ONTO-DESIDE

Guide 2

Decentralised sharing of data & information

For a circular and regenerative economy

Funded by
the European Union

Authors (alphabetical order + ORCID):
Eva Blomqvist: 0000-0003-0036-6662
Fenna Blomsma: 0000-0002-0996-4717
Ben De Meester: 0000-0003-0248-0987
Els de Vleeschauwer: 0000-0002-8630-3947
Huanyu Li: 0000-0003-1881-3969
Mikael Lindecrantz: 0000-0002-5525-6439
Vojtěch Svátek: 0000-0002-2256-2982

Edited by:
Eva Blomqvist
Fenna Blomsma

Version:
14th of November 2025

A publication by:
The Onto-DESIDE project

Acronyms/ terminology

API = Application Programming Interface
CE = circular economy
CEON = Circular Economy Ontology Network
CSV = Comma Separated Values
CQ = Competency Question
DPPO = Digital Product Passport Ontology
JSON = JavaScript Open Notation
JSON-LD = JavaScript Open Notation for Linked Data
LOV = Linked Open Vocabularies
MFM = Multi-Flow Method
OBDA = Ontology-Based Data Access
OBO = Open Biological and Biomedical Ontologies
OCP = Open Circularity Platform
OWL = Web Ontology Language
R2RML = RDB 2 RDF Mapping Language
RBAC = Role-Based Access Control (RBAC)
RDB = Relational Database
RDF = Resource Description Framework
RML = RDF Mapping Language
SHACL = Shapes Constraint Language
SPARQL = SPARQL Protocol and RDF Query Language
SSSOM = Simple Standard for Sharing Ontological Mappings
URI = Uniform Resource Identifier
W3C = World Wide Web Consortium
XML = eXtensible Markup Language
YARRRML = Yet Another R2RML and RML Language
XD = eXtreme Design

Table of contents

This guide—how it came to be, who it's for, and how to use it 6
Why, what and how of Circular Economy 10
For all circular strategies—from recycling to repair & from reuse to remanufacturing 14
Decentrally sharing dataan overview of the what, why and how 16
Step 1—Map information flows: value chains are more than resource flows alone 20
Step 1—Examples for different circular strategies 22
Step 2—Define technical requirements: what do users need data for? 24
Step 3—Build a data Inventory: what data is available or could be collected 28
Step 4—Enable data sharing: how and by/with whom? 30
Step 2, 3 and 4—Examples for different circular strategies 32
Step 5—Ensure semantic Interoperability and Ontologies 34
Step 5a—Ontology Requirements and Inventory 40
Step 5b—Ontology Extension 44
Step 5c—Ontology Alignment 50
Step 5—Examples for different circular strategies 54
Step 6—Ensure technical interoperability: data formats 56
Step 7—Define data transformation: connecting data and ontologies 58
Step 8—Set up querying: federated querying with SPARQL 60
Step 6, 7 and 8—Examples for different circular strategies 62
Step 9—Develop data access applications 64
Step 10—Plan for maintenance and evolution 66
Step 9 and 10—Examples for different circular strategies 70
Closing words 72
References 74

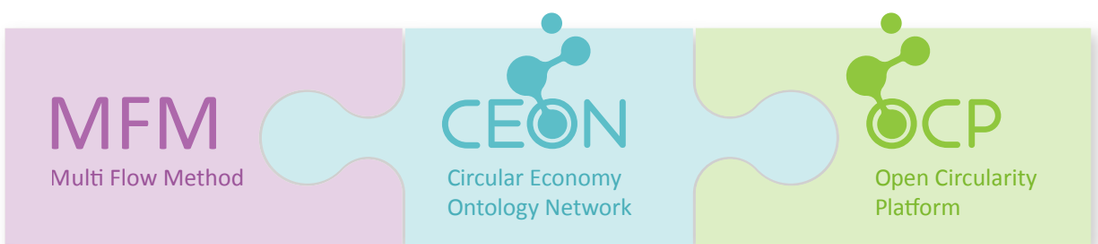
This guide—how it came to be, who it's for, and how to use it

Why this guide? The Onto-DESIDE project.
The Onto-DESIDE project aimed to accelerate the transition to a circular economy (CE) where materials, components, and products are reused to reduce waste and retain value. At the moment, circular value networks are difficult to design and scale because it is difficult to make sense of such systems as a whole. Second, industries struggle to form circular value networks due to inconsistent terminology, lack of semantic clarity, and limited tools for secure, automated data exchange.

To address this, Onto-DESIDE combined conceptual and technical innovation, by 1) creating innovation capacity for circular value chains, and 2) addressing key technical barriers to data sharing across industries. It developed the Multi-Flow Method (MFM), which integrates resource, energy, value, and information flows into a systemic view of circular value chains, using generative tensions to explore root causes to barriers and find ways to improve functioning and robustness. The project also introduced technical solutions: ontologies to model materials, products, actors and processes, ensuring vertical (within domains) as well as horizontal (across domains) semantic interoperability, together with a decentralised collaboration platform where data can be exchanged. However, a crucial aspect of supporting transformation is to provide guidance in using these new tools: the aim of this guide.

What was done and how
Onto-DESIDE applied a transdisciplinary and iterative methodology to develop the new tools and technologies for circular value networks. Academia and practice came together, using three diverse real-world industry use cases selected for their diversity and complexity—construction, electronics, and textiles—as testbeds to derive needs and validate the within- as well as cross-sector applicability of the solutions.

The project, running from June 2022 to November 2025, was structured into multiple work packages. One focused on the creation of the innovation method, a second on ontology development, and a third on the data-exchange platform. Each used their own methodology and domain-specific expertise, respectively, design science methods; agile ontology engineering practices including eXtreme Design (XD) resulting in the Circular Economy Ontology Network (**CEON**); and the application of mature open web standards to create a secure and decentralized interoperable data sharing infrastructure dubbed the Open Circularity Platform (**OCP**). Collaboration across these tasks makes them comprehensive and integrated. In uniting top-down research and standards analysis with bottom-up learning from use cases, the project created a solid and actionable foundation for advancing the circular economy.



Guide 1: *Circular value chain design, development & innovation*

Guide 2: *Decentralised sharing of data & information*

How to use the guides
There are two guides: one which focuses on circular value chain development and innovation, and a second technical guide that is dedicated to setting up a decentrally organised data-sharing infrastructure in such a way the data is interoperable and compatible.

Both guides focus on the practical steps to take towards better functioning circular value chains. Each guide discusses the relationship with the other, so it is clear where they connect. Depending on your needs and circular maturity level, you can drive straight into the technical parts, or you can first spend a moment thinking about the functioning of your circular value chain and how to design or improve it. It is up to you to decide what you need and where to start. Together, both parts of the Onto-DESIDE project outputs support the planning and automation of management and execution of circular value networks at scale, contributing to Europe's digital and green Twin Transition.

For more details, or more technical descriptions as well as templates, explainer videos, and other supplementary materials go to our website.

Please visit:
www.ontodeside.eu

ONTO-DESIDE

Ontology-based Decentralised Sharing of Industry Data in the European Circular Economy

- 12 partners, 7 countries:
  - Linköping University (SE)
  - *Interuniversitair Micro-Electronica Centrum (BE)*
  - *Concular Ug Haftungsbeschrankt(DE)*
  - *+Impakt Luxembourg Sarl (LU)*
  - *Circularise Bv (NL)*
  - *Universität Hamburg (DE)*
  - *Circular.Fashion Ug (DE)*
  - *Lindner Group Kg (DE)*
  - *Ragn-Sells Recycling Ab (SE)*
  - *Texon Italia Srl (IT)*
  - *Rare Earths Industry Association (BE)*
  - *Prague University of Economics & Business (CZ)*
- From: June 2022-November 2025
- Funding: Horizon Europe
  - *Grant agreement #101058682*

Three use case domains:
- Textile industry
- Electronics industry
- Construction industry

Funded by the European Union

Who the guides are for

Both of our guides are aimed at anyone who wishes to engage in circular oriented innovation. That is: anyone who wants to explore new or better circular value chains as well as get practical about data and information sharing to enable this in practice. Each guide is meant as an entry point into their respective topics, and they each target different roles—with an emphasis on the role and contribution of these different roles to the various steps in the process. Mainly these two guides provide an overview and explain what to expect whilst on this journey. In this, we focus on how different roles can work together. To this end at the top of each section, you find an indication of what roles are typically involved or who is needed to provide input to complete a step successfully. Of course, these roles can be different people, or be one and the same. Organising the guides around roles clarifies responsibilities and interfaces across the process, supporting structured collaboration, aligned expectations, and deliberate progress in circular-oriented innovation.

Guide 1—See: www.ontodeside.eu

Value chain design, development & innovation:

This guide has a strategic focus, and explores what currently shapes the value chain dynamics and how circular strategies can be (better) supported. The following roles are needed to successfully complete the process:

- *Project lead:* Coordinates the overall process in which the method is applied. Ensures the right people are involved, aligns the method with the project's goals, and takes responsibility for follow-up after sessions and working groups.
- *Facilitator:* Guides the group through the Multi-Flow Method. Ensures the process is structured, that flows, tensions, and patterns are captured in a way the group can work with, and that different perspectives are heard.
- *Decision maker(s)*: Stakeholder representatives with the authority to shape the value chain configuration or influence (strategic) decisions. To ensure relevance and actionability, the process should include different perspectives (e.g., suppliers, customers, recyclers, logistics providers).
- *(Flow) Experts:* Bring (technical and practical) knowledge of specific flows (material, information, value, energy). They explain how flows operate in practice and support the group in understanding constraints, dependencies, and opportunities.

Guide 2—*The guide in front of you now:*

Decentralised sharing of data & information:

This guide focuses on the technical side. It explains how to set up a decentralised, secure and automated data-sharing infrastructure that supports a chosen value chain configuration and the collaborations between the actors involved. The steps involved in setting up this infrastructure needs the involvement of different types of roles in the involved organisations. We identify 4 such roles:

- *Decision Maker*: May be the value chain manager, coordinating the setup of the whole value chain, or merely the internal manager in charge of ensuring the participation of a specific actor in the value network configuration. Additionally, a decision maker may be a CTO or CIO making decisions about the IT infrastructure setup and investments.
- *Data Steward*: Any role that produces, manages or maintains the data that is to be shared and used in order to make the value chain configuration work.
- *Developer*: Either an information architect/ data modeller, or a software developer/IT specialist. These are the roles that will do the practical work of modelling and transforming the data, as well as setting up the actual infrastructure and configuring it.
- *End-User*: The roles within the value chain organisations that hold the needs for receiving or sharing the data. For instance, this could be a person at a recycling facility, needing the information about incoming used materials in order to make decisions regarding where to dispatch a certain batch or container.

Why, what and how of Circular Economy

"Take-make-use-*lose*"
Our global economy operates largely on a linear model: extract, produce, consume, and dispose—repeat. This system assumes unlimited access to resources and an infinite capacity for waste absorption. But our planet can provide neither: we are rapidly depleting finite resources and are overwhelming natural systems with waste and emissions. Even recycling, often seen as a solution, only addresses a very small part of the problem and fails to fundamentally transform how we use resources. What's more, this extractive system entrenches inequality, undermines livelihoods, and worsens living conditions for many.

For example, resource extraction has already more than tripled since 1970 and is projected to rise another 60% by 2060 if the current path is followed, accounting for over 60% of global greenhouse gas emissions and 40% of pollution-linked health impacts[1]. Such scale places enormous pressure on ecosystems and communities. No wonder that the linear economy is sometimes also referred to as "Take-make-use-*lose*"[2].

Instead…
Our economies will have to change their extractive practices to sustainable and regenerative ones. Circular Economy (CE) offers one path through the application of Re-strategies like *re*think, *re*duce, *re*tain, *re*use, *re*pair, *re*furbish, *re*manufacture, *re*cycle—and a range of related strategies like composting & industrial symbiosis. The aim is to better meet the needs of the whole system–planet, people and businesses–and thereby encourage different ways of handling waste and resources, improving resource conservation, efficiency and productivity. Or: how can we live comfortably - without costing people and the planet?

CE is no longer optional, but a must-have
Mounting resource scarcity, increasingly volatile supply chains and resource prices, intensifying legislative and regulatory pressure, and rising stakeholder expectations mean that CE is no longer optional—it's essential for business resilience, compliance, innovation, and competitiveness[3,4,5]. Companies that continue to rely on a take-make-dispose model expose themselves to higher costs, operational disruptions, and reputational risks, while those that adopt circular strategies can secure materials, stabilize supply, and strengthen their license to operate. Thus, CE is becoming a key driver of both risk management and value creation.

Likewise, finance and investors are intensifying the shift of capital toward businesses that demonstrate circular strategies, recognizing them as lower-risk, future-fit, and better positioned to deliver long-term value[6]. Capital markets are increasingly embedding sustainability and circularity metrics into lending, investment, and valuation models, rewarding companies that proactively align with emerging standards. Those who fail to adapt may face shrinking access to capital, higher borrowing costs, and reduced investor confidence, while circular leaders stand to attract investment, partnerships, and preferential market positioning.

The Challenge
But… 'going circular' is complex. It requires systems thinking to understand how and why materials flow, where and why waste originates, and how circular strategies interact. It requires moving beyond simplistic models and truly solving problems—not shift them elsewhere or create new ones. And: not all circular strategies work at every scale or in every context, and some may even compete or create trade-offs. For example, choosing highly durable composites can hinder recyclability, and remanufacturing may initially require more materials—not less. And so on. The challenge is to design and operate sets of circular strategies that resolve, go around or balance these tensions and deliver real benefits. Doing so requires the right mix of competition and collaboration, clear and easily accessible data, and adaptive management across the value entire chain.

For this reason, circular innovation differs fundamentally from linear or 'business-as-usual' innovation. It involves creating virtuous loops—feedback mechanisms where resources re-enter the value chain—and generating emergent properties like sustainability and resilience. These benefits arise not from isolated actions but from how the entire system behaves.

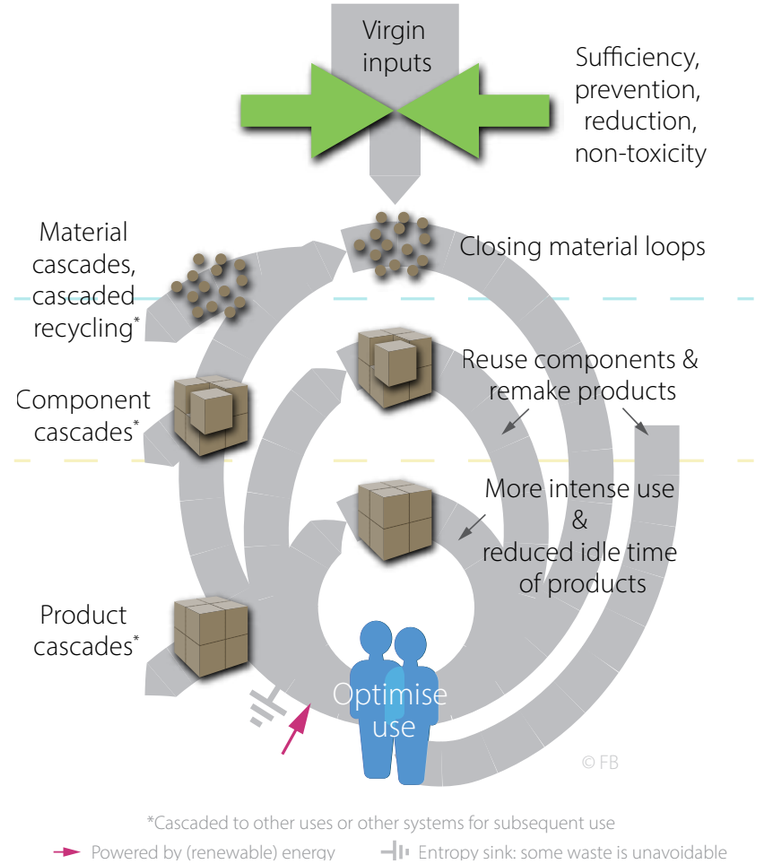


Figure: **Circular strategies in the use phase, and for products, components and materials.**

From a linear to a circular mindset

All this means that a different mindset is needed when engaging in circular oriented-innovation. Crucially: it means creating systems where multiple circular strategies operate synergistically—where, through collaboration, all actors benefit. Circularity Thinking helps cultivate this new circular mindset.

1. Flow Structure: One-Way vs. Feedback Loops

- Linear mindset: Resources flow in a straight line—extraction, production, use, disposal—with minimal interaction between processes.
- Circular mindset: Resources circulate and regenerate through feedback loops, re-entering the system several times (as products, components, and materials) and influencing upstream and downstream decisions over time.

