



WP4. Product Data Capturing, Exchange & Information Services

Task 4.2

Digital Product Memory Information Linking and Storage Service

Deliverable 4.2

Digital Product Memory for Tracing Products Across the Life Cycle



DISCLAIMER

The opinion stated in this report reflects the opinion of the authors and not the opinion of the European Commission.

All intellectual property rights are owned by CIRCTHREAD consortium members and are protected by the applicable laws. Reproduction is not authorised without prior written agreement.

The commercial use of any information contained in this document may require a license from the owner of that information.

ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 958448.

Project Data	
Project Acronym	CircThread
Project Title	Building the Digital Thread for Circular Economy Product, Resource & Service Management
Grant Agreement number	958448
Call identifier	H2020-LCCI-2020-EASME-twostage
Topic identifier	CE-SC5-31-2020 Develop, implement, and assess a circular economy-oriented product information management system for complex products from cradle to cradle
Funding Scheme	IA - Innovation action
Project duration	48 months (From 1 June 2021)
Coordinator	FUNDACION CARTIF
Website	https://circthread.eu
Deliverable Document Sheet	
Deliverable No.	D4.2
Deliverable title	Digital Product Memory for Tracing Products Across the Life Cycle
Description	<p>The task will implement a Digital Object Memory (DOME) service for storing and updating digital information of the physical product for operationalising use case 1. DOME presents the product as an object with properties, states, and history of use, for interfacing across the life- cycle with design, manufacturing, in-use, and end-of-use stages. The task builds upon the identified product status and life cycle quantitative information needs defined in T3.1 and T3.2 and building upon the H2020 knowledge base of projects incl. SemPROM, RES-COM, FAR-EDGE (DFKI), and TagItSmart! The design of the DOME will be assessed in terms of passive (only information retrieval) or active (information storage and/or decision execution) attached to the product. This covers physical information hardware attached to the product (e.g. memory, microsensors, radio modules for web connectivity, energy supply), passive tagging information systems (e.g. QR Codes, Data Matrices), quality degradation of the tagging system over time, and software needs (e.g. memory management, sensor management, user interfaces, security). The design will be compliant with the new EN 45559:2019 standard covering product vehicles of communication and confidentiality levels. Specific attention will be paid to the interface between manufacturing objects and product objects from product-assembly-component relationships, in developing the flexible semantic product memory model on the software side in a semantic language (RDF or OWL). To this end an ontology mapping will be generated for machinery/factory objects to products as represented in ontologies for Machinery Execution Systems (MES), Product Life Cycle Management (PLM) and Enterprise Resource Management (ERP) software on one end, and product information on the other end (CAD/CAM) utilising relevant standards (e.g. ASTM E3012 for environmental process characterisation). This will enable interfacing of the Digital Object Memory with machinery and process specific information (energy use, material use, material waste) both in manufacturing and end of use phases (recycling, disassembly, remanufacturing), to attach such</p>

	information given that the appropriate APIs are in place (as designed and implemented in T4.3). The task will also evaluate options for persistent storage, and how to manage active updating of the DOME, as it enters a new life cycle stage. The task will result in the design, technical framework and required vocabularies for the DOME service.		
WP No.	WP4		
Related task	T4.2 – Digital Product Memory Information Linking and Storage Service		
Lead Beneficiary	12 -DFKI		
Author(s)	DFKI		
Contributor(s)	ATOS, CAR, ECOW, IDP, MONC (DOM), MASS, SIS, SUP		
Type	Report		
Dissemination L.	Public		
Language	English – GB		
Due Date	31/12/2023	Submission Date	20/12/2023

Version	Action	Owner	Contributors	Date
V.0.1	TOC Proposal, Written	DFKI	Jonas Brozeit, Leonhard Kunz (DFKI)	22/06/2023
V.0.2	TOC Confirmed, Reviewed	ATOS, CAR, ECOW, IDP, MONC (DOM), SISTRADE, SUPSI	Roberto Castillo (ATOS), Anibal Renones Domínguez (CARTIF), Cristina Parrado Nunez (CARTIF), Rembrandt Koppelaar (ECOW), Leone Deborah (SUPSI), José Cunha (SISTRADE), Aitor Apraiz (MASS), Mikel Borrás (IDP), Carolina Mejia Nino (MONC)	28/06/2023
V.0.3	Formulate draft	DFKI	Jonas Brozeit, Leonhard Kunz (DFKI)	29.11.2023
V.0.4	Full draft	DFKI	Jonas Brozeit, Leonhard Kunz (DFKI)	15.12.2023
V.0.5	Review feedback	ECOW	Rembrandt Koopelaar (ECOW)	18.12.2023
V.0.6	Finalised deliverable	DFKI	Jonas Brozeit, Leonhard Kunz (DFKI)	20.12.2023

V.0.7	Submitted deliverable	CARTIF	Fernando Burgoa (CARTIF)	20.12.2023
-------	-----------------------	--------	--------------------------	------------

1 EXECUTIVE SUMMARY

CircThread seeks to make appliances like boilers and washing machines sustainable. Both from environmental and social perspectives. To achieve this, we want to swiftly increase appliance lifespan, repairability and reuse. And ensure that products are properly recycled when they are no longer repairable. We are working on this challenge with more than 30 organisations, thanks to grant funding from the European Union under the H2020 programme.

To solve this challenge, we will put in the hands of all actors a software platform for sharing critical information about appliances. Information shared between product designers, manufacturers, retailers, citizens, repairers, and recyclers. Radically improving our ability to make better lifespan improvement, reuse, and recycling decisions. Helping you and others with circularity decision making at all stages of a product's life cycle. Therefore, our open-source software platform will enable digital exchanges of data across the extended product life cycle.

The platform will become available in the cloud in 2025. Equipped with services for collaboration, trust, and security. And we will test it before launch in three pilots, in Slovenia, Spain and Italy. Together with manufacturers, repairers, retailers, collectors, recyclers, and many others.

The purpose of this deliverable is to document the progress and methods used in the development of some core modules of the CircThread platform, namely the DOME Resolver and the Event Interpretation and Information Recommendation Service. It aims to provide insights into these services within the wider context of the CircThread platform, highlighting the challenges and solutions encountered during the development process. The target audience for this report includes stakeholders involved in product lifecycle management and all service- and data provider of the CircThread platform.

On behalf of the Authors

Jonas Brozeit, DFKI

Leonhard Kunz, DFKI

Table of Contents

1	EXECUTIVE SUMMARY	6
2	Report Outline	11
2.1	Overview.....	11
2.2	Project context and use of results.....	12
2.3	Methodology.....	13
2.4	Structure of Report	14
3	Introduction	16
4	State of the art for Data carriers and the digital object memory.....	17
4.1	Digital Object Memory.....	17
4.2	Physical tagging technology	20
4.2.1	Barcodes.....	20
4.2.2	Radio-based Tags	22
4.3	Product unique identification	23
5	Development of a Digital Object Memory Service	26
5.1	Microservice 1: Resolver	27
5.1.1	Staying up to date with the data providers	28
5.1.2	Integrate Legacy identifiers.....	31
5.1.3	Resolver integration.....	33
5.2	Microservice 2: Event interpretation and information recommendation	36
5.2.1	System architecture	38
5.2.2	Description of data structure of “Generic User” database	40
5.2.3	Description of data structure of dynamic user behaviour database	41
5.2.4	Recommendation Logic with Neo4j and Resolver integration.....	43
6	Showcasing the service in the Smartfactory Kaiserslautern	46
7	Conclusion and next steps	54
7.1	Conclusions	54
7.2	Future development opportunities / Next steps.....	54
7.3	Other conclusions and lessons learned	55
8	References.....	56

List of Figures

Figure 1 Broad conceptual framework overview of Digital Product Memory Service.	12
Figure 2 Excerpt from personas of the requirements analysis.	13
Figure 3 Draft of the click prototype designed with Figma.....	14
Figure 4 Picture of DFKI Living Lab – Smartfactory ^{KL} showcase for physical-digital linking.	14
Figure 5 Schematic Representation of the developed microservices.	16
Figure 6 Distinction of DOME implementations regarding the level of implementation between edge and cloud.....	17
Figure 7: Distinction between active and passive DOME.....	18
Figure 8 Object Memory Model based on W3C Object Memory Model Incubator Group	19
Figure 9 Different barcode styles. a) is a Spotify code, b) the EAN 13 and c) the CarTRAK ACI plate for railroad identification.	20
Figure 10: Assortment of 2D-Matrix codes from the Code 16K a), which is based on the Code 128, a colour coded cryptographic code b) developed in a hackathon for signing in e- banking and the industry and consumer common data matrix code c) and QR-Code d)....	21
Figure 11: Composition of an RFID-tag.	22
Figure 12: Samples of labelled and unlabelled composite identifiers containing identification on the product model and the product unit level.....	24
Figure 13: Example for GS1 digital link format utilizing a URL-based direct pointer to a product related dataset.....	24
Figure 14: Blockchain-based identification verification using distributed ledgers to find the verified data owner.....	25
Figure 15: CircThread ecosystem architecture.	26
Figure 16: Description of the CircThread ecosystem using ADOME terminology.....	27
Figure 17: Schematic information request sequence to the CircThread platform through a client application using the product identifier.....	28
Figure 18: Passive resolver approach including the registration of all services with a central service registry.	29
Figure 19: First draft of the required service metadata within the service registry on the example of the product model metadata catalogue developed by project partners.	30
Figure 20: Regular expression type pattern storage format for the matching based parser.	33
Figure 21: Description of the digital link type identifier using the Resolver identifier storage format.	33
Figure 22: Open API documentation of the Resolver API.....	34
Figure 23: Resolver modules and their interaction in resolving a request to the service.	35

Figure 24 Different patterns of demand for information from the individual players in the life cycle.....	37
Figure 25 Motivation for a recommendation system for the circular economy.	38
Figure 26 Overview of the process of the Event Interpretation and Information Recommendations through to integration in the DOME Resolver.	38
Figure 27 Building blocks of the property graph model	39
Figure 28 Neo4j graph schema of the generic user.	40
Figure 29 Extract of an example of an event for the generic user from Neo4j.	41
Figure 30 Neo4j graph schema of the dynamic user database.	42
Figure 31 Example for the creation of a customized Event template.....	42
Figure 32 Extract from the dynamic Neo4j database to illustrate the relationship between the various events.....	43
Figure 33: Sample product of SmartFactory ^{KL} is a model-sized truck where the colors, geometries, materials and manufacturing processes can be configured individually.	46
Figure 34: First data carrier which is primarily customer facing.....	52
Figure 35: Second data carrier on the protected inside of the product.....	53

List of Tables

Table 1: Typical error correction levels.	22
Table 2: List of proposed definitions of terms used that are defined for the service registry.	31
Table 3: Created role-based personae and their characterization for the projected life cycle of the sample product.....	47
Table 4: Derived requirements and their classification into f=functional, nf=non-functional, c=conditional.....	48
Table 5: Prioritization of requirements using the Moscow-method with M=must have, S=should have, C=could have, W=won't have.	50

Index

AAS Asset Administration Shell
ADOMe Active Digital Object Memory
CE Circular Economy
CEAP Circular Economy Action Plan
CI/CD Continuous Integration and Continuous Delivery/Deployment
CPS Cyber-Physical System
DOMe Digital Object Memory, Digital Object Memory
DPP Digital Product Passport
DTDl Digital Twin Design Language
ESPR Eco-design for Sustainable Products Regulation
European Article Number European Article Number
GTIN Global Trade Item Number
HF High Frequency
IDM Identity, role and user management module
IoT Internet of Things
ISBN International Standard Book Number
JAN Japanese Article Number
LPG Label Property Graphs
NFC Near Field Communication
OPCUA Open Platform Communications Unified Architecture
PMMC Product Model Metadata Catalogue
RE Recommendation Engine
REST Representational State Transfer
RFID Radio-frequency identification
UHF Ultra High Frequency
UPC Universal Product Code
URL Uniform Resource Locator
WP Work Package

2 REPORT OUTLINE

2.1 Overview

Currently, consumers purchase products with the awareness that they will be replaced or discarded within a fairly short period of time, even if only one element of the product is broken. Addressing this problem, by improving the lifespan of products, is a key example that the EU transition to a circular economy aims to tackle. The European Green Deal (European Commission, 2019) is the cornerstone of this effort, which seeks European industry to lead the way in promoting a shift in this behaviour by reducing its carbon and material footprint and integrating circularity into the economy (European Commission, 2022). As part of the Green Deal, the Circular Economy Action Plan (CEAP) (European Commission, 2020) aims to promote the development of lead markets for carbon-neutral and sustainable products in the EU. The plan also aims to establish a sustainable product policy framework, with the goal of creating a just and prosperous society that has a modern, resource-efficient, circular, and competitive economy.

Demonstrating the benefits of a Circular Economy (CE) and challenging the dominant linear economy is a key aim of the CircThread project. The project will develop and demonstrate digital life cycle information exchange solutions. This exchange of information is essential for informed decision-making and enables the creation of services and business models that promote product recycling, prolong product life, and enhance product reuse. To operationalise these exchanges of information across the extended life cycle of products, an open-source platform is under development by the CircThread project partners.

The approach will be tested in three pilot clusters located in Spain, Slovenia, and Italy. The test will be based on seven use cases for newly manufactured and existing products throughout the product life cycle, described in detail in deliverable D2.2¹. The products considered are home appliances and home energy systems, including washing machines, boilers, batteries, and solar glass panels. Each use case represents a different information need to support decision-making in the CE to achieve a specific goal. The overall idea behind CircThread is that a CE can be enabled by sharing information and potential links throughout a product's life cycle between organisations that do not typically share information.

Within the CircThread project, work package (WP) 4 provides the metadata and data structures required for product representation and the services that enable information linkage throughout the product life cycle. Task 4.2 aims to develop a Digital Object Memory (DOME) Information Linking and Storage Service. By using a microservice approach, two key components have been developed as part of this service:

- The *DOME Resolver* serves as a central 'routing unit' for guiding a user to the desired web URIs containing product information. It connects the various service and data providers that are part of the CircThread platform. Specifically, a distinction is made between information provided by the Product Model Metadata Catalogue (PMMC) at product model and the Digital Product Passport (DPP)

¹<https://circthread.com/download/deliverable-2-2-use-case-information-requirements-evaluation/>

Manager at product item level, due to the nature of these services as a repository of information.

- The *Event Interpretation and Information Recommendation Service* is a term recommendation engine (RE) that extends the Resolver service. It suggests relevant terms from the CircThread vocabulary to the various actors in the product life cycle, allowing the creation of an event- and role-specific DPP.

2.2 Project context and use of results

This deliverable describes the **Digital Product Memory Information Linking and Storage Service, needed for Tracing Products** across the life cycle, developed under task 4.2. This service is needed for operationalising use case 1, that seeks to deploy the baseline of traceability and exchange across the product life cycle. To facilitate communication throughout the entire product value chain, from manufacturers to final product recyclers. The Digital Product Memory service allows for enhancing the digital-physical links for tracking products throughout the life cycle using a digital object memory with physical tagging technologies (e.g., QR-codes, NFC).

The use case will be tested in the pilots by tracking of the product throughout its extended life cycle with DOME capabilities. This system will be used for physical-digital tracking of the products at each life cycle stage, utilizing scannable information codes and the ability to link to a digital information set of the product. This will enable exchanges using CircThread as a Circular Digital Thread, providing a Circular Product Chain of Custody.

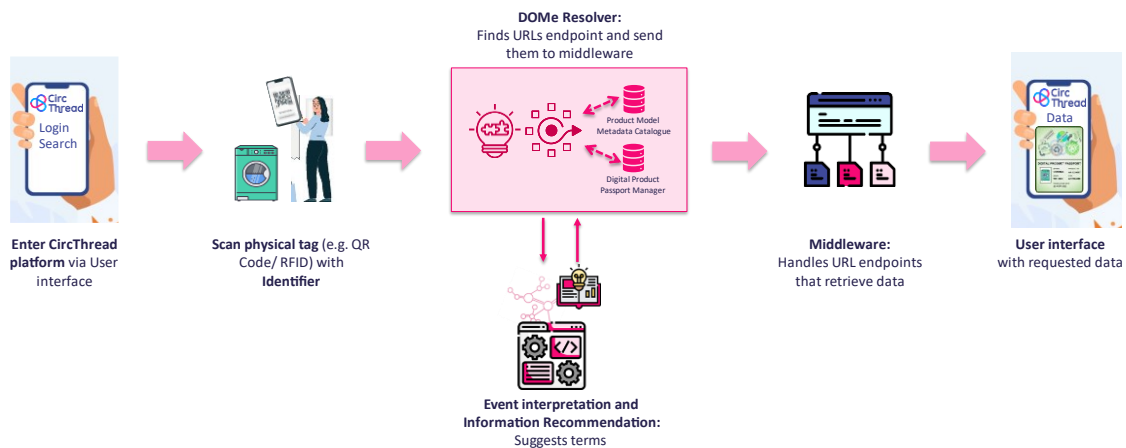


Figure 1 Broad conceptual framework overview of Digital Product Memory Service.

