currently loosely defined and it is possible to create intelligent clients and applications that work within the ecosystem but do not fully adhere to them.

An important aspect of the PodMerge approach is that the users have the choice of which intelligent client or applications to use when collaborating. Moreover, collaborators can use different ones and switch between them even when collaborating on the same resource. Hence, we provide interoperability between different collaborative real-time applications and tools. This allows you to build collaborative editing applications with a standardized collaboration interface and linked open data that are owned and controlled by its user. A constraint is that this is limited to the CRDT implementation used, where it is currently not possible to provide interoperability between applications that are not based on the same CRDT implementations. A visualization of clients that use different applications and intelligent clients, such as a browser extensions, can be seen in Figure **??**. Additionally, the figure shows the interactions of the client applications with each other in a peer-to-peer network and with the Solid Pod in a RESTful interface. The figure illustrates that the client on top is also the owner of the Solid Pod; hence, he can provide access to other clients who want to collaborate on one of his resources.

[7] https://solidproject.org/TR/

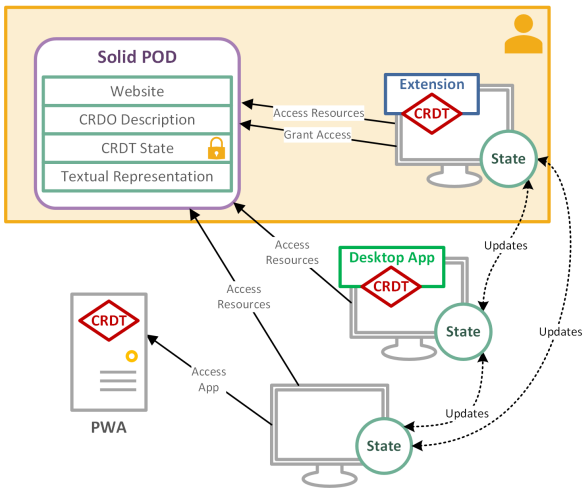

Figure 2: Interoperability of intelligent clients and applications: Client 1 is using a browser extension to collaborate, client 2 uses a desktop application, and client 3 accesses an external PWA which he can then use locally. All these applications use the same CRDT library which is visualised using the same diamond shape.

4.1.2 CRDT library. CRDTs are our main building block for real-time collaboration within intelligent clients. They allow multiple clients to concurrently modify shared data without conflicts, ensuring eventual consistency across all replicas. There are many existing implementations for CRDTs to provide shared editing for various data types and use cases. The PodMerge approach aims to offer an interface that can be used by different CRDT libraries. Thus, giving the user a choice of the CRDT library that best fits his use-case. This allows the use of established and familiar technologies for collaboration, reducing the complexity and dependency of adopting the PodMerge approach. As mentioned in the previous section, it is, however, not possible to directly collaborate on a shared resource with peers using different CRDT libraries.

4.1.3 Integration into the Linked Data Web.

4.1.4 Consuming a CRDO Description. The CRDO description acts as the collaboration interface of the resource to allow clients to infer the requirements and operations to participate in the collaborative peer editing. In the following, we illustrate the steps involved in PodMerge to collaborate on a Web resource that has an attached CRDO description:

(1) Discovery of a CRDO description through a link in a Web resource or a previously known URL.
(2) Reading of the CRDO description to determine the software requirements for the client to collaborate on the resource. If these requirements are met, continue.
(3) Reading the CRDO description to discover operations to get the recent document state of the collaborative Web resource. The software will create a copy of the Web resource on the local system.

(4) Establish a peer-to-peer connection to other collaborators to automatically receive and send the latest updates.
(5) Update of the Web resource on the local copy. These updates will propagate directly with peers.
(6) Push a new state of the resources to the Solid Pod using the discovered operations. Pushing the state includes updating the library-specific CRDT state, the textual representation of that state, and the specific metadata on the CRDO description such as contributors and latest change date.
(7) Optionally, as the owner of the Solid Pod or if you have been given admin privileges, you can invite other users to collaborate by giving read and write access to their WebId in the resource container.

4.2 Collaborative Co-editing in Linked Data Systems using CRDTs

In this section, we explore the use of CRDTs for real-time collaboration in an open and Semantic Web. First, we explain our reasoning for mainly supporting CRDT algorithms to provide concurrent operations on shared data. We briefly compare the two predominant technologies, OT and CRDTs, especially for their use in a Linked Data system such as the Solid ecosystem. Next, we look at the integration of CRDT implementations and libraries into the PodMerge approach and define a set of its current requirements and constraints. One significant aspect here is the representation of the CRDT document in an RDF compatible format. This is required to bring a co-editor editor collaboration experience to the open Linked Data Web. Finally, we further expand on the necessity of a textual representation of the CRDT document state that is available in the Solid Pod. We explain the trade-offs of this approach and its use as a core piece of the PodMerge approach.

4.2.1 Comparing OT and CRDT libraries for co-editing. One design goal of the PodMerge approach is to build on existing and popular technologies in the Web. There already exist a large variety of OT and CRDT libraries with JavaScript implementations for simple integration of co-editing capabilities into a web application. Thus, in contrast to most other methods for collaborative Linked Data, we are not proposing a new CRDT or OT algorithm but aim to integrate with many popular libraries. This gives the user more flexibility to choose the right algorithm for his use case that has been battle tested and provides a good documentation. Hence, we compared existing OT and CRDT libraries and developed criteria that make it possible to use such libraries in the PodMerge approach.

The first criteria in the selection of the libraries was, that they are all open source to give the community access to the codebase in case the library is discontinued or paylocked.

The next criterion is the support of a JSON document format and rich text types. The combination of the JSON document structure with rich text capabilities is especially interesting for our PodMerge approach in a Linked Data system. Using RDF to represent interconnected data in the Web, we can build RDF documents using JSON-LD and provide rich text editing on the specific nodes using the same CRDT document. More on this will be provided in the next section. In Table 1 CRDTs generally adopt the JSON format for its document structure more widely and there are more CRDT libraries that offer a combination of rich text and JSON.

While there are OT algorithms for distributed systems [], we have not found industry standard libraries with JavaScript implementations of those. All the analyzed OT libraries use a server-centric approach. Being dependant on a server either introduces complexity if you have to self-host it, or you introduce dependencies and give up full control over your data. CRDTs on the other hand allow one to represent and manage collaborative resources without relying on a central coordinater. This makes them well-suited for decentralized Linked Data systems — since we do not require an expensive origin server to integrate the changes. [].

The last two criterion are the developer friendliness and the maintenance of the libraries. With developer friendliness we mean that the library has good documentation and is relatively simple to integrate in a Web application. In maintenance we looked for activity in the repository or community in the last three months. Generally, we saw much more activity from the communities of the CRDT frameworks. Within these communities there is more active development and discussions. In addition, new libraries and frameworks generally utilize CRDT algorithms or build on existing CRDT libraries.