2. Value Creation: Localized vs. Emergent

- Linear mindset: Value is created and captured at specific points in the chain (e.g. sales, production)—with opposing and conflicting interests, resulting in value conflicts.
- Circular mindset: Value is emergent, arising from how the entire system functions—through resilience, sustainability, and shared innovation. Both the whole and the parts benefit equally.

3. Problem Solving: Fragmented vs. Systemic

- Linear mindset: Problems are solved in isolation, often within departmental or disciplinary silos. This leads to displacement and the creation of new problems.
- Circular mindset: Problems are addressed systemically, considering interdependencies, long-term effects, and cross-sector dynamics.

4. Strategy Use: Selection vs. Configuration

- Linear mindset: Strategies are chosen individually—reuse or recycling, efficiency or durability—as they benefit one actor, often without considering their interactions.
- Circular mindset: Strategies are combined into configurations, designed to work together synergistically and allowed to evolve over time, seeking the addition of more circular strategies through continuous improvement.

5. Innovation Process: Execution vs. Iteration

- Linear mindset: Innovation follows a fixed plan—analyse, design, implement—assuming predictability and with limited flexibility.
- Circular mindset: Multiple innovation modes operate alongside each other, where innovation also incorporates processes that are iterative, involving experimentation, learning, and the ability to pivot when assumptions prove incorrect.

6. Responsibility: Compliance vs. Stewardship

- Linear mindset: Responsibility is often limited to meeting regulations or minimizing costs.
- Circular mindset: Responsibility includes stewardship—ensuring that circular strategies address real problems and no new ones are created elsewhere in the system. And: that the needs of all parts of the system are served.

Value-, resource- & information-flows

Therefore, to design, improve, and operate a circular way of working it is essential to adopt a *value chain* perspective - sometimes also called a *value network*. This is because circularity cannot be achieved in isolation—materials, components, products, as well as benefits and impacts flow across multiple actors and stages. Only by seeing how decisions in one part of the chain affect others can businesses understand how shared benefits can be created and value captured and to design circular strategies that synergistically reinforce each other. This perspective also highlights trade-offs and tensions that must be managed collectively, rather than pushed onto individual actors, if the system is to function. *Data and information* play a critical role in this: they *provide the transparency needed* to track resource flows, identify where waste and inefficiencies occur, and coordinate action across suppliers, partners, and customers. Without accurate, shared and frictionless access to information, circular value chains cannot be designed effectively or operated at scale.

This guide

To help with this, the **Onto-Deside** project created the following guidance and support for:

- *Value chain design, development & innovation:* gaining insight into the root causes of barriers and enablers that shape the behaviour of value chains, and examining how this dynamic can (better) support circular flows.
- *Decentralised sharing of data & information:* understanding data needs and availability, formats, and aligning the data to a shared domain model, the Circular Economy Ontology Network (**CEON**), setting up an Open Circularity Platform (**OCP**)—data-sharing such that data becomes interoperable, but where control over what to share with who and when remains with the data owners.

The guide in front of you covers:

- *Decentralised sharing of data & information.*

Please find the other guide at:
>>> www.ontodeside.eu

For all circular strategies—from recycling to repair & from reuse to remanufacturing

To help bring the guidance to life and offer concrete examples to illustrate our methods, here are 4 short examples of different circular strategies that we'll refer back to throughout this guide. Although these scenarios (A to D) address different strategic priorities, they have the same needs in common. Each actor in the network must be able to selectively share its data, based on ever-changing business needs. Meanwhile, to track resource flows across stakeholders and make appropriate decisions, a common understanding of this shared data is needed.

(A) Beginning-of-life: using recycled input

What: Cross-sector recycling of apparel waste into feedstock for floor tiles.

Why: To unlock circular business models, and help to find the right recycled feedstock through product passports and secure data exchanges.

A *product manufacturer* creates a performance shoe using inputs from various *material suppliers*, each contributing data to a shared platform for product passports using standardized formats. Once the shoe reaches end-of-life, a *recycling operator* disassembles it, guided by digital instructions, and extracts the rubber outsoles and textile laces that are made into new bulk materials.

These recovered materials are listed on a *digital marketplace*, enriched with a certificate and metadata including composition, condition, and recycled content. Next, a *materials processor* identifies suitable batches and requests pricing via the platform. After purchase, the recycled inputs are turned into materials that an *interior outfit company* uses for acoustic floor tile layers. Certificates and material data travel along, and a new product passport is generated for the product.

(B) Middle-of-life: repair

What: Repair of an audio system through access to reliable spare parts and instructions.

Why: Automating sustainable asset management through digital tools to enable easier data management, whilst protecting sensitive data.

A *building owner* identifies a malfunction in the installed audio system. Using a data exchange platform they access repair instructions and discover that the *original equipment manufacturer* offers a repair service. The component is sent for repair, and the manufacturer replaces the faulty speaker with a newer model containing a higher amount of recycled content.

The repaired unit is reinstalled, and updated product data is published and added to the *building's digital twin*, including material composition and sustainability attributes. Digital product passports record both original and repaired versions, tracking components and their environmental impact—including recycled content, origin, and certifications—automating the management of building information.

(C) End-of-life: reuse

What: Reuse and resale of a door for use in other building projects.

Why: Linking supply & demand through a digital market place for second-life parts & components.

A *building owner,* preparing for demolition, assesses the reuse potential of installed elements, such as doors, for repurposing through resale. This information is used by the *demolition contractor* to negotiate a fair price for the building's demolition, and sets the frame for what demolition methods will be used.

To find a new use, the building owner lists the components, including the doors, on a *digital marketplace*—provided by an *intermediary* for sale to *construction companies* for reuse in new projects. Metadata such as dimensions, condition, and installation history are shared, and enriched with images, enhancing buyer confidence. Pricing information is managed securely via decentralized data pods, ensuring only authorized parties can access commercial terms and optimising the value for the building owner. Planning considerations are automatically taken into account.

(D) End-of-life: remanufacturing

What: Take-back of the floor tiles by the manufacturer for remanufacturing.

Why: Enabling manufacturers to take-back their products, ensuring access to future feedstock.

At the end-of-life stage of a building, a *building owner* initiates a demolition plan and assesses reuse and recovery options for installed components. Among these, the acoustic floor tiles—originally made with recycled feedstock from apparel waste—are identified as having high reuse potential, but not in their current condition.

The owner contacts the *original tile manufacturer,* who offers a take-back program. Through a data exchange platform—facilitated by an *intermediary*—the manufacturer provides pricing and logistics information for reclaiming the tiles. The tiles are returned, inspected, and remanufactured into new flooring systems, integrating both recovered and new materials. This process reduces raw material demand and preserves embedded value.

Decentrally sharing data—an overview of the what, why and how

Data & information sharing and management
Circular economy innovations along value chains depend on collaboration across multiple actors—from material suppliers to manufacturers, retailers, and recyclers. But collaboration is only effective if it is built on a foundation of reliable, consistent information and if it respects competition, too. This is why data and information sharing is not a side issue but a central enabler of circular value chain innovation.

Every circular practice—whether it is product-as-a-service, reverse logistics, reuse, or high-quality recycling—requires transparency about what resources exist, where they are, and in what condition. If one actor knows the material content of a component but cannot share it in a usable way with others, opportunities for reuse or recovery are lost. Similarly, without trusted information flows, it is hard to coordinate responsibilities, design for reuse, or match supply and demand in secondary markets.

The challenge is that different organisations often use different terms, classifications, and IT systems. What one company calls a "part," another might describe as a "module." And the same term, such as "product", may mean different things depending on the actor's perspective: what is someone's product may be another's material or component. Also: data that is meaningful in one system may be unreadable or misleading in another. This is why *aligned ontologies*—shared ways of structuring and describing information—are critical. Just as having a technical standard makes it possible to plug components together, having an information standard with clear semantics makes it possible to plug data together.

This is where a *data-sharing platform* comes in: it becomes the infrastructure that operationalises this alignment. It ensures that information about resources, processes, and transactions can flow securely and consistently across the value chain. Platforms can host material passports, product IDs, or usage histories, making them accessible in formats that others can understand, trust and act upon. With such systems, businesses can make informed choices about design, reuse, or recycling, and policymakers can monitor progress without imposing excessive reporting burdens.

Seen this way, data and information are the *bridge* between circular economy ambitions and their practical realisation. Value chain innovations create the demand for shared knowledge; ontologies and data platforms make it possible to meet that demand in ways that are consistent, scalable, and verifiable. Without them, circular strategies risk being isolated pilots. With them, they can be connected into frictionless circular ecosystems where information flows as seamlessly as materials—whilst keeping what matters safe.

This guide contains more information both on:

- Ontologies for CE—and the use of **CEON**, which is a reusable set of core ontology modules that can be extended to fit virtually any CE use case.
- How to set up a decentralised data sharing platform—and how the **OCP**, which is one such platform, allows for sharing data in a secure decentralised manner.

Both **CEON** and **OCP** are outputs of the Onto-DESIDE project and are **freely available.** Before going into the process of how to use these two tools, first we explain a little bit more about the foundations these two tools were built on.

Why *decentralised*?
In the Onto-DESIDE project we focused on decentralised data sharing solutions as it has certain benefits over centralised data sharing. Unlike centralised systems, where all information is collected, stored, and managed by one party, decentralised approaches allow each of the participating organisations to retain control and responsibility. This avoids the risks of bottlenecks, single points of failure, or power imbalances where one actor owns or controls the entire dataset. It also increases flexibility to integrate diverse data sources and adjust to evolving technical standards. In practice, decentralised approaches are often more acceptable to diverse actors, making collaboration possible in complex value chains.

This is especially relevant in a circular economy, where no single authority governs the system and collaborations can span multiple value chains, across multiple domains and participant constellations, and involve multiple circular strategies. Each actor brings its own priorities, IT systems, and sensitivities, which makes centralised data systems difficult to accept for many. For reasons of security and confidentiality, organisations also want to retain sovereignty over their data: deciding what to share, with whom, and under what conditions. Decentralised data sharing enables this by allowing data to remain in an organisation's own systems, on their own premises, while still making agreed portions accessible to partners. In this way, sensitive business knowledge is protected, collaboration barriers are reduced, and trust between partners can grow—creating the conditions for more open, yet secure, circular value chain innovation. By distributing control in this way, decentralised systems reflect the distributed and collaborative nature of circular economies themselves.

Scaling through automation and standardization
In order to scale the circular economy to cover virtually all material flows in our society, the information infrastructure needs to focus on automation and be built to scale. Each actor in a circular economy must connect with many other actors of different types and over long periods of time—an unmanageable task in practice unless current ways of exchanging information, such as phone calls and emails, are largely replaced by more automated solutions.

Apart from the technical scalability of the solutions, for instance, being able to handle big data, return query results in a timely manner and so on, this also means that automation needs to increase in all steps of the value chain setup and management. While decision making still needs human oversight, automation should be the target for all the frequent interactions along the value chain. This means that data sharing and access should be automated, with as little manual configuration as possible at setup time, and ideally no human intervention at execution time. For instance, adding new actors and their data sources to the network will likely involve some human mapping and configuration effort, e.g. to set up a sharing endpoint and provide access rights, but at runtime the exchange should be automatic.

This in turn implies that data sharing needs to be based on standards: not require every other actor in the network to adopt yet another technology or data format into their IT infrastructure, but to use what is already available and proven to scale. It also means that data sharing needs to be based on agreed vocabularies, i.e. ontologies, as the language for interchange of information. Ensuring not only technical but also semantic interoperability of the information to be shared.

What is available

The Onto-DESIDE project has developed and demonstrated one way to set up such an infrastructure: it is decentralised, secure, based on existing and emerging web standards, and supports a high degree of automation in data sharing and access. Over time, individual components can be replaced—for example, new software tools for hosting and managing data may appear—but the overall approach to designing and implementing the infrastructure, as described in this guide, remains valid. This guide therefore focuses on the key steps needed to decide how to set up such an infrastructure, how to build applications on top of it, and what kind of work is involved in actually implementing and running it in practice. The life-cycle process described in this guide consists of ten steps. It starts with understanding the needs of the intended value chain configuration and ends with maintaining and evolving the infrastructure once it is in use by value chain actors. The guide can be used by decision makers to understand which considerations are involved in setting up such an infrastructure, and by technical teams as a high-level overview and checklist when implementing it within an organisation or across a network of organisations.

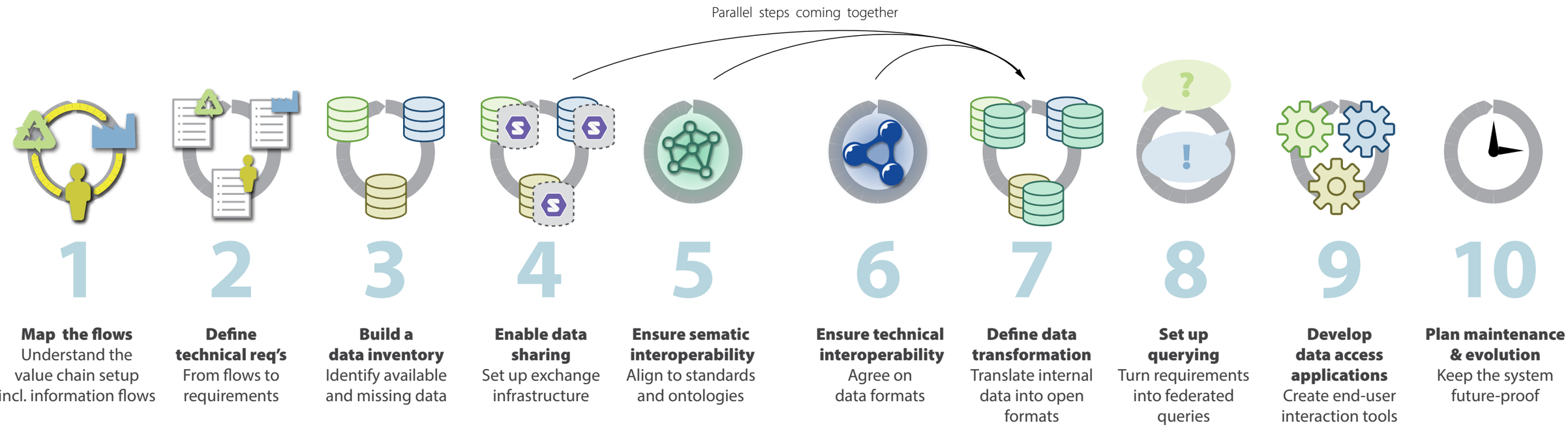


Figure: **The process for setting up your decentralised data infrastructure, starting with a strategic phase, followed by increasingly technical development.**

Step 1 - Map information flows—value chains are more than resource flows alone

Responsible role: Decision maker(s). **Participants:** End users who will practically implement the flows.

> See also Guide 1:
Value chain design, development & innovation

Resource flows
Guide 1 introduces a set of tools from the Circularity Thinking innovation methodology to support developing what (set of) circular strategies may be used in a given situation. It is a life-cycle and systems-based approach that "follows the flows": always asking where a resource comes from, where it goes, what forces drive this, what impacts it creates—and how these flows can be improved to become more circular and regenerative. This analysis helps to design how resources should flow in a value chain. But resource flows are only one part of the story: to ensure robust and well-functioning circular systems, the methodology also includes a range of other essential flows.

And also: value-flows…
For circular value chains to function at scale, value flows matter greatly. They determine whether circular strategies make business sense. A product may be recyclable, but unless customers, producers, and/or regulators agree on the value it creates, it may never be adopted. Value includes not just money but also environmental and social benefits. For instance, a take-back scheme for electronics only works if customers see value in returning devices, and businesses can capture value from resale, refurbishment, or recycling.

… energy flows…
Energy flows can be equally critical, as every resource loop consumes or saves energy. Some circular processes are far more energy-intensive than others—for example, melting metals back for recycling requires much more energy than reusing a part. Factoring in these energy costs can therefore be important when choosing which circular strategies make the most sense. At the same time, energy flows can also *enable* circularity: waste heat from one factory can power another, renewable electricity can drive recycling processes, and smart grids can synchronise production with peaks in renewable supply, making processes both more sustainable and cost-effective.

… and information flows!
But without reliable data on resource condition, origin, or composition, resources remain invisible, untrusted, and underused. And likewise, without trustworthy information about the actors involved and the processes they apply, circular collaboration cannot be orchestrated effectively. Companies need to know who potential partners are, which standards they follow, and how their practices align with environmental, social, and economic requirements. Without this visibility, it becomes difficult to discover promising partners, evaluate risks, or configure value chains that are efficient and resilient. In this sense, *information flows are the backbone* of circular systems.