Figure 1 presents a broad overview of the conceptual framework for this deliverable. The user accesses the CircThread platform and scans a physical tag or data carrier that contains a product identifier. Then, the DOME Resolver obtains the URL data endpoints from the various data- and service providers, such as the PMMC or DPP Manager. By suggesting relevant terms for a particular role or life cycle phase, the Event Interpretation and Information Recommendation Engine can support the user as well as the DOME Resolver. Providing a 'search engine' type functionality when scanning the physical data carrier. Finally, the Resolver sends the URL links to the middleware under development in CircThread WP5, through which the actual data is retrieved for visualization. This deliverable will describe the DOME Resolver for tracking the product using physical tagging systems and the technical framework for the information links to/from the product with the tagging interactions. Additionally, it includes the implementation of a

term recommendation system. The outcomes of D4.2 are relevant to all stakeholders in the product life cycle as well as to the scientific community and related projects. Within CircThread, this includes the pilot partner, all service- and data-providers as well as all actors across the product lifecycle.

2.3 Methodology

This section outlines the methodology used to develop the Digital Product Memory Information Linking and Storage Service. The initial research focus was on acquiring domain knowledge, which was achieved by analysing CircThread's household appliances and home energy products, as well as investigating tagging technologies. Valuable insights and knowledge about resolver technologies for product information management utilising a data carrier like a QR code, were also gained through exchanges with other organisations, including GS1 Germany. Several CircThread workshops followed, which collected the necessary attributes for the services and evaluated the vocabulary developed in D4.1. Additionally, discussions were held with other CircThread service providers to assess how their services could be best integrated into the DOME Resolver and to clarify the information they provide. The third part of the methodology was a requirements analysis for data carrier interactions and identifiers utilising persona-based survey evaluations (see Figure 2). The aim was to better understand stakeholder needs throughout the life cycle and adapt development accordingly. Use case diagrams were created to define various usage scenarios for the resolver, including for example the access for logged-in and non-logged-in users. This phase was carried out in coordination with WP5 leaders ATOS supported by EcoWise.

role	#	requirement	f	nf	c	M	S	C	W	
Product Designer	1	The identifier should seamlessly integrate into the product design without compromising aesthetics.	x			x				f - functional
	2	Compatibility with various product materials and designs for versatile integration options.	x			x				nf - non-functional
	3	The identifier should have a compact form factor to fit within the product's dimensions without being obtrusive.		x			x			c - condition
	4	Customizable sizing options for different product types and sizes.	x			x				M - Must have
Manufacturing Engineer	5	The identifier should be easy to integrate into the manufacturing process without causing disruptions or delays.	x			x				S - Should have
	6	Compatibility with existing manufacturing equipment and processes for seamless integration.	x			x				C - Could have
	7	The identifier should be made of durable materials resistant to manufacturing stressors like heat and pressure.		x			x			W - Won't have
	8	Material should be conducive to mass production techniques (e.g., molding, engraving).		x			x			
Warehouse Operator	9	The identifier should be easily scannable, even from a distance, using handheld scanners commonly used in warehouses.	x			x				
	10	The identifier should withstand rough handling, occasional impacts, and exposure to dust and moisture without losing functionality.		x						
	11	Compatibility with common handheld scanning devices used in warehouses			x					

Figure 2 Excerpt from personas of the requirements analysis.

After analysing the requirements, a click-prototype of the resolver (see Figure 3) was created to gather feedback and improve the service. The feedback led to a revision of the resolver concept, resulting in the current implementation. Additionally, it highlighted the need for a way to receive information in a customised manner. Consequently, a recommendation system was developed that suggests relevant terms from the CircThread vocabulary to the actors involved in the product life cycle, enabling them to create a DPP tailored to their specific event and role.

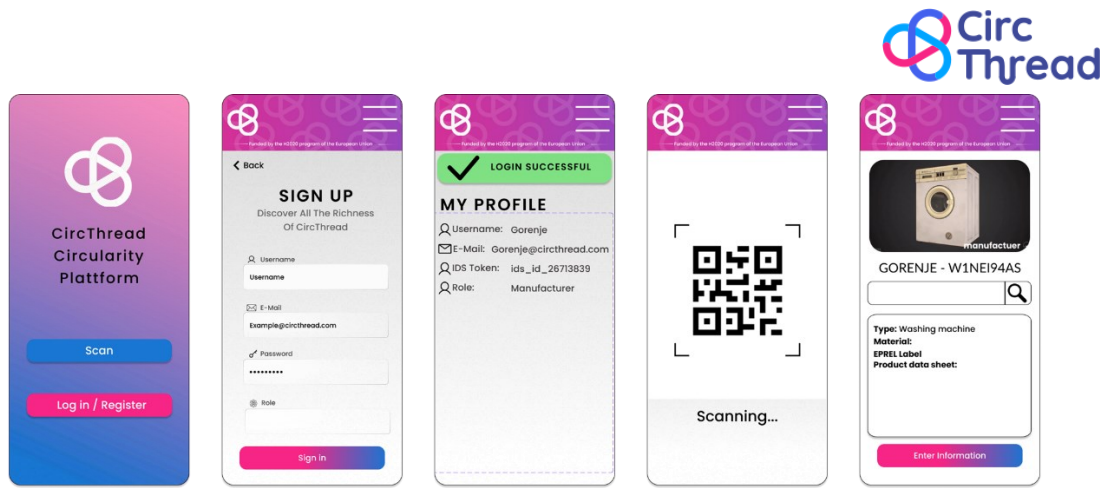


Figure 3 Draft of the click prototype designed with Figma.

During that process an iterative prototyping improvement process was implemented for both the DOME resolver and the Event Interpretation and Information Recommendation Engine. This involved using tools like GitHub and processes such as Continuous Integration and Continuous Delivery/Deployment (CI/CD), as well as conducting surveys to gather user feedback.

After completing the initial prototype implementations, a showcase was also set up at the DFKI Living Lab – Smartfactory^{KL}, as shown in Figure 4. This involved acquiring an RFID scanner to prove the connection between physical and digital elements. Now that an insight into the methodological approaches and processes has been gained, the next chapter will deal with the structure of this report.



Figure 4 Picture of DFKI Living Lab – Smartfactory^{KL} showcase for physical-digital linking.

2.4 Structure of Report

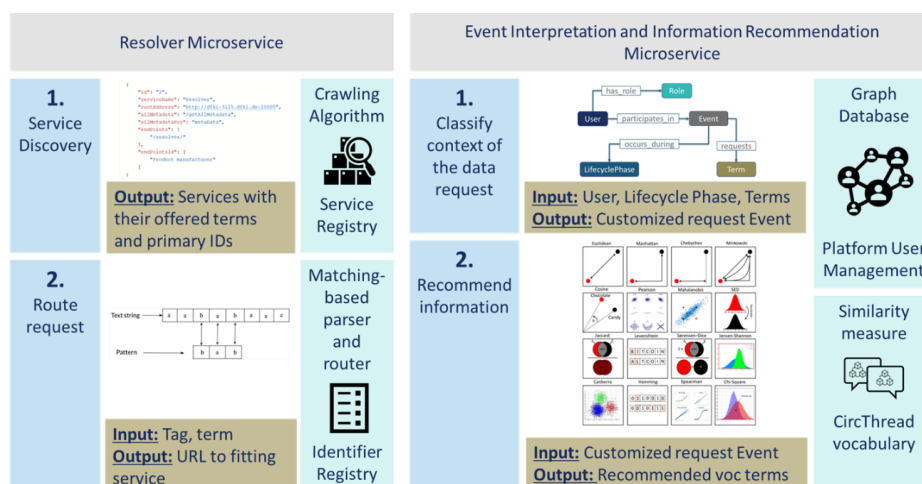
The outcomes of the work conducted under Task 4.2 between months M14 - M31 are presented in this report, which is divided into four sections:

- **Section 3** familiarises the reader with the topic and roughly describes what the implemented DOME Resolver and the term recommendation engine are.

- **Section 4** deals with the state of the art for DOME's and physical digital linking, including the physical tagging system and product identifier.
- **Section 5** describes in detail the development of the several microservices like the Resolver service and the Event Interpretation and Information RE.
- **Section 6** focuses on the transfer of the developed service to a sample product within the DFKI-IFS SmartFactory^{KL} show case.
- **Section 7** concludes the achievements and points to future development opportunities.

3 INTRODUCTION

The Eco-design for Sustainable Products Regulation (ESPR) is a part of the EU Green Deal and is introduced by the European Commission. It establishes a transformative framework for a multi-sectoral DPPs (European Commission 2019). Companies that produce particular products, such as mattresses, furniture, or textiles, will in the future due to this legislation be required to provide and share detailed product data in accordance with EU legislation. In parallel, the EU has agreed on a supply chain law in which companies will become responsible for upholding particular standards or requirements in their business chain, i.e. also for business partners of the company and in some cases also for downstream activities such as distribution or recycling (BMAS, 2023). The broader context of this legislation is the need to seamlessly link and store product information throughout a product's life cycle. The CircThread project partner DFKI has developed to this end the Digital Product Memory Information Linking and Storage Service. This service comprises two main subservices implemented as separate microservices under a single responsibility paradigm: a resolver and a term recommendation system, as shown in Figure 5.



Services are modular and independent of each other through separate APIs

Figure 5 Schematic Representation of the developed microservices.

Why is a recommendation system useful? One reason for this is that people are often overwhelmed with irrelevant information. For example, a recycler working on an assembly line is not interested in the price of a product, but instead if it contains hazardous components or materials. The RE should only suggest relevant terms from the CircThread vocabulary, which is used as the projects and IT services standardised terminology.

The DOME resolver is a central interface for linking different information sources and finding the corresponding data endpoints (URLs) of service and data providers. It has two main tasks. First, interpreting the user's intent when scanning the product identifier/data carrier and providing a route to the appropriate available information for updating and retrieving information. The second step is carried out through a scan request to the platform by connecting to the relevant service that provides the necessary information for the request. The project differentiates between information at the product model level, which is provided by the PMMC, and at the product unit level, which is provided by the DPP Manager. The middleware processes and manages the data retrieval process for the returned URL endpoints as part of WP5. The subsequent chapters provide a detailed account of the DOME resolver and recommendation service development.

4 STATE OF THE ART FOR DATA CARRIERS AND THE DIGITAL OBJECT MEMORY

4.1 Digital Object Memory

The concept of a Digital Object Memory (DOMe) is closely related to the definition of cyber-physical systems (CPS) and their interconnection on the Internet of Things (IoT). The former defines systems that integrate computational elements into physical objects with the goal to enhance the objects behavioural modalities and human-machine as well as machine-machine interactions (National Science Foundation, 2023). The latter focusses on the communication and data exchange between CPS (Li et al., 2015). The DOMe expands the capabilities of CPS by representing a digital information storage, that carries data regarding the object, which can be accessed, read and written during the life cycle of the product (Hauptert, 2021). This memory is a uniquely identifiable information storage, that can be associated with a specific individual object. In its operationalisation it can be placed on the object, such as a product like a boiler or a battery in the CircThread project.

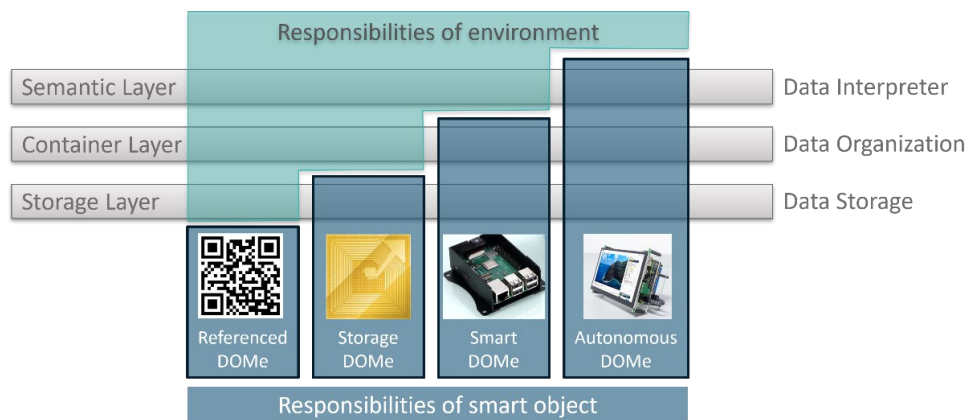


Figure 6 Distinction of DOMe implementations regarding the level of implementation between edge and cloud (Kröner et al., 2012).

The concept of CPS and DOMe as an integral part is in general very broad and leaves room for many discussions as to the concrete implementation. A lot of these discussions revolve around which tasks can be implemented on the edge level of the information storage itself, and which need to be offloaded to the cloud. The heterogeneity of the implementations of results from economic, social, technical or environmental constraints applying to different physical objects (Kröner et al., 2012). Figure 6 shows an approach to classify the DOMe implementations according to the on-object storage and computational capabilities. The classification distinguishes three levels of information storage the first of which is simple physical storage of data, the second containerizes and structures larger amounts of data, and the semantic layer adds semantic interpretation to the stored data (Kröner et al., 2012). Offering access to the different layers of on-object memory capabilities. Kröner et al. distinguish between 4 kinds of DOMe systems (Kröner et al., 2012):

- **Referenced DOMe:** These provide a basic physical level of storage, as the physical component on the product only provide a reference to a memory service. The actual data retrieval needs to be performed by the reading environment and is detached from the physical product. Nevertheless, these implementations require little to no on-object resources to maintain their function.

- **Storage DOME:** These possess the capabilities to store a limited amount of information storage or process data. Nevertheless, advanced tasks like memory management and interpretation need to be performed by the environment.
- **Smart DOME:** These typically integrate a small edge device capable of performing queries on the internal memory to selectively provide information to the reader. More complex operations of data processing must still be performed by the environment.
- **Autonomous DOME:** These are essentially Smart DOME equipped with a more powerful computational unit that enables data processing and the answering of complex queries, that require an understanding of the integrated memory semantics. They have the highest requirements in computational hardware, as well as power supply.

The lines between these categories however can be blurry and implementations of a DOME can belong to more than one form at the same time. For example, the memory of a passive RFID transponder could be used to store the reference to an information storage accessible through the scanning environment while the rest of the storage can be used to store information about the object directly in the storage (Kröner et al., 2012).

In a follow up work the authors introduce the term Active Digital Object Memory with which they want to address the distinction between a passive data storage accessible for read and write operations and an active component capable of data processing and active communication with its environment (Hauptert et al., 2012). They describe the extension of a passive DOME with an activity module and define the requirements for an ADOME (Hauptert et al., 2012):

The memory should provide a mechanism to execute platform independent processing logic that is either located in the memory itself or injected by a requesting application. Furthermore, the memory should provide the capability to execute processing logic periodically.

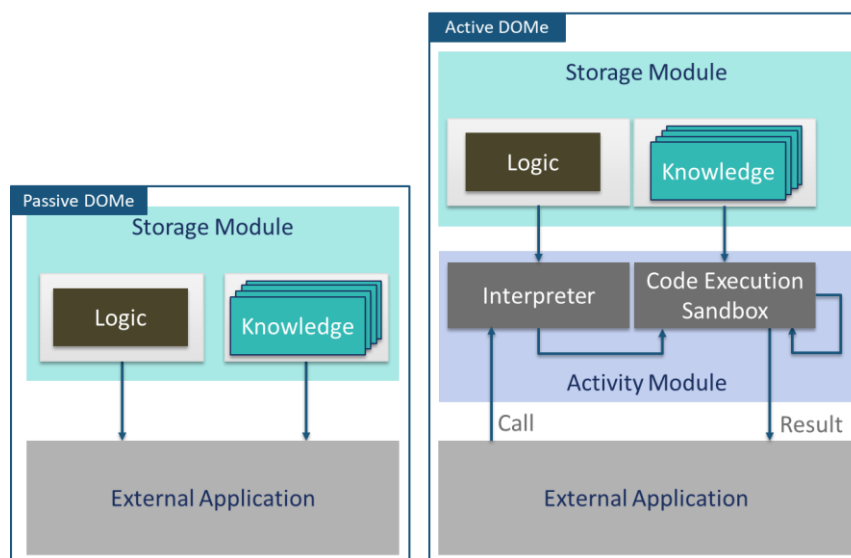


Figure 7: Distinction between active and passive DOME.

