Institutions: DYNE.org, FCHH
Author Name: Denis Roio (DYNE.org)
Contributors: Stefano Bocconi, Adam Burns, Puria Nafisi (Dyne.org), Michel Langhammer (FCHH)
Publication Date: 31/01/2023
Internal review: 13/03/2023 (V2)

BUILDING THE DIGITAL INFRASTRUCTURE FOR FAB CITIES, REGIONS AND NATIONS

D2.3c Standards for interoperability and appropriate end-to-end federation

INTERFACER

Funded by

EUROPÄISCHE UNION
Europäischer Fonds für regionale Entwicklung

Hamburg Behörde für Wirtschaft und Innovation

Consortium Partners

HELMUT SCHMIDT UNIVERSITÄT

FABCITY Hamburg

HIWW

dyne.org

interfacerproject.eu

Abstract

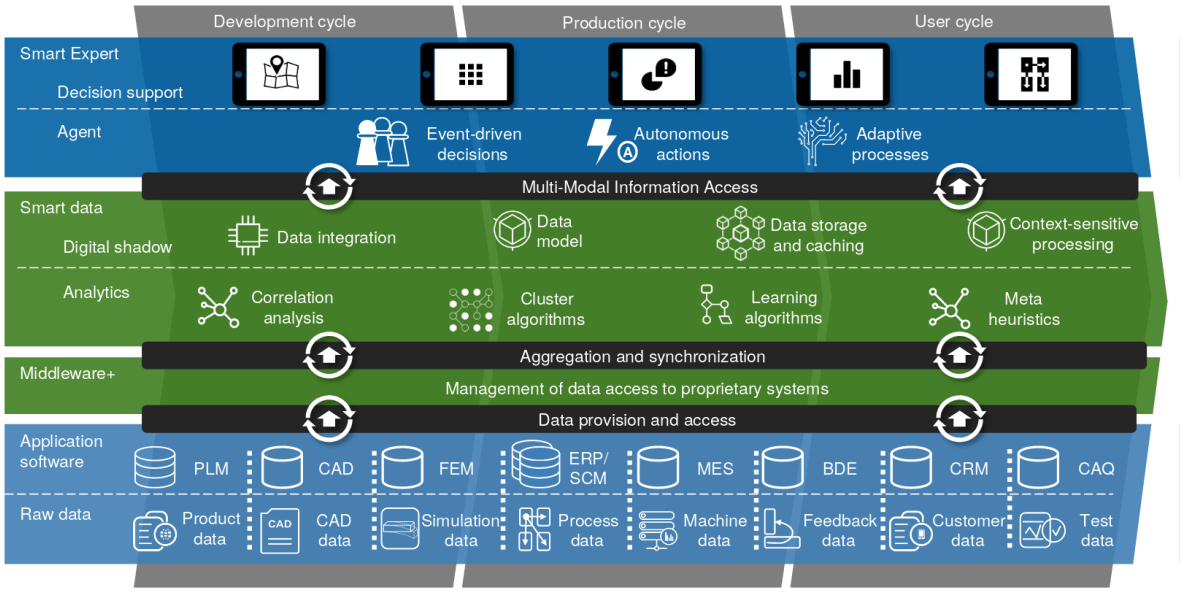
The Interfacer software architecture for the global federated network of FabCities utilises a REA graph database, an end-to-end crypto wallet for sovereign identities, a W3C compliant distributed identity controller and a scalable infrastructure for load distribution across federated nodes. This article discusses the implementation of this architecture, showcasing its benefits in terms of security, scalability, and decentralised control for the network of FabCities.

Table of Contents

1. Context

We start this document by describing the context in which our application is inscribed as a bird's view before diving into specific use cases. This approach will help us understand the salient features of the software architecture we have developed for Interfacer.

1.1 Smart Data



Quelle: WZL

Figure 1. Infrastructure of the internet of production (Brecher et al., 2017).

In the concept of the "Internet of production", this figure shows that a new so-called "smart" layer is inserted between expert systems and production tools. The name "smart data" indicates that the perception of how things are precisely done is blurred, perhaps due to the complexity of its role. It is a layer to liaise between agent-driven decisions and production processes and supports decisions taken through the lifetime of every product. In this figure, there are four relevant roles for the "smart data" layer:

1. Aggregation and synchronisation of data: quality and semantic organisation
2. Multi-modal information access: presentation of data-backed meaning

3. Digital shadow: mostly digital twin functionalities
4. Analytics: processing large quantities of data according to different criteria

While the definition of "smart" through these functionalities is still somehow "magical", it is possible to imagine a few practical and feasible interventions to implement this layer with sufficiently advanced technologies and a few components that still need to be developed.

1.2 Digital Twins

Another diagram helps establish the context of our intervention and is shown below:

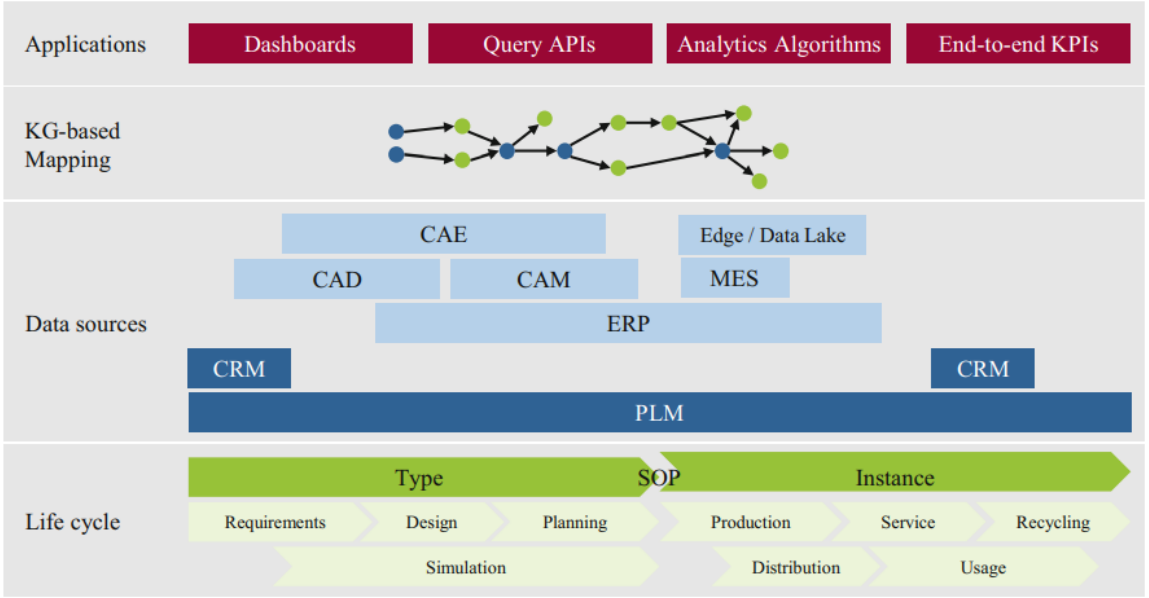


Figure 2. Phases of production process, Infrastructure of the internet of production, Towards Knowledge Graphs for Industrial End-To-End Data Integration,Martin Sjarov

Here the horizontal flow from left to right correlates to the previous figure representing the type definition of designed system standards and models of production processes till the start of production and its operations , loosely associated with the development, production and user cycle. This segmentation may or may not be helpful in our final perception of this context, as what we consider agile development makes this waterfall conception much more fluid in a spiralling form.