Just as materials, parts, and products circulate through supply chains, the data describing them must also flow—through mechanisms such as digital product passports, certification schemes, or open data platforms—to guarantee traceability, compliance, and quality assurance. Construction firms, for example, rely on material passports to verify that reused doors and windows meet safety standards, while fashion brands depend on accurate fibre content data to enable textile-to-textile recycling at scale. Only when such data travels consistently, transparently, and across organisational boundaries can circular systems expand beyond isolated pilots and mature into trusted, efficient, and resilient networks.

With enabling infrastructure
Each type of flow also requires its own enabling infrastructure. Logistics hubs and processing plants support resource flows; value flows need contractual, accounting, and financial mechanisms that allow costs and benefits to be distributed fairly; energy flows require renewable and smart grids to match supply and demand efficiently; and information flows depend on data standards and digital platforms to ensure that actors can exchange reliable, trusted data.

Truly circular value chains integrate all four flows—resources, value, information, and energy—with infrastructures designed to move them smoothly. To not do so, risks creating bottlenecks.

Understand the whole to set information needs
Guide 1, and in particular the Multi-Flow Method, helps circular innovators look at each flow in a value chain and understand what is driving it, and how the different flows influence one another. By working with what we call *generative tensions*, the method uncovers the underlying causes behind sets of barriers and enablers, instead of only dealing with their symptoms. Taken together, these insights provide a systemic perspective on how flows, actors, and constraints interact.

In practice, working through the steps of the method provides a systemic overview that shows where challenges lie and what information is needed to address them. Mapping the flows reveals where data is missing, siloed, or poorly connected, and where sharing is critical for effective collaboration.

Clarifying flows and their interconnections exposes leverage points for greater circularity and shows how information underpins collaboration.

This analysis highlights gaps in transparency, duplication of effort, and weak or misaligned incentives for data sharing, thereby ensuring that technical solutions are grounded in the lived realities of actors across the value chain.

The insights gained—such as identifying actors, clarifying information needs, mapping existing data, highlighting gaps, and exposing barriers—are an essential starting point. However, at this stage they remain too high-level to guide infrastructure design. Subsequent steps (2 and 3) are therefore required to translate them into detailed technical requirements and system specifications.

Step 1—Examples for different circular strategies

The process explained in Guide 1 *Value chain design, development & innovation* serves to map the current as well as the desired circular flows: resources, value, energy (if needed) and, of course, information flows—who moves what, who benefits, with which evidence. This shared picture is then examined through the lens of recurring tensions—Individual vs. Collective Interest, Robustness vs. Adaptability, Concentration vs. Distribution—so that real bottlenecks surface. This enables targeting the highest-leverage frictions rather than symptoms, to guide value chain design and improvement. For our four circular strategies, we illustrate below what insights can be used to kick-start the development of new data and information sharing infrastructures. See for more detail on the analysis Guide 1.

(A) Beginning-of-life: using recycled input

What: Cross-sector recycling of apparel waste into feedstock for floor tiles.

Current undermining tension—Individual vs. Collective Interest: Each actor optimizes for self-protection. Suppliers upload batch "proof" as PDFs with metadata at different levels of confidence and granularity ("rubber≈20%"), but keep sensitive fields—phthalates, heavy metals, formulations—offline. Recyclers list vague tags ("post-consumer, clean"); processors and buyers can't verify claims such as "≤0.1% phthalates" or binder compatibility without seeing company secrets. Suppliers won't risk exposure; buyers won't risk non-compliance. Deals fail not for technical reasons, but because evidence can't be shared selectively.

Improvement opportunity for data flows: Apply governed, selective disclosure. Certificates become machine-readable and are mapped to shared terms; sensitive results are issued as verifiable credentials with field-level, purpose-bound access and audit trails. A supplier can prove "Batch X meets limit Y" without revealing spectra or recipes. Predefined queries check compliance across decentralized stores, preserving ownership. With accountable, granular sharing, trust rises and qualified recycled batches can flow—opening trade across sectors. It is decided to create this.

(B) Middle-of-life: repair

What: Repair of audio system through access to reliable spare parts & instructions.

Current undermining tension—Concentration vs. Distribution: Repair data, diagnostics, and spare-part info are locked behind OEM portals and contracts. Access to repair guides, compatibility, and pricing is controlled by OEMs, while building owners must scrape PDFs or call helpdesks, and updating a building's digital twins is cumbersome. This erodes trust, invites errors, and tilts decisions toward replacement: unable to verify "fit-for-use," and nudged by warranties and liability, owners replace rather than repair—driving premature end-of-life and extra cost.

Improvement opportunity for data flows: Apply governed, selective disclosure. That is: encode diagnostics and compliance as verifiable credentials; grant time-limited, purpose-bound access to repair data; and automatically sync updated composition and sustainability attributes to the digital twin or product passport after repair. This aligns incentives—owners get proof, OEMs keep control—ensuring access to reliable parts and streamlined data updates, letting repair outcompete replacement. Creating this capability develops into the focus of the next nine steps in the process.

(C) End-of-life: reuse

What: Reuse and resale of a door for use in other building projects.

Current undermining tension—Robustness vs. Adaptability: Ahead of demolition, doors are assessed and listed, but rigid, document-based formats can't capture real-world variability. Key fields—dimensions, swing, interfaces, material, glazing, ratings, condition, and install history—are missing or incomparable so buyers can't test fit for new projects. Planning constraints aren't linked, and commercial terms sit in scattered PDFs. "Robust" formats (reports, photos) are too static to support confident pricing for reuse, so building owners accept conservative offers or default to disposal.

Improvement opportunity for data flows: Enable machine-readable door passports with geometry, interfaces, ratings, condition, provenance, images, and location, using shared vocabularies and verifiable credentials. Sync planning constraints automatically and manage contracts decentrally so only authorized parties can access them. Interoperable APIs let marketplaces and builders auto-check fit and code, improving negotiation with the demolition contractor and enabling confident resale into new projects. Putting this in place becomes the main concern of the project that follows.

D) End-of-life: remanufacturing

What: Take-back of the floor tiles by the manufacturer for remanufacturing.

Current undermining tension—Individual vs. Collective interest (Take-back): At end-of-life, owners and demolition crews optimise for speed and lowest cost, while the OEM needs predictable, quality-controlled returns to plan remanufacturing. Crucial evidence—lot IDs, composition/binder, contamination risk, install zones, uninstall technician, custody—sits in PDFs or isn't captured. With incentives split and proof missing, tiles are cherry-picked or downcycled, and OEMs can't secure stable and reliable feedstock for remanufacturing.

Improvement opportunity for data flows: Use governed, selective disclosure. Issue machine-readable passports with lot/composition/binder/condition; attach verifiable credentials for "fit-for-return" and chain-of-custody. Publish reverse-logistics slots and price bands via decentralized pods (with commercials only to authorised parties). Predefined queries in planning tools auto-route eligible tiles to the OEM, aligning incentives and making take-back predictable and scalable. Enabling these solutions is what the next steps are about.

Step 2—Define technical requirements: what do users need data for?

Responsible role: Developer. **Participants:** End users holding the needs.

With the systemic insights from step 1 in place, the next task is to translate them into actionable technical requirements (step 2). This involves capturing the identified needs in the form of user stories, which describe in detail how different actors are expected to interact, what information should be exchanged, and under what conditions. These user stories serve as the requirements of the decentralised data-sharing infrastructure and its potential applications. Alongside functional requirements that specify system capabilities, it is equally important to capture non-functional requirements, which relate to aspects such as security, speed, and reliability.

Why user stories?

User stories are a tool used in software development that make complex requirements tangible and easy to communicate. Instead of long technical specifications, they are short, plain-language descriptions of what an end user wants to achieve and why it matters. The format is deliberately simple—*"As a [user], I want to [do something] so that I can [achieve a goal]"*—yet powerful in keeping the focus on outcomes rather than features.

The value of user stories lies in bridging strategy (where Step 1 offered insights) and implementation (from Step 3 onwards). They translate abstract system needs into concrete scenarios that reflect how people will actually interact with the system, ensuring that technical design choices align with business priorities and user expectations. In the context of decentralised data sharing, this could mean expressing requirements around how suppliers, manufacturers, or recyclers access and exchange information in ways that build trust and efficiency across the value chain.

By capturing both what the system should do (functional requirements) and how it should perform in practice (non-functional requirements), user stories provide a structured yet flexible foundation for development. They make it easier for diverse stakeholders—business leaders, technical teams, and end users alike—to work from a shared understanding, reducing the risk of misaligned investments or impractical solutions that hinder instead of help.

How to make user stories

The process of producing user stories needs input in the form of documented value chain information flows as described in step 1. In the previous step concepts such as actors, process, actions, and needs are described on a high level. These are further detailed in user stories as text using a structured format.

Each user story is a specific expression of a distinct need or interaction and will therefore differ from others by focusing on the perspective of a particular actor, a particular task, and/or a particular information requirement. Therefore, creating a broad set of stories is encouraged, since diversity ensures that requirements are captured from all relevant viewpoints and that no critical gaps are overlooked. Documenting these stories also helps to surface assumptions and make implicit expectations explicit, which supports alignment across stakeholders.

But as the number of user stories grows, there will be some stories that make more sense to be implemented before others. Prioritisation is not only about urgency, but also about logical sequencing—some stories may only deliver value once others are in place, while some serve as enablers for multiple others. Such dependencies should be noted in the stories.

Elements of a user story

User stories contain a number of common elements and each story is assigned a unique identifier (such as a number). Below more on the template used.

- ***As a [user]...***

The "As A" part is to be interpreted from the perspective of a certain actor or role that needs to perform an action. For example: you could examine the role of a building owner, a deconstruction company or a recycler, but also detail it further to represent specific roles within those organisations, such as an architect or a purchasing agent.

- ***… I want to [do something]...***

The following "I want", is a detailing of what needs to be done. For example, when dealing with information flows: one actor may be interested in understanding what options for treatment of a broken product exist, another may want to know how to dismantle something, and another may be interested in knowing the composition of a batch of materials. However, actions could also be concrete physical things that should happen, such as shipping a product, or melting a batch of raw materials. The latter may however be less relevant to the information sharing infrastructure, so try to keep the focus.

- ***… so that I can [achieve a goal].***

The "So that" describes the intended result or end state after the action is performed. For example: execute a repair, plan for processing and logistics, or make decisions on purchasing materials and planning production. Again, keep in mind that the focus is here on the value chain setup and execution, so goals should be aligned with the mapped flows from Step 1.

- ***Additional information***

This is a more detailed description of the situation or context that may be added to allow you to understand the conditions under which a user story applies, and potentially any secondary consequences or constraints.

- ***Data need***

Here you place a description or list of the information that the user would need to complete the task described in the above. This can be a list of data sources needed, or if possible a detailed list of data points.

Functional and non-functional requirements

By capturing technical requirements in the form of user stories, the needs from a user perspective are captured together with additional contextual information that further adds to understanding the need and how to address it. These make up the (initial) functional requirements.

But apart from the functional requirements on the system there are also non-functional requirements, such as security, privacy, usability and performance. These should be considered too, as they are also essential for a smooth use and operation of the final system. In Onto-DESIDE we used grouped lists, where requirements were grouped under their respective heading, e.g. "security", "usability" and so on. For example:

<u>Security</u>

- It should not be possible to manipulate source data by an unauthorized actor.
- It should not be possible to manipulate data in transit by an unauthorized actor.
- The platform's source code can be uploaded to a public repository (e.g. GitHub, Bitbucket) under an open-source license.

Who to involve to make user stories
The work of writing user stories is a collaborative process between the stakeholders involved in the value chain and are preferably done together, in person, so that uncertainties and nuances in perspectives are discussed and agreed upon. It is important to make sure that the different user roles appearing under “As a [user]...” are actually involved in this process, so as to validate the needs and the understanding of the context.

User stories would most often be written by a developer or systems engineer, with end users describing their needs and how they should be achieved in the system. Additionally, there needs to be a prioritization in what requirements are most important and in what order they need to be addressed. To do this prioritization architectural needs as well as functionality needs have to be weighted against each other. This prioritization could be done by a senior developer or a systems architect, someone with the authority, overview, and experience to make the correct judgements.

Next steps
Next, these user stories will be the basis for further detailing data needs in Step 3. Step 3 focuses on cataloging what data is needed to meet the needs specified in the user stories. Additionally, the non-functional technical requirements (together with the outcome of Step 3) will be used in step 4 as input to designing a suitable data sharing platform and in Step 5 to derive ontology requirements.

User stories based on the use case examples
Based on the circular strategy examples in section two possible user stories using the Onto-DESIDE template could look like this:

Story: repair-05

Depends on stories:
repair-03, recycling-07, logistics-02

As a *[user]:* Building owner

I want to:*… [do something]*
To be notified when a piece of installed electronic equipment in my building is faulty and needs repair or replacement.

So that I can: *… [achieve a goal]:*
Investigate the status of the faulting equipment and initiate relevant actions to restore it to a fully functional state.

Additional information:
A facility management system is in place and caters for providing information about the building and its health. This system is the primary interface for the building owner in maintaining and acting on data related to the maintenance of the building and its components.

Data need:
- Structural information for locating equipment in a building
- Data on installed equipment/components and contact information to suppliers
- Maintenance and repair information for installed equipment

Story: deconstruction-01

Depends on stories:
recycling-02, logistics-01, manufacturer-07

As a *[user]:* Deconstruction company

I want to:*… [do something]*
To have detailed instructions on how to deconstruct and handle floor tiles so that they can be reused or used as recycled raw material in producing new floor tiles.

So that I can: *… [achieve a goal]:*
I can dismantle floor tiles in the correct way and plan for the correct logistics of them.

Additional information:
A deconstruction company is subcontracted by the building owner and will manage the deconstrution of the building part according to a contract. As part of this contract, different circular strategies will be considered.

Data need:
- Structural information for locating equipment in a building
- Data on installed equipment/components
- Instructions on how to dismantle and handle floor tiles
- Information on circular strategies applicable for the specific floor tile

> See also User Story Template:
www.ontodeside.eu

Step 3—Build a data inventory: what data is available or could be collected

Responsible role: Data Steward(s). **Participants:** End users, to verify that data meets their needs.

Once the information needs are clear, a data inventory is needed. This inventory establishes what data is available, which actor has the needed information, what restrictions on sharing it with others exist, and in case no data exists: could it be collected, how, and by whom? This step may also result in revising the requirements, since some of them may be unrealistic in light of the data collection they entail, or because of restrictions on data access.

What is a data inventory

During the previous step, the data needs surfaced. However, it is hardly ever the case that the information you need perfectly matches the information you already have. To assess this, we first need an overview of the information already available—we need to build a data inventory.

A data inventory is a structured overview of the datasets you have access to and are relevant to address the user stories defined in Step 2 – Technical Requirements. These datasets can be internal or external, private or public, and are typically heterogeneous. That is: they typically encompass a variety of data formats (e.g. JSON, XML, CSV) and data structures (e.g. different database table schemas for different applications), often requiring careful analysis and management to ensure consistency and usability across systems.

This structured overview allows you to understand which informational gaps exist in your organization, and allows you to prioritize: either focussing on filling those gaps to cover all user stories, or focussing on the user stories that can be fulfilled with the existing set of data at your disposal, thereby guiding efficient resource allocation and more informed decision-making throughout the development process.

How to build a data inventory

A data inventory is typically documented in a structured format, such as a spreadsheet as illustrated on the next page, to ensure consistency and ease of use.

The process begins by taking each user story as input and answering the following questions:

- What data is required to resolve a user story?
- And: does this data already exist?

If *the data exists*, a description of the relevant dataset is added to the inventory. For each dataset, the following aspects may be documented:

- Location, format and access—Where is the dataset stored, in which format, and how can it be accessed?
- Ownership and governance—Who owns the dataset, and what usage restrictions or policies are applied?

For each dataset, the data elements (e.g. columns, keys, or references) should also be described:

- Definition: What is the meaning of the element?
- Relevance: For which user story is this element relevant?
- Sensitivity: What is the sensitivity level of the element?
- Access restrictions: What access restrictions apply? Who is allowed to see this data element?
- Data granularity: Can the raw value be shared, or only a derived or aggregated value?

This level of detail ensures that the data inventory not only supports technical implementation but also complies with governance, security, and privacy requirements.

If the required data does *not exist*, the next step is to analyze whether it can be collected and captured, and if doing so would be worthwhile.

A living document

A data inventory is a living document: building it is not a one-time task, but an iterative process that evolves as new information becomes available. Any change in a dataset that is part of the inventory—such as new data elements or revised access policy, but also the creation of new datasets—leads to an update of the data inventory and to potentially revisiting the subsequent steps in this guide. By maintaining this iterative and structured approach, a data inventory remains accurate, useful, and aligned with project goals. In addition, it provides a reliable reference point for all stakeholders, reducing confusion and ensuring that decision-making is consistently based on the most recent information available.