The activity module, as depicted in Figure 7, can be seen as a data processing and logic component bound to the DOME that functions as an internal execution logic and interface between the DOME and the environment. As stated by Hauptert et al, the activity module

is not necessarily located on the DOME itself, therefore requiring the utilisation of actual processing power on the device itself as described with Smart or autonomous DOME. Alternatively, the module can also run on a server being connected to the object via an object identifier.

Hauptert describes an architecture created by the W3C Object Memory Model Incubator Group that proposes URLs as memory identifier pointing towards a structured memory, which is provided using the model shown in Figure 8 (Hauptert, 2021). The model structures the information in storage blocks individually identifiable by their own identifier.

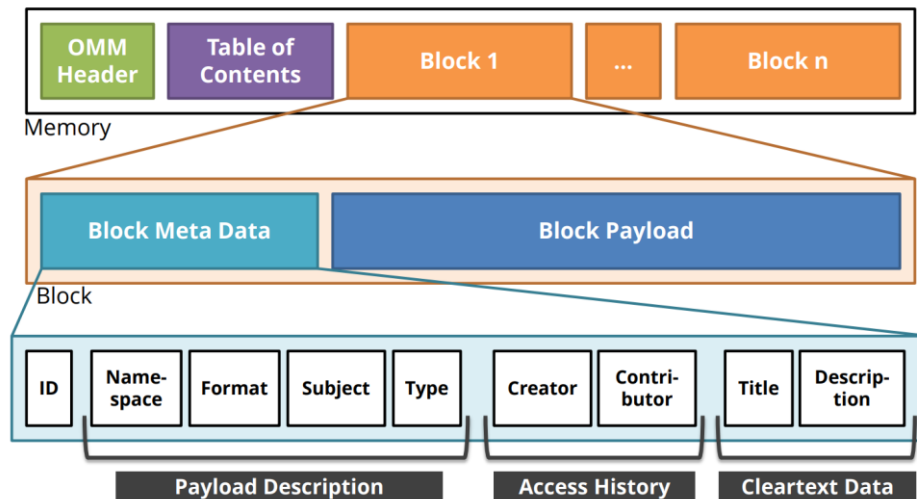


Figure 8 Object Memory Model based on W3C Object Memory Model Incubator Group (Hauptert, 2021).

Access to the different blocks should be provided via a RESTful interface and the structured memory URL. A proposed example of the URL structure is the following:

http://sampleoms.org/st/s_memory/<blockID>/payload/

Here 'st' in the path stands for storage access while the authors also propose to replace it with 'web' to point to a HTML-based user interface.

Birtel et al. applied this approach to structuring the DOME to production environments in the context of Industry 4.0 (Birtel et al., 2020). Their implementation of an ADOME allowed the product to interact with its means of production during the manufacturing process. However, they used an OPCUA-based implementation of the Asset Administration Shell (AAS) standard to create a hierarchical object storage. This approach has gained traction due to its relevance in the implementation of digital twins in the Industry 4.0 (Heidel, 2019).

The concept of the DOME is quite closely related to the idea behind the DPP. This is a key reason why the DOME service was included in CircThread project, as it provides for a similar approach to physical-digital interactions as envisioned in the data carrier utilisation of Digital Product Passports. The concept of the DOME was quite one of the first to consider an individual product record to be accessed throughout its life cycle (Alvarado et al., 2009), yet the concept never got much attention when it was developed in the 2000s. Nevertheless, we think that both concepts can be considered together, and the considered Digital Object Memory Storage and Linking service can be described using the established ideas of the DOME.

4.2 Physical tagging technology

A basic element of the DOME is the physical-digital link, especially when dealing with an off-object storage which raises the need to locate the right storage belonging to the object at hand. There are many technologies available to design a machine-readable physical interface. Common in the context of DOME and product identification are optical or radio-based data carriers (Hauptert, 2021). Even though these tags are cheap to produce and require no active energy upkeep, they also require external server and computation capabilities and are typically used as reference DOME.

4.2.1 Barcodes

Optical machine-readable barcodes can be categorized into 1D and 2D tags. Patented in 1952, the barcode was inspired by morse code patterns (Woodland & Bernard, 1952). Several different forms exist using the length of the bars, the spacing between the bars, their colour, and the width of the bars itself to encode simple codes of limited length. Some example of different barcode styles are shown in Figure 9.

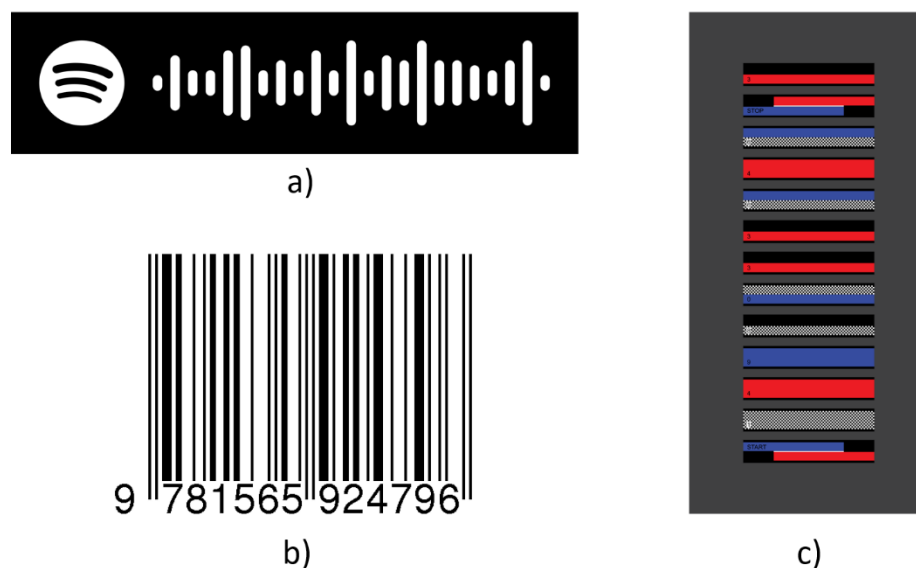


Figure 9 Different barcode styles. a) is a Spotify code, b) the EAN 13 and c) the CarTRAK ACI plate for railroad identification.

The storage capabilities of 1D barcodes can vary immensely depending on the used character set and the subsequent encoding as well as the length of the code. Typical codes include the Code 39 (ISO, 2023) and the Code 128 (ISO, 2007), the Universal Product Code (UPC) (Hong-ying, 2009) and European Article Number (EAN) (Hong-ying, 2009) with derived formats in Japanese Article Number (JAN) (Hong-ying, 2009) in Japan and the International Standard Book Number (ISBN) (Hong-ying, 2009) for books. Following is a little overview over these codes:

- **Code 39:** This is one of the most widely used 1D barcode types. It supports alphanumeric characters (0-9, A-Z, some special characters) and has a variable length. The typical storage capacity for Code 39 barcodes ranges from about 20 to 25 characters, although it can be extended by concatenating multiple Code 39 barcodes.
- **Code 128:** Code 128 is a higher-density barcode that can encode the entire ASCII character set, allowing for higher data density. It's capable of encoding a larger amount of data compared to Code 39. The storage capacity of Code 128 barcodes

can range from about 40 to 100 characters, depending on the specific variant and symbol size.

- **EAN/UPC:** EAN and UPC barcodes are commonly used in retail for product identification. They are numeric-only barcodes and have fixed lengths: 12 digits for UPC-A and 13 digits for EAN-13. These barcodes are primarily used for identifying products and don't hold substantial amounts of arbitrary data.

2D barcodes extend the limited storage space of 1D barcodes by encoding patterns in matrix form. The development started with stacked 1D barcodes and forms spanned the use of all kinds of patterns like rectangles, dots, hexagons, and other patterns. Figure 10 shows different forms of common and more exotic codes.

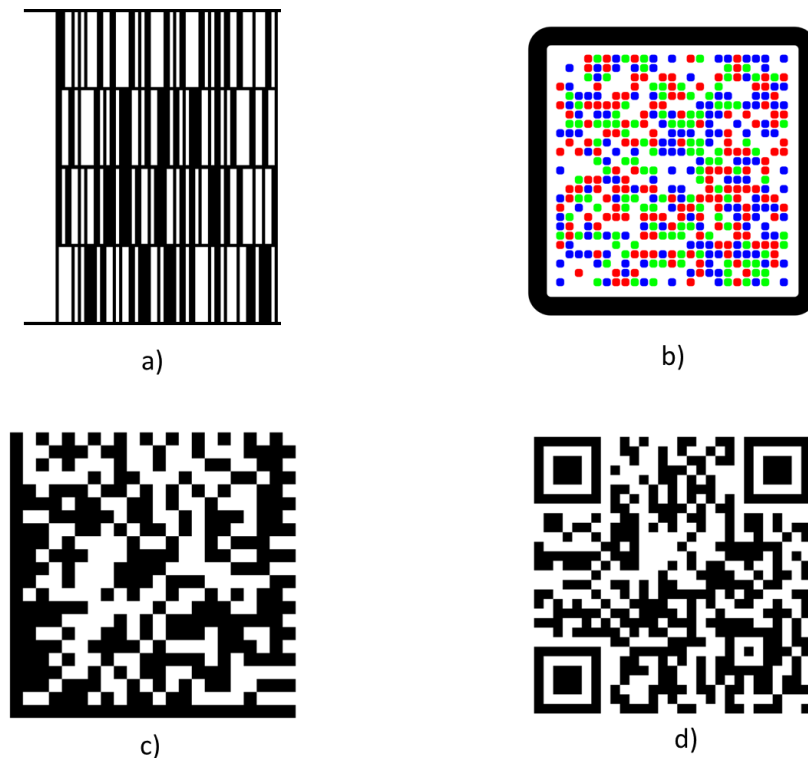


Figure 10: Assortment of 2D-Matrix codes from the Code 16K a), which is based on the Code 128, a colour coded cryptographic code b) developed in a hackathon for signing in e-banking and the industry and consumer common data matrix code c) and QR-Code d).

The wide range of patterns and variations in the representations make it difficult to make a general statement on the storage capabilities of the different kinds. In general, 2D-codes are not only capable to store single characters but encode data in the size of a few bytes.

Commonly used in the industry are data matrix codes while QR-codes have also been widely adapted to the consumer market (Tiwari, 2016). While data matrix codes can encode a limit of up to 2355 alphanumeric characters, QR-codes can store up to 4296 characters. This depends of course also on the size and the bits necessary to encode a character.

Additionally, 2D matrix code also offer mechanisms of error correction, that most 1D barcodes do not consider due to the limited pattern space.

The basic idea behind the error correction is to encode information redundantly within the code. As a result, error correction enables greater resiliency to errors occurring during

scanning or due to damages on the code. QR-codes and data matrix codes both use Reed-Solomon error correction algorithms for that (Reed, 2012). This group of error correction algorithms is widely used in storage systems and data transmission. Table 1 shows the four error correction levels that are usually used for QR-codes. The higher the error correction level is, the greater is the resiliency towards errors but the greater is also the reduction in storage capabilities.

Table 1: Typical error correction levels.

Error-Correction Level	Approximate Amount of Correction
L (Low)	7 %
M (Medium)	15 %
Q (Quartile)	25 %
H (High)	30 %

A code with a higher correction level reserves the correlating amount of correction in redundant information decreasing the actual maximum information content of the code accordingly. A higher error-correction level might nevertheless be attractive, when the code might be subjected to a rough environment getting damaged as this might otherwise affect the readability of the code.

4.2.2 Radio-based Tags

Radio-based tags store information using different physical effects. They consist of a radio transponder, a receiver and a transmitter which are electric and electronic components and are activated by electromagnetic fields (Domdouzis et al., 2007). Similar to optical codes, RFID-based identification happens using a Tag-Reader pair. A distinction is made between active, passive, and battery-assisted passive RFID-Tags. While passive tags operate completely on the energy received from the radio energy transmitted by the reader and are therefore the smallest and cheapest setup, battery-activated tags rely on an internal power source, that can enhance the response signal enormously or even enables independent sending (Domdouzis et al., 2007). Figure 11 shows the basic setup of a passive RFID chip.

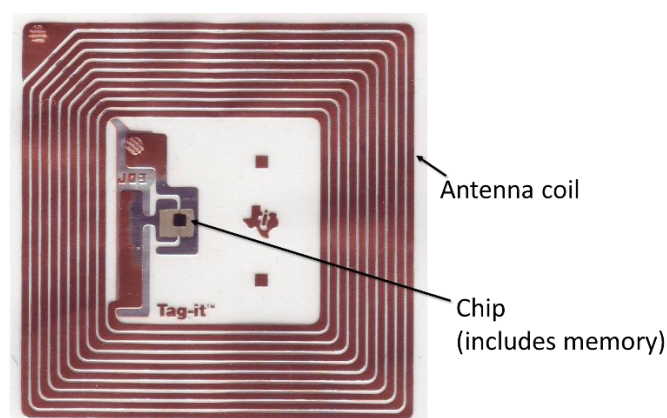


Figure 11: Composition of an RFID-tag.

Depending on the different tag technology, three types of reader setups are possible:

- Active Reader Passive Tag, where the energy for the reading process comes from the reader.

- Passive Reader Active Tag, where the energy for the reading process comes from the battery on the tag.
- Active Reader Active Tag, where the energy for the reading process comes from both the reader and the tag.

Most product tagging systems use the Active Reader Passive Tag setup as it is the cheapest alternative and usually sufficient for most scanning scenarios, e.g. in warehouses or production environments (Ting et al., 2011).

RFID-technology covers several different frequency bands from 120 kHz-24.125GHz (Domdouzis et al., 2007). A higher frequency band that has a higher bandwidth corresponds positive to the speed of data transmission and responsiveness but negative to the range of the transmitter. Depending on the application the frequency range is thus selected to optimise either range or speed and responsiveness. Common in payment systems or inventory tracking and supply chain management are frequencies in the High Frequency (HF) and Ultra-High Frequency (UHF) range (13.56 MHz up to 2.45 GHz). NFC as a subclass of the RFID communication technology is a popular technology for payment systems and operates within the HF range, typically around 13.56 MHz, and is used for close contact identification e.g. in payment systems. Typical ranges of passive tags within the HF and UHF spectrum range up to a few meters while NFC technology operates in the range of a few centimetres (Domdouzis et al., 2007).

Neither the tag nor the reader need to be openly accessible for the communication to work between the two parties, as the radio waves also cross through matter. This enables the embedding of the tag into the product itself and protecting it from direct physical damage and harsh environments. However, it does not cross easily through metals which impair the transmission of the signal. Different encapsulations and housings of the RFID chip enable further protection and resistance. Nevertheless, due to the magnetism-based technology utilized for the storage chip, the information on the chip might be susceptible to disturbance through electromagnetic fields (Domdouzis et al., 2007).

The storage capability of radio-based storage tags depends on the build in memory chip and typically range from a few bytes up to several kilobytes. The memory capabilities especially for passive tags, are constrained due to size and energy consumption limitations.

4.3 Product unique identification

The most important part of a DOME implementation with at least partly off-product data storage is the allocation of the right storage to capture the data of the individual product. The inclusion of an individual product identification on the DOME is a minimum necessity, so that the product item can be uniquely identified. A wide range of identification schemas exist for various purposes with a more or less standardized foundation. Nevertheless, a few key issues need to be addressed in general with identification systems:

- **Standardization vs Interpretation:** Many identification systems rely on the user to know what the identifier is and expresses. A simple example is the Global Trade Item Number (GTIN) for identification of product models which is often given as a plain 13-character number on a product. Without the knowledge about the meaning of the number the proper handling remains open to interpretation and vulnerable to misuse. This is especially problematic with identifiers that have no universal conventions and are defined very individually, e.g. product serial

numbers created by a manufacturer itself that form a unique manufacturer specific system.

The standardization of identifiers is one way to tackle this issue. GTIN, GLN, GSIN, UPC, EAN and SSCC (GS1 Germany GmbH, 2023) are some of the standards that are widely used in supply chain management and identification throughout the life cycle, and which are partly harmonized through the GS1 standardization non-profit organization. Through their standardization and widespread use, these identifiers are recognizable and interpretable by the user. This would also be a valid approach for more complex composite identifiers as shown in Figure 12, that need knowledge about the order of the combined sequences and their individual meaning and possibly dividers between the individual part of the identifier.