However, here we gain more insights as the figure explicitly uses the term "Digital Twin" for this "Integration layer". What is interesting is that in this layer are listed three key elements:

- Graph form of data, highlighting relationships and organising them in a vectorial form
- The need for an Ontology or at least something that helps establish a semantic layer
- The presence of significant data assets and their description

We can now establish that the architectural choices made for the Interfacer project, some also inherited from the Reflow project, match the expectations for this almost-magical image of a "Smart Data" and "Digital Twin" layer inserted between the production processes and decisions made about them.

It is, of course, a cybernetic construct that tries to rationally fill the gap between the strategic direction and the procedural execution in manufacturing and service production. It does so by organising the raw data from production processes into a graph structure that integrates data with its meta information. This approach goes beyond univocal correlation, but deeper into semantic models that can be a starting point for the intelligent organisation of meaning.

The" Digital Twins" concept offers a powerful vision for existing production infrastructure, but for developing a digital infrastructure for Fab Cities we need to adopt to a bottom-up approach and define the features of our architecture step by step.

1.3 Internet of entities

The growing complexity of this conceptual engineering paradigm won't be sustainable in the long term. It is already evident by the insertion of an extra layer carrying features that fill the cognitive gap between strategic and executive functions. It is necessary to adopt a different "data-centric" approach to simplify this conceptual design, facilitate the presence of graph organisation, and create what we call a "first-class citizen" in computer programming, a base data entity that every function can process.

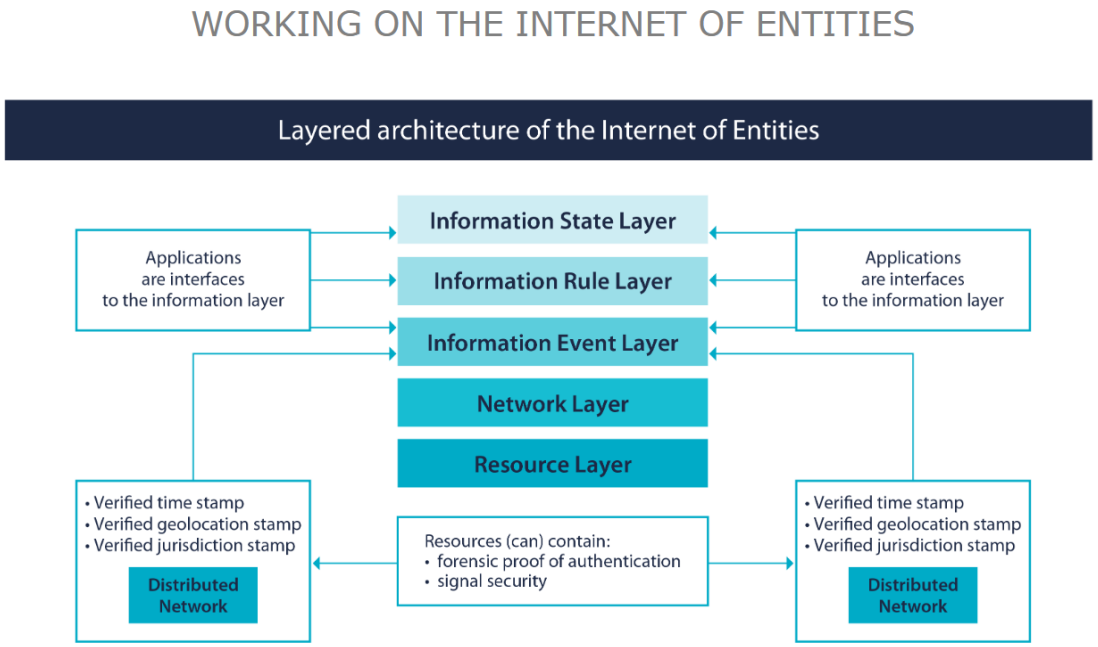


Figure3. FOCAFET and the Internet of Entities (2021)

Our first-class citizen is the node of a graph. This multi-layered entity integrates its "meta-data". It carries it in a graph of relationships from the "resource layer", closely linked to reality, and the "information state layer", resulting from applying all digital events, rules and interactions.

This diagram by the FOCAFET foundation applies to many different domains and works to show that the Context of Interfacer can also be designed from the inside out, around an evolved data entity, rather than just observing an industrial production flow from the top-down.

This "Internet of Entities" conception works well with applying a graph database. It defines the role of all applications as interfaces to the "information layer" made of events, rules and states.

In the case of Interfacer, we took a similar approach. Still, we applied a well-established pattern called REA accounting (Resource Event Agent) as a palette of information layer attributes for the entity. The REA is a fundamental building block for organising knowledge: the REA graph is where we apply events, states and rules and deploy any intelligent operation.

The only missing step from REA to the final product is applying a semantic model to the graph and permitting the transformation of *Sinn* into *Bedeutung*.

2. Methodology

Given the complexity of federated system design, it is essential to specify the methodology we adopted in the Interfacer project to co-design and communicate the outcome to developers in a sufficiently compartmentalised and understandable way to absolve their own tasks. Our work involves several different disciplines and expertise, even in the field of computer science alone. It has been a crucial feature of adopting a methodology that creates as few blockers as possible, facilitates agile development cycles in sprints and allows the integration of components in the leading architecture without locking down processes in a waterfall management style.

To simplify this overview, we break down our architecture design methodology into three sections:

- Fast prototyping
- Sequence diagram
- Pluggable crypto
- Purpose-driven

2.1 Rapid Prototyping

During the development of the Interface project, it became apparent that there was a need to test modelling choices (concerning Interfacer's chosen vocabulary, Value Flows) and back-end functionality independently and in a parallel fashion to the development of the front-end.

Programmatically performing user journeys had the advantage that flows could be tested without human intervention in a quick and repeatable way. The idea was to sort out any modelling uncertainties before the front end would be tested on participants and provide a well-thought solution ready to implement.

For this task, we have chosen the tool Jupyter Notebook[1], a very commonly used web-based interactive programming environment that can execute python code and visualise the results (see figure below).