The hard part of building and maintaining such a data inventory is the human factor. That is: finding out which datasets are available to you and reaching a common understanding of what the existing data is about. In some cases also confidentiality of data can be a challenge, e.g. if even the presence of some data elements and their structure is considered confidential. However, the maintenance of such a data inventory can be done using a simple spreadsheet.

Review user stories

During the process, it may become clear that some user stories are not feasible because the necessary data is missing or cannot be accessed. In such cases, the user stories should be revised or marked as unresolved. These unresolved elements can be revisited in future iterations if new datasets become available.

ID	Description	Format	Access	Owner	Remarks
DS1	Building structural info	XML	\\drive1\buildings.xml	facility manager	
DS2	Equipment registry	CSV	\\drive1\equipments.csv	facility manager	Read-only access for rexternal repairers is allowed on equipment level during the period of the repair.
DS3	Supplier contact info	JSON	https://procurement/api/v1/suppliers	procurerment manager	For internal use only
...					

Column	Definition	Sensitivity	Remarks	Linked User Stories
id	id of the equipement	medium		repair_05, deconstruction_01, procurement_07
location	id of location (room, corridor ...) where the equipment is located)	medium		repair_05, deconstruction_01, procurement_07
supplier	supplier of the equipement	medium		repair_05, deconstruction_01, procurement_07
model	id under which the product model is known by the supplier	low		repair_05, deconstruction_01, procurement_07
installation_date	date of the installation of the equipement	medium		repair_05, deconstruction_01, procurement_07
cost	total cost of the equipement at the moment of installation	high	Only cost ranges per 1000 euro may be shared.	procurement_07
...				

Figure: **Example of a data inventory.**

Step 4—Enable data sharing: how and by/with whom?

Responsible role: Developer. **Participants:** Decision makers responsible for including relevant actors.

This step builds on the user stories and the outcomes of the data inventory outlined in the previous steps. That is: now that you know what data is available or will be generated, and you know when and how actors want to exchange this information, and under what conditions—you are ready to decide what kind of data sharing infrastructure is needed. Translating these needs into practice requires more than internal IT systems—it calls for data infrastructure that supports secure, scalable, and standardised sharing across organisational boundaries. Next, we focus on what is needed specifically for *decentralised* data sharing.

Why additional data infrastructure is needed

Data can relatively easily be shared internally using existing organisational deployments, e.g. by providing access to an internal SAP system, or maintaining datasets on an internal FTP server. While sufficient within one company, these approaches break down when collaboration extends to multiple organisations. Getting updates of different datasets—which, as we've seen in Step 3, are likely to encompass various data formats—would require fetching updates via each different protocol and manually integrating those different changes. This is time consuming and easily results in errors.

The web and the Solid protocol

Instead, we can make use of the Solid protocol, which builds directly on the global document sharing infrastructure of the web. By exposing data through web APIs in a controlled manner, information becomes globally accessible while relying on open standards and widely available tools, often at little or no cost. This move from closed systems to web-based infrastructure makes cross-company data exchange feasible at scale. It simplifies technical integration, lowers the barriers for new partners to connect, and creates a more resilient ecosystem where information can flow consistently across organisational boundaries without costly custom solutions. For businesses, this means faster onboarding of new collaborations, reduced IT maintenance, and a clearer path to scaling pilots.

Within the Onto-DESIDE project, the Solid protocols have been applied to support decentralised data sharing. These protocols are a set of open standards (based on the RESTful architecture style for web API design) that specify how secure but granular data sharing can take place on the web. In essence, they define the requirements of a Solid server: how a web server should respond to data updates and request calls, and how this aligns with authentication and authorization protocols so that data can be shared securely. Think of a Solid server as a type of Google Drive or Dropbox, but then for individual data points instead of documents, with sharing across platforms.

In practice, this means that one or more Solid servers, all using the same API, can scalably manage the pieces of data to be shared with other (internal or external) parties. Deploying a Solid server allows to more easily adhere to the same standards, making it easier to integrate your own data with that of your partners—and vice versa. Because each organisation maintains its own Solid server, the approach naturally supports decentralised data sharing: every actor keeps control over its data while still enabling interoperability across the value chain.

Solid is a compelling infrastructure choice because it combines control with openness, aligning technical efficiency with business trust. It provides the foundation for scalable, trustworthy data-sharing ecosystems that transform collaboration into lasting competitive advantage.

Sharing data: servers, workspaces and pods

Within a Solid server, multiple workspaces can be defined. Each workspace is informally called a Solid pod. When you create an account on a Solid server, you receive an account ID in the form of a URI, called a WebID. Access rights are then managed by linking WebIDs to specific data resources and defining the type of access each account has (e.g. read-only, read-and-write, or administrator). Data can be partitioned into resources, giving you full control over storage, grouping, and sharing. The next step is choosing the infrastructure to host your Solid server.

Decide on the infrastructure

To make use of the Solid protocols, a Solid server is needed. As with any Software-as-a-Service, the strategic 'Build, buy, or rent' choice arises:

- ***Build**:* Develop software or technology from scratch, creating a highly customised solution and potential competitive advantage. This requires significant time, skill, and resources but allows full tailoring to specific needs.
- ***Buy**:* Deploy an existing implementation on your own infrastructure. This offers quick setup but limited customisation. Deploying an open-source implementation lowers the cost but comes without commercial support.
- ***Rent**:* Use a cloud or SaaS solution, paying for access and usage. This option provides flexibility, scalability, and reduced upfront investment, but may involve vendor lock-in.

For an initial pilot, we suggest using the MIT-licensed Community Solid Server[7], as this shows the web service's capabilities without requiring major effort or investment. For setup instructions, see the tutorial linked from our website.

More about Solid

<u>History</u>

Solid servers were initially presented as a way to manage personal data. Solid used to be short for *Social Linked Data*, for connecting personal data across the web. However, the Solid protocols have never been tailored only to personal data. Within Onto-DESIDE, we found that Solid also lends itself very well to industry and governmental data, *making it a strong candidate for supporting the kinds of information exchanges* described in the user stories.

<u>More about Data Spaces</u>

Readers that have heard about *Data Spaces* might wonder how Solid and Data Spaces are connected: the technologies and methods put forward by Data Spaces organizations such as IDSA or Prometheus-X mostly focus on domain-specific governance of data sharing. When going through their documentation, you will notice a lot of emphasis is put on how to create contracts between parties, to make sure data access rules are well established. Solid focuses on the technical protocols needed to let data flow easily between parties, once those data access rules are in place. As such, Solid and Data Spaces are complementary, but not yet fully integrated. Within this training guide, we will not go into detail how data access contracts are put in place, but focus more on how we can technically adhere to these contracts.

Step 2, 3 and 4—Examples for different circular strategies

As shown, Step 2 supports with clarifying the technical specifications. Step 3 involves creating a data inventory by identifying available datasets, their formats, ownership, and access restrictions. It helps clarify what data exists, what is missing, and what can be collected. Step 4 focuses on enabling data sharing by setting up decentralized infrastructure that allows secure, controlled exchange of standardized data across actors, while preserving data ownership and confidentiality. Here, we briefly illustrate—for each of the circular strategies introduced earlier—how the steps play out to reach the strategic objectives in each case.

(A) Beginning-of-life: using recycled input

What: Cross-sector recycling of apparel waste into feedstock for floor tiles.

Step 2 & 3—Requirements & Data Inventory: On a digital marketplace, the interior outfitter struggles to source recycled materials because the materials processor's data is inaccurate and inconsistent. Certificates arrive as PDFs and spreadsheets with varying terms. The actors first meet to clarify their needs, writing user stories that specify which data is needed, in which situation, and for what purpose.

The materials processor's data steward then builds a data inventory, cataloguing existing data, formats, ownership, gaps, and sensitivity. Most certificates list recycled content and composition; few include verified environmental impact scores. Toxic substance data is highly sensitive and only shared with trusted partners. A mitigation is to share thresholds only (as in REACH) and apply granular access restrictions.

Step 4—Enable Data Sharing: The data steward and marketplace developer enable data sharing by adopting a decentralised setup with Solid pods,based on the Open Circularity Platform (**OCP**), giving suppliers control over sensitive data. Fields needing protection get access rules. The processor subscribes to a hosted Solid service, configures access-controlled data fields, and registers its pod on the marketplace.

(B) Middle-of-life: repair

What: Repair of audio system through access to reliable spare parts & instructions.

Step 2 & 3—Requirements & Data Inventory: To enable sharing of repair options and instructions, OEMs and building-owner representatives make user stories—for example: "As a building owner, I want replacement-component options for a failed part, with costs and repair instructions, so I can decide between repair and full replacement." The OEM's data steward compiles a repair-asset inventory (manuals, speaker-module BOM, parts catalog, compatibility matrix, firmware notes, service logs), noting format, ownership and gaps. Certificates (safety, EMC, RoHS/REACH, recycled content) are registered, though few confirm post-repair performance. Sensitive fields—e.g., regulated-substance levels, raw diagnostics and pricing—are flagged for restricted, purpose-bound access.

Step 4–Enable Data Sharing: To share compliance data securely, the OEM chooses a decentralised setup using Solid pods on the Open Circularity Platform (**OCP**), which lets customers and suppliers retrieve data in a standardised way. The OEM can control access to each data element, ensuring only authorized actors view it. As a large company with many subsidiaries, the OEM hosts its own Solid server and offers this service to subsidiaries. They also maintain a registry of suppliers' pods to enable federated queries.

(C) End-of-life: reuse

What: Reuse and resale of a door for use in other building projects.

Step 2 & 3—Requirements & Data Inventory: The intermediary (marketplace owner) consults with other actors, buyers and sellers of reused building elements, what their data needs are, and they formulate a set of user stories. The intermediary's data steward then helps the building owner to create a data inventory for reuse-relevant data on doors, including sources such as pre-demolition audit (measurements, condition grades), installation history, certification (fire/acoustic), images, location and planning constraints. For each dataset, they record format, location, owner, update cycle, sensitivity and access rules. They flag sensitive fields (exact addresses, pricing) for restricted sharing, and log information gaps (e.g., missing hinge). This inventory clarifies what information exists, what is missing, and which additional data can be produced.

Step 4–Enable Data Sharing: The intermediary's developer sets up a Solid pod per building owner, as a part of an Open Circularity Platform (**OCP**) setup, so owners retain control over their data. The building owners can then share door passports, each in their own Solid pod, exposing some data publicly, and restricting access to commercial and location data to selected buyers. This enables secure, scalable data exchange across actors.

D) End-of-life: remanufacturing

What: Take-back of the floor tiles by the manufacturer for remanufacturing.

Step 2 & 3—Requirements & Data Inventory: First, the OEM consults owners, collectors and representatives of the demolition sector to map data needs for automating the take-back process. They capture user stories, e.g.: "As a building owner, I want to know whether the OEM offers a take-back system, and under what conditions, so I can choose the right end-of-life option." The building owner's data steward compiles a data inventory of installed components (materials per room with m², tile-type installation dates, installation specs such as binder/adhesive and underlay, batch IDs, removal methods). The OEM likewise inventories take-back data: needs, criteria and pricing. For each dataset, the inventory records format, owner, update cycle, provenance and access rules. Sensitive fields—such as exact locations and contractor rates—are flagged for restricted access.

Step 4–Enable Data Sharing: The building owner decides to publish his data on a Solid pod, using a hosted service providing access to the Open Circularity Platform (**OCP**), rather than transferring it to a central database of an intermediary. The OEM sets up their own **OCP** Solid server to host their pods, to hold data of various products. This way they both retain control over their data, sharing selected data elements with other actors.

Step 5—Ensure semantic interoperability: ontologies

Responsible role: Developer. **Participants:** Data Stewards + End Users to co-develop and verify the steps.

Before data can be shared through data infrastructure, e.g. Solid, it often needs preparation. To achieve true interoperability, data must not only be technically accessible but also semantically consistent. This means describing it with recognised standards and ontologies. We recommend relying on de facto standards within your field, complemented with the Circular Economy Ontology Network (CEON) developed in this project, to ensure cross-domain alignment.

The role of interoperability

Interoperability can exist at different levels, and basically means that entities can work together. In a technical ecosystem, interoperability usually means that a set of systems can operate together without requiring extensive human effort in the day-to-day processing and exchange of data. Achieving this requires looking at both technical and semantic interoperability.

- ***Technical interoperability*** ensures that the *form* (the syntax) of data can be exchanged. For example, agreeing on what file formats a system should import or export, or aligning on the API structure for web-based systems.
- ***Semantic interoperability*** goes further by ensuring that the *meaning* of the exchanged data is shared and understood. Even if two systems agree on a common format—e.g., CSV files or an API with certain parameters—there remains the question of what the data values actually represent. Semantic interoperability means knowing that we use the same meaning for the concepts exchanged, and recognising when data is incompatible.

In general, technical interoperability can be solved by building file format converters and publishing API specifications. Semantic interoperability requires in-depth investigations into not only the data itself but also the processes of using and producing the data. Digital technologies can help to support this process, e.g., by allowing us to describe and capture the meaning of our data in a precise and formal way—understandable by both humans and machines. This is where standards and ontologies become essential.

Achieving semantic interoperability

To move from ad-hoc interpretation to shared meaning, data should be described using recognised standards and ontologies. We recommend relying on de facto standards within your field, and complementing them with the Circular Economy Ontology Network (CEON), developed in this project, to support cross-domain alignment. CEON provides a common vocabulary that bridges differences between domains, making collaboration smoother and more reliable.

Ontologies

An *ontology* is a formal model describing a domain of interest. It offers a way of capturing meaning, i.e. defining concepts and relations in the domain that can be used to describe our data, so that we precisely know the semantics of it. Commonly such ontologies are called “domain ontologies”, to separate them from models trying to capture the nature of the whole world, or application ontologies that may be specific to one particular system or type of application.

In order to create and share an ontology, and to make it formal and precise so that it can not only be understood by human users, but also by systems, we need a formal language in which to express the ontology. The most common language used today is a web standard, OWL[8], developed for expressing and sharing ontologies over the web. The language is based on other common web standards, e.g. using URIs to identify things so that we can point to concepts and relations in the ontology from anywhere on the web. The ontology is stored in a file that should be accessible over the web, so that the URIs identifying its concepts and relations actually resolve. To allow for describing a domain in detail, the language allows to express axioms that can be used for automated reasoning. While reasoning is a powerful feature, it also adds complexity. OWL provides the possibility to derive new fact and check consistency, but it is not a constraint language. For validating data, and verifying constraints, other languages exist, such as SHACL[9]–but this will not be covered in this guide.

To identify and link to ontologies OWL makes use of the namespace notion, which allows us to give an identifier to the ontology as a whole, and then extend that namespace to identify its elements. Normally the URI of the ontology is also its URL, i.e. the location where it can be found, as well as the namespace for its content. A URL can represent a public address on the web, or for instance, a location on an intranet. However, to facilitate a shared model for interoperability between actors, ontologies need to be accessible by the actors that use the data described by the ontology.

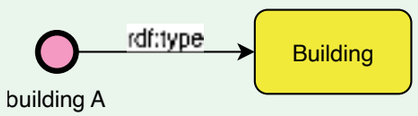
Figure: **Why are ontologies important?**

An example of an ontology

To understand and manage the ontologies you need in your value chain, it might be useful to first get a better understanding of what concretely can be expressed using ontologies. In this section we consider a simple example of an ontology consisting of just a few concepts and relations between them, to exemplify some of the constructs in the OWL language. The illustrations here are shown in a notation called Grafoo, though ontologies can also be visualised in other notations. However, an ontology is fundamentally a set of machine readable logical axioms, not merely a graph or diagram. The axioms are stored in an ontology file, using a syntax such as Turtle (see more details in Step 6) to make them understandable and usable by applications.

Concepts

The core element of an ontology is the concept. In OWL concepts are called classes, but since this is easily confused with the class notion in programming, we mainly use the term concept. A concept can be viewed as representing a set of things, its instances. So one way to view the concept "Building" is that it represents the set of all buildings. Another point of view is that it represents a type, i.e. it is a category that can be used to classify instances. The example below shows the concept "Building" and an instance of building, i.e. a specific concrete building, which we for the sake of the example have called "building A". The relation rdf:type is a built-in relation, coming from the RDF language[10] (as indicated by the prefix "rdf:"), that is used to connect instances, i.e. data nodes to concepts.

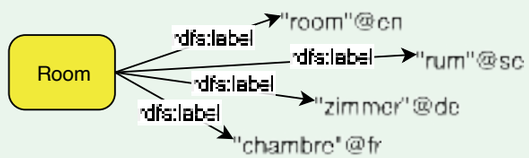


Relations between concepts

In an ontology we also want to express what possible relations may hold between instances of concepts. These are called object properties in OWL. The example below shows two concepts, "Building" and "Room", and a relation "hasSpace" that can connect instances of these concepts. In OWL, properties are first class citizens that can exist independent of any concept, but commonly we express their intended usage through some axioms in the ontology. This could be a restriction on the concept, or domain and range restrictions on the object property, which (in Grafoo) is illustrated by connecting the boxes.