Two common mentioned drawbacks of CRDT based approaches over OT are a large per-character memory overhead and a continuously growing document size. In recent benchmarks conducted on various CRDT implementations using real-world text-editing scenarios, it was found that modern frameworks have substantially mitigated traditional CRDT drawbacks. In particular, showing a negligible memory usage even for very large documents [16, 46], effectively countering claims of high per-character memory overhead [?]. Furthermore, save sizes with CRDT metadata were only 60% of the actual text size, and even decreased when the text was deleted, challenging concerns of the ever-growing document size [46]. Moreover, delta-state CRDTs [1, 2] have started to see wide adoption [10, 35, 43], reducing the message size in CRDTs significantly for state updates. Such findings underscore the efficacy and practicality of contemporary CRDTs for collaborative editing.

To summarize the finding from the comparison, CRDTs generally fulfill more of the criterion, with the two CRDT libraries Yjs and Automerge being the only ones that fulfill all of them. Both these libraries build on a JSON compatible document structure and also offer CRDTs in other data types such as rich text. Additionally, both have active communities, are well maintained, and actively used in production applications. Hence, we decided that for the beginning we will mainly support CRDT libraries. The inherent characteristics of CRDTs, such as decentralization, simplicity, robustness, and eventual consistency, align perfectly with the objectives and principles of PodMerge.

4.2.2 CRDTs for structured RDF data. Real-time co-editing applications, in the majority of their use cases, typically operate within data silos. The data in these applications is confined within a specific ecosystem; this makes collaboration or reading dependent on access to the resource through the application or with an explicit export to other formats that are further distributed [18].

In stark contrast, the inception of Linked Data on the Web envisioned a more interconnected future. The objective is to improve semantic interoperability, enable data integration, and transform the Web from a collection of documents to a Web of interconnected

Table 1: Comparison of open-source OT and CRDT Java Script libraries for JSON based and/or rich text editing.

Family	Library	Open Source	JSON	Rich Text	Distributed	Dev Friendly	Maintained
OT	ShareDB [8]	✓	✓	✓	×	✓	✓
	Etherpad [9]	✓	×	✓	×	✓	✓
	TogetherJS [10]	✓	✓	×	×	✓	×
	ot.js [11]	✓	×	✓	×	×	×
	Firepad [12]	✓	×	✓	×	✓	×
CRDT	Automerge	✓	✓	✓	✓	✓	✓
	Yjs	✓	✓	✓	✓	✓	✓
	RON (Swarm) [13]	✓	✓	✓	✓	×	×
	GUN [14]	✓	✓	×	✓	✓	✓
	JSON JOY [15]	✓	✓	×	✓	✓	✓
	Legion [44, 45]	✓	✓	×	✓	×	×
	Convergence [16]	✓	✓	✓	✓	✓	×

data [7, 9]. Our mission is to bring together these paradigms: We aim to harmoniously integrate the collaborative editing experience derived from state-of-the-art co-editing applications with the versatile data linking capabilities of RDF.

To date, CRDTs designed for RDF are somewhat in their nascent stage. While there are various efforts to implement such CRDTs [], their capabilities are limited. In today's implementations, the smallest directly alterable unit remains an RDF node — be it subject, object, or predicate. Consequently, more detailed edits on lengthier nodes of text or other data structure where more precise update mechanisms are required are still not an option. Our PodMerge solution addresses this by conceptualizing each node as a distinct CRDT, with the overarching document envisioned as an RDF-compatible CRDT tree. The extensibility of our approach allows us to support a range of CRDT libraries, with empirical testing conducted on the prominent CRDT libraries Yjs and Automerge [].

One mentioned criterion JavaScript CRDT libraries must conform to in PodMerge is the representability of the document structure in the JSON format. JSON's ubiquity in the Web ecosystem is undeniable and is attributed to its innate simplicity and versatility [32]. Predominantly, JSON's seamless serialization and deserialization capabilities in JavaScript have rendered it an attractive document data type for numerous CRDT solutions. However, the Semantic Web's aspirations demanded a format that retained JSON's human-centric readability while facilitating web-scale data interoperation. Hence, JSON-LD [17] was introduced to integrate data into the Semantic Web while being 100% compatible with traditional JSON [23]. Given JSON's entrenched support across CRDT libraries, JSON-LD emerges as the logical choice for building collaborative linked data applications that are interoperable.

Central to both the Yjs and Automerge libraries is the concept of documents, structured as compositions of diverse CRDTs such as maps, lists, or rich text. Inherently, these documents can be represented in JSON format, with CRDTs as distinctive properties. An example, illustrated in Figure 3, encapsulates a document as a map-based CRDT that contains key-value pairs or the so-called JSON properties. These properties can be intrinsic to the map CRDT or embody other nested CRDTs of various types. As a result, by

Figure 3: Visualisation of a JSON-LD document in the PodMerge approach: The document structure is based on JSON compatible CRDT libraries with the option to nest other CRDTs in properties. This allows one to create Linked Data nodes that are collaborative with targeted CRDT algorithms for the use case.

adding JSON-LD specific semantics like `@context` into the document, it becomes feasible to create CRDTs that can be represented in the JSON-LD format using prevalent CRDT libraries. This equips users with an unprecedented granularity of control, enabling co-editing on RDF-nodes with targeted CRDT employment; e.g., using rich-text CRDTs for nodes with longer text.

Although all JSON-LD documents are also JSON documents, JSON-LD introduces additional specifications on top of the JSON format. Algorithms and functions in the JSON compatible CRDT libraries are built to produce valid JSON documents. However, building valid JSON-LD documents is a profoundly more complex task because the JSON-LD specification does not only depend on the document structure and types as in JSON but also syntax tokens and keywords, combinations of these tokens that are possible, and rules for the semantic structure of the documents. Only providing functions in the CRDT library that do not violate the JSON-LD specification and implementing merging algorithms that can handle incompatible states from different copies goes beyond this paper. Since, our object is to use the existing libraries and not implement new ones, we create an other solution for this within the PodMerge

[17] https://www.w3.org/TR/json-ld/

approach: Users can freely collaborate on the document and violate the rules for JSON-LD in the process, however once the user wants to update the Solid Pod state of the document, he is required to validate the document state for JSON-LD. This is to ensure that the online version of the data is always valid JSON-LD and hence interpretable as Linked Data.

Our PodMerge solution accentuates the collaborative editing experience, provides users the means to interlink their Web resource with external data, and describes it with Semantic Annotations. The flexibility of the PodMerge methodology means that it has potential adaptability across all JSON-like data-structured CRDTs or even distributed OT libraries, laying the groundwork for compatibility with a wide spectrum of CRDT libraries.

4.2.3 Textual Representation. CRDT documents are typically stored and shared in binary implementation-specific formats for efficiency and compactness reasons, necessitating the specific CRDT framework to interpret the data. While this binary format is beneficial to store data and facilitate updates in the CRDT document, it is less human-readable than textual formats. However, for the Semantic Web we require a document that is discoverable and in a human-readable RDF format such as JSON-LD without specialized software. To address this, we introduce a textual representation of the binary CRDT document, allowing any Web client to access the most recent state of the CRDT; only clients actively collaborating require specialized software (e.g., a Web browser plugin) to resolve the CRDTs. The server stores both the binary CRDT document including the complete change history and a textual representation of the latest state. Updating the textual representation operates similarly to a Git commit-and-push, enabling users to contribute their local state. This means that the document state in the Solid Pod is not real-time. The online state is only updated when a collaborator consciously pushes his changes that are valid JSON-LD. Whenever such a push to the Solid Pod occurs, both the CRDT document and the textual representation must be updated.