[1] https://jupyter.org/

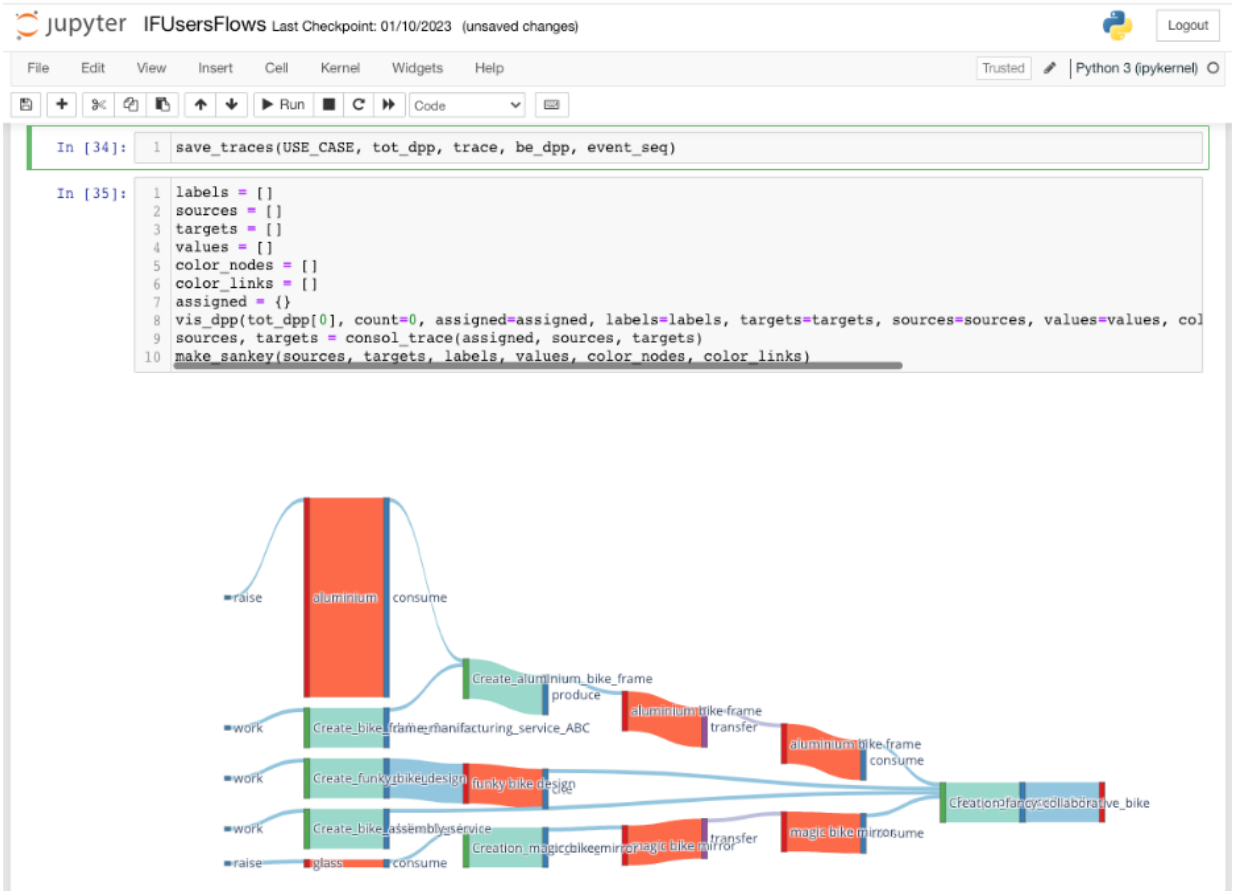


Figure 4. Screenshot of a Jupyter notebook with 2 code cells and one Sankey diagram visible[2].

We have implemented several flows (or use cases) in this tool to check (often using visualisations) that the results of the modelling (and the recorded track of objects) were consistent with our expectations and domain knowledge. To verify this, we also have used another type of visualisation closely related to a possible representation of the Digital Product Passport, shown below.

[2] Repository available at https://github.com/interfacerproject/Interfacer-notebook.

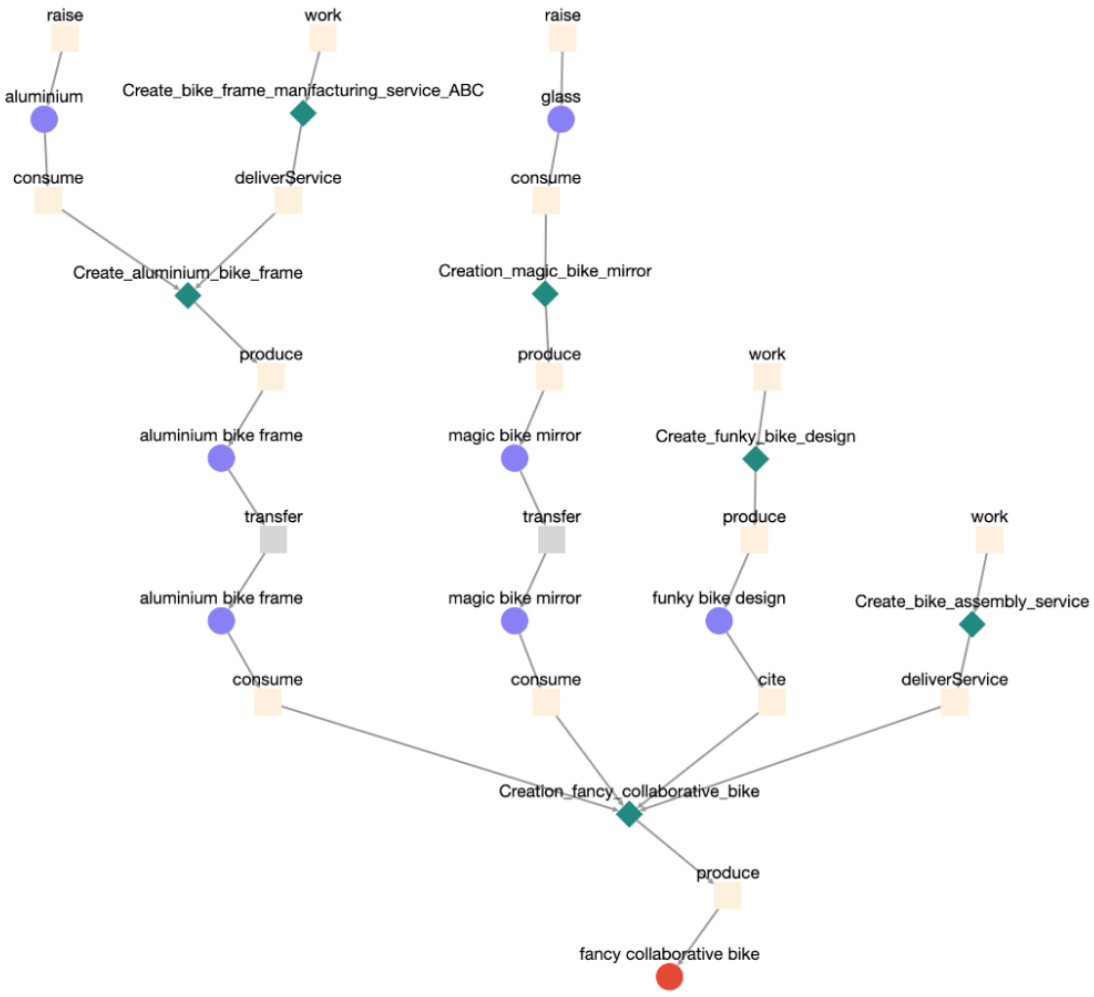


Figure 5. Screenshot of Jupyter Notebook, visualization of recorded track of objects modelling

The figure above is an example of a representation of the same flow. Squares represent events, circles, resources and diamond processes.