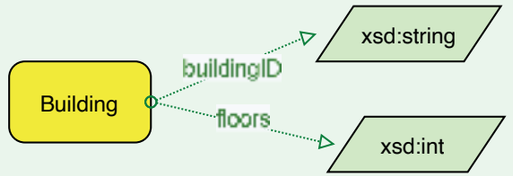
Lexical representations

Ontologies also help us to separate the lexical form of a concept, i.e. the different terms that can be used to refer to a concept or its instances, and the concept itself. Each box and arrow in our figures will have a unique identifier in the ontology file, independent of what label we give it. Thus this separates a concept definition from its naming, allowing us to, for instance, express synonyms, and to represent how different languages name the same concept. It can also be used to express different naming schemes, e.g. how different standards name the same concept differently. Below you see, the "Room" concept translated into English, French, Swedish and German:



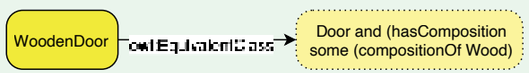
Attributes

Another type of relation is attributes. An attribute connects instances of a concept to literals, i.e.. data values. The actual data values are not part of the ontology, but in the ontology we express what attribute relations (called datatype properties in OWL) exist and the datatypes the data values have. Below we illustrate two datatype properties of a "Building", one that can hold string literals representing a building identifier, and one representing the number of floors as an integer.



Axioms and reasoning

So far we have only seen the features of OWL that allows us to "name things", i.e. to create a vocabulary for types of things and relations in our datasets. However, ontologies can be more than vocabularies. OWL is a logical language, which means that it allows axioms to derive new facts from existing data – a powerful feature, but also complex. For instance, we might want to classify elements in a building as potentially "recyclable" if they meet certain well-defined criteria. In the naïve example below we classify doors as being wooden if they are composed of some wood material, which illustrates such capabilities.



Namespaces and links between ontologies

Agreeing on one single naming scheme, or one single ontology, makes things easier for data exchange within a domain. However, when considering cross-domain scenarios this may simply not be feasible. Each industry doman has its own formats and standards, and in some cases already their own ontologies. So instead ontologies can support mapping between standards and ontologies, because ontology languages, such as OWL, are built for the web. So just as we can create links between web pages, we can also create links between ontologies, and even between single elements and definitions inside the ontologies.

Since all ontologies have unique identifiers, which can be resolved on the web, we can point to any ontology on the web from our data. This means that for instance, the "building A" in our first example on this page, may reside in a dataset on our company server, while the ontology containing the "Building" concept may be published in an entirely different location on the web. To make this work, we need to provide the "address" of each element we introduce, normally a URI. To make things more readable in a file, or an illustration like in this guide, we can shorten the URI to what is called a prefix - a short name for the longer address. In the example below you see two concepts representing a room, from two different ontologies - the "ex:" and the "wd:" are prefixes, which in the ontology file then has to be connected to the correct URIs.

Different uses of an ontology

An ontology is in itself merely a model of some domain, i.e. making it explicit how that domain views the world, how it defines different concepts, and how the concepts are related. This means that an ontology is NOT in itself a data format, it is NOT even a schema for creating data in any format. As such, an ontology does not *prescribe* any format or any structure of the data. It can simply be used as a terminology (or mapping between different terminologies), providing a *description* of the domain. Since ontology elements are uniquely identified by URIs in OWL, this means that descriptions do not have to reside in the same place as the data, but data can link to ontologies published elsewhere.

Using an ontology as a vocabulary

An example of this usage could be to apply an ontology to define the keys (names of attributes) in a JSON structure, using the JSON-LD syntax. In this example, see image below, the @context key identifies that it is the GS1 Web Vocabulary (an ontology, available under URI https://ref.gs1.org/voc/) that contains the definitions of the attributes “globalLocationNumber” and “organizationName” that are used to describe the entity identified by the @id URI, which this JSON document is about.

Figure: **A JSON example.**

Using an ontology for data access

Another use of an ontology is to express queries over data. As you will read about in step 8 of this guide, expressing queries over data needs to be done using some vocabulary, i.e. we need to know what we can ask for. In this guide we assume that all datasets are actually transformed into RDF (as discussed in step 7), but even if this was not the case we could express queries based on the ontology and then create a mapping to the data sources using their own specific schemas.

Using an ontology as a data schema

An ontology can also be used as a schema for expressing data, i.e. as the structure of a graph dataset. If we would like to create an integrated knowledge graph, for instance, containing data from different data sources. Then the ontology could be used as the schema of that knowledge graph. But it is important to keep in mind that this is just one possible use of the ontology, and which use is right for you depends on the decisions made in the other steps of this guide.

Example data, illustrated as a graph (knowledge graph) with individuals as dots, representing instances of the building and room concepts illustrated earlier, and values for the two datatype properties in the previous example can be seen below:

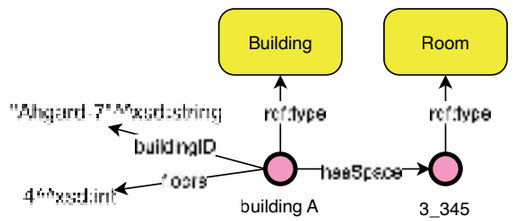


Figure: **A data schema example.**

Why does CE need ontologies?

CE inherently means collaboration, and collaboration means the need to understand each other. CE also implies complexity—complex systems that need to scale across organisations, material flows, and time. Most of our current IT systems have not been built for such scenarios, but targets the internal processes of the organisation. Many also have implicit information models and system-specific data formats that are hard to explain and share with others. To some extent standardisation has aimed to alleviate this problem, but usually on a per-industry domain basis, whereas CE goes across domains. In such a scenario it is important to make definitions and assumptions explicit, and to be able to map between a plethora of different standards, formats, and models, and make domain assumptions explicit to others—also to other technical systems. This is where ontologies play a crucial role!

Next steps in ontology support

With semantic interoperability understood, the task is to assess needs and determine ontology support in your case.

- ***Ontology requirements and inventory (5a):*** Identify technical requirements and review existing ontologies, including **CEON** from Onto-DESIDE. Carefully assess their coverage and fit for purpose.
- ***Ontology extension (5b):*** If **CEON** or other ontologies do not cover all requirements, extend them to meet specific needs.
- ***Ontology alignment (5c):*** Where multiple ontologies are in use, connect and align them to ensure consistency and interoperability.

Outcome: Step 5 delivers an ontology requirements specification, together with a network of extended and aligned ontologies, designed to support your value chain’s data sharing infrastructure effectively.

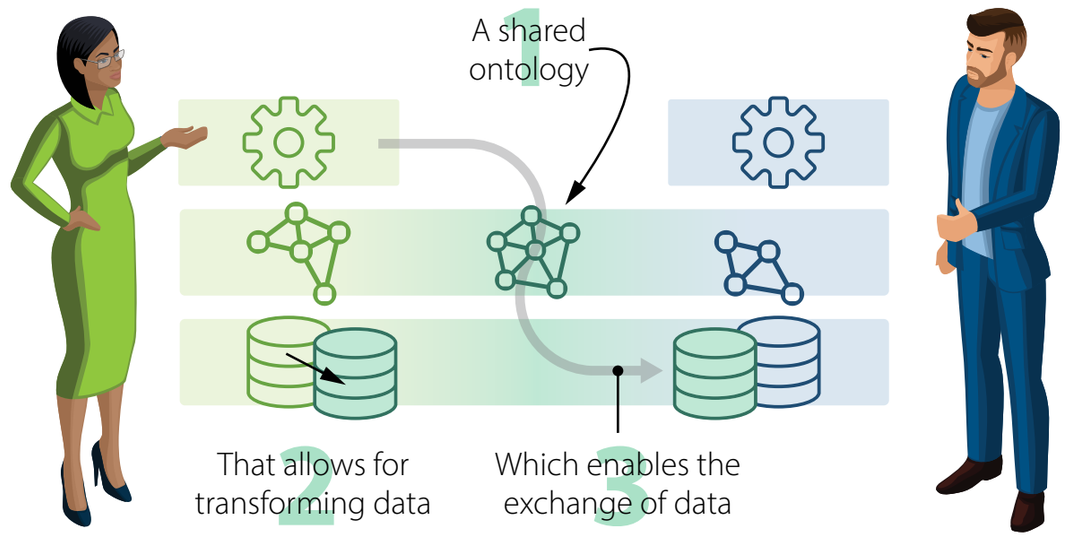


Figure: **How ontologies work—they provide a shared language for organising data.**

Step 5a—Ontology Requirements and Inventory

Responsible role: Developer. **Participants:** Data Stewards + End Users to co-develop and verify requirements, and verify the steps taken by the developer.

It is time to figure out what ontology (or ontologies) we need in the setup of our particular value chain collaboration. Based on the output of steps 1-3, i.e. an understanding of the value chain, its actors, needs, and the data required to make the intended flows happen, as well as the inventory of what data exists and where, it is now time to see how all this maps to an ontology. To have more detailed guidance in the following steps, you may want to look further into a specific ontology engineering methodology, such as LOT[11], and also think about how these steps will interface with your existing information management and systems development practices.

Ontology requirements

In step 2 we determined the overall system requirements of the data sharing infrastructure of the value chain, and in this step we figure out what this means in terms of ontology requirements. Ontology requirements are often expressed as Competency Questions. Competency Questions (CQs) are questions that should be possible to answer using the ontology, or data described using that ontology, and should be based on the information needs of the actors in the user stories. You could view it as listing what questions different users and actors in the value chain would be asking the overall distributed data of the value chain in specific situations. It is usually a good idea to express CQs in generic manner, i.e. without mentioning explicit data examples, but rather types or categories of data. For instance, while "What is the percentage of ALU content of this window construction?" would be a valid CQ, a better one, making the scope and level of detail of the intended ontology more clear, would be "What is the weight percentage of a specific substance in a certain building element?" for an ontology targeted at the construction sector, or even "What is the fraction of a specific substance in a specific physical object?" for a more generic ontology about materials. If expressing questions feels awkward, the requirements can also be expressed as simple sentences, specifying what the ontology needs to cover, such as "Physical objects and their material content, expressed using different metrics with associated units of measure."

In addition, CQs may represent reasoning requirements, i.e. things that should be derivable based on the ontology or its associated data, rather than merely "retrieval questions". For instance, a CQ like "What is the best end of life scenario for a certain used product?" will probably not imply a "lookup" of this in a database, but rather deriving different suggestions for end-of-life scenarios based on a set of parameters of the product in question. Such CQs may spark valuable discussions on the scope and task of the ontology. To what extent should this be derivable through "rules" expressed using the ontology, and to what extent should this be a manual task, or a task of an associated recommender system perhaps? Thus in some cases you will encounter needs that first seem to be ontology requirements, but in the end may result in other kinds of requirements, that may not be solvable by the ontology itself but rather requires a certain software solution or application to be built on top of the data. Adding lots of axioms in your ontology will make it complex, both for humans and machines. So be aware that this should be clarified before starting to build extensions to the ontologies, to avoid having overly complicated models that in the end still may not solve the need you actually had—ontologies do not do magic, they are merely models of the domain, and documentation for your data!

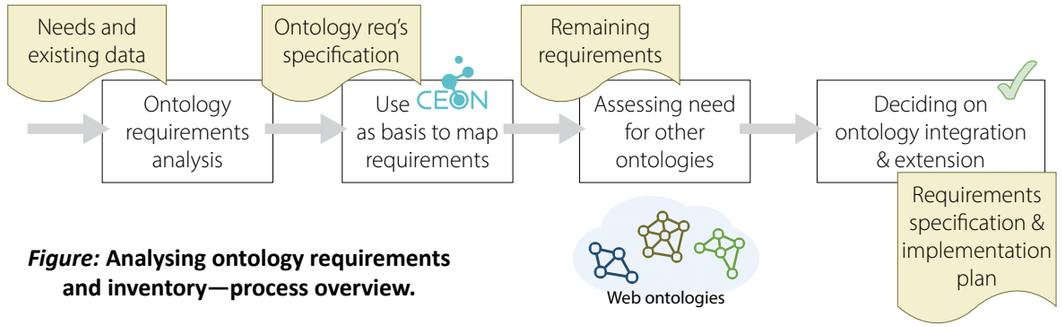


Figure: **Analysing ontology requirements and inventory—process overview.**

Requirements can also be based on data that is already available, and processes that are already in place, e.g. as discovered in the data inventory step. Studying such existing data may yield additional CQs representing typical queries that are already today used to retrieve the data. Or you may simply use sentences to list the "entities" that exist in those datasets, and their associated attributes. But also make sure to capture any hidden assumptions, e.g. units of measure that are not explicit, or whether there is a need to also capture the provenance of the data, apply change tracking or not, etc.

The output of the ontology requirements engineering process should be an ontology requirements specification document. Take care to give CQs identifiers so that it is easier to refer to them, and to indicate when they potentially change. CQs should also be verified with the actors identified as holding those needs, and may have to be prioritised if there is a long list of them.

Understanding **CEON**

Once requirements are in place, the next step is to identify to what extent these are already solved in existing ontologies. In the Onto-DESIDE project we have developed an ontology network specific to CE needs, called **CEON**. It is modular, to allow reuse and extension of the single modules, and to make it more flexible to varying needs.

To map the CQs to the **CEON** ontology network, the ontologies can be loaded into a tool for visualisation and editing (such as Protégé), or the online documentation can be used. Entering the URI of the ontology module in a browser window will normally take you directly to the documentation page, where you can see a visualisation of the ontology, as well as lists of all its classes (concepts), properties (relations) and their natural language representations (labels) and definitions (comments). For each CQ you can search for the specific terms included in the CQ to check whether they are actually present in the documentation, and may map to classes or properties. However, in most cases the terminology of your specific requirements may be too specific compared to the one used in the ontology modules, hence, you will need to either search for broader terms, or make a manual assessment by reading through the lists of concepts and relations and their definitions.

In some cases, the ontology may need to be

studied more in detail to make sure that a CQ is fulfilled, e.g. checking whether there are also sufficient properties connecting the concepts found, and that their definition (also in terms of domain and range) match the usage you have in mind. This can be done through the ontology visualisation, or by opening the actual ontology file in a tool of your choice.

For example, consider the case where you are the manufacturer of a floor tile systems, and your CQ reads “What is the material composition of a specific floor tile?”. “Material composition” can be found in the ontology documentation of the materials module of **CEON**, while there is no concept representing “floor tiles” in any module. Instead, there is a general concept representing “products”, where in this case the “floor tile” is your specialisation of this concept. Additionally, you check the ontology visualisation to make sure there is a connection between products, and materials. Hence, the conclusion would be that parts of the CQ is directly modelled through the materials module, while another part requires a specialisation of the products module, but it is possible to express their connection through existing classes and properties.

After completing this, you will have an assessment for each CQ, stating whether they are (a) directly modeled by **CEON** already, (b) need a specialisation or extension of **CEON** to be satisfactorily modelled, or (c) seems to be completely out of scope of **CEON**, or are modelled or defined in an incompatible way in **CEON**.

Other ontologies?

In some cases you may already be using other ontologies, or there are other ontologies already built for a specific concept that goes beyond **CEON**. In this case, it is good to first check whether **CEON** also already provides an alignment to such ontologies. In many cases, this may actually have been considered already. For instance, **CEON** contains references to GS1 Web Vocabulary, as well as the EMMO upper ontology for materials modelling. To allow flexibility these links are mainly included as “see also” references in the ontology modules, rather than explicit relations, while also a number of alignment modules are available. When checking if **CEON** already aligns with your ontology of choice, check for “see also” annotations, and review the alignment files available in GitHub.

Decision point - Is the ontology sufficient?

After this step, you should now make a decision how to proceed, depending on your assessment of on one hand the coverage of **CEON** and related ontologies over your CQs, and on the other hand if there are additional ontologies you need to align to.

If all your CQs are sufficiently covered by **CEON**, or **CEON** and ontologies where alignments are already provided, then you can proceed to step 6. If there are parts of your CQs that are not sufficiently covered, and you think extensions to **CEON**, such as adding more specific concepts, or extending the scope, are needed, then you should consider step 5 b – ontology extension. If you also would like to connect additional ontologies to **CEON**, or your extensions, that are at the moment not aligned with **CEON**, then consider also Step 5c – ontology alignment.