The opposite approach is to label the identifier in an interpretable manner. In the case of the serial number this could mean to provide a short descriptor with the identifier, e.g. S/N: or Serial No. as depicted in Figure 12. Using labels to mark the meaning of the identifier parts also helps to decode composite identifiers and prevents misinterpretation in the case of e.g. a not standardized order of the combined parts.

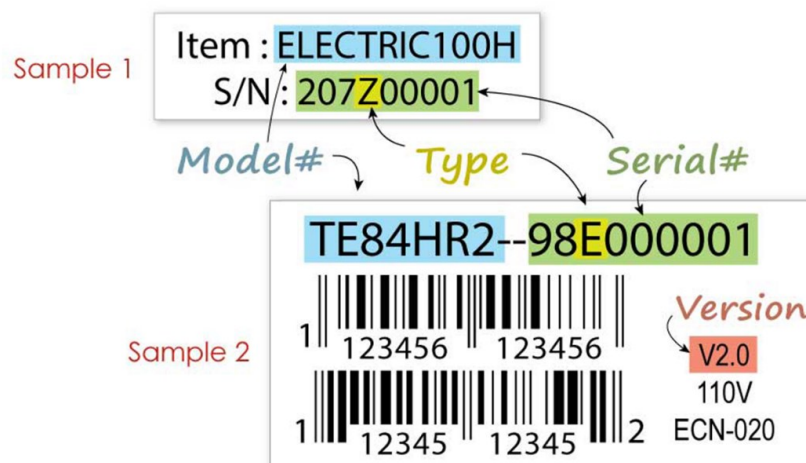


Figure 12: Samples of labelled and unlabelled composite identifiers containing identification on the product model and the product unit level (Shop Elite Screens, 2023).

- **Entry point:** Interpreting an identifier in the context of digital object memories first of all requires knowledge about the context of the digital data storage and its location. This relates to the question: Where do I type in the identifier to receive the assorted data? Modern approaches to identifier design, like the GS1 digital link as shown in Figure 13, therefore resort to the use of URLs that directly point to a data entry point.

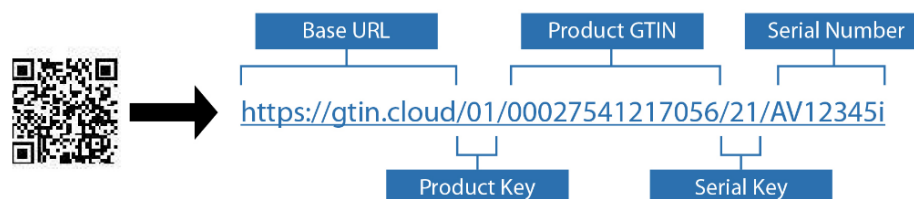


Figure 13: Example for GS1 digital link format utilizing a URL-based direct pointer to a product related dataset (Bar Code Graphics Inc., 2023).

Different challenges arise from the use of URLs as pointers, as the access to the assorted data must never change throughout the life cycle of the product or be consequentially forwarded to the correct entry point to not break the link between the physical and the storage layer of the DOME.

- **Verification:** To ensure a correct linkage between the physical product and its data storage the link must be unique and ensure some kind level of protection against fraud. Ensuring uniqueness of the identifier prevents confusion and incorrect allocation of product data. Organizations like the GS1 offer a registration service with their own identifiers and therefore guarantee the uniqueness of a purchased identification number.

The prevention of tempering with either the physical product tag or the digital record is a more difficult challenge. Especially the detection of fake products is an important issue here. One way to use encryption in the identifier or adding a cryptographic hash to the identifier, therefore complicating it to reproduce a valid identifier. Another promising approach to tackle this problem is currently being explored in the wake of the rise in blockchain powered solutions. The strength of these solutions regarding fraud protection lies within the distributed ledger about product transactions among all participants of the blockchain. The many-eyes-principle used in verifying transactions and the subsequent change to the ledger makes it difficult to tamper with all records at once therefore ensuring the integrity of the record of transactions. Figure 14 shows this approach developed in the MediLedger network trace medical supply chains (Chronicle Inc, 2021).

Blockchain-based product verification using MediLedger

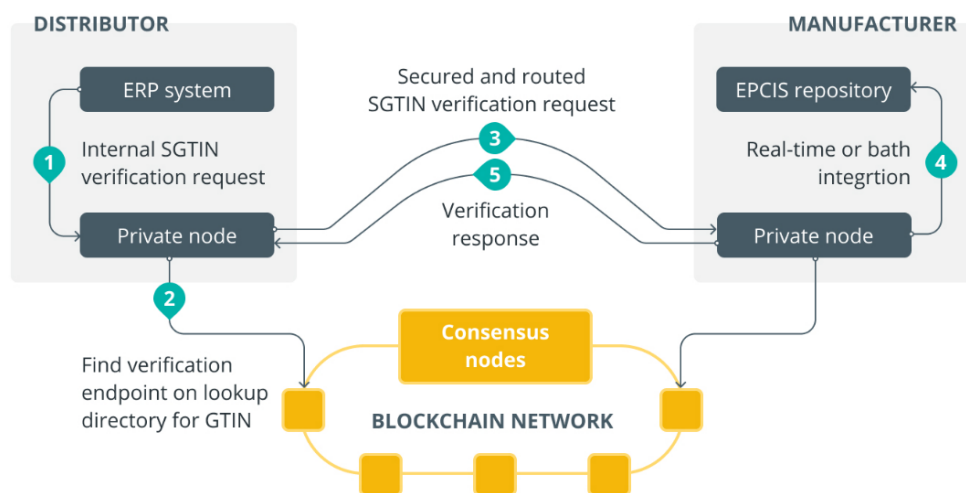


Figure 14: Blockchain-based identification verification using distributed ledgers to find the verified data owner (Onkar Singh, 2023).

5 DEVELOPMENT OF A DIGITAL OBJECT MEMORY SERVICE

As described in the Deliverable 5.1 of the CircThread project, the DOME service is an essential part of the CircThread platform layer as shown in Figure 15. It is intended to implement a primary referenced DOME approach. This means that the platform primarily supports systems in which the physical data carrier on the product contains only the identifier and no additional data. The decision to support this reduced data carrier content doesn't exclude the compatibility with other forms like storage or smart DOME but reduces the requirements to implement a platform-compatible identifier.

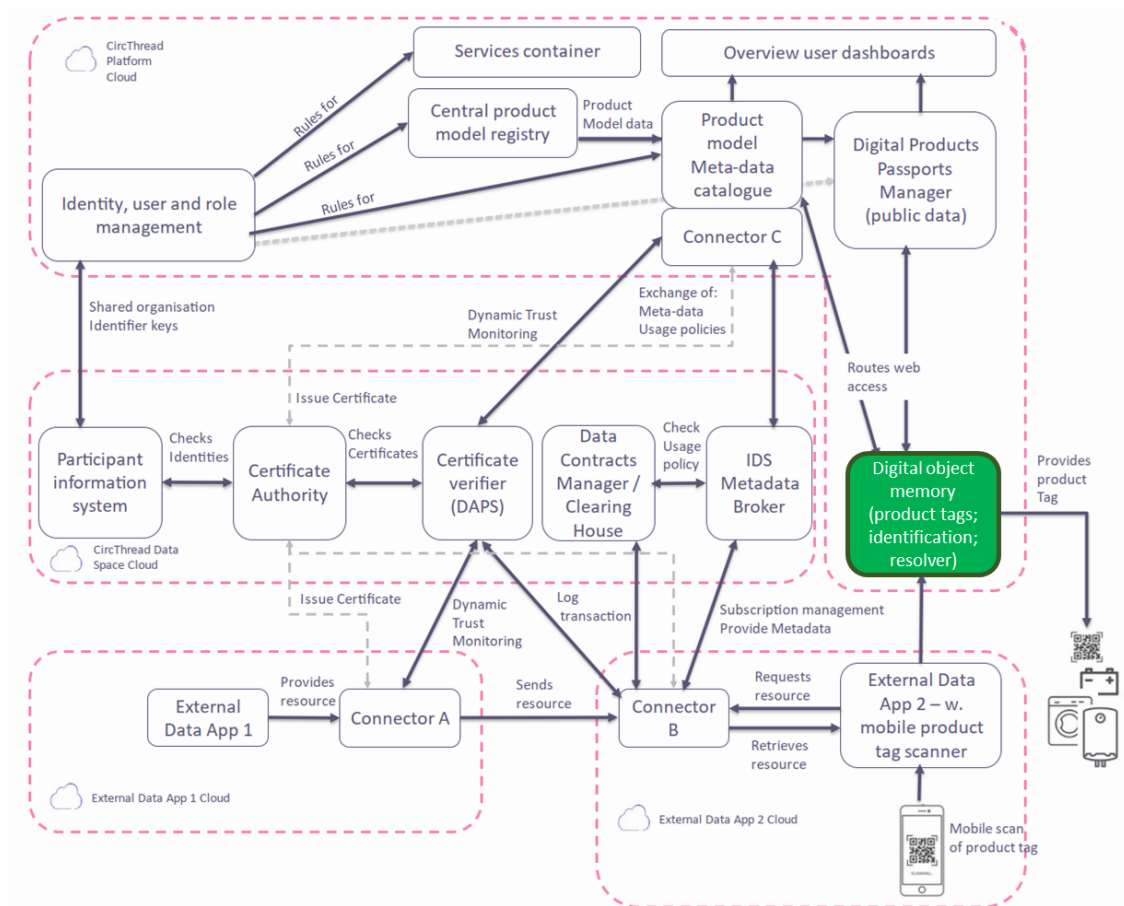


Figure 15: CircThread ecosystem architecture.

The CircThread ecosystem takes a very distributed approach to the data storage about a product. The platform distinguishes between product model and product unit level information provided by the Product Model Meta-Data Catalogue and Digital Product Passports Manager. It also follows a policy-regulated approach to individual data sharing via the Data Space Layer. Additionally, the platform is aiming at a modular approach regarding the assorted services. That means that services must be dynamically replaceable or similar ones coexist in parallel. The ecosystem will be open for these product services to be registered with the platform and made accessible via the platform middleware. The DOME service therefore has to manage the connection to several dynamically changing data providing services.

Additionally, the platform will be open to multiple applications in the external application layer connecting to the core modules. These external apps will be the entry point for the user to interact with the platform and the DPP itself. Users can be all actors active

throughout the product life cycle which have different interaction intentions regarding retrieved or updated information.

Under these conditions, there are two tasks the service must fulfil:

1. Interpret the intention of the user scanning the product identifier with regards to information update and retrieval and recommend suitable available information.
2. Resolve a scanning request to the platform by providing a connection to the service that provides the right information for the request.

Under these conditions, the Digital Object Memory service architecture can be described using DOME terminology as depicted in Figure 16. The product tag represents the physical layer on the product. The external application serves as a client to the platform and the Event Interpretation and Information Recommendation service and Resolver as well as the storage layer are part of the platform core modules. The Resolver and the Event Interpretation and Information Recommendation are considered as part of an activity module, instilling intelligence, and autonomous behaviours to the DOME of the product.

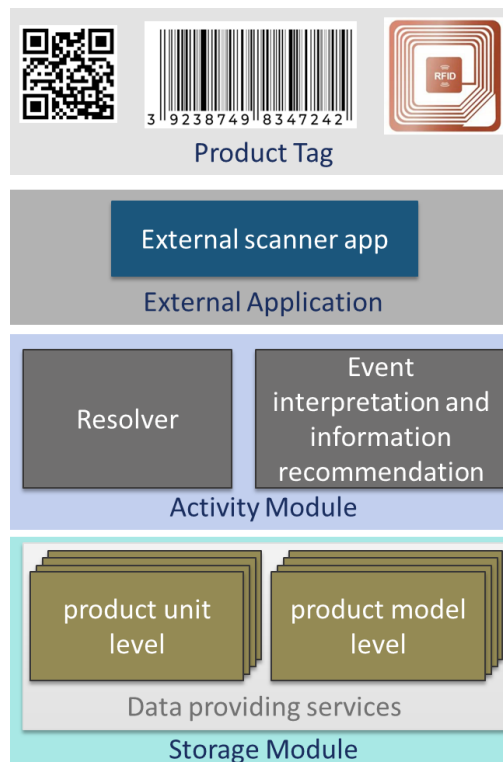


Figure 16: Description of the CircThread ecosystem using ADOME terminology.

The Digital Object Memory service with the assorted services of the platform can therefore be considered as an implementation of an ADOME. The following subchapters discuss the implementation of both parts of the activity module as core modules of the platform. They are implemented as separate microservices under the single responsibilities design paradigm.

5.1 Microservice 1: Resolver

The Resolver is a central part of the request and update structure of the DPP data on the CircThread platform. It operates as an interpreter of the product tag and router to the data providing services. As such, the Resolver only manages operations with the involvement

of the product identifier. This excludes the creation a new DPP with the consequential creation of an identifier which requires no allocation solely on the identifier and is solved via a direct connection to the storage module.

Figure 17 exemplifies the process of a data request to the DPP service in the following 5 steps:

- First, based on the scan of the identifier the client application starts a data request to the platform.
- Second, a request handler in the middleware forwards the request also containing the product identifier to the Resolver microservice.
- Third, the Resolver locates the responsible data providing service and optionally can execute callbacks regarding the availability and integrity of the data. It will then respond with an executable retrieval request URL for the data providing service to the middleware.
- Fourth, the middleware can then execute the request URL to retrieve the actual data.
- Finally, as a result the data is forwarded to the client application.

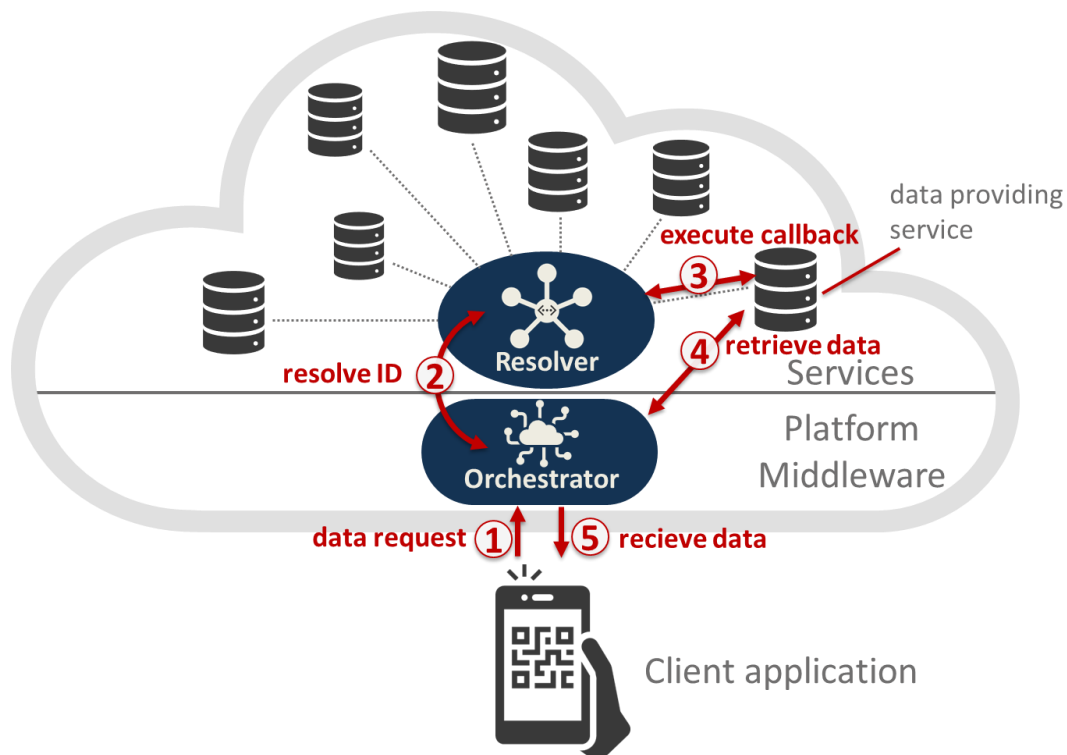


Figure 17: Schematic information request sequence to the CircThread platform through a client application using the product identifier.