At the time of writing, we have developed several notebooks corresponding to prototyping scenarios for data exfoliation, correlation and visualisation. They are a handy tool for creating dialogues with developers and users of the system.

2.2 Sequence Diagram

Sequence diagrams are a powerful tool in software development that help to describe complex processes involving multiple participants. They allow developers to visualise the interactions between different components in a system and how they work together to achieve a specific task. This visual representation provides a clear understanding of the flow of events and the interactions between different components.

In the Interfacer project, sequence diagrams describe the exchanges between various participants, such as the user, the database, and the user interface. The use of sequence diagrams helps to ensure that every aspect of the software's business logic is linked to the exchanges between these participants. By documenting these interactions, the software becomes more secure and better understood, as it provides a clear and concise picture of how the different components of the system work together.

Adopting this methodology also makes the Interfacer software better documented, as the sequence diagrams provide a clear record of the interactions between the different components. This methodology can be beneficial for future developers who may need to make changes to the software, as they can quickly understand how the system works and where changes need to be made. The Interfacer documentation details all exchanges and links every piece of code describing business logic to these exchanges; this way, our software becomes more secure and better documented, making it easier to understand and maintain.

2.3 Pluggable Crypto

Zenroom is a tiny secure execution environment developed in the flagship EU project DECODE (grant nr. 732546). It integrates into any platform and application, even on a chip or a web page. It can authenticate, authorise access and execute human-readable smart contracts for blockchains, databases and much more. It helps to develop cryptography keeping it simple, understandable and maintainable.

We are aware of many challenges when implementing a service infrastructure based on end-to-end cryptography. These challenges were barely visible until recent years when distributed ledger implementations (DLT) of token economies have exposed this technology to crypto-market speculation and has incentivised antagonist actors to exploit flaws.

Among the challenges faced by the industry is that of finding developers that can manage the growing complexity and a framework that accommodates engineers, mathematicians and law

experts to work smoothly together, leveraging interdisciplinary models of organising the workforce, the same way the world-wide-web evolved to include designers and data scientists into its growth. The trust on the reliability of this workforce and its ability to face the complexity of distributed systems becomes crucial: the increase in operational automatisation leaves most power and liabilities in the hands of the code that is executed and in those who write it. Proof of this is the fact that businesses offering audit services of smart contracts have that businesses offering smart contract audit services.

Our goal today is to implement the Interfacer infrastructure for the FabCity global network in a way that is ready to address the existence of a multiplicity of infrastructures and data sources and the political evolution of governance models adapting to different geo-political conditions and trans-national use-cases. We think of use cases that are cross-border driven, and we think about developers, service designers, security experts and institutional auditors that need to understand and trust our solution quickly. They have to interface it with existing infrastructure.

In Interfacer the Zenroom VM and the Zencode contract language are used at every end of the communication, in clients and servers, to ease the integration work and to facilitate the work of cryptographers in parallel with that of software integration engineers, as well the readability of critical code.

2.4 Purpose-driven

In developing the Interfacer project, we made a conscious effort to ensure that real needs and user feedback guided our work. The purpose-driven approach was achieved by closely observing the requirements of FabCities and conducting user interviews to gather information about the challenges and needs of potential users. Our goal was to create a solution that met the real needs of the people.

While it would have been tempting to add a multitude of features, we deliberately chose to focus only on the ones that the participants expressed in our research. This approach allowed us to remain focused on delivering a solution that would have the most significant impact rather than simply adding features for the sake of having more features.

By working hand in hand with the participants and incorporating their feedback into our development process, we created a purpose-driven solution that met their needs. This approach not only resulted in a more effective solution but also helped to build trust and a sense of partnership with the people using it.

In conclusion, our development of the Interfacer project was guided by a purpose-driven journey informed by real needs and user feedback, but that was limited so far to the city of Hamburg and to a limited sample of operating FabCity labs and SMEs.

3. Architecture

3.1 Federation

Federated software architecture is a distributed computing architecture that enables the integration of autonomous systems to work together as a single entity while maintaining the independence of each individual procedure. This approach allows large-scale systems to scale dynamically when for instance, one needs to add or remove new components.

The concept of a federation in computing goes back to the early days of distributed computing in the 1960s and 1970s. These early efforts aimed to create large-scale systems that could take advantage of multiple computers' processing power and storage capacity. The earliest examples of federated systems were networked mainframes: engineers used to link them together to form a giant virtual mainframe.

In the 1980s and 1990s, the rise of client-server computing and the Internet led to the development of new federated architectures, such as distributed object systems and distributed database systems. These systems allowed for integrating different types of computers and networks and sharing data and services across organisational boundaries.

Today we use the federated software architecture pattern to build a wide variety of systems, including cloud computing platforms, Internet of Things (IoT) systems, and social networks. This architecture allows the creation of large-scale systems composed of many different components, each controlled by another organisation or individual.

One of the critical advantages of federated systems is still to achieve **dynamic scaling** as we add or remove new modules. Federation allows the creation of large-scale systems that can accommodate growth and change over time without requiring extensive reconfiguration or redesign.

Another critical advantage of federated systems is the ability to maintain **data sovereignty**. In a federated system, each participant controls their own data and decides how it is shared and used. Data sovereignty allows participants to retain control over their information and ensure compliance with legal and regulatory requirements.

Overall, federated software architecture provides the ability to create large-scale systems that are flexible and scalable and respect the autonomy of the participant. This architecture is used in various areas, such as Cloud, IoT and Social networks, as they represent scenarios where potentially independent entities must work and share data.

3.2 Components

The software component architecture of Interfacer is made of software we developed for this project. Internal modules use well-established open-source tools like PostreSQL and Tarantool, among the languages we used, the most used are C, Typescript, Golang, Elixir and Lua. The modular architecture allows for load distribution, and every single component is tuned to perform well even on low-power hardware and, in some cases, ARM64 boards like the Raspberry Pi4. In all cases, the Linux kernel and GNU tools are used where necessary.

Deployment is compatible with the Fab City OS Installer by targeting Docker containers and, in some cases, by providing Helm charts.

Integration tests cover the software entirely, and, in most cases, unit tests are also present and included in the continuous integration infrastructure on GitHub. The crypto contracts governing the business logic are written in Zencode and interpreted by the Zenroom VM, which is embedded both client-side and server-side.