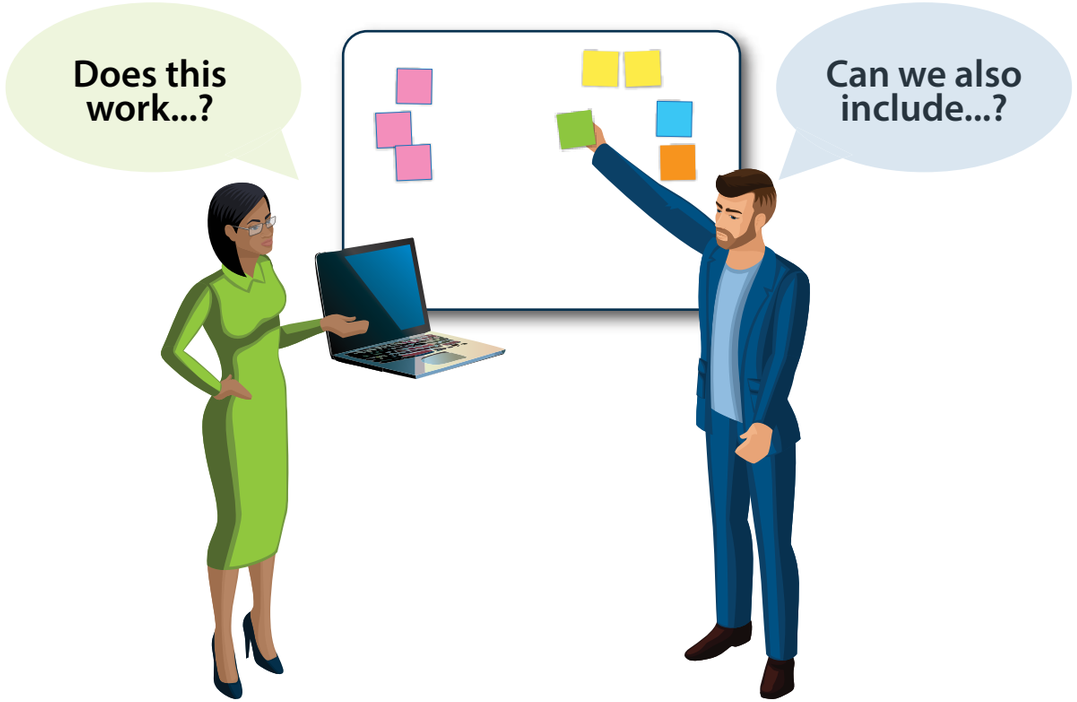


Figure: **A developer discussing with an end user to verify the requirements.**

Step 5b—Ontology Extension

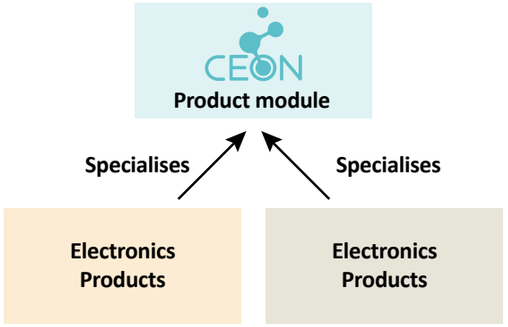
Responsible role: Developer. **Participants:** Data Stewards + End Users to verify steps taken by developer.

Given that the assessment of CEON and other existing ontologies resulted in the conclusion that some extensions are needed, a structured approach to such extensions is necessary. In this guide we only briefly describe the steps needed to create an extension of a CEON module. If a completely new ontology has to be created, from scratch, we recommend to study ontology engineering methodologies more in detail, and set up a suitable plan for performing the ontology development project based on that.

Why specialising an ontology?

Our focus here is on ontology extension, i.e. specialising and potentially slightly extending a specific CEON module, based on requirements identified in the previous steps. Specialisation means to add more specific concepts and relations that better describe our data, i.e. adding domain-specific modules as illustrated in the figure to the right, rather than the generic cross-domain concepts in the core CEON modules. The motivation for adding such concepts and relations can be both purely technical, i.e. the need to further specify the concepts so that certain consistency checking or inferences can be made, or also a matter of communication.

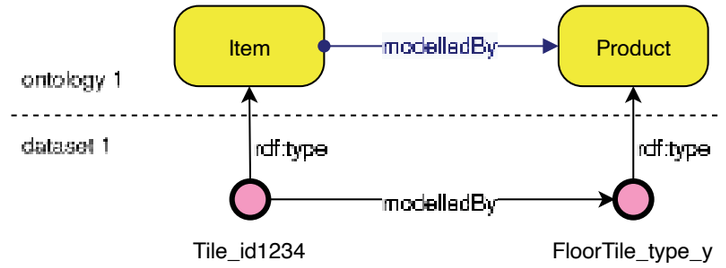
Consider CEON concepts such as "product" and “component”, which can include multiple kinds of things depending on the industry domain, and also the perspective of the organisation where the data originates. In the case of our running examples (A-D), we would rather like to have a more detailed taxonomy of products, such as “building element”, “door” and “floor system”, as well as more detailed component concepts, such as “floor panel” and “floor pedestal” as components of a floor system. The key idea of specialisation is that the new concepts and relations are added



mainly as subconcepts or sub-relations of already existing concepts and relations. While a general extension may also add concepts and relations completely independent of existing concepts and relations in the extended modules.

The benefits of an extension, rather than modelling something from scratch, is that the taxonomy can be used to reason over and query data regardless of at what level in the taxonomy it is described. Consider a dataset that is described using the core CEON module, describing a certain object as an item of a certain product type (see top of the figure to the right). Another dataset may use a more specific ontology that describes another object as a tile of a certain floor system product (middle part of the figure). When querying over both these datasets, asking for all tiles will only return the data tagged with the more detailed concept, i.e. only the result from dataset 2, while asking for all items will return both the items described using the generic concept, but also everything tagged with the subconcepts, such as “tile” (bottom of the figure). Hence, we use the semantics of the subclass relation between concepts, to allow us to seamlessly integrate data that is described with ontologies at different levels of detail and granularity – as long as they both rely on the same core concepts.

Other benefits include the possibility to include more specific relations or restrictions that apply only to the subconcepts. For instance, some characteristics, such as length, width, height, and material composition, may be relevant for almost all items, while other aspects, such as the maximum point load a tile can withstand makes much more sense to model at the tile level (or a potential intermediate level of built elements).



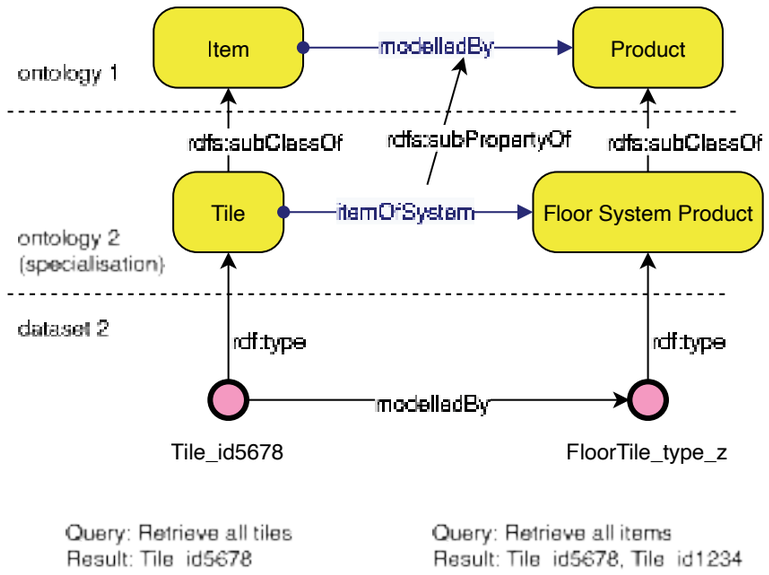


Figure: **An example of extending an ontology with domain specific concepts.**

Specialisation

In order to create appropriate specialisations, consider the ontological requirements (e.g. CQs) that were identified as not immediately covered by **CEON** or other existing ontologies. If the matching process in step 5a, understanding **CEON**, resulted in a number of identified connections to CEON, these can now be used to add subconcepts representing the missing specific concepts.

Start by creating a new ontology file, i.e. do not modify the **CEON** files directly, but create your own ontology file with a new identifier and then import the **CEON** module you want to specialise. The new file should have its own identifier, using a URI that can be resolved (whether externally or on your intranet) when you publish the ontology. Then add the specialisations in this file. Commonly a modelling tool, such as Protégé, would be used to allow for graphical modelling. In Protégé, imports can be added using the user interface, and then subclasses, subproperties, and additional axioms can be added, before the file is saved and published.

Where to actually "attach" your extension is not always straightforward. As mentioned above, you should have identified some connections between your requirements and **CEON**, or any other ontology you are extending. However, even if you have identified two concepts that you will specialise by creating subclasses of them, understanding how they are to be "connected" is not always obvious. In this context, it is important to remember the fact that OWL treats relations as first class citizens, i.e. properties exist in their own right, and not only as something "attached" to a concept. This is a great flexibility, but also a usability challenge as it may be hard to understand the intended usage of a relation, and thus also how to specialise it. Usually, some axioms are added to the ontology to prevent misuse of the relations, and to show the intended usage. This can be through domain and range restrictions on the relations themselves, or by setting a restriction on the class. Logically these have very different meanings (semantics), but unless you are going to use a reasoning engine (see more on this below) you can view this as cues about the intended use.

Another issue that may arise is that you seemingly cannot find any direct relation between the concepts you are adding subconcepts to. In some cases, this may of course be because they are not actually connected in the model. But in many cases this may simply be because the connection is not direct. In OWL only binary relations exist, which creates a challenge when we need to connect multiple pieces of information that depend on each other. This could be for instance when we want to express material content of an item, and need to specify both the matter involved, the amount of it, and a unit of measure. For example, saying that "my tile contains 70% calcium sulfate" means connecting a specific tile ("my tile"), a material (calcium sulfate), a number (70), and a unit of measure (%). This cannot be expressed through a direct relation. Instead we have to use a technique known as reification, i.e. we treat the relation itself as a "thing". This can be done by adding a class to the ontology representing the relation, e.g. in this case a MatterComposition class, that then relates to the item, the matter, the value and the unit. **CEON** is actually full of these reifications, in order to allow for very flexible modelling, and also to enable to track metadata on these kinds of statements.

Design choices and tradeoffs

To make your additions reusable and maintainable also keep in mind that *modularisation* is beneficial. Hence, adding too many new concepts and relations into the same extension module may not be the best choice. Consider to divide the extensions into subsets, either by criteria that make sense to users, such as extensions that will be used for certain queries or certain applications, or layer the extensions so that their granularity fits certain datasets to be described by the ontologies.

Other things to keep in mind is the tradeoff between being very *detailed* versus sticking with less detail but instead being more reusable. Look at the technical requirements defined in step 2 and think about how your module might need to generalise in the future, or whether it can be designed in a way that can cover more than one requirement (e.g. a broader set of CQs and/or more of the user stories) at once. There is no right or wrong here, but this needs to be a decision made by the developers, in close collaboration with data stewards and end users, who may be the ones knowing what to expect in the future in terms of changes to the data or new uses planned.

Another design choice involves what *level of axiomatisation* to add to the ontology. We actually already saw an example of inferences on the previous page, where we used the semantics of the rdfs:subClassOf axioms to infer that the instance of a subclass is also an instance of the superclass, i.e. the tile is both a tile and an item in the case of the extended ontology. OWL also allows for much more complex axioms than this. What is important to remember is that reasoning is a powerful tool, and can be very valuable to both check the consistency of the domain model, as well as infer new information. However, these complex axioms also add complexity to the model. Both complexity in terms of understanding the model, so it becomes harder to use, extend and modify in the future—it is easy to accidentally introduce conflicts and unintended consequences in an ontology with many complex axioms. But also complexity in terms of the computations that are done using the ontology.

For reasoning there are OWL inference engines already available, as general purpose software that allow you to derive the conclusions that you can draw based on the model and associated data. However, these reasoning engines will be slower the more complex the model is, and in some cases there may not even be guarantees on their termination. The same goes for queries that use inferencing as a pre-processing step, such as was the case of the tile example, where we first have to derive the full set of types (concepts) the tile belongs to, before the actual retrieval takes place. This is powerful, but time consuming. So carefully assess the need for such reasoning to take place, before adding too many axioms to your ontology extension.

Documentation and publishing

Once you finish your extension, make sure to properly document all the new elements by using rdfs:label for human readable labels, and rdfs:comment for human readable definitions and explanations of the concepts and relations added. Also document the ontology itself, by adding metadata to it, such as a version number (for keeping track of new versions when changes are made), author, and publisher etc. If the ontology is to be made publicly available, also include licensing information, for enabling its reuse by others.

Testing

As hinted previously, sometimes the models become large and complex, even when split up into a set of modules. So another piece of advice is not to underestimate the need for testing of your ontology. Most ontology engineering methodologies include some evaluation or testing step, but they are sometimes not explained in detail. A minimal set of testing methods may include at least:

- Verifying the syntax by loading the ontology into a tool, and checking consistency of the model using an OWL reasoning engine.
- Checking for structural errors using a validator tool, such as the OOPS! and FOOPS! services.
- Testing your requirements fulfilment by expressing test data according to the ontology structure, and expressing some CQs (or existing queries) as queries over the test data, to verify that expected results are returned.
- Verifying that expected inferences are made over the test data using a reasoning engine, and provoking errors by adding erroneous data that should not cause issues when encountered by the reasoning engine.

Ontology Engineering Methodology

Several well established ontology engineering ethodologies exist, and a useful summary and overview of typical steps and activities can be found in the description of the Linked Open Terms approach[12]. However, other types of approaches and methodologies also exist, such as the eXtreme Design (XD)[13], which is more suitable for modular ontology development and rapid prototyping of ontologies. More concrete, detailed advice on actual modelling can also be found in the tutorial from Stanford University[14].

Step 5c—Ontology alignment

Responsible role: Developer. **Participants:** Data Stewards + End Users to verify the alignments.

In some cases, you may want to reuse concepts or relationships from other domain ontologies that are not yet represented in CEON or that are more detailed than CEON. Ontology alignment, also called ontology matching, is thus a key technique for enabling semantic interoperability[15].

Why ontology alignment?

In some cases, the ontologies you are working with does not cover all your needs. For example, you might need more detailed categories of engineering materials in your application, but CEON does not currently model them. If you identify another ontology that provides such detailed categories, you will need ontology alignment to integrate it with CEON. Ontology alignment establishes mappings between common concepts and relationships across different ontologies. These mappings can capture equivalence (e.g. two concepts representing the same meaning), hierarchical relationships (e.g. one concept being more specific, a subclass, of another), or even more complex relations than that. For instance, both CEON and the Digital Product Passport Ontology (DPPO) have *Product* concept definitions. If we establish an equivalence mapping or sub-relation mapping (CEON:Product is a DPPO:Product), we make it possible to utilize more semantics in DPPO, as graphically illustrated below.

What is ontology alignment?

Ontology alignment takes two ontologies as input and produces an *alignment* as output, that is, a set of mappings between entities from the input ontologies. These entities may include classes, object properties, data properties, or individuals.

In our project, we established a pipeline for generating alignments between CEON and other relevant ontologies. As illustrated in the figure to the right, once you have relevant ontologies, you can start with using one or more ontology

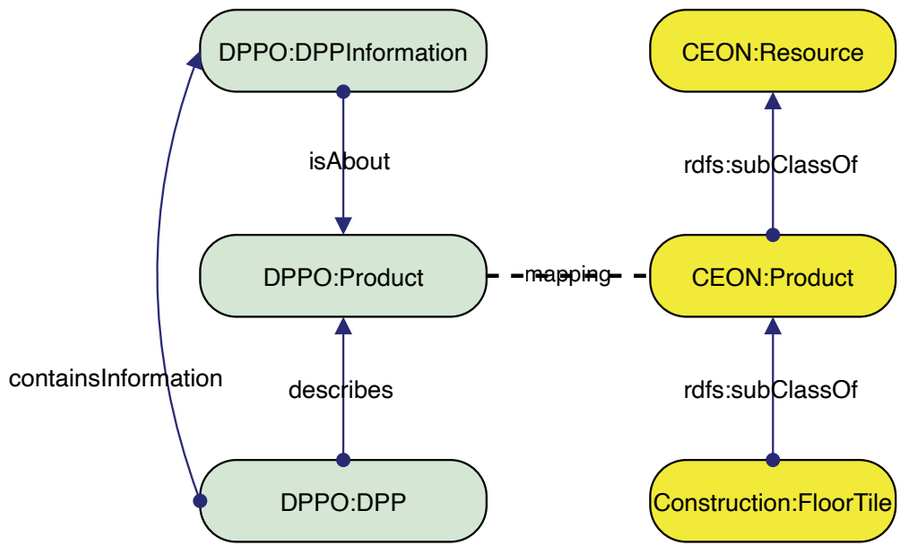


Figure: **An example of ontology alignment.**

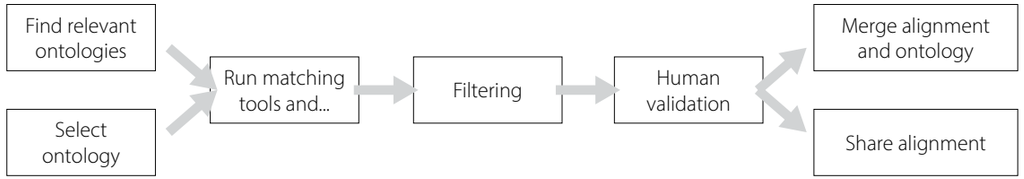


Figure: **A process overview for ontology alignment.**

matching tools to generate candidate mappings. These candidates then undergo an optional filtering step and a validation step to produce the final alignment. Filtering might mean that mappings generated by fewer than half tools are excluded to improve precision, if you use several ontology matching tools to produce candidate mappings. Then validation is a manual step, where a domain expert should check the output of the matching system, to make sure that no errors are introduced (since no such system is 100% accurate).