4.3 Collaborative Resource Description Ontology (CRDO)

Traditional collaborative editors rely on fixed contracts imposed by static APIs, which necessitates hard-coding into clients and limits interoperability among different systems. In contrast, the Semantic Web already provides means to support more open interactions on the Web through hypermedia controls, by using methodologies such as Hydra and HCTL. Hydra and HCTL define vocabularies and mechanisms in the context of Web APIs that enable clients to discover actions and operations dynamically at run-time by inspecting the data they receive. Essentially, they aim to make APIs more self-descriptive and adaptable to changes.

Similarly, we propose a novel vocabulary, Collaborative Resource Description Ontology (CRDO), that complements existing hypermedia controls vocabularies to create such hypermedia controls for real-time collaborative Web resources in a Solid Pod. CRDO provides terms to create descriptions for collaborative resources on the Web which advertise possible client actions for reading, manipulating, and synchronizing documents with peers. Beyond facilitating collaborative resource descriptions and hypermedia controls, our vocabulary also introduces terms to describe textual CRDT representations that are in an RDF format. The vocabulary provides classes, properties, and data types to declare the Web resource as a textual representation, link to the collaborative resource description, and annotate resource values with the utilized CRDT data type.

The CRDO is a vocabulary to support the PodMerge approach by decoupling the client from the server and improving its adaptability to changes using Web and framework specific operations that are machine-processable at run-time.

4.3.1 Collaborative Resource Description Ontology (CRDO). Figure 4 illustrates the Collaborative Resource Description Ontology (CRDO) with its classes and properties. A main concept of the CRDO is the `crdo:Description`, which is a class to further describe web documents with annotations such as a title, description, and last change date and supported operations to interact with that document. The `crdo:CollaborativeResourceDescription` is a subclass of the `crdo:Description`. It stands at the center of the CRDO because it is used to declare the entry point of a collaborative Web resource. This can be a website, a web application, or other web resource that uses or contains collaboratively editable data. This collaboratively editable data is the textual representation of a CRDT document as discussed earlier. Hence the `crdo:CollaborativeResourceDescription` can also contain `crdo:TextualRepresentaionDescription` and `crdo:DocumentDescription` which are both sub classes of the `crdo:Description`. The `crdo:DocumentDescription` is a description of the document that is used to collaborate on. In the PodMerge approach this is typically a CRDT document. This description lists possible operations to interact with that CRDT document and provide metadata such as the utilized CRDT framework, which is relevant because mainstream CRDT frameworks use implementation-specific formats and synchronization mechanisms. The `crdo:TextualRepresentationDescription` specifies possible operations for clients to interact with the current public version of the CRDT state in textual form.

We refer to the possible actions that are advertised to clients by both descriptions as `crdo:Operation`; these are a key concept of our collaborative resource vocabulary. To foster reuse and interoperability, the `crdo:Operation` class is based on the *hctl:Form* concept from HCTL. The *hctl:Form* defines a set of instructions and constraints that guide clients on how to construct valid requests to interact with hypermedia-driven web services. Since this definition and its implementation aligns with the core concepts of the operations for collaborative resources we use it as the super class for the `crdo:Operation` class. This enables us both to reuse its data and object properties but also provide additional constraints for more specific operations. For example, it introduces the `crdo:OperationType` class with individuals of the class such as `crdo:Read`, `crdo:Write`, or `crdo:Synchronize` that are more aligned with the types required in PodMerge. Such additional constraints are required because our vocabulary also introduces a differentiation between `crdo:WebOperation` and `crdo:FrameworkOperation` which are both sub classes of the `crdo:Operation`.

Instances of the `crdo:WebOperation` represent classic HTTP-based interactions and are essentially the same as operations

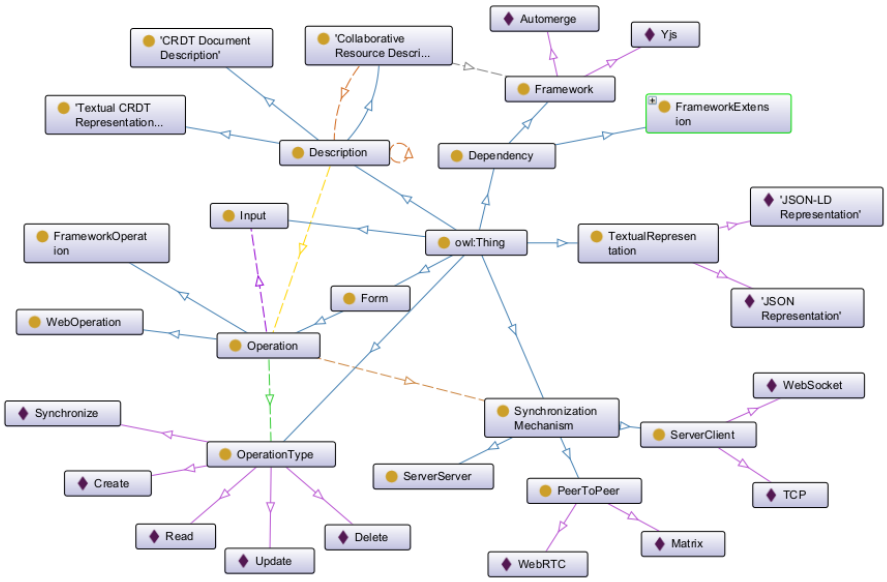

Figure 4: Classes and the connecting properties of the CRDO

using the hypermedia controls from *hctl:Form* with the addition of the CRDO specific operation types. Instances of the `crdo:FrameworkOperation` are used for interactions that are mainly handled or executed by the CRDT framework of the resource. Framework specific operations can be necessary, for example, to establish real-time synchronization with peers using WebRTC [18] or Matrix [19]. These protocols and communication frameworks can be annotated for operations of type `crdo:Synchronize` with the `crdo:SynchronizationMechanism` and its sub classes and individuals. The `crdo:FrameworkOperation` are not as strictly defined as the http-based operations because there are often different approaches between the framework to handle operations such as real-time synchronization over a specific protocol and often even multiple options to do so within the same framework. The idea is that the frameworks or its communities can provide adapters that better standardize the definitions for framework operations and simplify its integration by software developers.

Another concept provided by the CRDO that is required for clients to establish real-time collaboration on the web resource at run-time are the dependencies. The CRDO introduces the `crdo:Dependency` and its sub classes `crdo:Framework` and `crdo:FrameworkExtension`. Individuals of the `crdo:Dependency` must be packages that are available in the NPM registry [20]. Individuals of the `crdo:Framework` are more specifically declaring the main CRDT framework that is used for co-editing and the `crdo:FrameworkExtension` are extensions of that framework such as adapter for real-time synchronisation with peers. The dependencies can be properties of an instance of the `crdo:Description` or `crdo:Operation`.

4.3.2 Collaborative Resource Description (CRD). The Collaborative Resource Description is a file or multiple files in RDF that uses the proposed CRDO vocabulary to provide a comprehensive description all the operations and metadata required for clients to establish real-time co-editing at run-time. We show an excerpt of an example of such a CRD in Listing 1. Typically within the PodMerge approach the main entry point of the CRD and the description of the textual representation will be publicly available and as such discoverable in the Web. However, access on the description of the CRDT document might be restricted to vetted collaborators. In that case you have to swap the parts of the `crdo:DocumentDescription` into another file with different access policies under Solid.

4.3.3 Annotating the Textual Representation. Apart from enabling descriptions of resources, our vocabulary introduces terminology to depict textual CRDT representations within a Linked Data structure. In the PodMerge approach this is mainly the

[18] https://webrtc.org/
[19] https://matrix.org/
[20] https://www.npmjs.com/

JSON-LD format. An example of a JSON-LD document that is annotated with the CRDO vocabulary for textual representations is shown in Listing 2. The document is annotated with a type of `crdo:TextualRepresentation` to show that it is a textual representation of a collaborative document. Thus, when the document is discovered by a client or agent he can infer it as a collaborative document and dereference the `crdo:hasCollaborativeResourceDescription` value to access the description of the collaborative resource.

Furthermore, CRDO also introduces new data types for collaborative resources, such as `crdo:Map`, `crdo:Text`, or `crdo:List`. Listing 2 shows the CRDO data type for the properties `"articleBody"` and `"keywords"` These data types not only inform agents about the collaborative data types utilized for the properties, but also give them the option to use this as the basis for creating a new collaborative document. This can be useful if a user wants to collaboratively edit a copy of the data. With the right adapter that understands the CRDO data types, the collaborative framework can infer the document structure with the appropriate data types.

Listing 1: An excerpt from a description of a collaborative web blog in turtle that uses the PodMerge approach with Yjs CRDTs and WebRTC synchronization.

```
@base <https://imp.inrupt.net/local-first/blog/context>.
...
:CRD
  a crdo:CollaborativeResourceDescription ;
  crdo:title "A coffee blog";
  crdo:description "A coffee blog with ...";
  crdo:created "2022-11-14T19:10:23.824Z" ;
  crdo:modified "2023-06-01T09:12:43.124Z" ;
  crdo:document :DD;
  crdo:textualRepresentation :TRD.

:DD
  a crdo:DocumentDescription;
  crdo:framework crdo:Yjs ;
  crdo:operation
    [
      a crdo:WebOperation ;
      crdo:operationType crdo:Read ;
      hctl:forContentType "application/octet-stream" ;
      http:methodName "GET" ;
      hctl:hasTarget "https://imp.inrupt.net/local-first/blog/content.bin" ;
    ],
    ...
    [
      a crdo:FrameworkOperation ;
      crdo:dependency crdo:Yjs, crdo:YWebRtc ;
      crdo:operationType crdo:Synchronize ;
      crdo:syncMechanism crdo:WebRTC ;
      crdo:withInput [
        a crdo:Input;
        crdo:name "name";
        rdf:value "coffeeBlog"^^xsd:string ; crdo:isRequired true
      ],
    ].
  ...
:TRD
  a crdo:TextualRepresentationDescription;
  ...
```

4.4 Solid Pods as Repository for Collaborative Web Resources

To support open collaboration on Linked Data, we use the Solid Protocol and especially Solid pods as an integral part of the PodMerge approach. Solid pods serve as repositories for the web resources in a Linked Data format and files that are required for collaboration in PodMerge, such as the CRDT document. A key advantage of Solid is that it provides fine-grained, decentralized access control in the Semantic Web. This feature allows for the creation of a publicly readable version of the collaborative resource and textual representation, while restricting controls for collaboration. Another crucial advatage of Solid is that the control of the data is given back to the end user, which works great in combination with the PodMerge approach, since CRDTs do not require a central coordinator. Thus, the user is in full control not only of the local CRDT document, but also of the online counterpart and its textual representation. In this chapter, we first expand on the rationale behind aligning the PodMerge approach with the Solid protocol. Moreover, we look

Listing 2: An example of a textual representation of a Yjs CRDT document using the CRDO vocabulary for annotations

```
{
  "@context":
  {
    "@version": 1.1,
    "@base": "https://imp.inrupt.net/local-first/blog/",
    "schema": "http://schema.org/",
    "crdo": "https://imp.inrupt.net/ontologies/crdo.ttl#",
    "hasCRD": {
      "@id": "crdo:hasCollaborativeResourceDescription",
      "@type": "@id"
    },
    "framework": "crdo:framework",
    "headline": "schema:headline",
    "articleBody": {
      "@id": "schema:articleBody",
      "@type": "crdo:Text"
    },
    "keywords": {
      "@id": "schema:keywords",
      "@type": "crdo:List"
    },
    ...
  },
  "@id": "content.jsonld",
  "@type":  [ "schema:BlogPosting", "crdo:TextualRepresentation" ],
  "hasCRD": "context.ttl",
  "framework": "crdo:Yjs",
  "headline": "World of Coffee",
  "articleBody": "This blog post explores ...",
  "keywords": [ "coffee", "brewing", "history" ],
  ...
}
```

at how PodMerge leverages Solid's access control mechanism and propose a structured document layout for a Solid pod.

4.4.1 Reasons for Solid Protocol Integration in PodMerge. The Solid protocol integration into the PodMerge approach offers clear advantages when working with Linked Data resources. The primary motivation for this choice is the decentralized nature of both Solid and CRDTs. This decentralization ensures that no single entity has overarching control over the data which makes collaborations more democratized and reduces potential single points of failure.

Firstly, Solid's inherent design emphasizes data ownership and control. In traditional collaboration platforms, data is typically stored on a central server, with the platform provider having implicit control over the data. This often raises concerns about data privacy, ownership, and potential misuse. By using Solid, each user maintains ownership of their data in the Web. This is aligned with the principles of the PodMerge approach, which aims to give users control over both the local and online versions of their collaborative documents.

Secondly, the use of the Solid protocol facilitates interoperability. Linked Data, by its nature, is designed to be interconnected and referenced across various Web resources. The Solid protocol ensures that the data stored in pods is in a standard Linked Data format that can be easily accessed and referenced by heterogeneous applications. These same principles are applied in the PodMerge approach with its CRDO vocabulary. This alignment allows us to provide collaboration on Web resources that are not restricted to a single application or platform, thus promoting a more open and inclusive collaborative environment.

Moreover, combining Solid with the local-first software principles in PodMerge allows us to make data available and discoverable online while maintaining user control and privacy. Drawing a parallel, using Solid pods in tandem with local-first software mirrors the relationship between Git and GitHub. Just as GitHub offers an online repository for code with access management, Solid pods act as a repository for the PodMerge resources with even more controlled access settings for individual resources or collections. We use Solid pods as a mediator to facilitate collaboration on Web resources across heterogeneous applications, while also supporting the principles of local-first software by enabling direct collaboration between participants.

In conclusion, the integration of the Solid protocol into the PodMerge approach provides a robust, decentralized platform for collaboration on Linked Data resources. By combining the strengths of both Solid and local-first software prinpicples, PodMerge offers a powerful tool for open collaboration, ensuring data ownership, interoperability, and seamless collaboration.

4.4.2 Access to Collaborative Resources with WebIDs. The PodMerge approach uses Solid's WebID to identify user and provide access control based on it. A WebID is a unique URL-based identifier linked to a user's profile in the Solid ecosystem [26, 36]. It enables secure user authentication and links data across various services using a single identity. This means that users of the PodMerge approach are required to have a WebID and a personal Solid Pod to collaborate on access restricted resources or provide co-editing on own resource. The WebID and Solid Pod can be registered by a public Solid provider or by self-hosting a Solid server [21]. In addition, intelligent clients and applications must support user authentication based on the Solid WebID.

In Solid pods, WebIDs determine access permissions, allowing users to specify which WebIDs can interact with their data. This is done by providing each resource with an access Control List (ACL) that defines permissions for the various WebIDs or the public. These permissions can encompass actions like reading, writing, or appending. For making web resources collaborative with the PodMerge approach, we identified three main user groups; the public, collaborators, and project owners. The public typically can read the web resource, its description and the textual representation of the CRDT document. Collaborators can also read and write all the resources required for collaboration, including the CRDT document, the textual representation, and its descriptions. Project owners generally have full access to all resources, and the privileges set access to other users. Solid uses containers to group resources, similar to how folders group files in traditional file systems. The ACL of a container typically applies to all its contents, unless it is not overridden by specific resource ACLs. We can use this property to suggest a general container structure within the PodMerge approach that simplifies access management for user groups. An overview of this can be seen in Figure 5. Important to note is that the URI of a resource in Solid reflects its location within the Solid pods hierarchy; e.g. using our proposed approach the URI of the CRD assuming the file is named context is `https:/[pod URL]/[project]/podmerge/context#CRD`

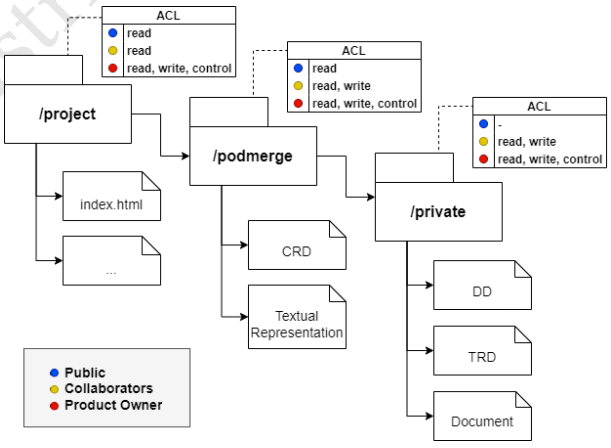

Figure 5: Suggested collection structure for the PodMerge approach within a users Solid pod. The ACL of a collection applies to all its resources.

5 DEMONSTRATIVE USE-CASE

To showcase the practical implementation of the PodMerge approach, we demonstrate real-time co-editing on a static website hosted in a Solid Pod, allowing collaborative editing of the website's content directly within a web browser. In addition, we integrate a GUI-based editor for Linked Data annotations, enabling users

[21] Infos to register a Solid pod and WebID: https://solidproject.org/users/get-a-pod

to update the Linked Data context of the website's content. The demonstration shows that it is possible to create real-time collaborative Linked Data systems with reasonable overhead using the PodMerge approach. It demonstrates the use of JSON-based CRDTs for open Web data with the addition of a textual representation and the creation of collaborative websites and structured data markups for users without the need for an expensive server setup. Additionally, we demonstrate the implementation of an intelligent client service in the form of the Chrome browser extension that can read the CRD and provide direct collaboration with peers.

We accomplish this by providing a CRD of the content using our proposed CRDO vocabulary. The CRD can be consumed by a software client; in our case, we developed a Chrome browser extension for this purpose. The extension utilizes the discovered CRDT state or creates a new CRDT state of the content using a CRDT framework. Moreover, it leverages the operations specified in the CRD to facilitate user collaboration through three main functionalities: (i) presenting a pop-up with actions to initiate the collaboration, synchronize changes in real-time with other peers using WebRTC, and commit the current state to the Solid pod; (ii) converting the website elements that utilize the collaborative content into editable elements and synchronizing their content with the state of the CRDT document; and (iii) introducing a Linked Data editor panel offering utility functions to read and update the term definitions of the content's context.

In this chapter, we first introduce the use case and its problem description further. Afterwards, we present the implementation of the use case. On one side, the setup and construction of the CRD and Solid pod repository. On the other side, the implementation of the browser extension that uses this CRD to provide co-editing on the Web resource and Linked Data annotations. Finally, we discuss the results of the demonstrative use case and its implications.

5.1 Use Case and Problem Description

In the contemporary digital age, web content increasingly demands collaborative editing and semantic enrichment to enhance user and machine understanding. Yet, conventional tools struggle to address the rising needs for decentralization and interoperability in this diverse landscape.

To bring this into perspective we created a demonstrative website — a simple Café site. The Café website serves as a digital storefront, featuring a menu, store information, and promotional content. In the context of our demonstration, the goal was to make this website's content collaboratively editable in real-time. This means that multiple users can simultaneously edit the website's content, updating menu items, or tweaking promotional content. Moreover, we want to provide a layer of semantic richness, allowing for annotations that improve its Search Engine Optimization (SEO) and make it readily discoverable on the Semantic Web. Building on the principles of Solid and local-first software, such a setup requires decentralized access and data management, allowing for co-editing across heterogenous applications or diverse client services.

However, achieving this is not straightforward. Currently, enabling real-time collaboration on Web resources often requires specialized software or platforms. Existing tools, while powerful, often centralize data and control, which is in contrast to the ethos of the modern decentralized web. Additionally, when we add fine-grained collaboration of RDF nodes, the building blocks of the Semantic Web, the challenges multiply. There are currently no well-known tools that allow seamless co-editing of RDF nodes. Using our PodMerge approach, we demonstrate a potential solution for seamless co-editing of the Café website in a decentralized Linked Data ecosystem.

5.2 Implementation

For the implementation, we divide it into two main parts: On one hand, there is the user who wants to enable decentralized real-time co-editing on his Web resource, in this case the Café website. Using the PodMerge approach, this requires the construction of a CRD, the setup in the user's Solid pod, and linking of the website elements to the collaborative data and CRD. On the other hand, there is the development of a client service that enables collaborative editing on the Web resource for the user and other participants. Using the requirements of the PodMerge approach, this includes interactions based on the operations in the CRD, the construction of a local CRDT document with direct peer communication, and the use-case-specific manipulation of the Café website.

5.2.1 Enabling Co-Editing with a CRD and Solid Pod. The initial situation is a simple static Café website in HTML that loads external content from a JSON file. In the current use-case implementation, the external data is required to be a JSON or JSON-LD file. The first step for the user is to create a CRD for the Café website using the CRDO and HCTL vocabularies. While this is currently a mostly manual process by the user, in the future, this can be greatly simplified by a setup or client service. In the CRD construction, the user links a TRD with operations to read and update the JSON content, this JSON file will be used as the initial CRDT state by the client service. Additionally, he links a DD with operations to read and update a binary document, the document does not yet have to exist as it will be created by the client service, but its final storage location in the Solid pod should be linked. In the DD, the user also includes the CRDT library and an operation for synchronizing the communication protocol with its required dependencies.

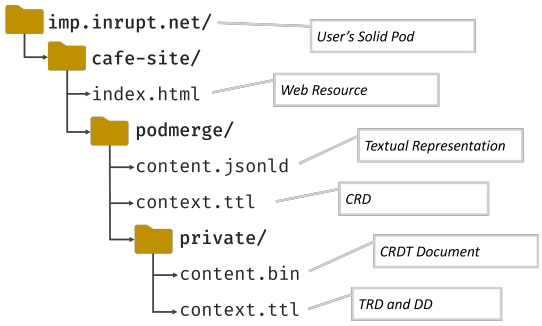

Figure 6: Collection and file structure of the user's Solid pod from the Café website demo. The resources can be accessed publicly using its path; e.g., the CRD can be accessed under https://imp.inrupt.net/cafe-site/podmerge/context.ttl.

In a second step, the user sets up his Solid pod collection and file structure and sets the ACLs based on the PodMerge recommendations. A visualisation of that structure can be seen in Figure 6. The Café website, pod structure and CRD of the use case can be publicly accessed under https://imp.inrupt.net/cafe-site/.

The last step is the linking of the CRD from the collaborative Web resource. In our case the collaborative Web resource is the Café website which requires a link to the CRD which is machine readable. We do this by including a link tag in the header of the website with a custom `rel` attribute using our CRDO vocabulary: `<link rel="https://imp.inrupt.net/ontologies/crdo.ttl#hasCRD" href="/podmerge/context.ttl#CRD">`.

5.2.2 The PodMerge Chrome Extension: Bridging Collaboration and Semantic Annotation. To showcase a client service that can read the Café website's CRD and provide co-editing using the PodMerge approach we implemented a Chrome browser extension. The current implementation of the browser extension is tailored to the Café use case of making simple static websites collaborative with semantic annotations. It can provide co-editing for CRDs that use the Yjs and Automerge CRDT frameworks. The Chrome extension encompasses three major components:

(1) **Browser Pop-up:** This tool provides users with authentication options via their WebID, enabling access to collaborative resources within the Solid pod. Post-authentication, users can initialize the co-editing session and connect to peers for real-time updates. Additionally, it allows users to push the current state to the Solid pod repository which updates the CRDT document and textual representation. Thus, creating a new public version of the website.
(2) **On-site Editable Elements:** Upon activation, the browser extension embeds a script into the target website that renders the elements editable through a CRDT document. It uses the CRDT framework to initialize a new document. The document's initialization can originate from various sources: local state, the discovered public document, or textual representation (if no existing CRDT document is found). The chrome extension uses a reactive pattern to update the document based on user updates and the website elements based external peer updates. In this demonstration, we use WebRTC to connect to the other peers. This is not fully serverless however, as we require a signaling server initially to coordinate this connection.
(3) **Embedded Linked Data Editor:** An addition to the content editing is an embedded GUI-based Linked Data editor, accessible via a toggle sidebar initiated by a floating button. This interface, a screenshot can be seen in Figure 7, provides user functionality to view and update the semantic annotations of the data properties, which are also synchronized real-time in the CRDT document using specifically crafted context functions. These functions ensure the validity of the JSON-LD format specification and automatically handle abbreviations of well-known Linked Data vocabularies. The Linked Data editor also gives the user the option to view the entire CRDT document in a human readable format.

The developed Chrome extension demonstrates the PodMerge approach by transforming static websites, like the Café case, into dynamic collaborative platforms. It seamlessly blends the decentralized authentication, real-time co-editing, and enriched semantic annotations to offer a comprehensive toolkit for modern digital collaboration, ensuring data consistency and immediate synchronization

5.3 Results and Discussion

The demonstration of decentralized co-editing through the Café website exemplifies the efficacy and versatility of the PodMerge approach. The Café website transformed from a static digital presence into an interactive platform with real-time collaborative editing on semantically enriched data. This dynamic shift showcases how effectively multiple users can co-edit a singular web resource in RDF, ensuring consistency and immediate synchronization. We demonstrate how PodMerge simplifies the user process of making the Café website collaborative without a central coordination server using a CRD and Solid pod. Moreover, we provide an example of a client service that can operate on this CRD and provide co-editing on the website with direct peer collaboration.

From a broader perspective, this demonstration holds significant implications for the Semantic Web community. It signifies the beginning of a new era where decentralized web resources can become inherently collaborative without relying on centralized servers or proprietary platforms. This leap potentially allows the construction of more open, transparent, and user-controlled co-editing sessions on heterogeneous applications. The PodMerge approach, although demonstrated on the Café website, is flexible enough to be applied across a wide number of use cases, making its application vast and versatile. It also opens up avenues for integration with other semantic web tools and CRDT frameworks, offering a comprehensive toolkit to web developers.

However, like any innovative approach, there were challenges and limitations. The primary challenge is currently a uniform interpretation and implementation of framework operations, which is crucial for creating a collaborative Linked Data system with heterogeneous user applications. Additionally, while the PodMerge approach is versatile, specific vocabularies, CRDT frameworks and CRD structures may pose initial setup challenges for those unfamiliar with these concepts. Hence, future implementations should focus on simplifying the CRD creation process, creating adapter for popular CRDT frameworks, and expanding the support to a broader range of use-cases.

6 CONCLUSION

- Hard to build a co-editor application as a user if you want to make a personal Web resource collaborative. We take away this complexity by providing a language to write a simple interface to do so and defining specifications to allow an ecosystem of applications that support this.

Our approach significantly simplifies the implementations of real-time collaboration in Linked Data Systems by providing vocabulary to describe the collaborative interface. Moreover, it ensures access control, interoperability, and discoverability of shared data types by leveraging the Solid protocol, hypermedia controls, and CRDTs with textual representations. Ongoing and future work on our current approach include testing and expanding our system's

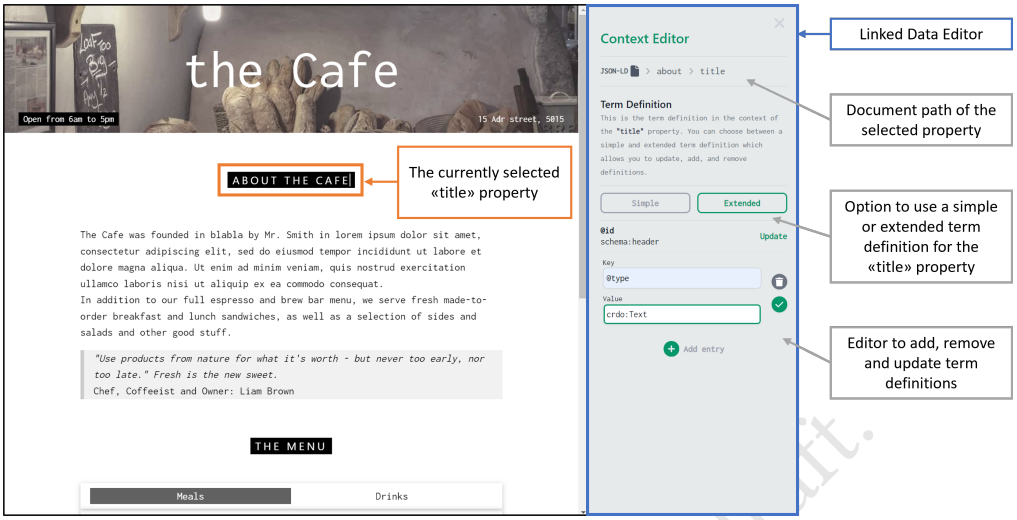

Figure 7: Screenshot of the Café website with the embedded Linked Data editor from the Chrome browser extension. The user is currently updating the content and context of the "title" property.

support for various CRDT implementation frameworks and formats as well as the validation of the approach in other realistic use cases to assess its versatility and effectiveness. We furthermore plan to publish a specification and stable implementation for broader adoption and feedback, and to explore the potential extension of our approach beyond the Solid ecosystem as well as its application in existing Linked Data systems.

7 FUTURE WORK

- How can Solid be further integrated -> copying public versions of websites to introduce a private version with real-time updates, annotations (simplifying LD annotations and structured data for the Web),
- Currently the update and versioning mechanism for the Solid Pod are very rudimentary. While there is a new state of the collaborative document associated with the textual representation of the resource, it is not clear what version that is. Also, metadata like version, last change date, contributors, etc. are possible to set, but there is no clear guideline yet.
- Implementing adapters for popular CRDT frameworks that can interact with Solid resources and read/write the crdo:Descriptions to be able to further simplify the use of PodMerge through applications that guide the user through this process

ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

REFERENCES

[1] Paulo Sérgio Almeida, Ali Shoker, and Carlos Baquero. 2015. Efficient State-Based CRDTs by Delta-Mutation. In *Networked Systems*, Ahmed Bouajjani and Hugues Fauconnier (Eds.). Springer International Publishing, Cham, 62–76.

[2] Paulo Sérgio Almeida, Ali Shoker, and Carlos Baquero. 2018. Delta state replicated data types. *J. Parallel and Distrib. Comput.* 111 (2018), 162–173. https://doi.org/10.1016/j.jpdc.2017.08.003

[3] Khaled Aslan, Pascal Molli, Hala Skaf-Molli, and Stéphane Weiss. 2011. C-Set : a Commutative Replicated Data Type for Semantic Stores. In *RED: Fourth International Workshop on REsource Discovery*. Heraklion, Greece. https://inria.hal.science/inria-00594590

[4] Sören Auer, Sebastian Dietzold, and Thomas Riechert. 2006. OntoWiki – A Tool for Social, Semantic Collaboration. In *The Semantic Web - ISWC 2006*, Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora M. Aroyo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 736–749.

[5] Florian Bauer and Martin Kaltenböck. 2011. Linked open data: The essentials. *Edition mono/monochrom, Vienna* 710 (2011), 21.

[6] Tim Berners-Lee. 2009. *Read-Write Linked Data.* Retrieved August 16, 2023 from https://www.w3.org/DesignIssues/ReadWriteLinkedData.html Last change: 2018/01/19.

[7] TIM BERNERS-LEE, JAMES HENDLER, and ORA LASSILA. 2001. THE SEMANTIC WEB. *Scientific American* 284, 5 (2001), 34–43. http://www.jstor.org/stable/26059207

[8] Tim Berners-Lee and Kieron O'Hara. 2013. The read–write Linked Data Web. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371, 1987 (2013), 20120513. https://doi.org/10.1098/rsta.2012.0513 arXiv:https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2012.0513

[9] Christian Bizer, Tom Heath, and Tim Berners-Lee. 2011. *Linked Data: The Story so Far.* IGI Global, Hershey, PA, 205–227. https://doi.org/10.4018/978-1-60960-593-3.ch008

[10] Amos Brocco. 2022. Melda: A General Purpose Delta State JSON CRDT. In *Proceedings of the 9th Workshop on Principles and Practice of Consistency for Distributed Data* (Rennes, France) *(PaPoC '22)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3517209.3524039

[11] Leslie Carr, Wendy Hall, Sean Bechhofer, and Carole Goble. 2001. Conceptual linking: ontology-based open hypermedia. In *Proceedings of the 10th international conference on World Wide Web*. 334–342.

[12] Meran Haymo. 2013. *Wikidocs - Real time collaborative editing for HTML.* Retrieved August 23, 2023 from https://www.slideshare.net/draftkraft/wikidocs-real-time-collaborative

[13] Tom Heath and Enrico Motta. 2008. Revyu: Linking reviews and ratings into the Web of Data. *Journal of Web Semantics* 6, 4 (2008), 266–273. https://doi.org/10.1016/j.websem.2008.09.003 Semantic Web Challenge 2006/2007.

[14] Luis Daniel Ibáñez, Hala Skaf-Molli, Pascal Molli, and Olivier Corby. 2012. Synchronizing Semantic Stores with Commutative Replicated Data Types. In *Proc. of WWW '12 Companion'* (Lyon, France). ACM, 1091–1096. https://doi.org/10.1145/2187980.2188246

[15] Luis-Daniel Ibáñez, Hala Skaf-Molli, Pascal Molli, and Olivier Corby. 2014. ColGraph: Towards Writable and Scalable Linked Open Data. In *The Semantic Web – ISWC 2014*, Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble (Eds.). Springer International Publishing, Cham, 325–340.

[16] Kevin Jahns. 2020. *Are CRDTs suitable for shared editing?* Retrieved September 24, 2023 from https://blog.kevinjahns.de/are-crdts-suitable-for-shared-editing/

[17] Kevin Jahns. 2021. *How we made Jupyter Notebooks collaborative with Yjs.* Retrieved August 23, 2023 from https://blog.jupyter.org/how-we-made-jupyter-notebooks-collaborative-with-yjs-b8dff6a9d8af

[18] Ramesh Jain. 2003. Out-of-the-box data engineering events in heterogeneous data environments. In *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405).* 8–21. https://doi.org/10.1109/ICDE.2003.1260778

[19] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. 2004. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics* 2, 1 (2004), 49–79. https://doi.org/10.1016/j.websem.2004.07.005

[20] Martin Kleppmann and Alastair R Beresford. 2018. Automerge: Real-time data sync between edge devices. In *1st UK Mobile, Wearable and Ubiquitous Systems Research Symposium (MobiUK 2018). https://mobiuk. org/abstract/S4-P5-Kleppmann-Automerge. pdf.* 101–105.

[21] Martin Kleppmann, Adam Wiggins, Peter Van Hardenberg, and Mark McGranaghan. 2019. Local-first software: you own your data, in spite of the cloud. In *Proc. of Onward! 2019.* 154–178. https://doi.org/10.1145/3359591.3359737

[22] Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. 2009. Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections. In *The Semantic Web: Research and Applications*, Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 723–737.

[23] Markus Lanthaler and Christian Gütl. 2012. On Using JSON-LD to Create Evolvable RESTful Services. In *Proceedings of the Third International Workshop on RESTful Design* (Lyon, France) *(WS-REST '12)*. Association for Computing Machinery, New York, NY, USA, 25–32. https://doi.org/10.1145/2307819.2307827

[24] Markus Lanthaler and Christian Gütl. 2013. Hydra: A Vocabulary for Hypermedia-Driven Web APIs. *LDOW* 996 (2013), 35–38.

[25] Barry Levine. 2014. *Atlassian buys Wikidocs and the tech to make websites and apps collaborative.* https://venturebeat.com/entrepreneur/atlassian-buys-wikidocs-and-the-tech-to-make-websites-or-apps-collaborative/ Accessed: 23 September 23.

[26] Essam Mansour, Andrei Vlad Sambra, Sandro Hawke, Maged Zereba, Sarven Capadisli, Abdurrahman Ghanem, Ashraf Aboulnaga, and Tim Berners-Lee. 2016. A Demonstration of the Solid Platform for Social Web Applications. In *Proc. of WWW '16 Companion* (Montréal, Québec, Canada). WWW Conferences Steering Committee, 223–226. https://doi.org/10.1145/2872518.2890529

[27] year = 2021 Mariën, Jan, title = Real-Time Collaboration in Jupyter Notebooks. [n. d.]. Master's thesis.

[28] Moulay Driss Mechaoui, Nadir Guetmi, and Abdessamad Imine. 2015. Towards Real-Time Co-authoring of Linked-Data on the Web. In *Computer Science and Its Applications: 5th IFIP TC 5 International Conference, CIIA 2015, Saida, Algeria, May 20-21, 2015, Proceedings 5*. Springer, 538–548.

[29] Marios Meimaris, George Alexiou, and George Papastefanatos. 2014. LinkZoo: A Linked Data Platform for Collaborative Management of Heterogeneous Resources. In *The Semantic Web: ESWC 2014 Satellite Events*, Valentina Presutti, Eva Blomqvist, Raphael Troncy, Harald Sack, Ioannis Papadakis, and Anna Tordai (Eds.). Springer International Publishing, Cham, 407–412.

[30] Petru Nicolaescu, Kevin Jahns, Michael Derntl, and Ralf Klamma. 2015. Yjs: A framework for near real-time p2p shared editing on arbitrary data types. In *International Conference on Web Engineering*. Springer, 675–678.

[31] Alexandre Passant and Philippe Laublet. 2008. Meaning Of A Tag: A collaborative approach to bridge the gap between tagging and Linked Data. *LDOW* 369 (2008).

[32] Dunlu Peng, Lidong Cao, and Wenjie Xu. 2011. Using JSON for Data Exchanging in Web Service Applications. 7 (12 2011).

[33] Igor O. Popov, M. C. Schraefel, Wendy Hall, and Nigel Shadbolt. 2011. Connecting the Dots: A Multi-pivot Approach to Data Exploration. In *The Semantic Web – ISWC 2011*, Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 553–568.

[34] Matthias Ressel, Doris Nitsche-Ruhland, and Rul Gunzenhäuser. 1996. An Integrating, Transformation-Oriented Approach to Concurrency Control and Undo in Group Editors. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work* (Boston, Massachusetts, USA) *(CSCW '96)*. Association for Computing Machinery, New York, NY, USA, 288–297. https://doi.org/10.1145/240080.240305

[35] Arik Rinberg, Tomer Solomon, Roee Shlomo, Guy Khazma, Gal Lushi, Idit Keidar, and Paula Ta-Shma. 2022. DSON: JSON CRDT Using Delta-Mutations for Document Stores. *Proc. VLDB Endow.* 15, 5 (jan 2022), 1053–1065. https://doi.org/10.14778/3510397.3510403

[36] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. 2016. *Solid: A Platform for Decentralized Social Applications Based on Linked Data.* Tech. Rep. MIT Computer Science and Artificial Intelligence Laboratory and Qatar Computing Research Institute.

[37] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. 2014. Adoption of the Linked Data Best Practices in Different Topical Domains. In *The Semantic Web – ISWC 2014*, Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble (Eds.). Springer International Publishing, Cham, 245–260.

[38] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. Conflict-Free Replicated Data Types. In *Stabilization, Safety, and Security of Distributed Systems*, Xavier Défago, Franck Petit, and Vincent Villain (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 386–400.

[39] Maher Suleiman, Michèle Cart, and Jean Ferrié. 1997. Serialization of Concurrent Operations in a Distributed Collaborative Environment. In *Proceedings of the 1997 ACM International Conference on Supporting Group Work* (Phoenix, Arizona, USA) *(GROUP '97)*. Association for Computing Machinery, New York, NY, USA, 435–445. https://doi.org/10.1145/266838.267369

[40] Chengzheng Sun and Clarence Ellis. 1998. Operational transformation in real-time group editors: issues, algorithms, and achievements. In *Proc. of CSCW 1998.* 59–68.

[41] Chengzheng Sun and Clarence Ellis. 1998. Operational Transformation in Real-Time Group Editors: Issues, Algorithms, and Achievements. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '98)*. Association for Computing Machinery, New York, NY, USA, 59–68. https://doi.org/10.1145/289444.289469

[42] David Sun, Chengzheng Sun, Agustina, and Weiwei Cai. 2019. Real Differences between OT and CRDT in Building Co-Editing Systems and Real World Applications. *CoRR* abs/1905.01517 (2019). arXiv:1905.01517 http://arxiv.org/abs/1905.01517

[43] Bartosz Sypytkowski. 2021. *Deep dive into Yrs architecture.* Retrieved September 24, 2023 from https://www.bartoszsypytkowski.com/yrs-architecture/

[44] Albert van der Linde, Pedro Fouto, João Leitão, Nuno Preguiça, Santiago Castiñeira, and Annette Bieniusa. 2017. Legion: Enriching Internet Services with Peer-to-Peer Interactions. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 283–292. https://doi.org/10.1145/3038912.3052673

[45] Albert van der Linde, João Leitão, and Nuno Preguiça. 2016. -CRDTs: Making -CRDTs Delta-Based. In *Proceedings of the 2nd Workshop on the Principles and Practice of Consistency for Distributed Data* (London, United Kingdom) *(PaPoC '16)*. Association for Computing Machinery, New York, NY, USA, Article 12, 4 pages. https://doi.org/10.1145/2911151.2911163

[46] Matthew Weidner, Joseph Gentle, and Martin Kleppmann. 2023. The Art of the Fugue: Minimizing Interleaving in Collaborative Text Editing. (2023). https://doi.org/10.48550/arXiv.2305.00583 arXiv:2305.00583 [cs.DC]

[47] E James Whitehead and Meredith Wiggins. 1998. WebDAV: IEFT standard for collaborative authoring on the Web. *IEEE Internet Computing* 2, 5 (1998), 34–40.

[48] Yi Xu and Chengzheng Sun. 2016. Conditions and Patterns for Achieving Convergence in OT-Based Co-Editors. *IEEE Transactions on Parallel and Distributed Systems* 27, 3 (March 2016), 695–709. https://doi.org/10.1109/TPDS.2015.2412938

[49] Hafed Zarzour and Mokhtar Sellami. 2013. srCE: a collaborative editing of scalable semantic stores on P2P networks. *International Journal of Computer Applications in Technology* 48, 1 (2013), 1–13.