Our staging systems adopt a devops approach of infrastructure as code based on a number of Ansible roles included in the source code.

The figure below illustrates the components' disposition and their inter-dependency. This documentation is kept brief and one should refer to each software repository for more details, they are linked here and listed in Appendix 1.

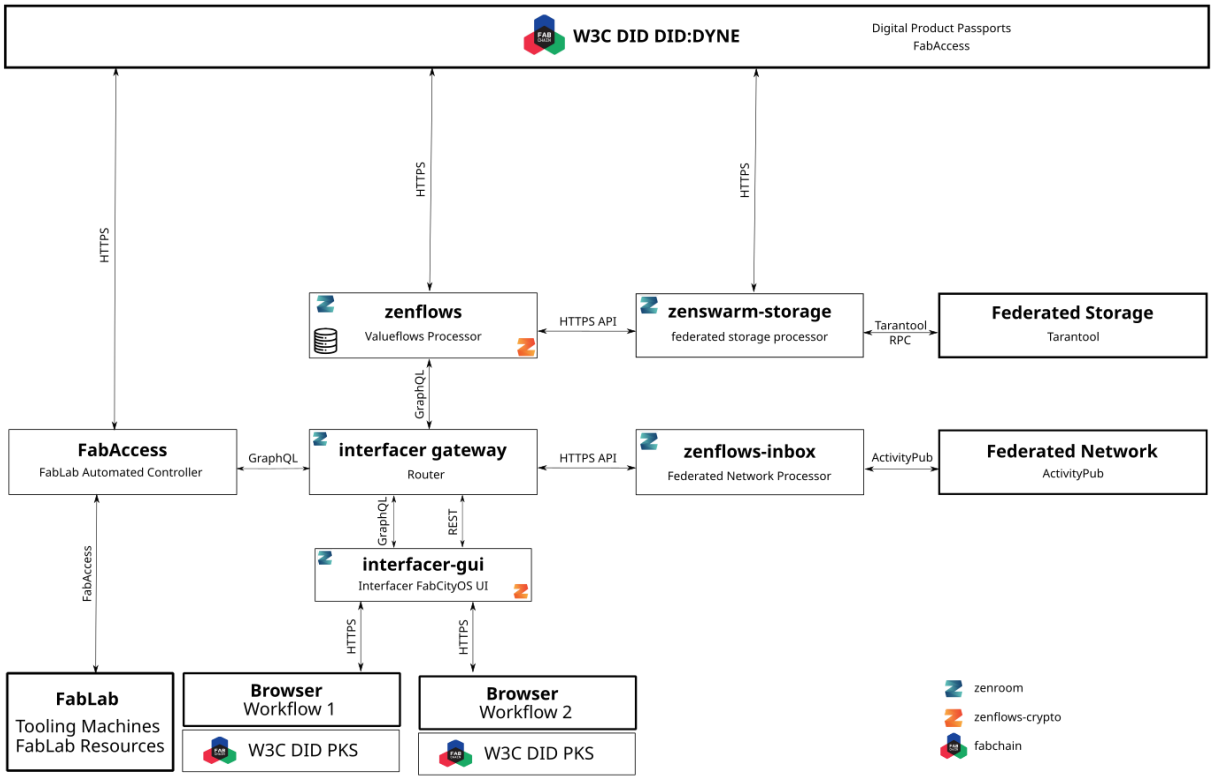


Figure 6. Architecture of Fab City OS Core

W3C DID DID:DYNE

The World Wide Web Consortium (W3C) Distributed Identifiers (DIDs) is a standard that enables verifiable, decentralised digital identities for any subject (e.g., a person, organisation, thing, or data model). The Dyne.org Foundation has registered a specific schema with the W3C standards body and implemented its software controller to support the Interfacer project. See the Distributed Identity section for further details.

zenflows

zenflows is a modular server that leverages commons-based peer production by documenting and monitoring the life cycle of products. Its function is to enable a federated network of organisations to bundle, systematise and share data, including information and knowledge about designs, services and physical artefacts.

interfacer gateway

interfacer gateway is a public Internet gateway service that acts as a proxy to route information flows to each of the back-end services, enabling a secure, modular, and scalable deployment architecture.

zenswarm-storage

zenswarm-storage is a distributed caching and sharding service offering high availability of data: its nodes are used by applications to store arbitrarily large data objects pegged to a blockchain without overloading ledger or blockchain nodes. See the Global Distribution section for further details.

zenflows-inbox

zenflows-inbox is a modular service that provides incoming and outgoing messages to allow updates to be processed from a distributed and federated network. It supports the W3C ActivityPub standard for decentralisation.

FabAccess

FabAccess is a service that enables a resource access control system, enabling FabLabs to deny or allow access to the machines and tools in the labs only to authorised persons. See the Local Fab Access section for further details.

interfacer-gui

interfacer-gui is a Progressive Web Application (PWA) that is run in-browser and acts as Graphical User Interface (GUI) and a crypto-wallet (W3C DID PKS or Personal Key Store). See the Self-sovereign identity section for further wallet details.

4. Use-cases

4.1 Self-sovereign identity

The notion of self-sovereign identity has two functional roles in our implementation.

The first role is political and gives individuals complete control of their identity, meaning that only individuals can sign documents, designs, transactions or resource commitments with their key in personal possession. Our on-line infrastructure will not hold participants' keys or passwords; the participants will store them in their hands in a sort of wallet.

The second role is functional and responds to the needs of a federated infrastructure. Participants should not be tied to one FabCity instance but may be able to move and sign into different

governance systems within the same federation. Participants should be able to do this autonomously.

Our implementation of self-sovereign identity goes well beyond the design of a mere crypto-wallet. It provides a user experience modelled after the FabCity use case, consisting of complex interactions between resource events and agents that the participants can sign as agents.

At the core of this implementation is the crypto-model of a key created client-side by answering personal questions, which are then processed through cryptographic transformations to develop or recover the same key. The only link between these keys and the global network of FabCities is a "secret salt" configured server-side and is common to all federated instances.

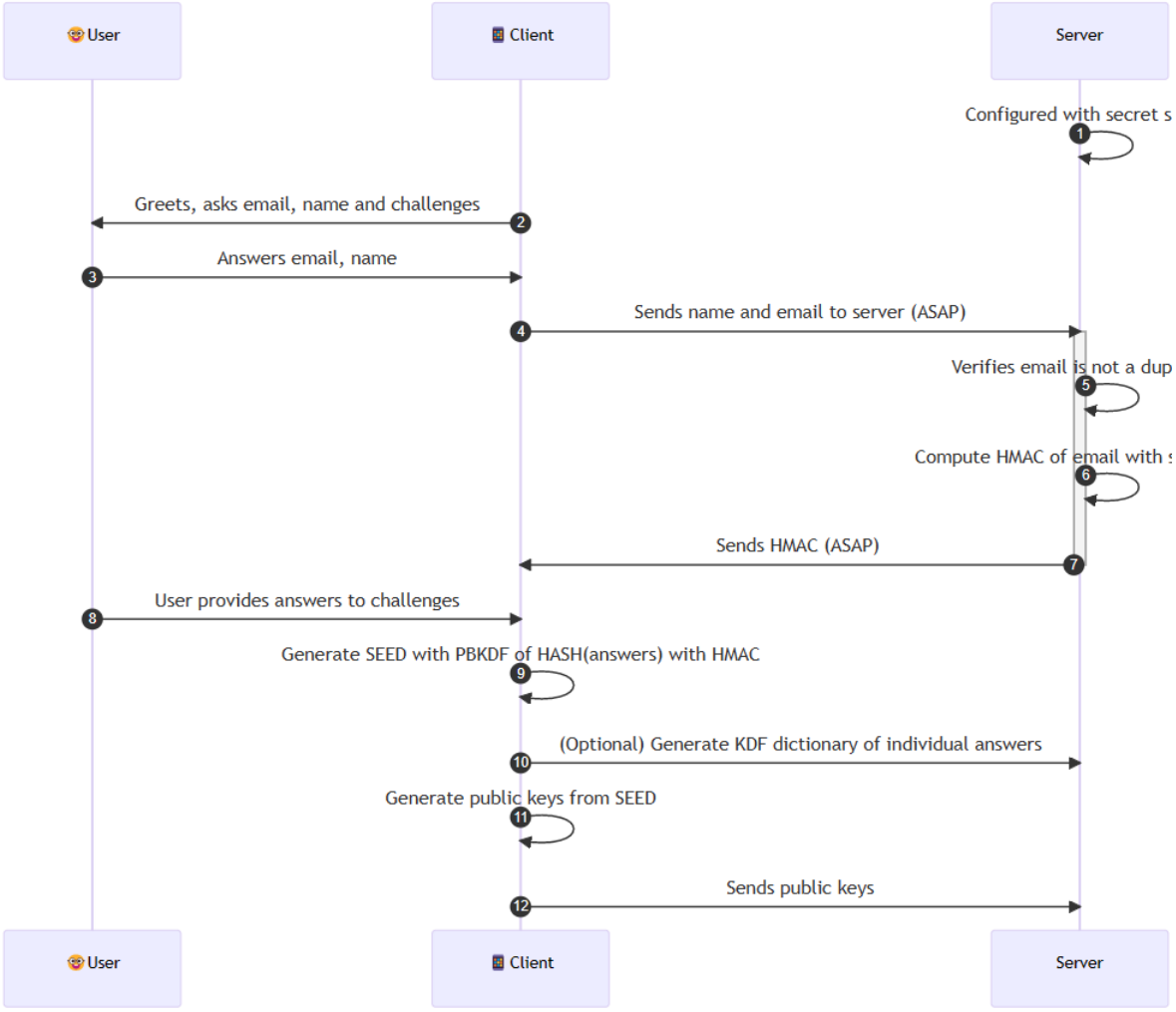


Figure 7. Fab City OS GUI & zenflows workflow diagram

Notes

1. Secret salt is generated at server install and saved as an HEX string in its configuration
2. Interactive GUI poses all questions on one page: email, name and 5 challenges
4. As soon as user answers name and email, reactive page sends them to server (ASAP)
7. As soon as Client receives HMAC the Submit button is green
8. May happen in parallel while Client and Server are handshaking the HMAC (ASAP)
9. May need User confirmation that the answers given to challenges are OK
10. Useful to facilitate seed recovery: the server can check validity of single answers
12. Start with EDDSA public keys, the seed is reused for more key types when needed

4.2 Global Distribution

A content distribution network (CDN) is a distributed server system that works together to provide fast delivery of content, such as images, videos, and other digital files. A decentralised and federated CDN is a network that is not controlled by a single entity but is made up of many independent nodes that work together to provide content delivery.

In the Interfacer project, a decentralised and federated CDN is used to ensure the global availability of data. We store the data on several agnostic network supports, such as the small storage server developed for Interfacer and popular web3 solutions like IPFS. These decentralised storage solutions make the software scalable, as it is possible to add new storage nodes to the network without overloading the infrastructure.

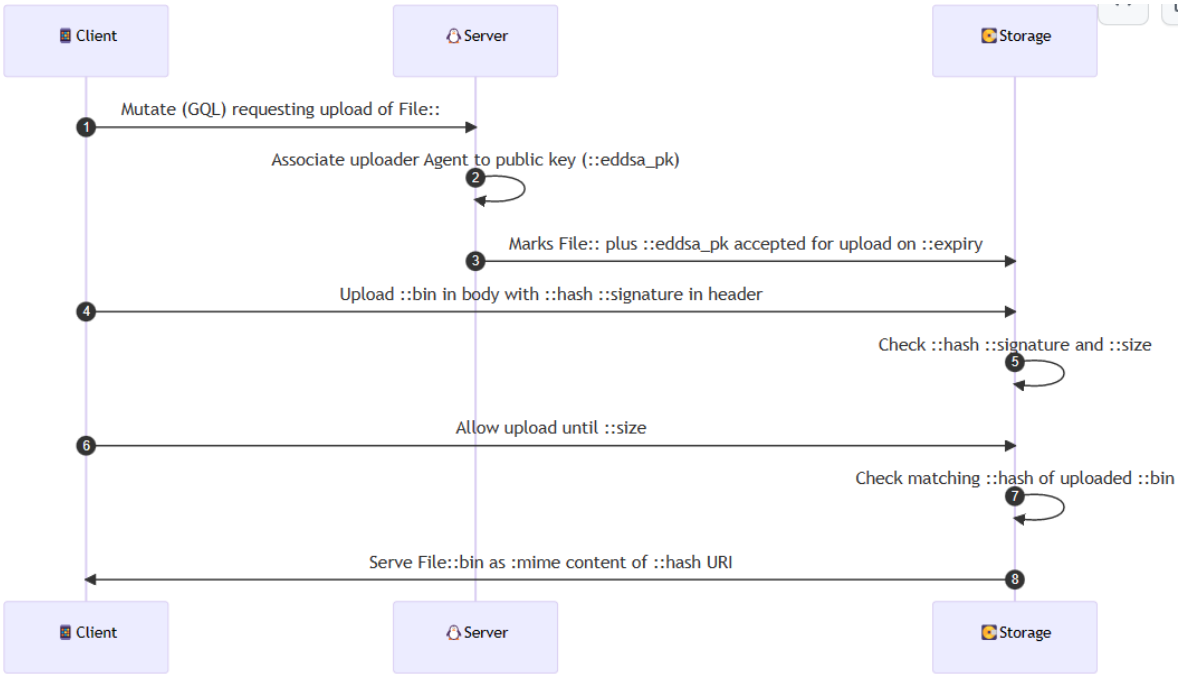


Figure 8. Fab City OS GUI & zenswarm-storage workflow diagram

Notes

1. Clients can make signed mutations on servers containing the File field detailed above
2. Servers signs a message to Storage about hash and size as accepted for upload (expiry)
3. Clients may upload to Storage the content of File of declared size at any later time (expiry), upload is made in multi-part and header with hash is content-disposition
4. Storage checks if hash and size are accepted for upload
5. Storage may abort the upload or allow it reading data only until size
6. Storage checks hash of uploaded data
7. Storage saves the data as File::bin and serves it on HTTP GET as File::hash

The only authenticated communication happens between Client and Server and between Server and Storage, not between Client and Storage.

The Storage has the public key of the server, not that of clients, which simplifies key exchange.

Any Client may hit the upload API endpoint of Storage without signalling, and verification is made on the expiring key/value of hash and size.

Cryptographic signatures of metadata are used to ensure the integrity of the data stored on the network. These signatures act as a digital fingerprint of the data, confirming its authenticity and preventing tampering. This means that users can trust the data they receive from the network, as it has been verified as genuine.

In conclusion, adopting a decentralised and federated CDN and using cryptographic metadata signatures in the Interfacer project ensures global data availability, making the software scalable at meagre marginal costs. The Interfacer project can provide a secure and reliable content distribution solution to users worldwide by allowing the network to grow and add more nodes.

4.3 Local Fab Access

FabAccess provides a resource access control system, enabling FabLabs to deny or allow access to the machines and tools in the labs only to authorised persons. FabAccess delivers essential functions for all the labs and spaces equipped with appliances like milling machines, cutters and all the other tools that require safety courses before being handled.

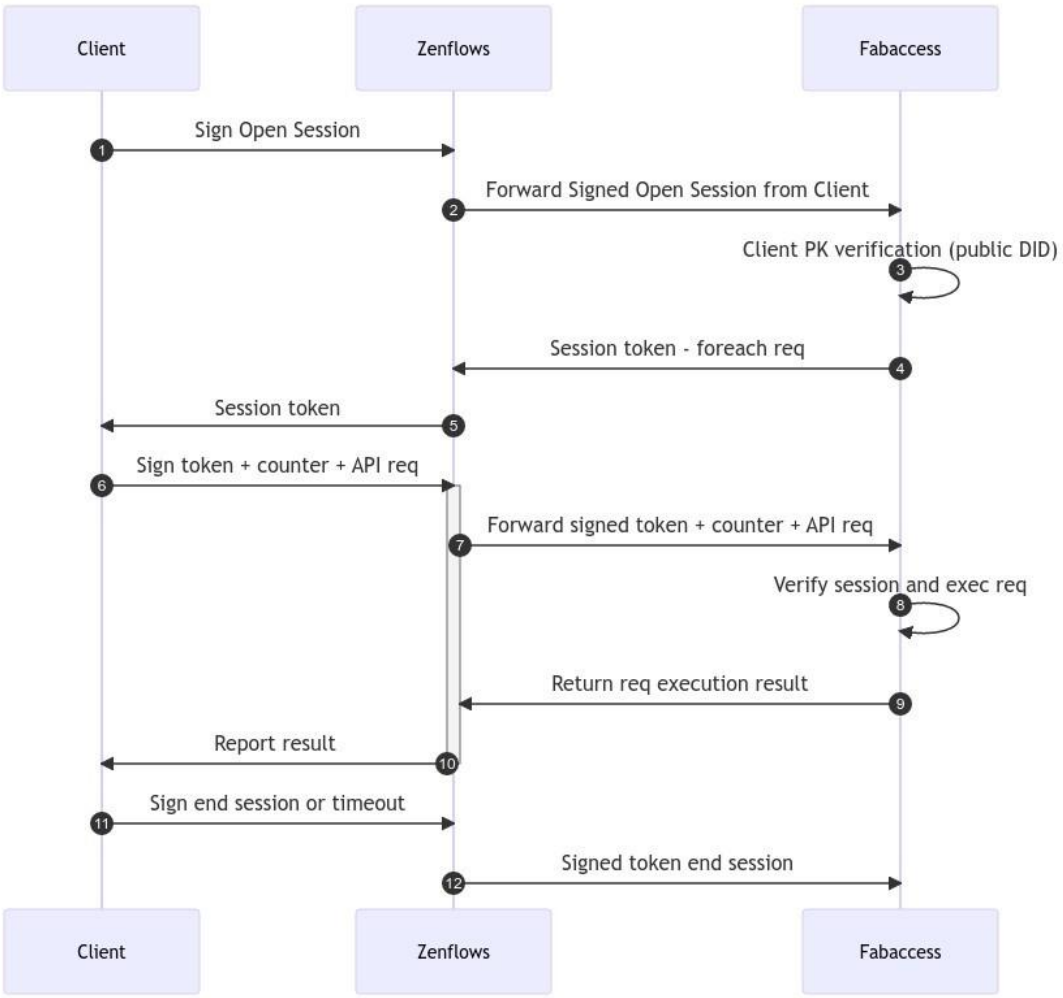


Figure 9. Fab City OS GUI & FabAccess workflow diagram

Notes

1. Clients may sign a request to open a FabAccess session
2. Zenflows may verify then forward the request to FabAccess
3. Fabaccess verifies the request using the PK found on DID
4. SESSION START: Fabaccess sends a session token to Zenflows
5. Zenflows forwards the session token to the Client
6. ITER: Client signs the token + a FabAccesss API request
7. ITER: Zenflows forwards to FabAccess the signed tok+API+count
8. ITER: Fabaccess verifies the signed tok+API+count and executes
9. ITER: Fabaccess describes execution / returns results

10. ITER: Results are forwarded to the Client
11. Client may request end of session
12. Zenflows signals the end of session or timeout

5. Distributed Identity

Distributed identity systems, such as the W3C DID (Decentralised Identifier) standard, play a critical role in federated systems by providing a way for different components of the system to **identify** and **authenticate** one another. In our setup, the DID controller allows for the interoperability of various federated modules and seamless data migration between components.

In a federated system, each component operates independently but must still be able to communicate and share data with other federation members. For communication and data sharing to happen, we need a way to identify and authenticate each component and its users. We developed a distributed identity system following the W3C DID international standard to allow each member to manage its identifiers and authentication mechanisms transparently in Interfacer and future projects in the FabCity legacy.

The W3C DID standard defines a way to create and manage decentralised identifiers (DIDs) for entities (people, organisations, devices, etc.) on a blockchain or other decentralised systems. These DIDs provide a globally unique and persistent way to identify an entity, regardless of where it operates. Additionally, the W3C DID standard allows access control and data management of the identity. It means that the identity owner, the entity, is in control of the data, access and sharing of the represented entity: such a condition permits the data sovereignty and the autonomy that is the fundamental aspect of a federated system.