In our project, we have already provided sets of mappings between CEON and several relevant domain ontologies in the fields of materials, manufacturing, and products. If these ontologies are of interest, you can directly reuse the existing mappings.

Alternatively, if you are working with your own ontology or have identified another ontology that you wish to integrate with CEON, you can follow the alignment pipeline to generate new mappings. This process enables the creation of an ontology network that includes CEON, your ontology, and the "connections" (mappings) between them. In this case, the first step is to select a relevant ontology matching tool and use it to generate candidate mappings.

How to find other related ontologies?

There are several public repositories where you can search for relevant ontologies. Examples include *DBpedia Archivo*[16], *BioPortal*[17], *Linked Open Vocabularies*[18] *(LOV)*, and the *GitHub repository maintained by our project*[19]. These repositories allow you to search for ontologies by concept or relationship names and to download them in various formats. Such repositories provide an important starting point for identifying ontologies that can be aligned with CEON, or any other ontology.

Ontology matching tools and what they do

A variety of ontology matching tools are available to generate candidate mappings. For example, *LogMap*[20] is a well-known tool designed for scalability, i.e. working with large scale ontologies. It applies both lexical and structural matching strategies and includes unsatisfiability detection to ensure that the resulting ontology network (ontologies plus alignments) does not contain inconsistent concepts. This feature is crucial, as it helps to verify whether the integrated network is logically consistent. Another widely used tool is *MATCHA*[21], which also employs matching strategies based on lexical and structural information contained in the ontologies. You can of course also select another matching tool of your choice, and apply it using the same principles. However, take care to ensure you can use the output of the tool for further validation, and be aware that different tools have different benefits and drawbacks and no tool is perfect—see them as an assistant, but not as providing the actual truth.

Validating and merging

Once you run an ontology matching tool, it is necessary to involve human validation to check the alignment output. Then, depending on the application, you may either require ontology alignments to demonstrate semantic interoperability between two ontologies, or you may need to integrate the alignments with the ontologies themselves—for example, when creating semantic mappings to transform data into RDF. In the latter case, a merging tool is required.

There are many ways of merging. Two common approaches to ontology merging (where an alignment can also be considered an ontology) are the following: (i) use *Protégé*[22], which provides a merging function accessible directly through its user interface; (ii) use *ROBOT*[23], a widely adopted tool in the Open Biomedical Ontologies (OBO) Foundry. ROBOT can be used as a command-line tool or as a library for any language running on the Java Virtual Machine. The result of this step (in case you decide to perform the merging) is an integrated ontology that includes the input ontologies, together with the alignment.

Sharing ontology alignments

Whether you decide to merge your ontologies or not, you may also want to publish the alignment you arrived at, for others to benefit from. For instance, this could be so that your business partners can also use the same alignments, and thus the same set of interconnected ontologies as you do.

Most ontology matching tools follow a common practice of representing mappings in the *RDF format*. A mapping is typically expressed as a four-element tuple consisting of two entities, the relationship between them, and a confidence score indicating the strength of that relationship. In our project, we further explored the use of the *Simple Standard for Sharing Ontological Mappings (SSSOM)*[24], which extends this practice by supporting richer metadata for describing entity mappings. To facilitate this, we provide a program that converts mappings from RDF format into an SSSOM-compliant CSV format. The figure below shows how to represent a mapping with basic metadata supported in the RDF format and additional metadata supported by SSSOM. The following figure shows the previous mapping example in the tabular format, where the metadata contains basic RDF format-based metadata and additional metadata supported by SSSOM.

We suggest that you share your mappings together with any ontologies you have created, or want to make available, and publish them with a dereferenceable URI at your website.

A detailed tutorial on using an ontology matching tool (e.g., LogMap) to generate candidate mappings, with an optional step for converting them into the SSSOM format, is linked from our website.

RDF format-based metadata | Additional SSSOM format-based metadata

entity1	enity2	measure	relation	justification	tool	Reviewer
CEON:Product	DPPO:Product	1.0	Equivalence Mapping	LexicalMatching	LogMap	Reviewer1

Figure: **An example of an ontology alignment expressed in RDFs and SSSOM.**

Step 5—Examples for different circular strategies

Continuing our circular strategy examples, we look at how Step 5 plays out. Step 5a identifies what concepts are needed to describe data, expressed as Competency Questions. Step 5b extends existing ontologies like CEON if gaps are found, adding specific classes or properties. Step 5c aligns multiple ontologies to ensure semantic consistency across systems, enabling interoperable data sharing and federated querying in circular value chains. They feature as needed for each example.

(A) Beginning-of-life: using recycled input

What: Cross-sector recycling of apparel waste into feedstock for floor tiles.

Step 5a—Ontology Requirements & Inventory: To improve the marketplace's data infrastructure, the data steward and developer define Competency Questions (CQs): 'What is the recycled content?' and 'Are toxic substances below legal limits?' The CQs are mapped to Onto-DESIDE's CEON ontology. CEON covers broad concepts (e.g., 'Material,' 'Composition') but lacks terms for recycled footwear components and batch-level certification metadata. The developer records which CQs are supported and which require extensions, laying the groundwork to refine the ontology around real data needs.

Step 5b—Ontology Extension: To describe recycled shoe rubber accurately, the developer reused CEON and extends it by adding a subclass RecycledFootwearRubber under Material, and a property hasBatchCertificate to link materials to compliance data. The extension is then verified with the data stewards and end user representatives. This allows detailed descriptions of batches, such as "contains 22% recycled rubber, certified below legal thresholds for phthalates." These extensions ensure semantic clarity and support traceability across systems, enabling consistent data interpretation and reuse.

(B) Middle-of-life: repair

What: Repair of audio system through access to reliable spare parts & instructions.

Step 5a—Ontology Requirements & Inventory: The building owner must verify the repaired speaker's compliance and recycled content. The OEM's data steward frames competency questions (CQs)—e.g., "What are this device's components?", "How much recycled content does it contain?", "Does it meet toxic-substance thresholds?"—and maps them to CEON. CEON covers general notions (Product, Product Component, Material Composition) but lacks electronics-specific and repair-history concepts. The steward records which CQs are supported and which require extensions.

Step 5b—Ontology Alignment: To cover smart electronics devices, the developer decides to reuse the SAREF ontologies (The Smart Applications REFerence Ontology) and additionally some compliance data that uses external certification vocabularies (ontologies). The developer creates mappings between CEON and these ontologies to ensure semantic interoperability. For example, saref:Device can be aligned with CEON's Product concept, and hasMatter is aligned with equivalent relations in certification schemas. This allows data from different systems to be interpreted consistently, enabling secure and verifiable sharing of repair-related environmental data across actors without requiring changes to internal systems.

(C) End-of-life: reuse

What: Reuse and resale of a door for use in other building projects.

Step 5a—Ontology Requirements & Inventory: Before putting such data online in the Solid pods it needs to be described by a shared model. Thus the actors jointly define CQs to represent common concepts in the data and data needs of the actors: "Does this door fit opening X within ±y mm?" "Is the fire rating valid for intended use?" "What is handedness and frame/hardware compatibility?" "What is the condition and provenance?" The intermediary's developer then maps CQs and current data to shared terms from the CEON ontologies and identifies gaps. While a Product concept is present in CEON, there is no notion of a building element, and locations within a building, such as floors and rooms etc.

Step 5b & C—Ontology Extension & Alignment (as needed): The developer extends CEON with door-specific properties/classes (e.g., hasHandedness, hasTolerance, hasHingePattern, hasBackset, hasFrameType, hasRatingCredential, and hasCondition), some which can be reused from IFC, to describe item-level passports. Additionally, location parameters relevant to the door placement are reused from the Building Topology Ontology (BOT) ontology, a common ontology used in the smart buildings domain which is already used by several of the building owners.

D) End-of-life: remanufacturing

What: Take-back of the floor tiles by the manufacturer for remanufacturing.

Step 5a—Ontology Requirements & Inventory: The developer first defines CQs to represent data needs: "Which tiles in Building X are eligible for take-back (lot, binder, wear ≤Y, contamination ≤Z) by what OEM?", "What m² per eligible lot can be collected within time window W?", "What packaging/handling is required per batch?", "What is the verified chain-of-custody from removal to OEM dock?". They then map current terms and data to CEON concepts and properties and list gaps.

Step 5b—Ontology Extension (as needed): While most concepts and properties needed are already available in CEON, for the sake of understandability the developer decides to add tile-specific concepts and properties, e.g., RemanufacturableTile (subclass of Item), hasLotID, hasBinderType, hasWearGrade, hasContaminationScore, etc.

Step 6—Ensure technical interoperability: data formats

Responsible role: Developer.

In the previous step, we introduced semantic interoperability and an ontology network to describe a data model to create a common understanding of what kinds of data we want to exchange across parties (Step 5). However, to achieve true data interoperability, we need to map existing datasets onto that ontology. The next step introduces some technical best practices of how to achieve that, but in this step, Step 6, we first introduce some background technical information: interoperable data formats.

Graph data

Existing tabular or tree-based data structures inherently have limited flexibility: changing a database schema typically requires a database migration, and creating new combinations of data within a database introduces the need for joins and join tables. To make sure we can map our data in such a way that it becomes interoperable for many different parties and their use cases, we would benefit from a more flexible data model.

We therefore recommend a *graph-based datamodel*, with its primitives only consisting of nodes and edges. By only using nodes and edges, we have a *very flexible means of introducing new structures* ("it's just adding a new edge and node to the existing graph") *and introducing new links* ("it's just adding a new edge between two existing nodes in a graph").

Labeled property graphs and companies such as Neo4J use a similar graph-based datamodel. However, labeled property graphs are stored in a centralized database. The graph-based datamodel makes it easier to introduce new data and new data links in the existing (centralized) graph, however, the problem of introducing new data stores and new cross-party data links is NOT solved.

The Resource Description Framework (RDF)

We apply a different graph-based datamodel, called the Resource Description Framework (RDF). This framework allows structuring data in a graph, but further introduces *global identifiers, i.e. web identifiers, URIs*. In RDF, data is structured as *triples*: a subject (node), a predicate (edge), and an object (node), where subjects and predicates are URIs (thus, becoming globally identifiable), and each object can be either a literal (value) or a URI. Since OWL is based on RDF, the ontology described in Step 5 can also be encoded as an RDF graph, and thus allow instance data to be linked to other instance data or to ontologies.

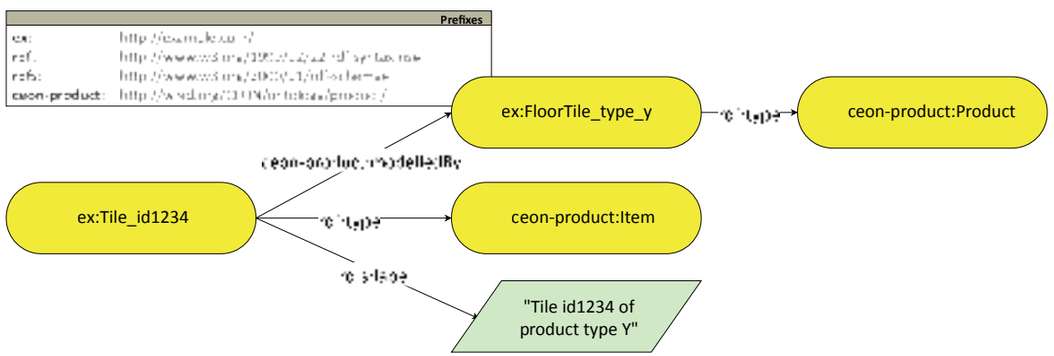


Figure: **Graph data described using RDF.**

RDF data can be represented in many different (syntactic) formats: you can represent an RDF graph in JSON using the JSON-LD specification, in XML using the RDF/XML specification, and others, such as Turtle: a dedicated format to represent RDF data. Turtle is specifically designed to be human readable and compact, so that it is relatively easy to understand an RDF file in this syntax, even when inspecting it using a text editor.

Below, we see a piece of RDF data that represents a specific type of product: a subject (URI, noted with angle brackets <URI>) links via a predicate (URI) to an object (URI).

```
<http://example.com/FloorTile_type_y>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://w3id.org/CEON/ontology/product/Product> .
```