5.1.1 Staying up to date with the data providers

The position of the Resolver between a dynamic landscape of services and the middleware poses the challenge of maintaining an overview of the data providing services and their APIs. Two approaches have been discussed to tackle this challenge:

1. **Actively managed Resolver:** In this approach the Resolver actively manages the knowledge about the data providing services and their APIs. This requires the resolver to actively manage a directory of these services (as discussed, their number might fluctuate over time) and changes to these need to be implemented

into the Resolver actively. The approach would imply that the Resolver needs to receive notice of all changes to the service landscape which is not only difficult to implement but also difficult to enforce. Again, there would be two solutions to this problem. Either the Resolver would run a crawler exploring changes to the service landscape and the APIs and autonomously interpret the changes or the other service providers would actively need to report changes to the Resolver service. The different possibilities need to be discussed in the light of the modular platform philosophy. Therefore, it might be possible that multiple resolver services exist in parallel, making it difficult for the service providers to keep an overview of all services you need to communicate changes to your API too.

2. **Passive Resolver:** This approach involves the creation of a new middleware module that is also dedicated to the task of centralized service discovery (Team, 2015). This means the platform provides a Service Registry which forces service providers to give a description of their API in order to join the platform and offer their service. A service requiring a complete overview of the assorted services and their APIs could be authorized to consume this registry and receive an up-to-date overview of all assorted services including the Resolver services as depicted in Figure 18. This approach offers more flexibility, as the service providers have a central authorised body to which they must report changes. Doing so supports the philosophy of the flexible modular platform.

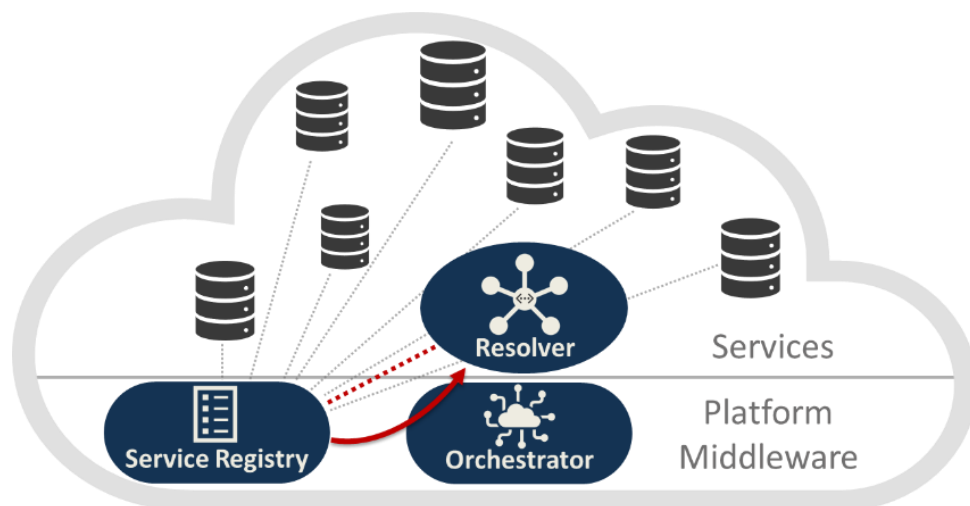


Figure 18: Passive resolver approach including the registration of all services with a central service registry.

We decided for the passive resolver approach which raises the question of what information the service providers would need to provide during the process of signing their service up to the platform. A first draft is depicted in Figure 19. It contains a service type descriptor, information about methods for optional callbacks and the address parameters to generate the target URL for the service response. With this registry, automatic processing of the services and generation of REST-based data retrieval requests is possible. However, the depicted registry example is only a first draft used for testing the described automatic API processing and for discussion with the other partners. A detailed version is currently developed together with the platform middleware and improved iteratively together with the other partners.

The vocabulary terms used to describe the metadata information about the services are being defined consequentially and as such are added to the platform vocabulary developed in deliverable D4.1. The finalized definitions are therefore released at a later point of the project, but Table 2 already shows the definitions of some proposed terms. These need to be harmonized with existing definitions and ontologies e.g. the hydra core vocabulary².

<pre>{ "Product_Model_Metadata_Catalogue":{ "SERVICE_TYPE": "product model level data provider", "ROOT_ADDRESS": "http://95.60.60.92:5000", "ACCESS_TOKEN_PARAMETER": "?accessToken=",</pre>	general service information
<pre> "ENDPOINTS": { "ALL_METADATA":{ "REST_TYPE": "GET", "ENDPOINT": "/getAllMetadata", "ID": "GTIN", "KEY": "metaData", "ACCESS_TOKEN_REQUIRED": 1 }, "ALL_INSTANCES":{ "REST_TYPE": "GET", "ENDPOINT": "/getModelList", "ID": "GTIN", "KEY": "idModel", "ACCESS_TOKEN_REQUIRED": 1 }, }, "DATA_RETRIEVAL":{ "REST_TYPE": "GET", "ENDPOINT": "/getModelData", "ID": "GTIN", "KEY": "idModel", "ACCESS_TOKEN_REQUIRED": 1 }, "DATA_UPDATE":{ "REST_TYPE": "POST", "ENDPOINT": "/postModelEditData", "ID": "GTIN", "KEY": "idModel", "ACCESS_TOKEN_REQUIRED": 1, "UPDATE_VALUE_STRUCTURE": { "idDataCatalogue": 0, "dataModel": "string", "categorized": true, "categories": "string", "mandatory": true, "accessibility": true, "dataResponsible": "string", "source": "string", "unit": "string", "validDate": "string", "observations": "string" } } } }</pre>	information for callbacks
	request-related information

Figure 19: First draft of the required service metadata within the service registry on the example of the product model metadata catalogue developed by project partners.

² <https://www.hydra-cg.com/spec/latest/core/>

Table 2: List of proposed definitions of terms used that are defined for the service registry.

Term (labels)	Definition
all instances	This call is meant to retrieve all product or product model instances on which the data provider has information
all metadata	With this call you can retrieve all metadata terms the data provider can give about a product
data retrieval	This designates endpoints, with which single data sets about an instance can be retrieved
data update	This describes endpoints with which single data entries about instances can be altered in a targeted manner
endpoints	Endpoints serve as entry points for communication with a web service or API, each typically corresponding to a specific function or resource provided by that service. They define where clients can make requests and receive responses.
key	Key refers to the identifier or label that is used to reference a specific piece of data within a collection or structure, usually associated with its corresponding value.
rest type	This defines the type of REST method under which a call to an endpoint is made (GET, POST, PUT, DELETE)
root address	Root address of the API under which the server is accessible
service name	Name of the service
service type	This denotes the type of service (product model level data provider, product unit level data provider, assorted service, ...)
update value structure	Parameter structure in which an update call needs to be made to the REST endpoint of a data provider in order to execute an update request.

5.1.2 Integrate Legacy identifiers

The core task of the Resolver service is the resolution of the product item's individual identifier. Today already established identification systems exist, primarily developed by the GS1, based on a business model which is a commercial paid-for-per-number-of-identifiers service. The creation of open-source identification schemata as an alternative to proprietary systems is necessary if the aim is to remain independent of commercial identification service providers. Thus, allowing for multiple systems, commercial and non-commercial to co-exist together. Additionally, as the current draft of the DPP standardization effort (Docsroom, 2023) does not stipulate the definition of a common identifier schema, we expect the creation of different identification schemata in the future and the further existence of already established solutions.

This means that the Resolver must be able to interpret not only established identifiers but also be extendable to new ones. Also, the CircThread platform as minimum will support referenced DOME architectures where the physical data carrier does only contain the identification code and not necessarily additional information on the interpretation. Therefore, the Resolver needs a flexible interpretation schema for many different combinations of simple strings containing for example identifiers on the product model and the product unit level.

The central component of the Resolver is therefore designed as a **pattern identifying algorithm connected to a database of registered identifier patterns**. In other words, the rulesets by which identification happens are stored in a database, setup in a structured manner, so as to make expansion of the rules possible. New identifier schemata can in this setup be registered with the Resolver service through the services API to expand the support for it. Coming back to the example from Figure 12 the Resolver must interpret the code *TE84HR2--98E000001* and conclude that *TE84HR2* represents a model level id, -- a divider and *98E000001* the unit level id. The schema for the composition of the code could in this example be:

*TE<number><number><letter><letter><number>--
<number><number>E<number><number><number><number><number><number>*

To formalize this, we want to describe the pattern above using regular expressions first used by Stephen Cole Kleene in 1951 (Ashby, 1951):

(TE\d{2}[A-Z]{2}\d)--(\d{2}E\d{6})

It uses the following notations:

- \d{2}: represents 2 digits
- [A-Z]{2}: represents 2 capital letters.
- \d: represents a single digit
- (): encapsulates a subexpression and groups the different parts of the code.

A comprehensive overview of the notation can be found in Stubblebine, 2003.

The first task after formalizing the code is to match an arbitrary identifier string against the exemplified schema. The second task would then be to parse it at the right position and identify the parts correctly.

Most modern programming languages already support the pattern matching using regular expressions and offer operations to be executed on matching parts. Python's build-in module *re-regular expression operations*³ uses the backtracking algorithm (Friedl, 2006). The library supports the following operations:

- final: Returns a list with all matches
- finditer: Returns an iterator of match objects
- match: Checks whether a match occurred at the beginning of the string
- fullmatch: Checks whether the entire string is a match
- search: Returns a match object if there is any
- split: Returns a list where the string has been split at each match
- sub: Replaces one or many matches with a string

Using this basic functionality of many programming languages we can build a matching based parser and a format to describe new patterns for an identifier database. The format also follows the two-step approach of format matching and parsing. Figure 20 shows the format for the described sample pattern above in a JSON type representation. The expression features a mandatory field called *ENTIRE_PATTERN* which describes the order of the composite identifier including delimiters. It also includes the names of the elements

³ <https://docs.python.org/3/library/re.html>

the identifier is composed of. This way we can use the *fullmatch* operation to confirm that the identifier fits this pattern. For the next step we extract the relevant parts from the identifier using the *search* operation.

```
"sample pattern": {
  "ENTIRE_PATTERN": "{product_model_number}--{serial_number}",
  "product_model_number": "(TE\\d{2}[A-Z]{2}\\d)",
  "serial_number": "(\\d{2}E\\d{6})"
}
```

Figure 20: Regular expression type pattern storage format for the matching based parser.

One issue that needs to be discussed with the *fullmatch* operation is that the order of the composite sub elements must be consistent with the pattern stored in the data base. It can be discussed that this case will not occur because identifiers without labelled element need to be standardized to ensure interpretability. Nevertheless, we implemented a fallback solution that tries to match all given elements of a stored schema. When all elements fit, it is probable, that the identifier matches the pattern in a different order. That does of course not cover the case where only some of the elements are actually used in the identifier.

This case becomes more relevant when we have more variable but labelled product identifiers like the digital link by GS1 (*GS1 Digital Link Standard*, 2023) that are more modular. The pattern description is much broader here as elements can be changed in their order. Also, the format is widely used so the patterns for some elements might also be very general. Nevertheless, the elements are recognizable by standardized labels. The element follows after a delimiter with which it is also separated from the next label. Figure 21 shows the description for the digital link type.

```
"digital link type": {
  "ENTIRE_PATTERN": "https?://([^/]+)/.*\\d{2}/.*",
  "GTIN": "01/(\\d{13})",
  "consumer_product_variant": "22/([^/|$]+)",
  "batch or lot number": "10/([^/|$]+)",
  "serial_number": "21/([^/|$]+)"
}
```

Figure 21: Description of the digital link type identifier using the Resolver identifier storage format.

The naming of the different identifiers is not predefined but needs to be consistent with the identifier naming used with the data providing service and consequentially the CircThread Service Registry. From this point the only term that needs to be added to the platform vocabulary is the definition of the term *entire pattern*.

5.1.3 Resolver integration

The entire Resolver microservice is developed as a webservice providing a RESTful API as interface which offers two categories of endpoints. The first type is related to the execution of Resolver requests as described in Figure 17. The other type is related to the identifier type data base and covers the usual CRUD (Create, Read, Update, Delete) operations to manage the identifier database. As long as the platform has no implemented identity management, the Update and Delete options are not open publicly and will later

be added with platform and service admins access only. Figure 22 shows the, under these conditions preliminary, documentation of the Resolver API.

Resolver API 1.0.0 OAS 2.0

Resolver Requests

GET

/resolver/resolveDataRequest/{product_identifier}

Resolve Data Request

Parameters

Try it out

Name	Description
access_token * required string (query)	<input type="text" value="access_token"/>
term_list * required string (query)	<input type="text" value="term_list"/>
product_identifier * required string (path)	<input type="text" value="product_identifier"/>

Responses

Response content type application/json

Code	Description
200	Successful operation

POST

/resolver/resolveDataUpdate/{product_identifier}

Resolve Data Update

POST

/identifier/registerNewIdentifier

Register New Identifier

GET

/identifier/getAllIdentifiers

Get All Identifiers

Figure 22: Open API documentation of the Resolver API.

The Resolver has several components that play together in processing a request. While the service interface handles incoming requests and operations on the identifier database, background processes in the update module handle the interaction with the platform service registry. The update module executes a query to the service registry to check for updates and copy the current state to another internal database one every 24 hours. This is meant to reduce traffic with the registry, as not every request to the Resolver entails a request to the registry and makes the service resilient more to communication failures with external services (e.g. server downtimes). The information on the data providing services is therefore collected from this internal database a processed by the Resolver core. These relationships are outlined in Figure 23.

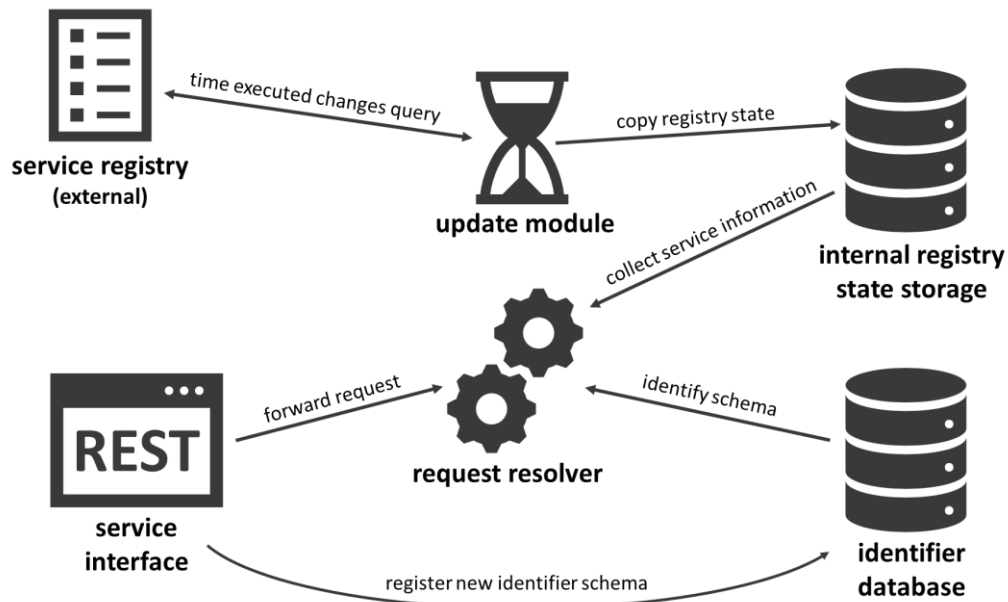


Figure 23: Resolver modules and their interaction in resolving a request to the service.

Technically, the service is developed using Python and MySQL databases. The service as developed and supported by the project partners is internally available on the CircThread GitHub⁴.

⁴ <https://github.com/CircThreadH2020/resolver> (access will be provided on request)

5.2 Microservice 2: Event interpretation and information recommendation

Recommendation engines (RE) are a type of information filtering systems that have been well researched in recent years. They either try to provide a set of personalized recommendations for products or a service that are expected to be useful for a particular user, or they try to predict whether a particular item is of interest to a user or not. This is done, for example, based on his or her previous preferences and the preferences observed among similar users. The recommendation is often provided in the form of a list of ordered items. The term 'item' refers to the entity recommended by each RE (e.g., products, terms, songs, movie, web pages, etc.). A user can interact with the RE through a particular interface to receive, evaluate or select the recommended items (Gatzioura et al., 2019; Ricci et al., 2011/2011; Zhang et al., 2021). Four main techniques have been established over time: Content-based, collaborative, graph-based, and hybrid filtering (Adomavicius & Tuzhilin, 2005; Gatzioura et al., 2019; Melville & Sindhvani, 2017; van Capelleveen et al., 2021).