The DID component in the federated architecture of the Interfacer project allows for the seamless integration of different elements in the federated system. Each member can use the DIDs to identify and authenticate other components and their users. It also responds to the need for an IAM/SSO (Identity Authentication Management / Single Sign-On, briefly debated in deliverable D2.1a section 3.1 with an early assessment). The Interfacer project envisions participants in the FabCity network as able to identify and authenticate themselves on various systems connected to the federation and access their services based on global and internal business logic. But this is not the only requirement emerging in Interfacer. While IAM/SSO primarily relates to people, there is also the need to authenticate digital and physical products (digital twins) with a DPP (Digital Product Passport), whose authentication needs to be publicly verifiable.

Our implementation of the W3C DID standard provides a core building block for federated systems by allowing for interoperability and seamless data migration between components, including digital product passports of digital twins. Our system offers a way to identify and authenticate any entity in the FabCity federated network, regardless of where they operate within the federated system or even outside of it, by using decentralised and distributed identifiers. It allows for each component's independent and autonomous operation while enabling communication and data sharing.

5.1 Core Implementation

Our free and open-source implementation of the decentralised identifier standard by the World Wide Web Consortium features Zenroom, a portable virtual machine for smart-contract language execution. All contracts governing the DID are written in a simple human-like language (Zencode) to make it easy to manage such delicate tasks as identification, authentication and data sharing.

With this software, one can create and manage multiple decentralised identity domains and use it to interact with various on-line services and applications.

Zencode is a vital part of our implementation, as it allows us to easily express complex data transformation and cryptographic operations in an easy way to understand and write. It makes it possible for you to customise and manage your decentralised identity without needing to have advanced technical knowledge.

This document will provide a light overview of the decentralised identity implementation; interested readers can read the code, run a sandbox on their computer and customise it on-line following the instructions we provide on https://did.dyne.org and on https://github.com/dyne/W3C-DID.

5.2 Interoperability

Here are some architectural choices taken in the implementation of our DID, briefly explained:

We strictly respect the **end-to-end cryptography** architecture pattern, and no private keys, not even for central admins, are kept on the DID controller service on-line. It means that even in case of a service break-in, the data cannot be tampered with, and there will be no leak of private information.

We use the **filesystem storage** provided by the host operating system, most likely GNU/Linux or any flavor of BSD, which leverages our solution's interoperability.

We **record history using Git**, an international de-facto standard for revision history and accounting for modifications of large codebases. The adoption of Git is made possible by the choice of filesystem storage.

Whenever needed we can distribute identities with a peer-to-peer **content delivery strategy powered by IPFS**, the "Inter-Planetary File System" capable of delivering files to any peer connecting to the network, similarly to the Bittorrent protocol. This is again made possible by the choice of filesystem storage.

We **express complex data transformations and cryptographic operations in Zencode**. Using a human-like language for the DID contracts provides an easy way to understand and write them. It makes it possible to customise and manage decentralised identities without needing advanced technical knowledge but applying what is most important to solve challenges: domain knowledge.

5.3 Compliance

Our W3C DID implementation supports:

- A list of API endpoints, as an array "serviceEndpoint".
- Geolocation fields as "Country" and "State"
- Public keys for:
  - Secp256k1 ECDSA, widely used for single signatures
  - ED25519 EDDSA widely used for single signatures
  - BLS381 "Reflow" [REFLOW], for multisignature and advanced zero-knowledge proof operations
  - Dilithium2, for quantum-proof signatures
  - Ethereum public addresses ("blockchainAccountId"), following the eip155 standard
- The JWS signature of the DID Document operated by an admin inside the "proof" field in order to ensure data integrity.

6. Conclusions

The Interfacer project has not only laid the base for the implementation of a federated and global network of FabCities. It makes the future creative and production labs capable of sharing knowledge, designs and values without imposing centralised governance and with the scaling potential of the recent web3 technologies.

The resulting software architecture implemented in Interfacer also has the potential to accommodate machine learning processes in a more deterministic and accountable way, applying it to the implementation of a semantic model and training it to use our vocabulary for its reasoning.

The automation of heuristic processes and strategic reasoning is an enormous potential to be explored at the dawn of a new generation of machine learning technologies. Today we improperly dub "Artificial Intelligence" technologies limited to putting words in context rather than developing strategic reasoning on a graph of relations.

Let us quote Dr. Anna Ivanova's concise words: "the word-in-context prediction objective is not enough to master human thought". Now that Interfacer enhances the integration of resource, network and information layers in a graph of entities governed by a vocabulary, it may be well possible to build the infrastructure for the decentralised industry of the future.

Another potential opened by the Interfacer federated architecture is the legal validation of signatures made by self-sovereign identities recognised by any governance joining the network. The standardisation of Interfacer signatures is not a technical perspective but one that has to do with policy and compliance. We may find it easy to establish this standard as we, in turn, adopted solid standard primitives in our implementation.

At last, there is a potential for open exploration to extend the economic model we experimented with and briefly piloted for the agile flow of values and remuneration in single FabCities and within the federated network. This aspect has been implemented technically but needs further modelling in-vivo to capture all complex economic relationships that participants may establish between different kinds of products and services. The adoption of this model by large organisations is promising, and plans include a more rigorous description of its economic underpinnings in an upcoming Interfacer publication, as well as the adoption by the FabCity network and the Dyne.org community.

Annex 1: References

DID Primer. Drummond Reed; Manu Sporny. Rebooting the Web of Trust 2017. URL: https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-fall2017/blob/master/topics-and-advance-readings/did-primer.md

Reflow: Zero Knowledge Multi Party Signatures with Application to Distributed Authentication. D. Roio; A. Ibrisevic; A. D'Intino. Dyne.org. Pre-print. URL: https://ui.adsabs.harvard.edu/abs/2021arXiv210514527R/abstract

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc3986

Augmented BNF for Syntax Specifications: ABNF. D. Crocker, Ed; P. Overell. IETF. January 2008. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc5234

Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

Dissociating language and thought in large language models: a cognitive perspective. M. Kyle and A. Ivanova et al. URL: https://doi.org/10.48550/arxiv.2301.06627

FOCAFET and the Internet of Entities (2021), F. Kleemans URL: https://focafet.org

J. B. Hoffmann, P. Heimes and S. Senel, "IoT Platforms for the Internet of Production," in *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4098-4105, June 2019, doi: 10.1109/JIOT.2018.2875594.

Towards Knowledge Graphs for Industrial End-To-End Data Integration: Technologies, Architectures and Potentials (2021), M.Sjarov, J.Franke URL: https://link.springer.com/chapter/10.1007/978-3-030-78424-9_60

Annex 2: Code repositories

Interfacer (https://github.com/interfacerproject) (16 code, documentation and research repositories, still growing)

dyne/W3C-DID: Dyne.org's W3C-DID implementation (https://github.com/dyne/W3C-DID-data) (stable implementation)

FabAccess · (https://gitlab.com/fabinfra/fabaccess) (home of the BFFH federated service)