Turtle provides a couple of shortcut mechanisms to make the data more easy to read, as seen in the next example. First, we see some prefix declarations, to make URIs shorter to write. The next triple links a tile's identifier to its type (Item) via the rdf:type ontology term), and via the rdfs:label predicate to its name (literal, noted with double quotes ""). The final triple specifies after which Product is modelled by. As all triples describe the same subject, we can use a shortcut via the semi-colon.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ceon-product: <http://w3id.org/CEON/ontology/product/>
PREFIX ex: <http://example.com/>
ex:Tile_id1234 rdf:type    ceon-product:Item;
   rdfs:label "Tile id1234 of product type Y" ;
   ceon-product:modelledBy ex:FloorTile_type_y.
```

These triples could be illustrated in the graphical notation (Grafoo) that you have seen earlier in this guide: see figure to the left.

More about RDF and knowledge graphs

RDF was developed by the World Wide Web Consortium (W3C) in the late 1990s as part of the effort to create a "semantic web" where web data can be understood and processed by machines. RDF became a W3C Recommendation in 1999, marking its official adoption as a standard for describing web resources. RDF is the backbone of linked data on the web, which encourages the interlinking of web resources to create a web of data, rather than isolated documents. What is nowadays called knowledge graphs is commonly seen as an extension of the linked data concept. RDF is one of the most common formats for knowledge graphs.

Step 7—Define data transformation: connecting data and ontologies

Responsible role: Developer. **Participants:** Data Stewards, to verify the correct mapping of the data.

Once relevant data, required to solve the user stories (Step 2), has been identified in the data inventory (Step 3) and the infrastructure to share the data has been set up (Step 4), the next step is to ensure that heterogeneous datasets, each with their own formats (e.g. databases, CSV, JSON, XML) and model (e.g. product, item, resource or material as term to express the same thing), can be represented in a uniform way that is semantically meaningful across all actors. Within the Open Circularity Platform (OCP), this is achieved by mapping company-specific data sources into RDF according to the Circular Economy Ontology Network (CEON) (Step 5), which serves as the global schema.

Data mappings

At the core of this process is the mapping component, which defines how data from a source schema is translated into RDF resources that follow the ontology. To describe these mappings, we use the RDF Mapping Language[25] (RML), an extension of the W3C standard R2RML[26]. Unlike R2RML, which is restricted to relational databases, RML supports a variety of data sources, making it well-suited for real-world scenarios where companies rely on diverse legacy systems. To make RML easier to configure, we use YARRRML[27], a human-friendly syntax that is converted into machine-readable RML rules by the YARRRML Parser.

The mapping files created by each actor specify:

- Which data is shared,
- How it is translated to RDF aligned with CEON,
- How the data is split across resources,
- How it is stored on a Solid Pod, and:
- Which access control rules apply.

A detailed tutorial explaining how to write these mapping files in YARRRML is linked from our website.

In its essence, creating and managing mapping files goes as follows. First, specify the globally unique and permanent identification schema you will apply, i.e. specify how to go from your local identifiers to URIs. A simple URI scheme could be https://[organization domain]/data/[element concept type]/[element local id]. This URI scheme includes semantics in the identifier. Such a human-readable URI makes debugging your setup-up much easier. However, this is not deemed a best practice: in this case, encoding the type of the element within your URI makes it confusing later if the type of the element changes (and global identifiers should not change). Just using generic identifiers such as UUIDs is preferred.

Second, the internal data is iterated upon. In a CSV file, this means going over every row of the file. In a relational database, this means executing a query and going over every query result. For a JSON file a JSONPath expression, e.g. $.items[*], can be defined to iterate over each item entry individually. Depending on which data source you are accessing, different query languages are applicable (e.g., SQL for a relational database, JSONPath for JSON files, XPath for XML files).

Third, for each iteration, the mapping from the relevant data fields to an RDF representation is described, e.g. the internal column ‘name’ is mapped to ontology term dcterms:title, etc.

Generating RDF data

Once a mapping file is done, this can be used to instruct a mapping engine to perform the actual data transformations. The RMLMapper is a mature mapping engine.

During execution, the RMLMapper processes the mapping rules, generates RDF data, and stores this data on the Solid Pod.

Optionally, the data can be wrapped in a Verifiable Credential[28] (VC) envelope. This additional step ensures that data consumers can verify the authenticity and integrity of shared resources using standard VC mechanisms.

Finally, also data lifecycle events must be handled. Updates to the source must trigger a re-execution of the mapping, automatically replacing outdated RDF resources. When source data is deleted, obsolete RDF resources must be removed from the Solid Pod, maintaining consistency between the source and the published data.

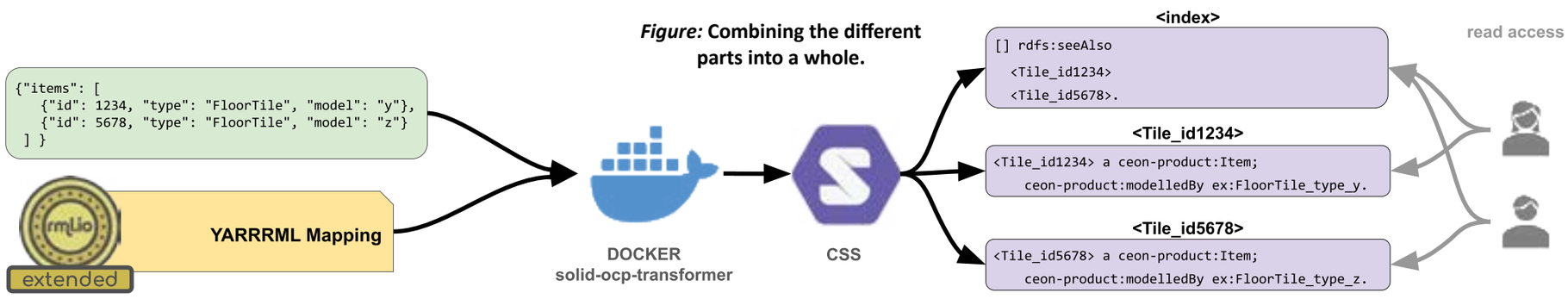
Mapping pipeline

In Onto-DESIDE, we have built a Docker image, available as *rmlio/solid-ocp-transformer* on Docker Hub, that encapsulates a complete pipeline—from source data to CEON-aligned RDF with access control and verifiable claims. This simplifies deployment and guarantees reproducibility. With one command, a system administrator can run the full process:

1. Transform heterogeneous data into RDF,
2. Split it into subsets,
3. Add Verifiable Credentials,
4. Upload resources to the Solid Pod,
5. Configure access control rules, and:
6. Delete obsolete data.

By following this transformation step, all actors in the OCP can share interoperable, trustworthy, and access-controlled data, while still maintaining full control over their own systems and sources. This approach ensures that the platform not only integrates heterogeneous datasets but also establishes a secure and verifiable foundation for data exchange within the circular economy.

Figure: **Combining the different parts into a whole.**



Step 8—Set up querying: federated querying with SPARQL

Responsible role: Developer. **Participants:** End users to verify that queries meet information needs.

Once data has been described by ontologies and expressed in a standard format, Steps 8 and 9 are about putting the data into use. First, given that your data is now in RDF form, it is time to create the SPARQL queries that are to be used for accessing the data from an application point of view.

The SPARQL query language

Now that we have all data in the standardised RDF format, integrating data becomes a matter of sending queries to all relevant data sources. SPARQL is a query language to select specific data from RDF datasets. By querying the relevant datasets via SPARQL, data can be integrated on the fly, and the right answers are generated dynamically.

A SELECT query is a typical SPARQL query to find values that satisfy conditions. The syntax can be seen as a combination of Turtle and SQL. RDF data is selected using triple patterns: a triple pattern is a triple in which each of the components can be a variable. Further processing or filtering of the selected RDF data is done using additional SPARQL modifiers, e.g. LIMIT (returns only the first n results), OPTIONAL (specifies a left join, i.e. includes extra information in your results if it exists, but it won't remove a result if that information is missing), FILTER (selects based on an expression), and ORDER BY (sorts results based on an expression).

Consider again the example of a specific floor tile, linked to its type, that was illustrated in Step 5b. The query at the bottom of this page is an example to find all items modelled by a certain type of product, in this case "floor tile type y", and to return both the item itself and its label.

The first three lines declare *prefixes*, which are shortcuts so we don't have to repeat long URIs. The next line *selects* what information the query should return, using variables which start with a question mark. In this case, we want two data elements: ?item, i.e. the item itself, and ?label, i.e. the human-readable name of that item.

The last five lines, the *where* block, describes the *pattern* the query looks for in the RDF data. In this case, each query result must satisfy the following three conditions: anything bound to the variable ?item must (i) be of the type <http://w3id.org/CEON/ontology/product/Item>, (ii) must have a label, where this label will be bound to the variable ?label, and (iii) must be modelled by the product with URI <http://example.com/FloorTile_type>.

```
PREFIX  ceon-product: <http://w3id.org/CEON/ontology/product/>
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  ex:  <http://example.com/>
SELECT  ?item ?label
WHERE
        { ?item  a ceon-product:Item;
      rdfs:label ?label;
      ceon-product:modelledBy ex:FloorTile_type_y.
        }
```


Federated querying

A *SPARQL engine executes the query,* running it against RDF data. Many established SPARQL engines–such as those built in triple stores such as Virtuoso, GraphDB, or QLever–are optimized for querying their own triple store, but cannot easily perform federated queries, especially in cases where the data sources are dynamic.

Comunica is a special kind of SPARQL engine, capable of processing such SPARQL queries over a (dynamic) set of data sources (so-called *federated querying*): the SPARQL query above could be asked to multiple resources, over multiple pods, and Comunica will be able to correctly combine the information coming for all different resources and integrate them on the fly.

Figure: **An example of setting up queries in Comunica.**

Step 6, 7 and 8—Examples for different circular strategies

Steps 6–8 turn aligned data into information that can flow across systems. Step 6 ensures technical interoperability by expressing data from different organisations in a common graph model (RDF) using serialisations like JSON-LD, so heterogeneous sources can be combined without redesigning databases. Step 7 defines mappings from internal formats (CSV, JSON, XML, databases) into this shared model, specifying what is shared, how it is represented in the ontology, and how it is stored and governed on decentralised infrastructure such as Solid pods. Step 8 implements information needs as SPARQL queries executed across multiple decentralised sources, enabling on-the-fly integration for circular decision-making. Below, we continue the circular strategy examples to show how these steps work in practice.

(A) Beginning-of-life: using recycled input

What: Cross-sector recycling of apparel waste into feedstock for floor tiles.

Step 6—Ensure Technical Interoperability: The suppliers store their data in formats like CSV and JSON, which vary in structure. To standardize this, the developer proposes to convert this data into RDF, linking each field—such as rubber content or batch ID—to the extended **CEON** concepts. This ensures that all actors interpret the data consistently, enabling smooth exchange and integration across the digital marketplace.

Step 7—Define Data Transformation: Using YARRRML mapping files, the developer describes how CSV headers and JSON keys from the suppliers' source data—such as "rubber_content" and"certificate_id"—map to **CEON** properties like MatterComposition and hasBatchCertificate. Sensitive fields, such as toxic substance levels, get restricted access. Lastly, all the data is uploaded to the Solid pods of the suppliers.

Step 8—Set Up Querying: To retrieve relevant information, the developer creates SPARQL queries, to be executed over the **OCP**. One query asks: "Which batches contain recycled rubber certified below legal thresholds for phthalates?" These queries run across all decentralized Solid pods, allowing processors and buyers to verify compliance and material suitability.

(B) Middle-of-life: repair

What: Repair of audio system through access to reliable spare parts & instructions.

Step 6—Ensure Technical Interoperability: The OEM stores product data in a relational database, incl. component IDs, repair history, and certificates. To ensure technical interoperability, the OEM enables the representation of this data in RDF, a flexible graph-based data format that can easily be shared through the **OCP**. However, the original database stays as it is, and the data needed to respond to the needs of the other actors is only transformed to RDF on a per-need basis.

Step 7—Define Data Transformation: Using YARRRML mappings, the OEM maps database attributes (e.g., "component_id", "certificate_reference") to **CEON** properties (e.g., hasProductComponent, hasCertificate). The resulting RDF is published to the OEM's Solid pod. Access control is applied to sensitive data entities, such as proprietary data, to support secure, standardised sharing of product and repair data.

Step 8—Set Up Querying: SPARQL queries retrieve key information—such as, which speaker units were repaired, level of recycled material, and certificate validity. These run across the **OCP**'s decentralised Solid pods, including the OEM's, enabling building owners and external repair partners to verify compliance and repair status.

(C) End-of-life: reuse

What: Reuse and resale of a door for use in other building projects.

Step 6—Ensure Technical Interoperability: To secure technical interoperability, RDF has been selected as the common data format, and several of the building owners can already exchange RDF data directly to/from their facility management systems. The graph-based structure of RDF, and use of globally unique identifiers, allows for seamless integration of new information types, supporting diverse building owner use cases.

Step 7—Define Data Transformation: The developer creates YARRRML mappings describing how building-owner data that isn't natively in RDF is transformed to RDF using **CEON** terms, with door-specific ontology extensions and alignments. For example, door IDs are mapped to URIs and image URLs linked via the property hasImage, defined in a **CEON** extension. The mapping also creates access-control rules for sensitive data (pricing, precise location), and the data is then published on Solid pods.

Step 8—Set Up Querying: The developer also supplies reusable SPARQL queries for buyers and planners to run over the **OCP**—for example, "select doors within ±10 mm of WxH for Opening #ID." An intermediary executes these queries across building owners' Solid pods on the **OCP**, returning a shortlist of units suitable for reuse.

D) End-of-life: remanufacturing

What: Take-back of the floor tiles by the manufacturer for remanufacturing.

Step 6—Ensure Technical Interoperability: All actors of the decentralized network—building owners, contractors, intermediaries and OEMs—decide to keep their data in their original databases but to convert this to RDF for exchange through the **OCP**, to secure technical interoperability. As they reuse persistent globally unique URIs to identify batches and product models, or even individual tiles, the data of the various actors can be automatically combined to optimize the take-back process and create traceability.

Step 7—Define Data Transformation: The developer creates YARRRML mappings describing how building-owner data that isn't natively in RDF is transformed to RDF using **CEON** terms, with door-specific ontology extensions and alignments. The mapping links door ids to URIs and image URLs are linked with the property hasImage, defined as a **CEON** extension. The mapping also creates access-control rules for sensitive data (pricing, precise location), and the data is then published on Solid pods.

Step 8—Set Up Querying: The developer also writes SPARQL queries which run over all data exposed by the set of Solid pods in the **OCP**, to answer questions such as "Find eligible tiles by lot, with area ≥ A and pickup window next 14 days".

Step 9—Develop data access applications

Responsible role: Developer. **Participants:** Decision makers to make decisions on investing in applications + End users to evaluate and guide development of user interfaces.

Although SPARQL (as presented in the previous step) is a mature standard to query data from RDF-based data storage, it is for technical purposes and application access to data. End-users of the data need to access data in a different way. In this step, we describe how applications can access the data, and exemplify this through a data viewer built in the project. This could be seen as the starting point for building your own applications on top of the data sharing infrastructure, or for connecting your existing applications to the OCP platform.

From backend to user-facing solutions

Up till now, we mostly introduced the backend solutions needed to easily share and integrate data for and from multiple stakeholders in a network. However, there are some additional steps to be taken. First, some governance framework needs to be set up, to make sure that the shared data is discoverable by the relevant partners in the network, e.g. managing the configuration of the value network, and its data sources. Second, data sharing contracts should be set so that access to the right data is agreed upon. Third, user-facing applications are needed to make sure that end users experience the benefits of easily integrated data from these backend solutions in an intuïtively understandable way.

Governance

First, concerning governance, existing initiatives such as dataspaces can be applied. These initiatives introduce the concept of a data catalogue: a registry where all data that may be discovered by the network is described. Note that this catalogue does not contain the actual data, but only the metadata. Within this step, we propose a source index, managed by a network administrator. Such a source index contains pointers to the location of the data of all included actors within the network, as a starting point for queries. Every change, e.g. every time a resource is added or removed, should be reflected in the source index. For the Onto-DESIDE demonstrators, we have maintained domain-specific source indexes, i.e. containing the URIs of data sources from resp. for the construction, electronics and textile domain, and a cross-domain index, covering these three domains.

Access control

Second, concerning access control, the Solid specifications provide a way to set access control rules on specific user accounts, or user account groups, thus providing role-based access control (RBAC). Granting or denying read or write access can be made simple via dedicated user-facing applications. We provided LOAMA as a demonstrator of how such an access management application could look like. A link to this application is available on the project website. Considering the resale of doors, a building owner can use LOAMA to grant read access to the resources with commercially sensitive data such as pricing with a specific buyer, without disclosing this information to other interested parties.

Frontend applications

Third, end users experience the benefits of these backend solutions through frontend applications, which can present the integrated data (i.e. the query results) to the end user in an intuïtively understandable way. Within the next section, we introduce such a demonstrator frontend application.

Using Miravi

Miravi is an exemplary application on top of the technically and semanticly interoperable open and access controlled data. Applications like Miravi showcase the potential of combined data to enable resource flows and accelerate circular economy.

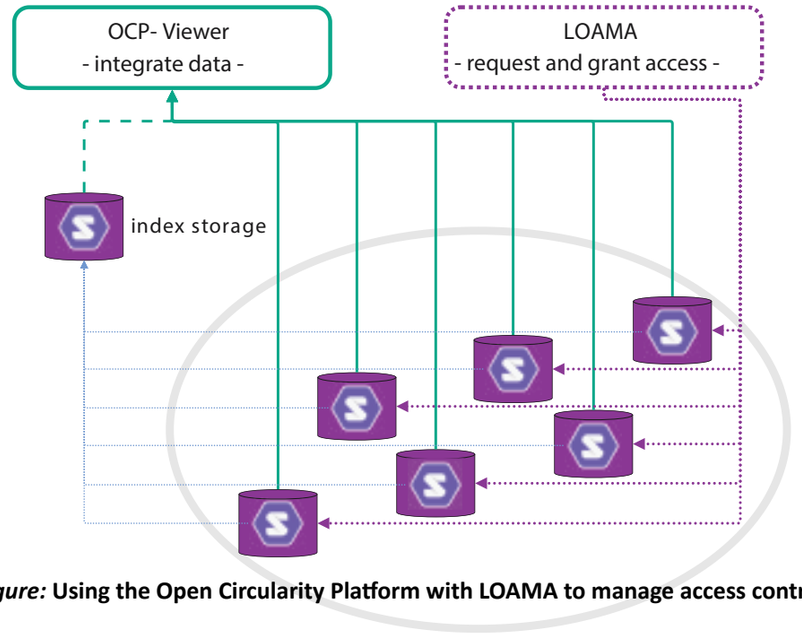


Figure: **Using the Open Circularity Platform with LOAMA to manage access control.**

Miravi allows you to configure a data dashboard customized to a specific use case, integrating data from decentralized open and access controlled data sources, including but not limited to Solid pods. During usage, an end user browses the user-friendly web UI which provides access to the Solid-based decentralized data-sharing platform via predefined SPARQL queries. The predefined queries can start from an index to traverse an evolving set of query sources and can make use of variables to be filled in by the end user for more flexible querying. Miravi also allows the end user to export the query results, and to create, save, and load custom queries. Miravi can be easily set up, independent of the networks the Solid pods reside in, following the instructions of the extensive readme of Miravi's GitHub repository.

As a demonstrator for the Onto-DESIDE project an Open Circularity Platform Viewer was configured using Miravi. The Viewer provides a set of predefined queries the user can execute over the Solid pods with example data from the three domains under focus in the project, i.e. the textile, electronics, and construction domain. The predefined queries provide the needed data to resolve the domain specific and cross-domain use cases defined to evaluate the project, e.g., for the repair of an audio system, a building owner can discover repair instructions and track the digital product passport information of the original and repaired product in OCP Viewer.

Step 10—Plan for maintenance and evolution

Responsible role: Developer/ End user. **Participants:** