FactStack: Interoperable Data Management and Preservation for the Web and Industry 4.0

Lars Gleim[1], Jan Pennekamp[2], Liam Tirpitz[1], Sascha Welten[1], Florian Brillowski[3], Stefan Decker[1,4]

Abstract:

Data exchange throughout the supply chain is essential for the agile and adaptive manufacturing processes of Industry 4.0. As companies employ numerous, frequently mutually incompatible data management and preservation approaches, interorganizational data sharing and reuse regularly requires human interaction and is thus associated with high overhead costs. An interoperable system, supporting the unified management, preservation, and exchange of data across organizational boundaries is missing to date. We propose FactStack, a unified approach to data management and preservation based upon a novel combination of existing Web-standards and tightly integrated with the HTTP protocol itself. Based on the FactDAG model, FactStack guides and supports the full data lifecycle in a FAIR and interoperable manner, independent of individual software solutions and backward-compatible with existing resource oriented architectures. We describe our reference implementation of the approach and evaluate its performance, showcasing scalability even to high-throughput applications. We analyze the system's applicability to industry using a representative real-world use case in aircraft manufacturing based on principal requirements identified in prior work. We conclude that FactStack fulfills all requirements and provides a promising solution for the on-demand integration of persistence and provenance into existing resource-oriented architectures, facilitating data management and preservation for the agile and interorganizational manufacturing processes of Industry 4.0. Through its open-source distribution, it is readily available for adoption by the community, paving the way for improved utility and usability of data management and preservation in digital manufacturing and supply chains.

Keywords: Web Technologies; Data Management; Memento; Persistence; PID; Industry 4.0

1 Introduction

While the management and preservation of manufacturing data regularly play a crucial role to fulfill legal and contractual accountability requirements, today's industrial data management is frequently considered an overhead factor instead of a valuable tool for data reuse, e.g., in the context of process optimization. While many aspects of data reuse have

[1] Databases and Information Systems, RWTH Aachen University, Germany · gleim@dbis.rwth-aachen.de
[2] Communication and Distributed Systems, RWTH Aachen University, Germany
[3] Institute of Textile Technology, RWTH Aachen University, Germany
[4] Fraunhofer FIT, Sankt Augustin, Germany

doi:10.18420/btw2021-20

been studied in prior work [Gl20b; Ka17; LGD20], low-overhead data management solutions for industry are missing to date [Pe19b]. In the following, we introduce a representative use case scenario in the aerospace domain to motivate the remainder of the paper.

Data Management and Preservation for Aircraft Manufacturing. The manufacturing of parts in the aerospace industry has strict certification requirements throughout the manufacturing process and supply chain, requiring detailed data about each process step to be collected, validated, and archived for years. For example, US-American regulations require the secure storage of *type design* case files, comprising drawings and specifications, information about dimensions, materials, and processes, for more than 100 years [Fe06]. Such strict requirements make sophisticated data management and preservation systems indispensable. At the same time, managing data in compliance with such regulations is traditionally associated with significant costs due to overheads incurred e.g., through manual data handling and inspection processes [Po17]. Considering a modern aircraft manufacturing supply chain, massive amounts of production data need to be managed and preserved [Pe19c], involving human paper-based signature mechanisms and leading to high associated overhead costs. In contrast, the efficient digital collection, management, exchange, and preservation of this data could lead to significant cost savings and productivity gains as part of Industry 4.0, not only in aviation but in many industries producing safety-critical components or otherwise facing strict certification and data retention requirements (e.g., textile, food processing, or plastics industry).

Use Case Scenario. Especially the manufacturing of structural elements in the aerospace industry relies on a large variety of technical textiles, such as light-weight, yet stiff carbon-fiber-reinforced plastics. We consider a common and simple aerospace scenario as illustrated in Fig. 1. Manufacturer **A** produces a light-weight carbon-fiber wing profile **R-001**, manufacturer **B** produces airscrew **PX9**, both collecting manufacturing process information along the process. Manufacturer **C** assembles an airplane **A1-001**, employing wing profile **R-001** sourced from **A** and airscrew **PX9**, sourced from **B**. **C** further conducts regular maintenance work on airplane **A1-001** throughout its lifetime, collecting corresponding maintenance data throughout its lifetime. Although material and workpiece identifiers within individual companies are usually standardized and production process data are often collected locally, individual resources are typically allocated to a specific cost center within the company's management and ERP system and cannot be easily linked to information in external systems, e.g., about which product, workpiece, or application may have been used during the manufacturing process. When, e.g., **C** buys **R-001** from **A**, existing manufacturing data, such as collected by **A** during the production process, is seldomly or only insufficiently passed on. Additionally, data collected during later stages of the product lifecycle, such as the maintenance data collected by **C**, is typically not passed back throughout the supply chain although it may serve as a valuable tool, e.g., in the context of wear and fatigue analysis of parts and products. Today, especially product quality data is mostly shared on paper and typically discarded after respective quality checks have been passed. Additionally, the quality of fiber-reinforced products is typically controlled only after post-processing

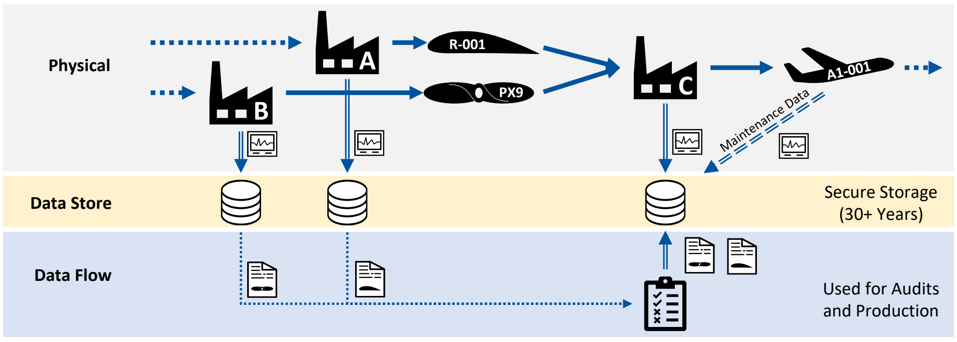

Fig. 1: Aerospace use case scenario: Manufacturer **A** produces wing profile **R-001** and Manufacturer **B** airscrew **PX9**. Both collect quality data as part of their respective certification requirements during the process. Manufacturer **C** acquires the parts, employing them in the manufacturing of plane **A1-001**, keeps all related certification data and maintenance records in secure storage for 30+ years to comply with legal regulations.

is finished and changes are no longer possible [Me12]. A product is then either certified for the intended application during quality control or scrapped, while it may be perfectly reusable for other subsequent applications.

Similar scenarios can be described for other domains in industry [Da19; Ni20; Pe19c; Pe20a; Pe20b]: Today, the collection, exchange, and preservation of data is frequently limited by the overhead cost of this data management (or fears of a loss of control over valuable data) while exactly the same data could be used for subsequent manufacturing process optimizations. Thus, an interoperable and principled data management and preservation system is needed to reduce data management overheads and therefore the associated costs and risks.

Principal Requirements. Data management and preservation for Industry 4.0 must enable the integration, exchange, and preservation of a wide variety of different types of data from all kinds of information systems employed throughout both the automation pyramid and the product lifecycle. Building upon the FAIR principles [Wi16] of scientific data management, an implementation should notably ensure that data is findable, accessible, interoperable, and reusable. Recent work has argued that these principles are equally applicable for data exchange throughout the supply chain and in Industry 4.0 [Gl20c]. For the realization of these principles, a number of specific *services* that need to be provided by any data management solution have been identified in prior work [GD20a; Gl20c; Hu00; Wi16], notably including:

1. **identification**, enabling globally unique and reliable referencing and citation of resources,
2. **versioning**, ensuring the immutability of individual resource revisions to avoid

references from becoming incorrect due to content changes and enable change monitoring and state synchronization,
3. **persistence**, allowing individual resource revisions to be archived and persistently identified through so-called *persistent identifiers* (PIDs),
4. an **access** mechanism for resource retrieval and modification which should be open, free, and universally implementable,
5. **discovery** mechanisms, to make resources, metadata, and archives findable, and
6. accurate **metadata**, to ensure interoperability and reusability through clear semantics, e.g., by keeping track of data *provenance*, i.e., information about data origins, influences, and evolution over time.

To address these requirements, Gleim et al. [Gl20c] recently proposed the FactDAG data interoperability model, for which we present a suitable implementation in this paper. Importantly, this implementation should further: provide the identified services in a manner that ensures **backward-compatibility** with existing resource oriented infrastructure and patterns as far as possible, be optionally adoptable, provide **interoperability** across software vendors and domains, employ non-proprietary, free, universally implementable and established **standards** whenever possible, and incur low overheads—both technical and otherwise—to support **sustainability**.

Contributions. To provide this implementation, we propose *FactStack*, an interoperable approach to data management and preservation based upon a novel combination of existing Web standards and tightly integrated with the HTTP protocol itself. Thereby, we directly realize the FactDAG data interoperability layer model, which we proposed in prior work [Gl20a; Gl20c], in a FAIR and interoperable manner, independent of individual software solutions and backward-compatible with existing resource oriented architectures. FactStack digitally supports data management throughout the full data lifecycle [Ba12; Co19] and directly integrates data management into the technology stack of the Web, instead of just using HTTP as an access mechanism. We further provide an open-source reference implementation of this approach, paving the way for its rapid adoption by the community and, subsequently, the proliferation of best practices for data management and preservation in digital manufacturing and supply chains. We demonstrate the scalability of the system to high-throughput applications and qualitatively highlight its applicability to industry using a real-world use case in the aerospace domain.

Paper Organization. The remainder of this paper is structured as follows. Sect. 2 provides an overview of related work and fundamental technologies. Sect. 3 conceptualizes our data management and preservation system, based upon open and standardized Web technologies. Sect. 4 then describes FactStack, our open-source implementation of this system, and evaluates its performance, before we discuss the impact of the proposed solution for data management and preservation in Sect. 5. Finally, we conclude our work in Sect. 6.

2 Related Work and Foundational Web Technologies

As related and foundational work, we first introduce existing data management solutions and the specific data characteristics and infrastructure requirements in the context of Industry 4.0. We then detail how Web technologies and standards provide fundamental primitives and building blocks for the realization of interoperable data management. We discuss essential aspects of interorganizational interoperability and outline the role and importance of provenance information for reliable data reuse. Finally, we summarize the FactDAG data interoperability layer model [Gl20c] as the theoretical foundation of our practical data management solution.

The authors further conclude, that existing systems are typically lacking semantic enrichment of data, e.g., using Semantic Web technologies [BHL01], which allows for them to be shared and reused across application, enterprise, and community boundaries, and enables the creation of machine-actionable knowledge. Based on their success in the realization of scalable, interoperable, and extensible enterprise solutions [Bl13], Web technologies are already integral components of many existing data management systems (such as the aforementioned). In combination with Semantic Web technologies, they provide a promising basis for the development of interoperable and sustainable data management solutions for Industry 4.0 and the Web [GD20a].

Web Technologies for Interoperability. Interoperability in the Web is based on a number of fundamental standards, notably including: the global Domain Name System (DNS) [Mo87], the HTTP protocol and its implementation of the Representational State Transfer (REST) architectural pattern [Fi00], the Uniform Resource Identifier (URI), as well as its directly resolvable incarnation, the Uniform Resource Locator (URL) [BFM05]. Building on top of these foundations, Linked Data and the Semantic Web enable data interoperability on the Web. Using the Resource Description Framework (RDF) [WLC14] data model and its serializations and enable machine-to-machine data interchange, the semantic enrichment of data, and the ability to interlink data across organizational boundaries. Deploying these standards supports interoperability. Notable standardized extensions of the basic HTTP protocol for distributed data exchange and management on the Web include (i) the Linked Data Platform (LDP) standard [SAM15], and (ii) the HTTP Memento protocol [VNS13]. The **Linked Data Platform** defines how Linked Data resources can be read and written using HTTP REST methods, i.e., HTTP `GET`, `POST`, `PUT`, `PATCH`, and `DELETE`. Besides resource *access*, the LDP enables the creation of containers that can be used to organize resources and to express relationships between them. Thus, it enables simplified resource *discovery*, as well as providing a mechanism to provide a dedicated metadata record for arbitrary Web resources using the HTTP `rel="describedby" Link` header. Using the LDP protocol, Linked Data and Web resources can be managed similarly to regular files in a local file system while enabling the augmentation of arbitrary resources with semantic *metadata*. A detailed introduction to the LDP can be found in [SAM15]. The **Memento protocol** introduces a mechanism to manage and retrieve persistent versions of Web resources by using timestamps as a resource version indicator and access key. Resource versions may be redundantly stored on multiple servers and managed independently of each other, enabling sustainable and distributed resource archiving [VNS13]. The Memento protocol provides primitives to address resource *versioning*, *persistence*, *access*, and *discovery*. As such, prior work already suggested the Memento protocol as a promising candidate for the implementation and standardization of data management and preservation systems [GD20a; Va14; Va18]. A detailed overview of the Memento protocol is provided in [VNS13].

Interorganizational Interoperability. An important factor limiting the adoption of interorganizational data exchange is uncertainty about the reliability of data, accountability, and liability questions for damages incurred by inaccurate data [Pe19a]. To this end, the concept of *data provenance* plays an important role in the realization of trust, accountability, and better interpretability of data and the processes that lead to their creation in collaborative manufacturing and supply chain systems. The term provenance, sometimes also called data lineage, refers to metadata regarding the formation history, origins, and influences that impacted the state of individual data. An open and extensible standard for provenance data is the W3C PROV data model (PROV-DM) for provenance interchange on the Web [MM13]. A primer on this model and its primitives can be found in [Gi13]. Provenance records are, e.g., successfully employed to build and analyze scientific workflows through process mining [Ze11], to ensure the reproducibility of such workflows [Ko10], to establish trust across heterogeneous sources of data [LLM10] and to further data reuse [Yu18]. Provenance data is special, in the sense that it is metadata that is relevant and collectible for practically any kind of resources and directly relates to the data authoring and management process. As such, it may serve as a generic kind of interoperable 'glue', relating resources throughout their formation history.

FactDAG Model. The conceptual *FactDAG* data interoperability model proposed by Gleim et al. [Gl20c] similarly employs data provenance to interlink resources and data throughout supply chain processes and in Industry 4.0. By using a persistent identification mechanism called *FactID*, FactDAG simultaneously addresses the requirements of *identification*, *versioning*, and *persistence*, constructing persistent identifiers from unique triples of global authority ID, internal resource ID and respective revision ID of a given immutable resource revision, also referred to as a *Fact*. The model further employs *Authorities*, entities (e.g., companies or organizations) that are responsible for Facts, *Processes*, which describe prototypical interactions with Facts, and *ProcessExecutions*, which refer to their instantiations and are introduced to capture individual influences and results (i.e., newly created Facts or Fact revisions). Additionally, a single relation (called *influence*, oriented forward in time) is used to express provenance relations between the elements of the FactDAG, thus constructing a provenance-based, directed acyclic graph of *Facts*, the *FactDAG*. Thus, the model allows tracing back the origins of Facts throughout time, revealing the resources, authorities, and processes involved in its conception and throughout the data management process. By globally and persistently identifying immutable revisions of resources, the model allows for information to be reliably referenced in global collaboration scenarios. The deep incorporation of provenance information into the model provides companies

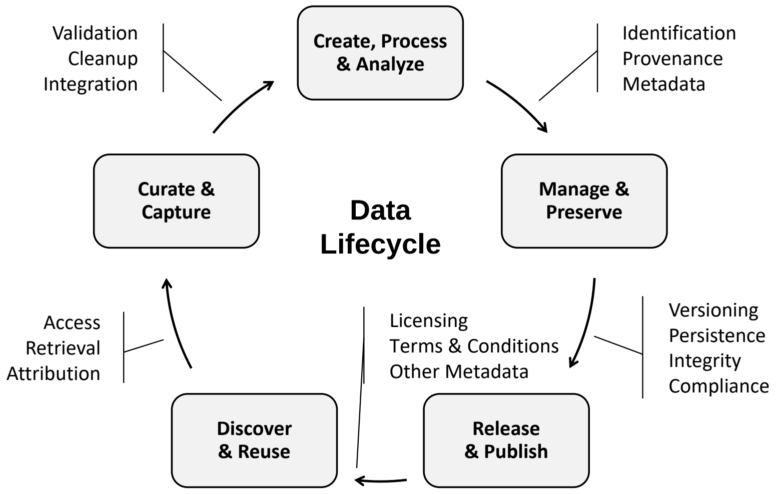

Fig. 2: The data management lifecycle consists of five steps providing an abstract model for data management processes, both in industry and academia. Adapted from [Ba12; Co19].

with a solid base of relevant metadata for the establishment of accountable, reliable, and sustainable data integration, even in interorganizational scenarios. For additional details, we refer to the specification of the FactDAG model [Gl20c].

While we believe that the abstract FactDAG model provides a promising basis for the implementation of a data management and preservation system for Industry 4.0, it lacks both a concrete implementation, as well as a principled integration with best practices of data management to date. Thus, we propose a concrete implementation concept based on the fundamental data management lifecycle in the following section.

3 A Concept for Interoperable Data Management and Preservation

Data represent corporate assets with potential value beyond any immediate use, and therefore need to be accounted for and properly managed throughout their lifecycle [Fa14]. Various *data lifecycle models* [Ba12; Co19] have been proposed in recent years to serve as a high-level guideline for the data management process—from conception through preservation and sharing—to illustrate how data management activities relate to processes and workflows, to assist with understanding the expectations of proper data management and to ensure that data products will be well-described, preserved, accessible, and fit for reuse. The recurring elements of such models can be summarized in a five-step data management lifecycle model as illustrated in Fig. 2, consisting of the steps: (i) creation, processing, modification, and analysis, (ii) metadata management and data preservation, (iii) release and publishing of data, including proper licensing, documentation, etc., (iv) discovery for reuse of available data, and (v) the retrieval, curation, and capture of this data for subsequent processing. In the following, we consistently refer to these steps using the names provided in Fig. 2. Traditionally, data management infrastructure is mainly employed as a kind of mediating service between the Release & Publish and the Discover & Reuse phase of the data lifecycle, while the remaining steps are carried out independently by human actors. In contrast, we aim to support the full data lifecycle process, integrating it directly with the fundamental infrastructure of the Web.

Technical Approach. Based on the principles of the FactDAG model [Gl20c], we strive to realize an interoperable data management and preservation system for usage throughout the data management lifecycle, the product lifecycle and the full supply chain, which satisfies the requirements identified in Sect. 1. As already motivated in Sect. 2, this system should build upon existing open Web standards whenever possible, ensuring compatibility and interoperability with existing systems and deployed solutions, as well as profiting from an ecosystem of developers with corresponding proficiency [St20] and the wide variety of available authentication and authorization mechanisms [TCS18]. The implementation should allow for incremental adoption, enabling the management and preservation of existing data according to the principles of the FactDAG model in an ad-hoc, on-demand fashion. Additionally, it should be backward-compatible, allowing clients that have no use for, do not support, or are unaware of data management principles in general, to simply ignore all related additional information. Provenance data should be collected and processed automatically whenever possible, especially during the Curate & Capture and Manage & Preserve phases, to minimize the amount of explicit markup and metadata management required and prevent easily avoidable user errors.

For the realization of these goals, we build upon two recent proposals by Gleim et al.: a PID system employing dated URIs in conjunction with a resolution mechanism based on the HTTP Memento protocol [GD20b] (addressing aspects of data *identification*, *versioning*, *persistence*, and *access*), as well as an alignment of FactDAG provenance with the W3C PROV standard for provenance information [Gl20d] (addressing aspects of data *provenance* and *discovery*). In the following, we outline and extend upon these proposals, integrating them with the W3C Linked Data Platform and other Web standards to form a comprehensive data management solution, addressing all requirements as formulated in Sect. 1.

3.1 FactID: Time-based Persistent Data Identification

To fulfill the *identification*, *versioning*, and *persistence* requirements as defined in Sect. 1 within the FactDAG model, a suitable persistent resource identification mechanism for the implementation of the FactID scheme is needed. As mentioned in Sect. 2, a FactID consists of the three components authority, internal resource ID and revision identifier. Inspired by the original FactID proposal [Gl20c], we map all three components to a single URI to enable backward-compatibility with the Web infrastructure, as follows:

Authority `auth`. All data in the FactDAG model is placed under the exclusive and authoritative control of an organizational body, as identified by its global (but not persistent) authority ID $auth \in \mathit{Authority}$. We map *Authority* to the set of all DNS domain names [Mo87].

Internal ID `iID`. All resources available under the control of `auth` are identified by their respective internal resource ID $iID \in \mathcal{P}$. We map $\mathcal{P}$ to the set of all URI Paths [BFM05] (including query and fragment suffixes). Combining `auth` and `iID` in a tuple creates a global (but not persistent) resource identifier, which we practically materialize as traditional HTTP URLs of the form `http://`*auth*/*iID*.

Revision ID τ. Individual resource revisions are further identified by their respective revision ID $\tau \in \mathcal{T}$. We map $\mathcal{T}$ to the set of all RFC3339 [NK02] arbitrary precision UTC timestamps. While other revision identification mechanisms (such as content hashing) are conceivable, we specifically employ UTC timestamps due to their globally agreed-upon semantics. Timestamps are further unaffected by content-variations (e.g., due to content-negotiation) and allow for the intuitive ordering of resource revisions and their direct interpretation as time series data. Subsequently, the triple $(auth, iID, \tau) \in \mathit{Authority} \times \mathcal{P} \times \mathcal{T}$ yields a persistent global identifier – a FactID – for the immutable state (i.e., revision) of the resource identified by the tuple $(auth, iID)$ at the point in time τ. We employ the term *Fact* to refer to this immutable data state.

FactID URI Scheme. Many PID approaches require the assignment, registration, and management of PIDs outside of the Web infrastructure and already existing URL identifiers. This causes overhead for identifier mapping and discovery [Va14]. Thus, Gleim et al. [GD20b] proposed a system capable of reusing existing URLs as PIDs by combining dated URIs (for identification) with an HTTP Memento-inspired resolution mechanism (for versioning and persistence). We employ this approach to realize a URI scheme for FactID through the following mapping:

While it is possible and common practice to resolve specific resource versions through URLs including HTTP query parameters, such an approach is hard to standardize in a backward-compatible manner. While a URL of e.g., the form `http://`*auth*/*iID*/`?v=`τ may be employed to uniformly express a persistent identifier according to the semantics of the FactID, the query parameter `v` is likely already used with different semantics in other contexts, creating the potential for naming conflicts. To avoid this problem, we adapt Larry Masinter's '`duri:`' *dated URI* proposal [Ma12] for the identification of specific resource revisions, resulting in the '`factid:`' URI scheme: A FactID of the form `factid:`τ`:http://`*auth*/*iID* persistently identifies the immutable state of the resource `http://`*auth*/*iID* at the point in time τ, also referred to as a *Fact* or *Memento*. We refer the interested reader to Masinter's RFC proposal [Ma12] for an additional discussion of the benefits and implications of employing dated URI.

HTTP-based Data Retrieval. To materialize and implement a practical resolution mechanism for such a FactID, we employ the HTTP Memento protocol as an *access* mechanism.

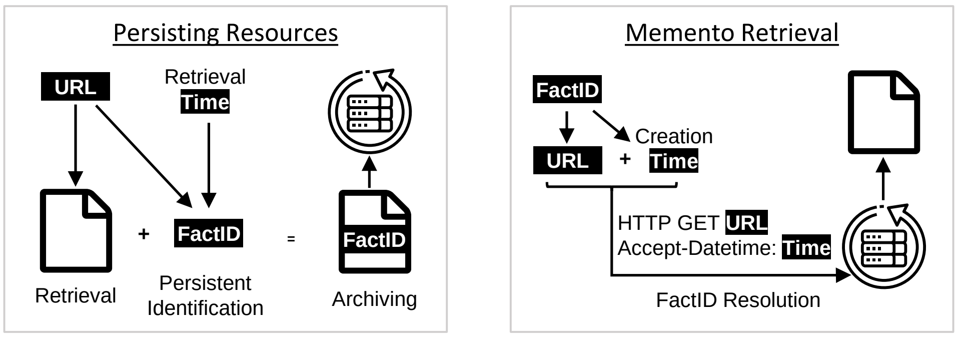

Fig. 3: Persisting and retrieving data using FactID. A FactID uniquely identifies a Fact (i.e., an immutable resource revision) by combining its URL with a timestamp. Such a FactID can be used to retrieve that Fact via the Memento protocol. Adapted from [GD20b].

Given a fixed ID, the resolution function $res : \mathit{Authority} \times \mathcal{P} \times \mathcal{T} \rightarrow \mathit{Fact}$ (with *Fact* as the set of all Facts) retrieves the Fact f identified by a given FactID through HTTP datetime negotiation via the HTTP Memento protocol [VNS13]. To maintain backward-compatibility, res defaults to resolving the current state of the resource identified by the tuple $(auth, iID)$, i.e., the URL `http://`*auth*/*iID*, if no revision ID is provided. The current resource state, as resolved via HTTP, may be lifted to a Fact by a consumer through incorporating the current timestamp as revision ID, as outlined in the original Fact construction procedure in [Gl20c] and illustrated in the left half of Fig. 3. This way, it is possible to enable the on-demand incorporation of persistence into existing systems implementing REST semantics. Given such a `factid`, the original resource state may then be retrieved from an archive through an HTTP GET request employing the Memento `Accept-Datetime` header with the Memento's creation time as specified in the τ part of the `factid` as depicted in the right half of Fig. 3. An overview of the different retrieval patterns and further features supported by the Memento protocol is given in [VNS13].

Data Persistence. As postulated by Kunze and Bermes [KB19], persistence (and analogously immutability) is purely a matter of service. It is neither inherent in an object nor conferred on it by a particular naming syntax but only achieved through a provider's successful stewardship of resources and their identifiers. Since the architecture of the HTTP Memento protocol "is fully distributed in the sense that resource versions may reside on multiple servers, and that any such server is likely only aware of the versions it holds" [VNS13], the service of data *persistence* may subsequently be provided by the authoritative data source, by any data consumer, by third parties such as governmental institutions or archiving providers or any number thereof, as long as volitional and legally permitted. By allowing for the *discovery* and retrieval of immutable data revisions over time, the Memento protocol further enables state synchronization between storage and archive locations as the data changes over time, thus additionally supporting redundant storage, e.g., for long-term data preservation. A detailed discussion of these applications may be found in [GD20b].

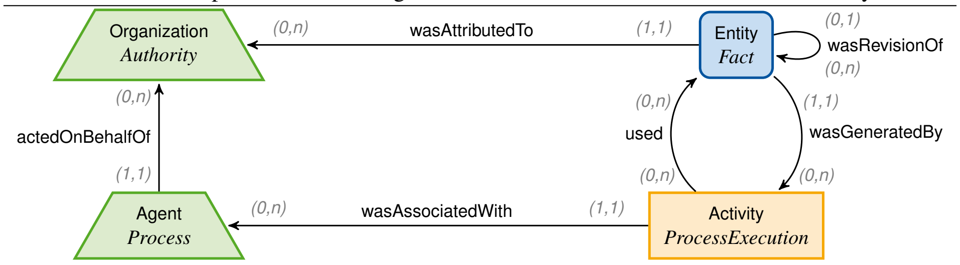

Fig. 4: The elements of the FactDAG model (in italic) and their provenance relations expressed using PROV-O primitives with corresponding *(min,max)*-cardinalities [Ab74]. The shapes represent the PROV core classes Entity, Activity, and Agent (with Organization as a subtype), respectively. Adapted from [Gl20d].

3.2 Automated Provenance Annotation and Distribution

As already discussed in Sect. 2, *provenance* is a particularly important category of *metadata* for data management. In alignment with the principal requirements identified in Sect. 1, we strive to minimize the metadata management overhead, by collecting *provenance* information automatically whenever possible. To ensure *interoperability* with existing tooling and reuse existing *standards*, we employ the recently proposed alignment [Gl20d] of FactDAG provenance to the W3C PROV-O ontology standard [LSM13]. The mapping, illustrated in Fig. 4, thus allows for the expression of FactDAG provenance information as RDF metadata. For any given Fact f with FactID $fID = (auth, iID, \tau)$, the following provenance relation may be directly derived:

- f is an instance of the `prov:Entity` class.
- $auth$ is an instance of the `prov:Organization` class.
- f is `prov:wasAttributedTo` its authoritative source $auth$.
- If f is a direct revision of predecessor Fact f', then f `prov:wasRevisionOf` f'.
- f is a `prov:specializationOf` its respective original resource, identified by the URL derived from the tuple $(auth, iID)$.

Additionally, information about any `prov:Entity` which was `prov:used` or `prov:wasGeneratedBy` a given `prov:Activity` may be automatically collected through the usage of a runtime library for *Fact* management, which we detail in Sect. 4. Nevertheless, further metadata and provenance information may have to be collected manually and can be added using RDF-compatible metadata vocabularies or other domain-specific ontologies, following the FAIR principle that metadata shall use a formal, accessible, shared, and broadly applicable language for knowledge representation and be described with a plurality of accurate and relevant attributes [Wi16].

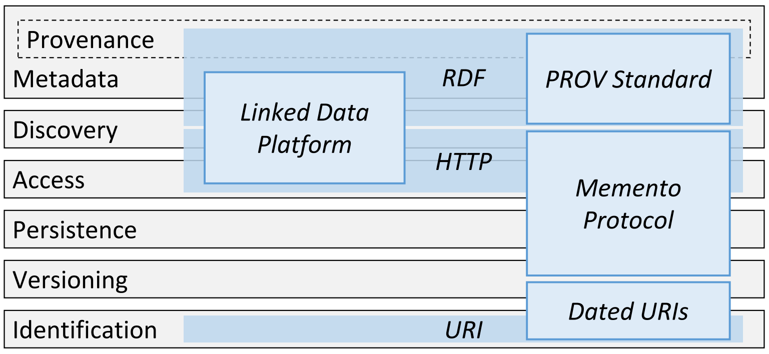

Fig. 5: The novel combination of existing protocols and Web standards provides a unified data management solution, addressing all principal requirements identified in Sect. 1.

3.3 Fact Discovery and Creation

The final missing conceptual component is a standardized discovery and read/write mechanism for data resource management. Due to its potential for interoperability with existing HTTP REST APIs and conceptual simplicity, we implement the Linked Data Platform specification [SAM15] for this task. Thus, resources (as identified by their respective HTTP URLs of the form `http://`*auth*/*iID*) can be organized in a hierarchy of LDP Containers within their authoritative source *auth*, enabling for resource discovery within it through exploration. In contrast, resource creation and modification are handled through the specified LDP HTTP REST methods.

To support a wide variety of structured, semi-structured, or unstructured data formats, including binary blobs of arbitrary file-type, the LDP specification further provides the option to augment Non-RDF LDP resources with a respective RDF metadata resource, linked through the HTTP `rel="describedby" Link` header, which we employ in practice, to store provenance information and further metadata. Both data and metadata can then be discovered through one single URL (or FactID respectively) and retrieved via HTTP.

Overall Concept. By combining dated URIs, the HTTP Memento protocol and the Linked Data Platform standard with PROV-O provenance and extensible RDF metadata, we ultimately propose a concept for semantic data management directly based on core technologies of the Web—URI, HTTP, and RDF—as illustrated in Fig. 5. By considering resource *versioning* and *persistence* as additional service layers of the basic Web technology stack and implementing them as extensions of URI and HTTP, we ensure *backward-compatibility* and *interoperability* with existing resources on the Web. By reusing existing *standards* where possible, and enabling on-demand resource versioning and persistence through consumers and third parties, the concept effectively addresses all requirements identified in Sect. 1, providing a promising foundation for its long-term *sustainability*.

Based on this concept, we present our reference implementation of an interoperable data management and preservation solution for the Web and Industry 4.0.

4 FactStack: A Concrete Realization of the FactDAG Model

Following the conceptual approach presented in Sect. 3, we realize *FactStack* as a concrete open-source implementation of the FactDAG model for interoperable data management and preservation. Based upon the basic REST paradigm at the core of the Web, FactStack employs standardized and open Web technologies to provide a uniform data management API for arbitrary data resources on the Web, while enabling persistent data preservation through the HTTP Memento protocol.

Our realization consists of three open-source components, available for practical usage: A server component[1], adapted from the Trellis LDP project and implementing the LDP and Memento protocols for data storage and management, a JavaScript client library[2] simplifying both the interaction with the LDP server and the management of data provenance, as well as an optional broker, which enables real-time subscriptions to changes of LDP resources, i.e., newly created data revisions.

Data Storage. For the realization of a Fact authority, we employ a data storage server[1] based on the LDP implementation of the Trellis open-source project[3]. Trellis provides both direct integrations with a number of freely available storage backends, as well as the ability to integrate with existing information systems as its data store. The project further implements the HTTP Memento protocol [VNS13] for resource versioning, which we adapted to support RFC3339 [NK02] timestamps with up to nanosecond precision, as per the recent proposal of Gleim et al. [GD20b]. By default, all resources are identified by traditional URLs of the form `http://`$auth$`/`iID, their Mementos by corresponding FactIDs and revisions managed through the Memento protocol. To enable backward-compatible linking to Facts with standard URLs and resolution over plain HTTP, the server assigns an additional unique Memento URL $URL\text{-}M$ (cf. [VNS13]) of the form `http://`$auth$`/`iID`/?v=`τ to each Fact. Finally, we implemented Memento headers to also be returned in response to LDP PUT and POST requests, as proposed by [GD20b], avoiding race conditions between competing resource updates and Memento header retrieval, thus ensuring efficient atomic resource updates.

Data Management. To guide the data management process in client applications, the *FactLib.js* library[2] mirrors the data lifecycle (cf. Fig. 2) in code. Facts are retrieved or created within the context of an activity and are subsequently registered as *used*, respectively *generated by* this activity, i.e., automatically recorded as corresponding provenance links. The library further handles the transparent and unified retrieval and creation of both RDF

[1] Available at: `https://git.rwth-aachen.de/i5/factdag/trellis`
[2] Available at: `https://git.rwth-aachen.de/i5/factdag/factlibjs`
[3] `https://www.trellisldp.org/`

and Non-RDF Facts and their respective metadata, as well as automatically adding collected and inferred provenance information (cf. Sect. 3.2) as RDF metadata using the W3C PROV standard. For RDF resources, the provenance information is directly part of the resource stored in the LDP and can be found and retrieved by all clients that resolve the FactID to that resource. For binary resources, the Factlib.js library automatically discovers and manages metadata through the HTTP `rel="describedby" Link` header (cf. Sect. 3.3), retrieves it via HTTP and delivers it to the client application as part of the Fact. While authorities only store and provide access to Facts under their own control, clients can read and write from and to resources associated with different authorities. Clients may learn about Facts under the control of third-party authorities, e.g., by following provenance links (i.e., traversing the FactDAG), through explicit membership links provided by the LDP implementation or through other generic RDF triples or links. Each authority server may employ its own authentication and access authorization mechanisms, as well as providing its own data licensing terms, in order to maintain control over access to its data. Once a client successfully retrieved a resource, they may optionally (if legally allowed) archive it with any number of external Memento archiving providers (such as their own organization's) to serve as long-term persistence providers for arbitrary Facts. This distributed and usage-based archiving mechanism allows for flexible and use case driven trade-offs between persistence guarantees and associated costs, further contributing to the long-term *sustainability* of the data management approach as a whole. Retrieving Facts from third-party archives does, however, raise associated questions regarding authenticity and integrity, which we plan to consider in future work.

WebSocket Subscriptions. Since many applications in Industry 4.0 may profit from push-based real-time updates of changes to resources, e.g., to react to events with low latency, a useful, practical feature consists of subscription support. Whenever new revisions of resources are created, a subscribed client receives a corresponding notification. Since for any pair of authority ID and internal ID, a series of data revisions could exist over time, all data within the FactDAG model is effectively *time-series data*. As such, every data point (as identified by authority and internal ID) is a stream of Facts and processing of facts is stream processing, which may, in turn, result in new Facts. To implement subscription support, we employ a broker-based approach to communicating change notifications in *Activity Streams 2.0* [SP17] format using a *STOMP*[4] message encoding and a WebSockets [MF11] transport.

Performance. Finally, we conduct a performance evaluation of a single-node deployment of the server application on a workstation with Intel i7-8700K CPU, 64 GB of RAM and NVMe SSD. We configure Trellis to store Mementos in the file system and employ a local Apache ActiveMQ Artemis[5] broker to support resource subscriptions via its Stomp over WebSockets implementation. We measure the average response time for Fact creation using HTTP PUT requests under different loads via Apache JMeter[6], as well as the average

[4] `https://stomp.github.io/stomp-specification-1.2.html`
[5] `https://activemq.apache.org/components/artemis/download/release-notes-2.14.0`
[6] `https://jmeter.apache.org/`

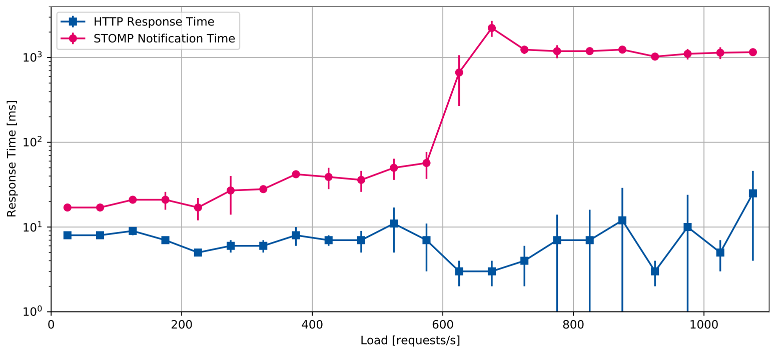

Fig. 6: Response times for random Fact creation under different loads as well as the time the system needs to notify a subscriber of the changed data (with 99 % CI). The results indicate sufficiently low and stable average response times for throughputs of up to 1000 insertions/s, as well as for notifications to the subscriber for throughputs up to 600 insertions/s.

time until a change notification is received back by a subscriber which consists of a basic collection script based on stomp.py[7]. To simulate random access to resources, 100 000 different resources are initially created containing five RDF triples each, as may be expected for small resources, such as single sensor values. The local JMeter client application issues PUT request to the server to update resources randomly chosen from this pool to apply the desired load and only starts the measurement of the average response times after an initial warm-up period. The stomp.py script subscribes to all resources with the ActiveMQ broker and records the timestamps of received notifications and the associated resource. The notification time is computed afterward, by comparing the timestamp of the request with the recorded timestamp of the collection script. To guarantee independent measurements, the whole system including the stored data is reset after each measurement.

The results plotted in Fig. 6 indicate a relatively stable average response time of around 10 ms for throughputs of up to 1000 insertions per second. For a throughput up to approximately 580 requests per second, the average response time is between 6 ms and 12 ms and the subscriber receives the notification in under 60 ms after the response confirming the resource modification is received by the sender. Around the 600 requests per second mark, the performance of the ActiveMQ broker deteriorates significantly, stabilizing at a notification latency of roughly 1 s for 680 requests per second and above. We attribute this behavior to the performance of the collector. If the collector cannot keep up with the processing of the incoming messages, the broker performance may be significantly reduced, as documented by the ActiveMQ project.[8] Therefore, the maximum achieved load does not indicate the maximum capacity of the broker, but only the performance with regard to a

[7] `https://github.com/jasonrbriggs/stomp.py`

[8] `https://activemq.apache.org/components/artemis/documentation/latest/slow-consumers.html`

single collector, which may be achieved by multiple collectors independently. In a practical scenario, a client would not typically subscribe to all resources, but only to the subset of those resources that are relevant to its immediate use case application. Thus, the ability to process up to 580 change notifications per second on a single client already provides a sufficient capacity for many practical scenarios. In order to expand the overall capacity beyond 1000 requests per second, e.g., for scenarios with multiple data producers and high update frequencies, users may instead profit from the horizontal scalability of Trellis LDP's server architecture. In future work, we further plan to evaluate the performance of different data persistence backends and potential alternatives to Trellis for additional performance optimization.

With FactStack, we provide a concrete, open-source implementation of the FactDAG model for interoperable data management and preservation, facilitating the rapid adoption and evaluation by the community. After demonstrating the system's scalability to high data-throughput scenarios, we discern its practical value for data management in Industry 4.0 in the following.

5 Applying the FactStack to Data Management in Industry 4.0

To ascertain FactStack's value for practical application scenarios, we illustrate its data and control flow by mapping it to the research data management lifecycle presented in Fig. 2, resulting in the workflow shown in Fig. 7.

Starting with the Release & Publish phase, data is made available as resources on the Internet through regular HTTP Web services, each its own Authority identified by its domain name. In the Discover & Reuse phase, these resources may then be discovered either through existing Web indexing and information retrieval techniques, by exploring the LDP resource hierarchy of a given Authority, or by following the fundamental provenance relations of the FactDAG.

To reuse a discovered resource R with URL U, a consumer retrieves it via HTTP as part of the Curate & Capture phase and persistently identifies it using the mechanism describe in Sect. 3.1. Therefore, the consumer checks for the presence of an HTTP `Memento-Datetime` header upon retrieval of R to determine if the resource already is a Memento. If so, the resource is already persistently identified by FactID `factid:`$\tau_M : U$, where τ_M is the Memento's creation time as indicated by the `Memento-Datetime` header. Otherwise, the resource is lifted to a Fact as described on the original FactDAG paper [Gl20c], persisting the state of R at the time of retrieval τ_{cur} as a new Memento identified with FactID `factid:`$\tau_{cur} : U$. The consumer may optionally opt in to archive an immutable copy of this persistently identified Fact in an archiving service of its choice, e.g., under its own control, or potentially at a third party or regulatory institution, ensuring the reliable preservation of consumed resources. If a client requests a non-Memento resource multiple times and is able to validate that the resource did not change in the meantime (e.g., through a strong HTTP

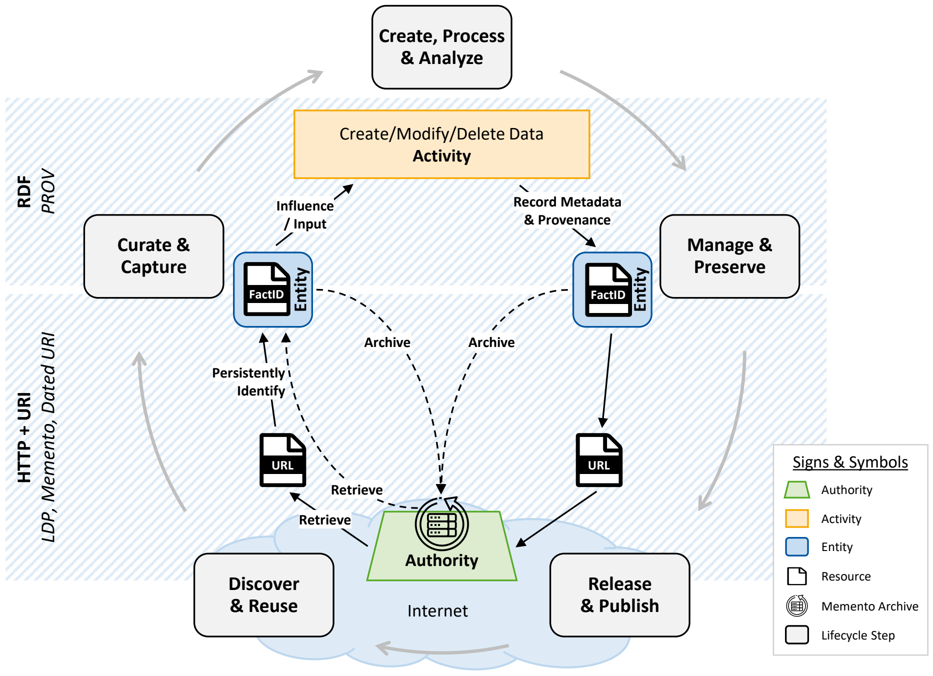

Fig. 7: The FactStack data management and preservation lifecycle provides a mapping from the FactStacks data and control flow to the research data management lifecycle presented in Fig. 2 using a combination of open Web standards.

ETAG provided by the server), it may choose to either reuse the previously created Fact or to create an additional Memento analogously to the previous procedure.

During the Curate & Capture phase, an arbitrary number of resources may be collected, validated, cleansed, integrated, and subsequently provided as an input to the Create, Process & Analyze phase, in which data is created, modified, or deleted. To capture this process, it is modeled as a PROV Activity A, capturing all resources used as an input to or resulting from the execution of the activity as corresponding Entities. All Facts S provided as input to A are then related to it using the `prov:used` predicate, while any resources generated in the process are similarly persistently identified and immutably archived and related to A using the `prov:wasGeneratedBy` relation. Notably, if a generated resource S is a new revision of a previously existing resource R, this information is captured using the triple S `prov:wasRevisionOf` R.

During the Manage & Preserve phase, additional metadata may be added to the resource in order to capture more of its semantic context and provenance, while newly created resources are uniquely identified for future reference. Finally, resource and metadata

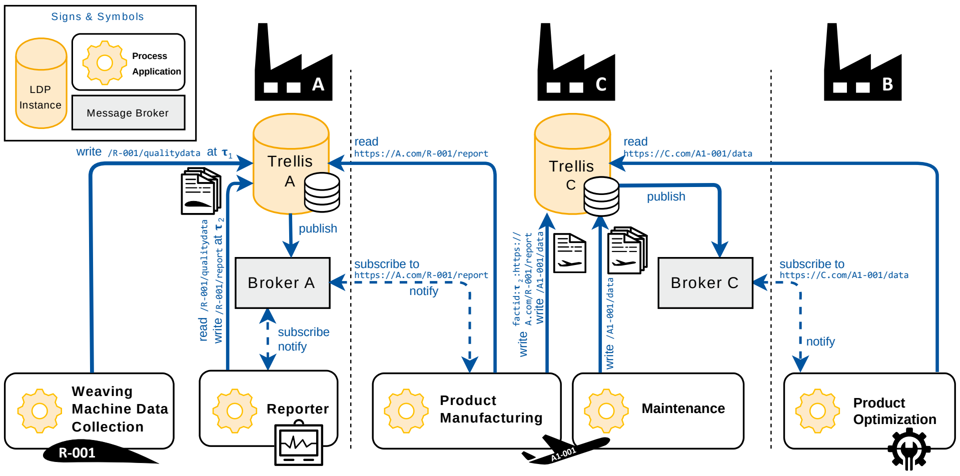
Signs & Symbols
LDP Instance
Process Application
Message Broker
A
C
B
write /R-001/qualitydata at τ_1
Trellis A
read https://A.com/R-001/report
publish
Broker A
subscribe to https://A.com/R-001/report
notify
read /R-001/qualitydata
write /R-001/report at τ_2
subscribe
notify
Trellis C
read https://C.com/A1-001/data
publish
write factid:τ_2:https://A.com/R-001/report
write /A1-001/data
write /A1-001/data
Broker C
subscribe to https://C.com/A1-001/data
notify
Weaving Machine Data Collection
R-001
Reporter
Product Manufacturing
A1-001
Maintenance
Product Optimization

Fig. 8: To exemplify, FactStack enables continuous data sharing along the supply chain.

(including any other applicable information such as licensing terms, etc.) are then stored together, identified by a single PID, during the Release & Publish phase, creating a new Memento or Fact in the process. Herein, the metadata may either be merged directly into the primary data – such as possible with RDF sources – or by adding it to the resource's LDP Metadata resource accessible using HTTP content-negotiation or discoverable through HTTP `rel="describedby" Link` header (cf. Sect. 3.3), which enables structured RDF metadata to be stored for other text or binary file formats.

As all newly generated data are now persistently identified, immutably preserved, published, and discoverable on the Web, a full data management lifecycle was completed.

Use Case Application. Revisiting the use case example introduced in Sect. 1 and visualized in Fig. 1, we can now illustrate the concrete impact of data management and preservation using FactStack in Fig. 8. In this scenario, manufacturer **A**, identified by its authority domain name **`A`**`.com`, collects manufacturing data as part of the production of workpiece **R-001**, which it stores identified with internal resource ID `/R-001/qualitydata` in its data storage system Trellis A, creating a corresponding Memento at the point in time τ_1. The internal Reporter process analyzes this data and creates a quality report with internal resource ID `/R-001/report` for this workpiece at the point in time τ_2, certifying the part for usage in aerospace applications and again creating a corresponding immutable Memento. Company **C**, identified by its authority domain name **`C`**`.com`, now acquires workpiece **R-001** and retrieves the associated certification report Memento, recording its persistent FactID `factid:`τ_2`:https://`**`A`**`.com/R-001/report`. **C** then stores a copy of this immutable Fact in its own data storage system Trellis C, where the Fact is still identifiable and retrievable through the standardized Memento protocol, using its original FactID, even if **A** deletes its copy or goes out of business. Even if a Fact becomes *globally* unavailable, knowing its FactID still allows for the derivation of basic metadata (cf. Sect. 3), as required by the FAIR data principles. Additionally, **C** subscribes to the resource `https://A.com/R-001/report` to be automatically notified of any future updates to the resource, allowing for immediate reaction to change. **C** further aggregates and maintains all data related to airplane **A1-001** (both from Product Manufacturing and later Maintenance) in the RDF graph resource `https://C.com/A1-001/data`. Airscrew supplier **B** (which in this example does not provide any data using FactStack itself) then subscribes to this resource in order to continuously incorporate the maintenance data collected by **C** in its own Product Optimization process, thus (at least in theory) enabling the continuous improvement of its airscrew product designs.

Following FactStack's data management lifecycle as illustrated in Fig. 7, metadata about each Process Execution in the described use case scenario is recorded through a corresponding PROV Activity. Links to all used (i.e., read) and generated (i.e., written) Facts (as identified by their corresponding FactIDs) are maintained as part of the RDF metadata of the corresponding generated resources. Subsequently, the origins and influences of any resource managed using FactStack can easily be traced back through the captured provenance relations, even across organizational boundaries and as resources change over time.

Discussion. To summarize, FactStack allows for the integration, exchange, and preservation of any type of data exchangeable on the Web and from any information system complying with the basic HTTP REST interface pattern. By implementing data identification, versioning, persistence, access, discovery, and metadata management through a novel combination of existing protocols and Web standards, it provides a backward-compatible and sustainable solution for data management and preservation. FactStack thus meets the system requirements posed in Sect. 1 and provides a promising solution for the management and preservation of the constantly evolving and diverse data of the Web and Industry 4.0.

Nevertheless, there are also some notable limitations. Although mandated by the FAIR data principles [Wi16], FactStack does not currently register nor index (meta)data in a searchable resource and does not enforce clear and accessible data licensing, nor domain-relevant community standards. Additionally, FactStack's reliance on HTTP and its LDP and Memento protocol extensions can lead to high numbers of HTTP requests when managing data, since neither protocol supports request batching. Especially for resource discovery and RDF metadata management, significant performance improvements could likely be accomplished through the usage of the SPARQL query and update language [PPG13; SH13].

6 Conclusion and Future Work

In this work, we presented FactStack, an interoperable data management and preservation approach for evolving data on the Web and in Industry 4.0. Based upon open and tightly integrated with standardized Web technologies, FactStack realizes the FactDAG data interoperability model approach, providing on-demand support for persistent data archiving, identification, retrieval, and synchronization through an interoperable HTTP API, backward-compatible with existing REST services. By employing dated URIs according to the FactID scheme, we enable the persistent identification of arbitrary Web resources, resolved, managed, and preserved through a combination of the HTTP Memento and Linked Data Platform standards. We further implemented the semi-automated provenance collection with the W3C PROV-O ontology to enable the standard-compliant collection of data and process provenance as Linked Data.

To illustrate FactStack's application in Industry 4.0, we focused on an exemplary, representative use case scenario in textile engineering for aerospace, highlighting corresponding opportunities for improved data management and preservation and interoperability. We support the practical adoption of the FactStack by releasing our implementation, which demonstrated scalability to high-throughput applications in the presented performance evaluation, as open-source software. FactStack promotes best practices for data management by directly supporting the full data management lifecycle and enables the continuous exchange and reuse of data using Web technologies throughout the supply chain and across domains, supporting the establishment of transparency and accountability through adequate and interoperable metadata and provenance management.

For future work, we plan to investigate the integration of the FactStack with existing enterprise resource planning and manufacturing execution systems to showcase FactStack's universality and deployability. Additionally, future work should address related questions of authenticity, integrity, and trust within the FactDAG model, as well as improving the performance of the LDP server. Finally, we plan to implement and evaluate easy to use front-end applications for the intuitive collection of FactDAG data to simplify adoption for end-users and validate its merit in practical data management and preservation scenarios.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

References

[Ab74] Abrial, J.-R.: Data Semantics. In: Proceeding of the IFIP Working Conference on Data Base Management. Elsevier, pp. 1–60, 1974, ISBN: 978-0-7204-2809-4.

[Ba12] Ball, A.: Review of Data Management Lifecycle Models, tech. rep., University of Bath, 2012.

[BFM05] Berners-Lee, T.; Fielding, R. T.; Masinter, L. M.: Uniform Resource Identifier (URI): Generic Syntax, IETF RFC 3986, 2005.

[BHL01] Berners-Lee, T.; Hendler, J.; Lassila, O.: The Semantic Web. Scientific American 284/5, pp. 34–43, 2001.

[Bl13] Bloomberg, J.: The Agile Architecture Revolution: How Cloud Computing, REST-Based SOA, and Mobile Computing Are Changing Enterprise IT. Wiley, 2013, ISBN: 978-1-118-41787-4.

[Co19] Corti, L.; Van den Eynden, V.; Bishop, L.; Woollard, M.: Managing and Sharing Research Data: A Guide to Good Practice. SAGE, 2019, ISBN: 978-1-5264-8238-9.

[Da19] Dahlmanns, M.; Dax, C.; Matzutt, R.; Pennekamp, J.; Hiller, J.; Wehrle, K.: Privacy-Preserving Remote Knowledge System. In: Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP '19). IEEE, 2019, ISBN: 978-1-7281-2700-2.

[Fa14] Faundeen, J. L.; Burley, T. E.; Carlino, J.; Govoni, D. L.; Henkel, H. S.; Holl, S.; Hutchison, V. B.; Martin, E.; Montgomery, E. T.; Ladino, C. C.; Tessler, S.; Zolly, L. S.: The United States Geological Survey Science Data Lifecycle Model, USGS Open-File Report 2013-1265, 2014.

[Fe06] Federal Aviation Administration: Aircraft Certification Service Records, N1-237-05-003, 2006.

[Fi00] Fielding, R. T.: Architectural Styles and the Design of Network-Based Software Architectures, PhD thesis, University of California, 2000.

[GD20a] Gleim, L.; Decker, S.: Open Challenges for the Management and Preservation of Evolving Data on the Web. In: Proceedings of the 6th Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW '20). CEUR Workshop Proceedings, 2020.

[GD20b] Gleim, L.; Decker, S.: Timestamped URLs as Persistent Identifiers. In: Proceedings of the 6th Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW '20). CEUR Workshop Proceedings, 2020.

[Gi13] Gil, Y.; Miles, S.; Belhajjame, K.; Deus, H.; Garijo, D.; Klyne, G.; Missier, P.; Soiland-Reyes, S.; Zednik, S.: PROV Model Primer, W3C Working Group Note, 2013.

[Gl20a] Gleim, L.: FactStack: Interoperable Data Management and Preservation for the Web and Industry 4.0. In: RDA 16th Plenary Meeting — Poster Sessions. 2020.

[Gl20b] Gleim, L. C.; Karim, M. R.; Zimmermann, L.; Kohlbacher, O.; Stenzhorn, H.; Decker, S.; Beyan, O.: Enabling ad-hoc reuse of private data repositories through schema extraction. Journal of Biomedical Semantics 11/1, 2020, ISSN: 2041-1480.

[Gl20c] Gleim, L.; Pennekamp, J.; Liebenberg, M.; Buchsbaum, M.; Niemietz, P.; Knape, S.; Epple, A.; Storms, S.; Trauth, D.; Bergs, T.; Brecher, C.; Decker, S.; Lakemeyer, G.; Wehrle, K.: FactDAG: Formalizing Data Interoperability in an Internet of Production. IEEE Internet of Things Journal 7/4, pp. 3243–3253, 2020, ISSN: 2327-4662.

[Gl20d] Gleim, L.; Tirpitz, L.; Pennekamp, J.; Decker, S.: Expressing FactDAG Provenance with PROV-O. In: Proceedings of the 6th Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW '20). CEUR Workshop Proceedings, 2020.

[Hu00] Hunter, G. S.: Preserving Digital Information: A How-to-do-it Manual. Neal-Schuman Publishers, 2000, ISBN: 978-1-55570-353-0.

[Ka17] Karim, R.; Heinrichs, M.; Gleim, L. C.; Cochez, M.; Porter, E.; Gioia, A. L.; Salahuddin, S.; O'Halloran, M.; Decker, S.; Beyan, O.: Towards a FAIR Sharing of Scientific Experiments: Improving Discoverability and Reusability of Dielectric Measurements of Biological Tissues. In: Proceedings of the 10th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences (SWAT4LS '17). Vol. 2042, CEUR Workshop Proceedings, 2017.

[KB19] Kunze, J. A.; Bermès, E.: The ARK Identifier Scheme, IETF draft-kunze-ark-24, 2019.

[Ko10] Koop, D.; Santos, E.; Bauer, B.; Troyer, M.; Freire, J.; Silva, C. T.: Bridging Workflow and Data Provenance Using Strong Links. In: Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM '10). Vol. 6187, Springer, pp. 397–415, 2010, ISBN: 978-3-642-13817-1.

[LGD20] Lipp, J.; Gleim, L.; Decker, S.: Towards Reusability in the Semantic Web : Decoupling Naming, Validation, and Reasoning. In: Proceedings of the 11th Workshop on Ontology Design and Patterns (WOP '20). CEUR Workshop Proceedings, 2020.

[LLM10] Li, X.; Lebo, T.; McGuinness, D. L.: Provenance-Based Strategies to Develop Trust in Semantic Web Applications. In: Proceedings of the 3rd International Provenance and Annotation Workshop on Provenance and Annotation of Data and Processes (IPAW '10). Vol. 6378, Springer, pp. 182–197, 2010, ISBN: 978-3-642-17818-4.

[LSM13] Lebo, T.; Sahoo, S.; McGuinness, D.: PROV-O: The PROV Ontology, W3C Rec. 2013.

[Ma12] Masinter, L. M.: The 'tdb' and 'duri' URI schemes, based on dated URIs, IETF draft-masinter-dated-uri-10, 2012.

[Me12] Mersmann, C.: Industrialisierende Machine-Vision-Integration im Faserverbundleichtbau, PhD thesis, RWTH Aachen University, 2012, ISBN: 978-3-86359-062-8.

[MF11] Melnikov, A.; Fette, I.: The WebSocket Protocol, IETF RFC 6455, 2011.

[MM13] Missier, P.; Moreau, L.: PROV-DM: The PROV Data Model, W3C Rec. 2013.

[Mo87] Mockapetris, P.: Domain names - concepts and facilities, IETF RFC 1034, 1987.

[Ni20] Niemietz, P.; Pennekamp, J.; Kunze, I.; Trauth, D.; Wehrle, K.; Bergs, T.: Stamping Process Modelling in an Internet of Production. Procedia Manufacturing 49/, pp. 61–68, 2020, ISSN: 2351-9789.

[NK02] Newman, C.; Klyne, G.: Date and Time on the Internet: Timestamps, RFC 3339, 2002.

[Pe19a] Pennekamp, J.; Dahlmanns, M.; Gleim, L.; Decker, S.; Wehrle, K.: Security Considerations for Collaborations in an Industrial IoT-based Lab of Labs. In: Proceedings of the 3rd IEEE Global Conference on Internet of Things (GCIoT '19). IEEE, 2019, ISBN: 978-1-7281-4873-1.

[Pe19b] Pennekamp, J.; Glebke, R.; Henze, M.; Meisen, T.; Quix, C.; Hai, R.; Gleim, L.; Niemietz, P.; Rudack, M.; Knape, S.; Epple, A.; Trauth, D.; Vroomen, U.; Bergs, T.; Brecher, C.; Bührig-Polaczek, A.; Jarke, M.; Wehrle, K.: Towards an Infrastructure Enabling the Internet of Production. In: Proceedings of the 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS '19). IEEE, pp. 31–37, 2019, ISBN: 978-1-5386-8500-6.

[Pe19c] Pennekamp, J.; Henze, M.; Schmidt, S.; Niemietz, P.; Fey, M.; Trauth, D.; Bergs, T.; Brecher, C.; Wehrle, K.: Dataflow Challenges in an *Internet* of Production: A Security & Privacy Perspective. In: Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy (CPS-SPC '19). ACM, pp. 27–38, 2019, ISBN: 978-1-4503-6831-5.

[Pe20a] Pennekamp, J.; Bader, L.; Matzutt, R.; Niemietz, P.; Trauth, D.; Henze, M.; Bergs, T.; Wehrle, K.: Private Multi-Hop Accountability for Supply Chains. In: Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops '20). IEEE, 2020, ISBN: 978-1-7281-7440-2.

[Pe20b] Pennekamp, J.; Buchholz, E.; Lockner, Y.; Dahlmanns, M.; Xi, T.; Fey, M.; Brecher, C.; Hopmann, C.; Wehrle, K.: Privacy-Preserving Production Process Parameter Exchange. In: Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC '20). ACM, pp. 510–525, 2020, ISBN: 978-1-4503-8858-0.

[Po17] Ponemon Institute: The True Cost of Compliance with Data Protection Regulations, White Paper, Ponemon Institute, 2017.

[PPG13] Passant, A.; Polleres, A.; Gearon, P.: SPARQL 1.1 Update, W3C Rec. 2013.

[SAM15] Speicher, S.; Arwe, J.; Malhotra, A.: Linked Data Platform 1.0, W3C Rec. 2015.

[SH13] Seaborne, A.; Harris, S.: SPARQL 1.1 Query Language, W3C Rec. 2013.

[SP17] Snell, J.; Prodromou, E.: Activity Streams 2.0, W3C Rec. 2017.

[St20] Stack Overflow: Developer Survey 2019, `https://insights.stackoverflow.com/survey/2019`, 2019 (accessed December 12, 2020).

[TCS18] Trnka, M.; Cerny, T.; Stickney, N.: Survey of Authentication and Authorization for the Internet of Things. Security and Communication Networks/, 2018, ISSN: 1939-0114.

[Va14] Van de Sompel, H.; Sanderson, R.; Shankar, H.; Klein, M.: Persistent Identifiers for Scholarly Assets and the Web: The Need for an Unambiguous Mapping. International Journal of Digital Curation 9/1, pp. 331–342, 2014, ISSN: 1746-8256.

[Va18] Vander Sande, M.; Verborgh, R.; Hochstenbach, P.; Van de Sompel, H.: Toward sustainable publishing and querying of distributed Linked Data archives. Journal of Documentation 74/1, pp. 195–222, 2018, ISSN: 0022-0418.

[VNS13] Van de Sompel, H.; Nelson, M.; Sanderson, R.: HTTP Framework for Time-Based Access to Resource States – Memento, IETF RFC 7089, 2013.

[Wi16] Wilkinson, M. D.; Dumontier, M.; Aalbersberg, I. J. J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J.-W.; da Silva Santos, L. B.; Bourne, P. E., et al.: The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3/, 2016, ISSN: 2052-4463.

[WLC14] Wood, D.; Lanthaler, M.; Cyganiak, R.: RDF 1.1 Concepts and Abstract Syntax, W3C Rec. 2014.

[Yu18] Yuan, Z.; Ton That, D. H.; Kothari, S.; Fils, G.; Malik, T., et al.: Utilizing Provenance in Reusable Research Objects. In: Informatics. Vol. 5. 1, MDPI, 2018.

[Ze11] Zeng, R.; He, X.; Li, J.; Liu, Z.; van der Aalst, W. M. P.: A Method to Build and Analyze Scientific Workflows from Provenance through Process Mining. In: Proceedings of the 3rd Workshop on the Theory and Practice of Provenance (TaPP '11). USENIX Association, 2011.