1. **Content-Based Filtering:** This approach leverages the characteristics of items to recommend other items similar to those a user liked, viewing, rated highly in the past. It mainly operates on the premise of similarity between items.
2. **Collaborative Filtering:** Unlike content-based filtering, collaborative filtering relies on user behaviour. It identifies patterns in user behaviour to make recommendations, often grouping users with similar tastes and suggesting items based on what other users in the same group have liked. It leverages the preferences, ratings, and actions of other users on the network to find items to recommend.
3. **Graph-based filtering:** Graph-based filtering utilises a graph structure to visualise the relationships and interactions between users and objects. This approach applies nodes for users and objects and edges to represent their relationships. It enables in-depth analysis of relationships and patterns that may not be discernible in other methods. Specifically, it is valuable for intricate recommendation scenarios where the connections between distinct entities are significant. Graph-based filtering is a versatile approach that can tackle challenges like the cold-start problem, which is explained in the following section.
4. **Hybrid Filtering:** Hybrid filtering combines the strengths of the different recommendation techniques to provide more robust recommendations. Different strategies can be used, overcoming the limitations of the approaches.

While these four techniques are already used on a plethora of digital platforms, social networks, and e-commerce, REs have not been explored for Circular Economy Information Systems or DPPs. This is also reflected in the current literature, where either only REs or the DPP are discussed and researched in the context of a CE, but not in conjunction with each other (Gatzioura et al., 2019; van Capelleveen et al., 2021). Consequently, there is a need to develop an intelligent RE that effectively supports users in handling the information overload and reduces the complexity of decision making. By selecting and analysing the right data, RE support users' decision-making processes and increase their ability and the quality of their decisions by enabling them to find items they are likely to want to receive (Adomavicius & Tuzhilin, 2005; Gatzioura et al., 2019). Groundbreaking

research could be conducted in this area to investigate how recommendation algorithms can be adapted to create customized DPPs for CE actors or to support decision-making processes in a CE by using DPPs.

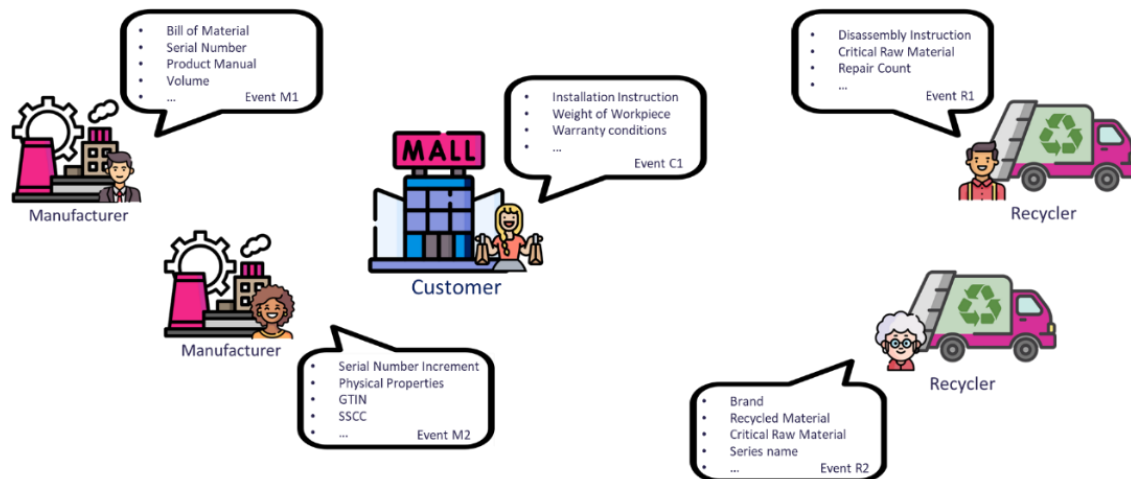


Figure 24 Different patterns of demand for information from the individual players in the life cycle.

So why does it make sense to have a recommendation system for the CE or a DPP at all? Looking at the product life cycle, it becomes clear that the different stakeholders and actors in the product life cycle need different information depending on a specific event. It is even possible that participants with the same role in the life cycle need varying information, as shown in Figure 24. This is confirmed by the CircThread pilots, where partners require varying information for individual events. In the context of CircThread, they will receive this information from multiple service and data providers that are part of the modular and dynamic CircThread platform. With the CircThread vocabulary (cf. Deliverable D4.1⁵ and GitHub⁶), a standardized terminology has been developed that is used by the various services to describe and receive the information from the services. To explain it more precisely with a key-value pair, on the one hand there are various terms (keys) that describe information (e.g. brand, GTIN, deviceType etc.) and on the other hand data that is the actual piece of information (value), such as "Gorenje; 8563465501523; washing machine".

⁵ <https://circthread.com/download/deliverable-4-1-product-meta-data-catalogue-structure-for-linking-product-information/>

⁶ https://github.com/CircThreadH2020/ct_product_metadata_vocabulary (access will be provided on request)

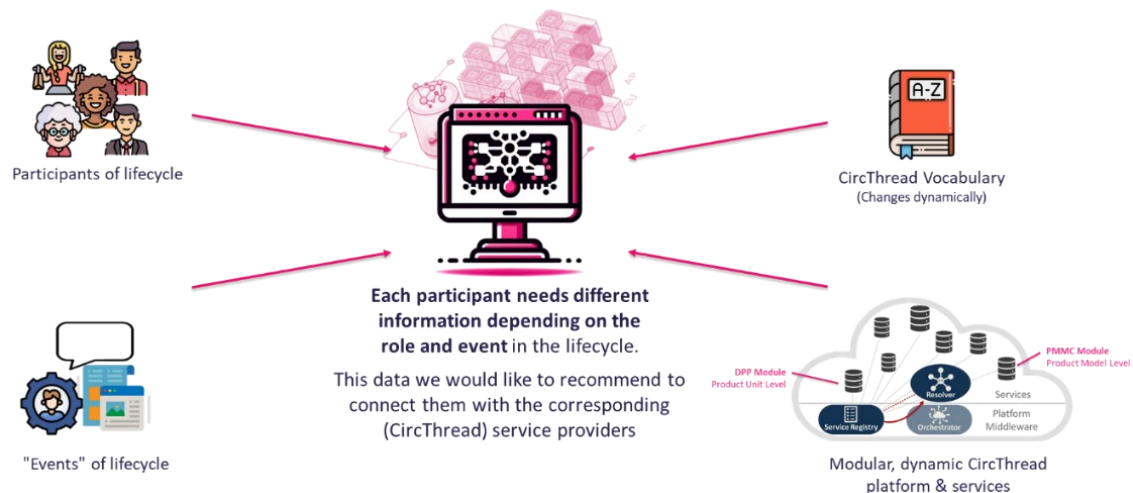


Figure 25 Motivation for a recommendation system for the circular economy.

The goal of the *Event Interpretation and Information Recommendation service*⁷ is to suggest key terms to the various stakeholders, depending on the role and life cycle phase they are in (cf. Figure 25). This will create the foundation for an individual, industry specific DPP that provides customized information and does not overload the stakeholders with unnecessary information. Accordingly, only relevant terms will be recommended. The DOME Resolver will then provide the URL endpoints (value). The following sections explain the architecture of RE and its integration into the Resolver in detail.

5.2.1 System architecture

This chapter aims to present the architecture of the sophisticated recommendation system. It first deals with the creation and interpretation of so-called Events and the recommendation of information, and then with the integration into the Resolver. Figure 26 provides an initial system overview.

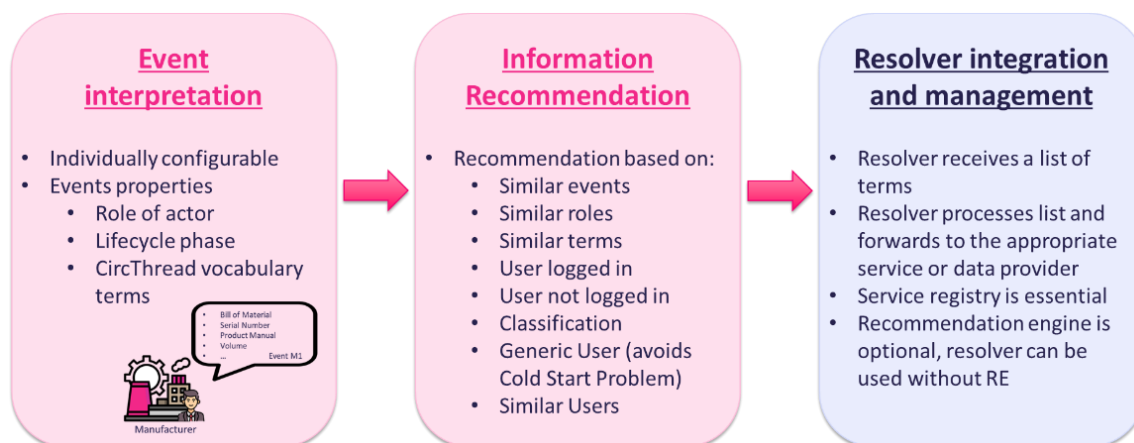


Figure 26 Overview of the process of the Event Interpretation and Information Recommendations through to integration in the DOME Resolver.

Technically, both a front-end and back-end have been implemented for the open-source recommendation system. On the one hand, Python FastAPI was used for RESTful API

⁷https://github.com/CircThreadH2020/circthread_event_interpretation_and_information_recommendation_engine (access will be provided on request)

The two Neo4j databases as well as their FastAPI integration will be explained in Chapter 5.2.2 and 5.2.3. Please note that the service can be used as an extension to the Resolver service but is not explicitly required by the Resolver.

5.2.2 Description of data structure of “Generic User” database

The “Generic User” Neo4j databases was developed to counteract the well-known cold start problem. It is a common significant challenge in recommendation systems that arises when there’s insufficient data available to make accurate and personalized recommendations. This issue commonly occurs in scenarios such as when new users enter the system, lacking a history of preferences and patterns, making it difficult for the engine to generate tailored suggestions (Lam et al., 2008; Lika et al., 2014). It’s also evident when dealing with new users, where the lack of recent or frequent interactions leads to a dearth of preference data (Le Son, 2016). Moreover, the introduction of new items to the recommendation pool that haven’t been widely interacted with poses a similar challenge, as there’s limited behavioural data to inform recommendations (Hasan & Khatwal, 2022; Yuan et al., 2016). In the context of the CE and the product life cycle, the cold start problem manifests itself in the fact that individually tailored information must be provided for each participant in the product life cycle. This applies to various life cycle phases – from production and use through to disposal or recycling. Each participant, whether producer, consumer, recycler, or disposer, needs specific information based on their role in the life cycle, the current phase of the product, specific events in this phase or their personal preferences. The main problem of the cold start is that it is difficult to recognise these specific information needs and generate appropriate recommendations when little or no data is initially available for a new participant or product.

To overcome this challenge, the so-called “Generic User” was developed. This is a predefined Neo4j graph database that reflects the various interests of the stakeholders and contains some of the terms they require. It provides initial term recommendations for new users. Figure 28 shows the schema of the generic user.

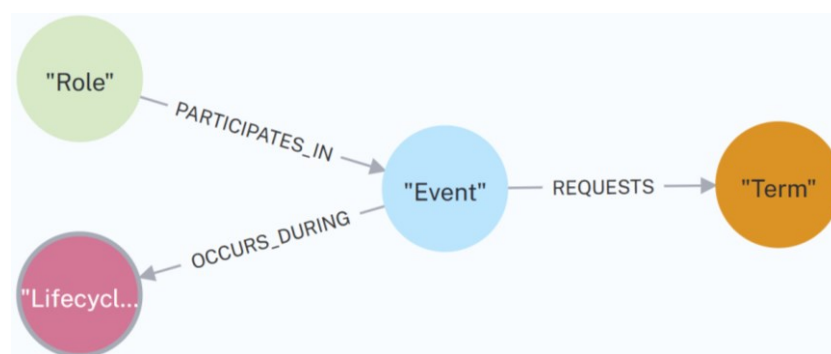


Figure 28 Neo4j graph schema of the generic user.

There are four types of nodes:

- *Role* (Green node) with property: name.
- *Event* (Light blue node) with property: name.
- *LifecyclePhase* (Pink node) with property: name.
- *Term* (Yellow node) with property: name. They are words from the CircThread vocabulary, the standardise terminology (cf. Deliverable D4.1 and GitHub).

Moreover, there are three relationships between the nodes:

- One *Role* can PARTICIPATE_IN one or more *Event(s)*.
- An *Event* REQUESTS (with property: request_by *Role*) one or more *Term(s)*.
- An *Event* OCCURS_DURING one or more *LifecyclePhase(s)*.

Several predefined Events have been created based on legal requirements or standards, such as IPC family 257x, and iteratively improved in workshops together with experts and CircThread partners. The Event templates are linked to Roles and LifecyclePhases so that recommendations can be created based on them without prior knowledge of a user. By logging into the IDM (identity, role and user management module) on the CircThread platform, the role and current life cycle phase can be queried, allowing relevant terms to be suggested to new users on a personalized basis based on their role. The generic user is therefore well suited to avoiding the cold start problem. Over time, the generic user is improved by the second, dynamic Neo4j database based on threshold values.

Figure 29 shows an exemplary excerpt of an event (blue node) of the generic user. “UC1_Product_information_entering_the_recycling_process_and_impact” is one of the events defined in deliverable [D2.3](#). The events of this deliverable were taken as the starting point for the generic user. Through workshops and research, relevant terms (yellow nodes) such as firstSerialNumber, deviceCategory etc. were assigned to the events. The various “REQUEST” nodes have a “requested_by” attribute, which is linked to the role of an actor in the life cycle phase.

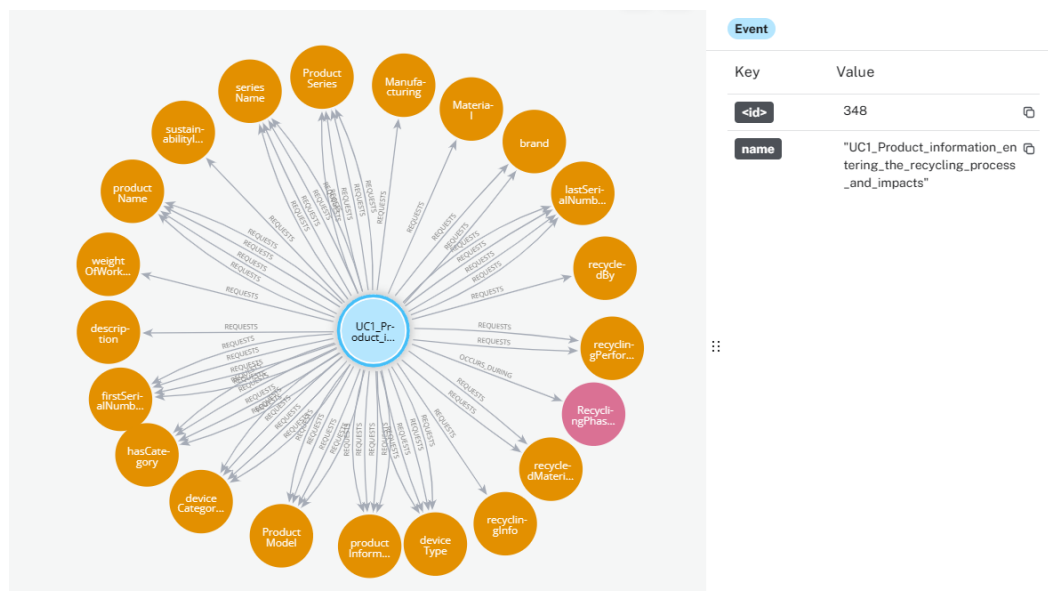


Figure 29 Extract of an example of an event for the generic user from Neo4j.

As already mentioned, the generic user is improved over time by the second Neo4j database, which is based on different prerequisites. These are explained in the next chapter.

5.2.3 Description of data structure of dynamic user behaviour database

It is evident that both life cycle information and the CircThread platform, including its vocabulary, will change and evolve over time. To accommodate this, a dynamic database is implemented alongside the predominantly static generic user database. This second Neo4j database dynamically captures current user behaviour through individually created event

templates, continuously improving recommendations, and refining the initial generic user model based on interaction thresholds. The dual database strategy is essential for effectively managing new term recommendations and adapting to constantly changing user data and terminology, allowing the service to play a critical role in promoting sustainable product life cycle management and a customized DPP. Figure 30 shows the schema of the dynamic user database.

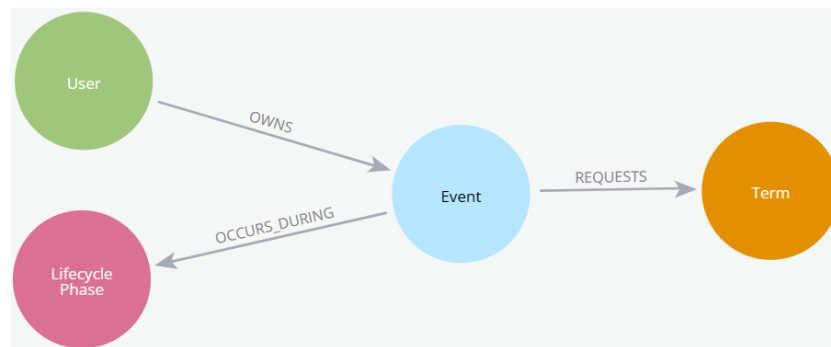


Figure 30 Neo4j graph schema of the dynamic user database.

There are four types of nodes:

- *User* (Green node) with properties: role, user_id
- *Event* (Light blue node) with property: name.
- *LifecyclePhase* (Pink node) with property: name.
- *Term* (Yellow node) with property: name. They are words from the CircThread vocabulary, the standardise terminology (cf. Deliverable D4.1 and GitHub).

Moreover, there are three relationships between the nodes:

- One *User* can OWNS one or more *Events*.
- An *Event* REQUESTS (property: request_by: *Role*) one or more *Terms*
- An *Event* OCCURS_DURING one or more *LifecyclePhase(s)*

Create Event Template

Template Name

Lifecycle Phase

DesignPhase

Add Term

Add Term

Save Template

Recommendations for Role

- + GTIN
- + EPRELabel
- + Product brand
- + collectionInstruction
- + Product serial number
- + EPREInfo
- + BOM
- + criticalRawMaterial
- + lastSerialNumber
- + serialNumberIncrement

Figure 31 Example for the creation of a customized Event template.

To utilise this module and establish custom Event templates, users are required to register or log in via the front-end. Presently, login authentication is managed by a SQL database that retains only the username, password, and role. In WP6, the login process will transition to the IDM system, which will enable role inquiries to be made. The Neo4j user database is dynamic and only stores the internal, non-public ID alongside the user's role

from the SQL database. This means that any links to the user's true identity cannot be established.

An example of a user-defined event template creation is illustrated in Figure 31. Here the user can specify the name of each event and its corresponding life cycle phase and add various vocabulary terms to create a template for a particular event. On the right side of the graphic, an initial recommendation is made by querying the life cycle role in the backend and suggesting recommendations based on the generic user. This ensures that new users receive useful term recommendations. Once confirmed, the created event is stored in the dynamic Neo4j database, which enables the updating and deletion of existing events, and the linking of different events via user or life cycle phase classes (see Figure 32). Moreover, it enables the use of further recommendation algorithms, which will be discussed in the next chapter. As part of the continuous improvement of the generic user, a comparison is made daily between the addition and deletion of terms. Any term which exceeds or falls below a certain threshold value is added to or removed from the generic user, forming the basis for recommendations.

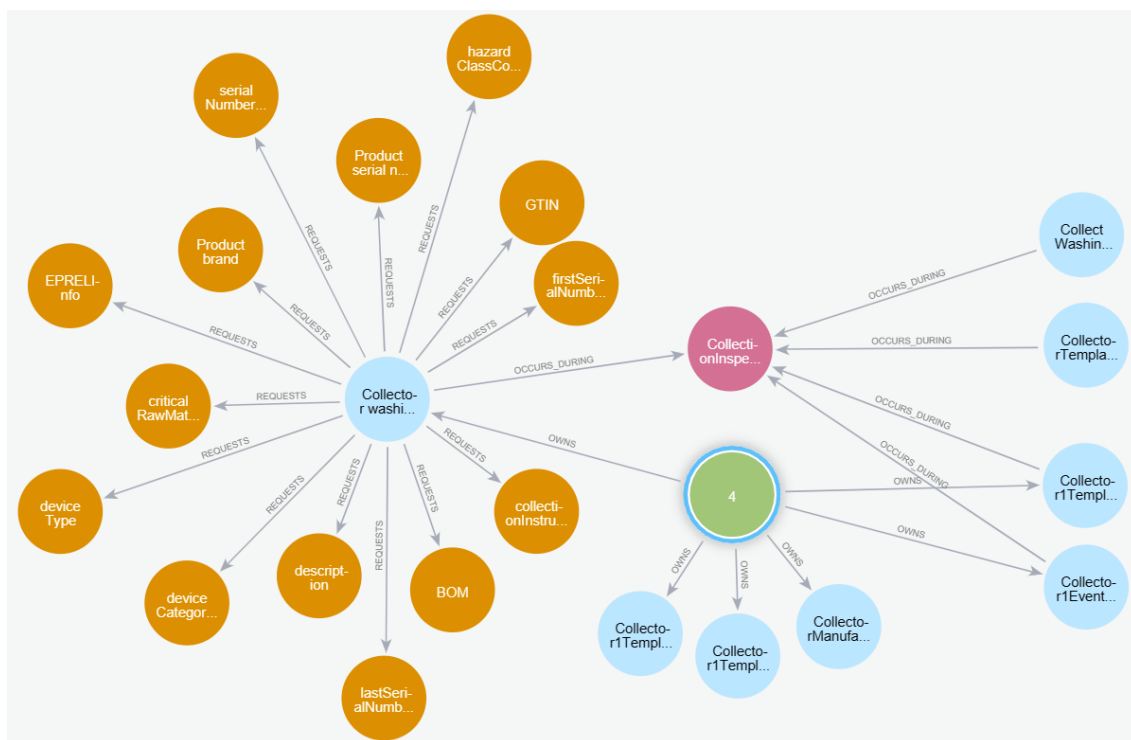


Figure 32 Extract from the dynamic Neo4j database to illustrate the relationship between the various events.

5.2.4 Recommendation Logic with Neo4j and Resolver integration

The Event interpretation and information recommendation service has a variety of capabilities for generating term recommendations. A hybrid system has been developed for this service, which is a mixture of content-based, collaborative, and knowledge-based filtering. Using Neo4j Aura DB at present, the recommendations are mainly based on frequency and similarity. A future upgrade to Neo4j Aura DS (Data Science) is planned, which will include additional data analysis and recommendations by using the Neo4j Graph

Data Science Library⁹. Examples of these features are the Jaccard index or PageRank algorithms. Despite this, the system currently utilises the adaptability of Neo4j's graph database and the potency of Cypher queries to generate varying relevant term recommendations. Each function is concentrated on a particular aspect of the fundamental data. Additionally, certain functions are designed to access user-specific data, typically stored in cookies. The user must, therefore, be logged in to ensure that this data can be accessed and utilised appropriately. By using query parameters such as suggestion limit, user role or life cycle phase, queries can be customised, and results can be filtered according to the user's needs. This enables precise and relevant recommendation generation. The following system functionalities are already implemented:

- **Association Rule Learning:** This function generates recommendations using the association rule learning method. It examines, which terms are frequently requested together in events to identify combinations that occur frequently. Query parameters are not necessary. The number of terms returned can be limited with suggestion limit, as with the other following functionalities.
- **Most occurrences per life cycle phase:** This algorithmic process identifies the most commonly requested terms for a designated phase within the life cycle. It scans for occurrences within this stage and outputs the most frequent terms linked to it. Required parameters are the name of the life cycle phase and suggestion limit. Login is not necessary.
- **User preferences based on role and life cycle phase:** This feature provides tailored suggestions based on each user's role and life cycle stage and returns terms that have been used most frequently. Please note, this feature requires information about the user's role, which is extracted from the user's cookies. Users must log in beforehand to ensure that the cookies contain accurate information. One use case would be, for example, a user in the "Collector" role searching for terms during the "Collection Inspection Phase". The system returns terms that were frequently used by other users in similar roles and phases. Query parameters are the user role and suggestion limit.
- **Most occurrences per role:** This function identifies the most commonly requested terms for each user role by analysing user events. Similar to user preferences, this function is dependent on the user role, which is derived from the user profile. Therefore, the user should either be logged in or specify the role via the user interface or query parameter.
- **Term similarity by similar roles:** The function takes a term as an input parameter. Using this term, it first identifies all users who have an event that requests this term. Then the function extracts the roles of these users and uses this information to find other terms that have been requested by users with the same roles. Thus, the role is determined implicitly, based on the users' interactions with the specific term within the database, rather than by a direct input to the function. Login is not required, and the parameters are just a term from the CircThread vocabulary and suggestion limit. This function is particularly useful for generating role-specific term recommendations.
- **Term similarity by context:** The function identifies terms that are frequently associated with a given term. The query first identifies all events that request the specific term and then searches for other terms that are requested by the same

⁹ <https://neo4j.com/docs/graph-data-science/current/>

events. This procedure is based on the assumption that terms that are frequently used in the same context (represented by the events) as the source term are related in terms of topic or content. A login is not required, and the parameters are merely a term from the CircThread vocabulary and a suggestion limit.

- **Term similarity by life cycle phase:** The function focuses on identifying terms that appear in the same life cycle phases as an entered term. The underlying cypher query tracks events that request the original term and determines the life cycle phases in which these events take place. It then searches for other terms that occur in the same phases, providing insights into phase-specific terminologies. There is no need to log in, and the parameters are only a term and optionally a default limit.

As a result of the functions, a list of terms (keys) is generated either as a list or URL link and sent to the Resolver to obtain the corresponding URL endpoints (value) of the respective service.

6 SHOWCASING THE SERVICE IN THE SMARTFACTORY KAISERSLAUTERN

During the development of the digital object memory service the DFKI team decided to introduce a sample product, needed to discuss the challenges and issues in relation to the implementation of a DOME. A suitable testbed was found with the SmartFactory^{KL} ¹⁰ which is a DFKI assorted living lab¹¹ and represents a network of industrial partners collaborating on the intersection of research and industry. They demonstrate the integration of upcoming software technology with industrial grade hardware and serve as a testbed of industry 4.0 concepts and technologies. Their local demonstration environment features the production of a sample product produced in individual configuration on modular production equipment. This sample product is show in Figure 33.



Figure 33: Sample product of SmartFactory^{KL} is a model-sized truck where the colors, geometries, materials and manufacturing processes can be configured individually.

During the production process a digital thread of the product is created automatically. It contains all relevant information about the product, production processes and used resources. This information is modelled using the Asset Administration Shell. In collaboration with the research projects SmartMA-X¹² and Green AI Hub¹³ we try to preserve this information beyond the manufacturing phase and implement a DPP for the sample product.

Discussed in the context within the CircThread project is the implementation of the data carrier as physical-digital link as physical layer of the ADOME (compare Figure 16). As the considered sample product has only a projected life cycle derived from that of existing commercial products and their life cycles, we first needed to define relevant actors for the product throughout its projected life cycle. For that we used a characterization of user roles derived from the persona method which is a popular approach in user-centred design and marketing (Lidwell et al., 2010) in a simplified manner and identified relevant stakeholders of the product life cycle an especially the data carrier. The persona which are

¹⁰ <https://www.smartfactory.de/>

¹¹ <https://www.dfki.de/en/web/applications-industry/living-labs>

¹² <https://www.smartfactory.de/en/gaia-x-project-begins-in-kaiserslautern-2/>

¹³ <https://www.green-ai-hub.de/en/>

reduced to simple exemplary roles according to their function in the life cycle are described in Table 3.

Table 3: Created role-based personae and their characterization for the projected life cycle of the sample product.

role	Characterization	
Product Designer	Scanning Equipment	Primary Stakeholder
	Scanning Environment	Scanning mostly for testing purpose
	Scanning Intention	
Manufacturing Engineer	Scanning Equipment	Primary Stakeholder
	Scanning Environment	Scanning mostly for testing purpose
	Scanning Intention	
Warehouse Operator	Scanning Equipment	handheld scanners designed for warehouse use
	Scanning Environment	warehouse environments with varying lighting conditions, potential exposure to dust, moisture, and fluctuating temperatures
	Scanning Intention	Rapid scanning of identifiers to manage inventory, track product movement, and ensure accurate order fulfilment, focusing on speed and precision.
Maintenance/Repair Technician	Scanning Equipment	Handheld scanners with robust capabilities specifically designed for consumer appliances
	Scanning Environment	Residential settings and service centres where washing machines are located, varying lighting conditions, and handling of the product within customers' homes.
	Scanning Intention	Focus on diagnosing issues, identifying necessary replacement parts, and ensuring correct installation and functionality of the product
End User	Scanning Equipment	Consumer-grade smartphones with scanning capabilities, using QR code reader apps or integrated camera functions.
	Scanning Environment	Diverse settings, ranging from homes to retail stores, with varying lighting

		conditions and potential external interferences.
	Scanning Intention	Accessing product information, user manuals, and eco-friendly features for making informed purchasing decisions and using the product effectively.
Recycling Plant Manager	Scanning Equipment	Automated sorting systems equipped with specialized scanners designed to identify and sort products based on their identifiers.
	Scanning Environment	Industrial-scale recycling facilities with conveyor belts, noise, and dust, where identifiers are scanned during sorting processes.
	Scanning Intention	Identifying products for efficient sorting, recycling, and processing based on material composition and recyclability data retrieved from the identifier.
Product collection coordinator	Scanning Equipment	Handheld scanners or mobile devices with scanning capabilities for on-site identification of products during collection activities.
	Scanning Environment	Diverse environments, including retail outlets, homes, and collection points, with varying lighting and noise conditions.
	Scanning Intention	Quickly identifying products, verifying their life cycle stage, and ensuring accurate collection and transportation, focusing on efficient logistics and traceability.

Through the different personae we were able to define requirements for the physical-digital link and classify them as functional, non-functional, and conditional requirements as shown in Table 4. We could then also prioritize them using the MoSCoW method developed by Dai Clegg (Clegg & Barker, 1994) as depicted in Table 5. The results were discussed with the partners from the consortium to capture especially the expertise of the industrial actors. The frame for this was provided by a joint workshop on 15.11.2023, to which all consortium partners were invited.

Table 4: Derived requirements and their classification into f=functional, nf=non-functional, c=conditional.

Role	#	requirement	f	nf	c
Product Designer	1	The identifier should seamlessly integrate into the product design without compromising aesthetics.	x		
	2	Compatibility with various product materials and designs for versatile integration options.	x		

	3	The identifier should have a compact form factor to fit within the product's dimensions without being obtrusive.		x	
	4	Customizable sizing options for different product types and sizes.	x		
Manufacturing Engineer	5	The identifier should be easy to integrate into the manufacturing process without causing disruptions or delays.	x		
	6	Compatibility with existing manufacturing equipment and processes for seamless integration.	x		
	7	The identifier should be made of durable materials resistant to manufacturing stressors like heat and pressure.		x	
	8	Material should be conducive to mass production techniques (e.g., moulding, engraving).		x	
Warehouse Operator	9	The identifier should be easily scannable, even from a distance, using handheld scanners commonly used in warehouses.	x		
	10	The identifier should withstand rough handling, occasional impacts, and exposure to dust and moisture without losing functionality.		x	
	11	Compatibility with common handheld scanning devices used in warehouses			x
Maintenance/Repair Technician	12	The identifier should be easily accessible for scanning during routine maintenance checks without requiring disassembly.	x		
	13	Clear placement on the product, ensuring visibility and accessibility even in confined spaces.		x	
End User	14	The identifier should be scannable using common devices like smartphones with user-friendly apps.	x		
	15	Compatibility with a variety of scanning apps and platforms for user convenience		x	
Recycling Plant Manager	16	Compatibility with automated sorting systems in recycling plants for efficient identification and processing.	x		

	17	The identifier should be easily separable from product and itself made from material that is recyclable	x		
	18	Environmental sustainability in the choice of materials.		x	
	19	The identifier should be easily recognizable and scannable, ensuring quick identification of products during the collection process.	x		
Product collection coordinator	20	Integration with handheld scanning devices for rapid identification of products in various locations.	x		
	21	Identifier should be scannable even if the product is damaged on the surface			x

Table 5: Prioritization of requirements using the Moscow-method with M=must have, S=should have, C=could have, W=won't have.

role	#	requirement	M	S	C	W
Product Designer	1	The identifier should seamlessly integrate into the product design without compromising aesthetics.	x			
	2	Compatibility with various product materials and designs for versatile integration options.	x			
	3	The identifier should have a compact form factor to fit within the product's dimensions without being obtrusive.		x		
	4	Customizable sizing options for different product types and sizes.	x			
Manufacturing Engineer	5	The identifier should be easy to integrate into the manufacturing process without causing disruptions or delays.	x			
	6	Compatibility with existing manufacturing equipment and processes for seamless integration.	x			
	7	The identifier should be made of durable materials resistant to manufacturing stressors like heat and pressure.		x		

	8	Material should be conducive to mass production techniques (e.g., moulding, engraving).		x		
Warehouse Operator	9	The identifier should be easily scannable, even from a distance, using handheld scanners commonly used in warehouses.	x			
	10	The identifier should withstand rough handling, occasional impacts, and exposure to dust and moisture without losing functionality.				
	11	Compatibility with common handheld scanning devices used in warehouses				
Maintenance/Repair Technician	12	The identifier should be easily accessible for scanning during routine maintenance checks without requiring disassembly.	x			
	13	Clear placement on the product, ensuring visibility and accessibility even in confined spaces.	x			
End User	14	The identifier should be scannable using common devices like smartphones with user-friendly apps.	x			
	15	Compatibility with a variety of scanning apps and platforms for user convenience				
Recycling Plant Manager	16	Compatibility with automated sorting systems in recycling plants for efficient identification and processing.	x			
	17	The identifier should be easily separable from product and itself made from material that is recyclable	x			
	18	Environmental sustainability in the choice of materials.			x	
Product collection coordinator	19	The identifier should be easily recognizable and scannable, ensuring quick identification of products during the collection process.	x			
	20	Integration with handheld scanning devices for rapid	x			

		identification of products in various locations.				
	21	Identifier should be scannable even if the product is damaged on the surface	x			

Using the derived requirements are not exhaustive but seem substantial enough to design a solution for the data carrier and its placement on the product. We derived two-point solution, involving two different technologies.

The first data carrier as a low-threshold access solution to the DPP is a QR code on the outside of the product. It can be scanned using consumer grade hardware like a smartphone and is designed to cover the retail and use-phase, specifically for customer information and decision support but also supports maintenance and repair actions. The downside of the positioning and the technology is its susceptibility to staining, environmental lighting and also damage and even loss of the data carrier itself through the wear and tear of the product. Figure 34 shows the QR-code attached to the outside of the product.



Figure 34: First data carrier which is primarily customer facing.

Therefore, a second data carrier is being placed on the protected inside of the product. The RFID tag which serves as this second data carrier can only be accessed by disassembling the product at least partially. It is targeted especially at the stakeholders at the end of the products life cycle (Collectors, Recyclers, Remanufacturers, ...) but can also be used in retail (warehousing, inventory control, ...). It is not only a meant as a backup in case the optical code becomes to damaged or lost and cannot be recovered but also enables batch scanning which is especially useful during the end-of-life processes where large batches of products need to be separated and dangerous or valuable materials identified and recovered. The data carrier itself has the same content as the QR-code on the outside. But it can be discussed that additional data can be stored on it to enable also offline data retrieval and information or identity validation. Figure 35 shows the truck with the data carrier protected within the cabin which is only accessible by disassembling the cabin and the chassis.

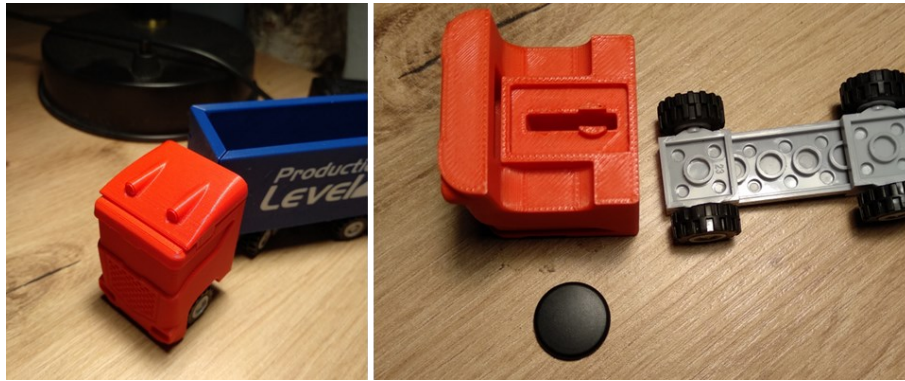


Figure 35: Second data carrier on the protected inside of the product.

As the primary data storage of the sample product is the AAS of the product, the data carrier contains a URL directly to a webserver hosting it. With a final version of the data providing services of the CircThread platform the product related data will be transcribed from the AAS to the platform. The implementation can later be used in other research projects to study and implement the DPPs effects on the end-of-life actors and be used as a showcase on industry fairs to disseminate the project results.

7 CONCLUSION AND NEXT STEPS

7.1 Conclusions

This chapter concludes the work from task T4.2, which focussed on the development and implementation of two microservices. The first, the DOME Resolver, is established as a central 'routing unit,' essential for directing users to the desired web URIs containing product information. Its functionality hinges on the use of a product identifier, typically scanned from a physical tagging technology such as a QR code. Upon receiving this identifier along with a list of terms, the resolver performs matching-based parsing so that it discerns the appropriate service and provides the URL endpoints. To follow the modular approach of the CircThread platform, a service registry was proposed to describe and use the service dynamically. The URL endpoints are then processed and visualized through the platform's middleware, ensuring seamless user interaction and data retrieval. The service delineates information originating from two key sources: the PMMC and the DPP Manager, catering to both product model and product unit levels.

The DOME Resolver is complemented by the second microservice, the Event Interpretation and Information Recommendation Service, a term recommendation engine. This service runs on a dual Neo4j database system. The first is the generic user database designed to mitigate the cold start problem in recommendation systems and is subject to continuous improvement. The second is the dynamic database where users can create custom event templates, incorporating properties such as name, life cycle phase and a set of terms from the CircThread vocabulary. Based on the recommended terms, event- and role-specific digital product passports can be created, providing tailored support for stakeholders in the product life cycle.

Taken together, these microservices underpin the CircThread platform's innovative approach to integrating and personalising product information and life cycle management. They also account for the intended modular and dynamic nature of the platform and the DPP.

Furthermore, this work tries to bring together the concepts of DOME and the DPP to align the ideas accordingly. As such the Digital Object Memory service describes the meta-concept of the platform and matches the terminology. Of course, the implementation of the described microservices is heavily intertwined with the other platform modules and the iterative improvement will take place with more and more of the platform available.

7.2 Future development opportunities / Next steps

The immediate next steps are to integrate the service with the platform middleware while defining not still open modules for e.g. the service registry and the identity management. Once the use case 1 of the CircThread project is completely operational, we are planning to conduct an evaluation regarding usability and user experience of the service. This goes hand in hand with an evaluation of the recommendations by the corresponding microservice. Already some measures to improve the system further are to transfer to Aura DS including data science tooling, also the use of user feedback and cookies could support further intuitive personalization of the recommendations.

Practically, the services will be improved iteratively, and current version made available to the partners using the CircThread GitHub organization. We will also provide containerized

versions of the microservices for hosting the service cluster when bringing the platform online.

The implementation of the showcase at the SmartFactory^{KL} will also be used at fairs to demonstrate and discuss the solutions with a broad public. In collaboration with other research projects, we will expand this showcase stronger integrating the end-of-life stakeholders.

7.3 Other conclusions and lessons learned

The integration of the SmartFactory^{KL} data structure raised further issues regarding the integration of digital twin formats like the AAS or the Digital Twin Design Language (DTDl). As these digital twin formats gain increasing importance in manufacturing systems, compatibility to lifecycle-overarching data management infrastructure must be ensured. Otherwise, isolated data silos might lead to high integration efforts in the future.

8 References

- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749. <https://doi.org/10.1109/TKDE.2005.99>
- Alvarado, J. E., Augusto, J. C., Cech, P., Cook, D. J., Higuera, A. G., Ikonen, V., Kroner, A., Mikulecky, P., & Schneider, M. (2009). *Workshops Proceedings of the 5th International Conference on Intelligent Environments*. IOS Press.
- Ashby, W. R. (1951). *Automata studies: Annals of Mathematics Studies. Number 34* (5. rist). *Annals of mathematics studies: Vol. 34*. Princeton University Press.
- Bar Code Graphics Inc. (2023). *Change is Coming – Item Barcodes Are Becoming Web Enabled*. <https://www.gs1digital.link/>
- Birtel, M., Motsch, W., Popper, J., & Ruskowski, M. (2020). Method for the development of an Asset Administration Shell in a product-driven modular production – realizing an active digital object memory. *IEEE*, 1, 583–588. <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9274753&tag=1>
- BMAS. (2023). *CSR - EU supply chain law initiative*. <https://www.csr-in-deutschland.de/EN/Business-Human-Rights/Europe/EU-supply-chain-law-initiative/eu-supply-chain-law-initiative.html>
- Chronicle Inc. (2021). *The MediLedger Network*. <https://www.mediledger.com/>
- Clegg, D., & Barker, R. (1994). *Case method fast track: A RAD approach* (1. pr). *Computer aided systems engineering*. Addison-Wesley Pub. Co.
- Comyn-Wattiau, I., & Akoka, J. (2017). Model driven reverse engineering of NoSQL property graph databases: The case of Neo4j, 453–458. <https://doi.org/10.1109/BigData.2017.8257957> (Original work published 2017)
- Di Pierro, D., Ferilli, S., & Redavid, D. (2023). LPG-Based Knowledge Graphs: A Survey, a Proposal and Current Trends. *Information*, 14(3), 154. <https://doi.org/10.3390/info14030154>
- Docsroom. (2023). *DocsRoom - European Commission*. <https://ec.europa.eu/docsroom/documents/54874>
- Domdouzis, K., Kumar, B., & Anumba, C. (2007). Radio-Frequency Identification (RFID) applications: A brief introduction. *Advanced Engineering Informatics*, 21(4), 350–355. <https://doi.org/10.1016/j.aei.2006.09.001>
- European Commission. (2019). *A European Green Deal*.
- European Commission. (2020). *Circular economy action plan – For a cleaner and more competitive Europe*. Publications Office of the European Union. <https://doi.org/10.2779/05068>
- European Commission. (2022). *Making sustainable products the norm in Europe*. Publications Office of the European Union. <https://doi.org/10.2779/014417>
- Friedl, J. E. F. (2006). *Mastering regular expressions* (3. ed.). O'Reilly.
- Gatzioura, Sánchez-Marrè, & Gibert (2019). A Hybrid Recommender System to Improve Circular Economy in Industrial Symbiotic Networks. *Energies*, 12(18), 3546. <https://doi.org/10.3390/en12183546>
- GS1 Digital Link standard. (2023). <https://ref.gs1.org/standards/digital-link/archive>
- GS1 Germany GmbH. (2023). *Mit Identifikationsnummern weltweit eindeutig | GS1 Germany*. <https://www.gs1-germany.de/gs1-standards/identifikation/>

- Hasan, S. N., & Khatwal, R. (2022). Cold Start Problem in Recommendation System: A Solution Model Based on Clustering and Association Rule Techniques, 1–8. <https://doi.org/10.1109/IMPACT55510.2022.10029293>
- Hauptert, J. (2021). *OMM: A Structure Model for Digital Object Memories (DOMe)*. <https://www.w3.org/2014/02/wot/papers/hauptert.pdf>
- Hauptert, J., Hauck, C., & Kröner, A. (2012). *UbiComp '12: Proceedings of the ACM conference on ubiquitous computing, Pittsburgh (PA), USA, 05-08.09.2012. ACM Conferences*. A.C.M. <https://dl.acm.org/doi/pdf/10.1145/2370216.2370463> <https://doi.org/10.1145/2370216>
- Heidel, R. (Ed.). (2019). *Beuth Innovation. Industrie 4.0: The reference architecture model RAMI 4.0 and the industrie 4.0 component* (1st edition). Beuth; VDE Verlag. https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/AAS-ReadingGuide_202201.pdf?__blob=publicationFile&v=1
- Hong-ying, S. (2009). The Application of Barcode Technology in Logistics and Warehouse Management. In *2009 First International Workshop on Education Technology and Computer Science*. IEEE. <https://doi.org/10.1109/etcs.2009.698>
- ISO (June 2007). *Automatic identification and data capture techniques: Code 128 bar code symbology specification* (15417:2007). <https://www.iso.org/standard/43896.html>
- ISO (May 2023). *Information technology Automatic identification and data capture techniques: Code 39 bar code symbology specification* (16388:2023). <https://www.iso.org/standard/77799.html>
- Kröner, A., Hauptert, J., Kawsar, F., Speed, C., Ploetz, T., & Schreiber, D. (2012). Digital Object Memories for the Internet of Things (DOMe-IoT). In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM. <https://doi.org/10.1145/2370216.2370462>
- Lam, X. N., Vu, T., Le, T. D., & Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In W. Kim & H. J. Choi (Eds.), *Proceedings of the 2nd international conference on Ubiquitous information management and communication* (pp. 208–211). ACM. <https://doi.org/10.1145/1352793.1352837>
- Le Son, H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, 58, 87–104. <https://doi.org/10.1016/j.is.2014.10.001>
- Li, S., Da Xu, L., & Zhao, S. (2015). The internet of things: A survey. *Information Systems Frontiers*, 17(2), 243–259. <https://doi.org/10.1007/s10796-014-9492-7>
- Lidwell, W., Holden, K., Butler, J., & Elam, K. (2010). *Universal principles of design: 125 ways to enhance usability, influence perception, increase appeal, make better design decisions, and teach through design* [2nd ed.]. Rockport Publishers. <https://learning.oreilly.com/library/view/-/9781592535873/?ar>
- Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4), 2065–2073. <https://doi.org/10.1016/j.eswa.2013.09.005>
- Melville, P., & Sindhvani, V. (2017). Recommender Systems. In C. Sammut & G. I. Webb (Eds.), *SpringerLink Bücher. Encyclopedia of machine learning and data mining* (expanded and updated second edition, pp. 1056–1066). Springer. https://doi.org/10.1007/978-1-4899-7687-1_964

- National Science Foundation. (2023). *Cyber-Physical Systems (CPS): NSF 21-551*. National Science Foundation. <https://www.nsf.gov/pubs/2021/nsf21551/nsf21551.htm>
- Onkar Singh. (2023). *An overview of fake product detection using blockchain technology*. <https://cointelegraph.com/explained/an-overview-of-fake-product-detection-using-blockchain-technology>
- Reed, I. (2012). *Reed–Solomon error correction*. <https://cloudflare-ipfs.com/ipfs/qmxoypizjw3wknfijnklwhcnl72vedxjqkddp1mxwo6uco/wiki/reed%e2%80%93solomon.html>
- Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In F. Ricci (Ed.), *SpringerLink Bücher. Recommender Systems Handbook* (1st ed., pp. 1–35). Springer. https://doi.org/10.1007/978-0-387-85820-3_1 (Original work published 2011)
- Sharma, C., & Sinha, R. (2019). A Schema-First Formalism for Labeled Property Graph Databases, 71–80. <https://doi.org/10.1145/3365109.3368782> (Original work published 2019)
- Shop Elite Screens. (2023). *How to Find the Serial and Model Numbers on Your Projector Screen*. <https://shop.elitescreens.com/pages/where-can-i-locate-my-screen-serial-number-s-n>
- Stubblebine, T. (2003). *Regular expression pocket reference* (First edition). O'Reilly.
- Team, W. (2015, October 12). Service Discovery in a Microservices Architecture. *NGINX, Inc.* <https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/>
- Ting, S. L., Kwok, S. K., Tsang, A. H., & Ho, G. T. (2011). The Study on Using Passive RFID Tags for Indoor Positioning. *International Journal of Engineering Business Management*, 3, 8. <https://doi.org/10.5772/45678>
- Tiwari, S. (2016). An Introduction to QR Code Technology. In *2016 International Conference on Information Technology (ICIT)*. IEEE. <https://doi.org/10.1109/icit.2016.021>
- van Capelleveen, G., Amrit, C., Zijm, H., Yazan, D. M., & Abdi, A. (2021). Toward building recommender systems for the circular economy: Exploring the perils of the European Waste Catalogue. *Journal of Environmental Management*, 277, 111430. <https://doi.org/10.1016/j.jenvman.2020.111430>
- Woodland, N. J., & Bernard, S. (1952). *Classifying apparatus and method*. <https://patents.google.com/patent/us2612994a/en>
- Yuan, J., Shalaby, W., Korayem, M., Lin, D., AlJadda, K., & Luo, J. (2016). Solving cold-start problem in large-scale recommendation engines: A deep learning approach, 1901–1910. <https://doi.org/10.1109/BigData.2016.7840810>
- Zhang, Q., Lu, J., & Jin, Y. (2021). Artificial intelligence in recommender systems. *Complex & Intelligent Systems*, 7(1), 439–457. <https://doi.org/10.1007/s40747-020-00212-w>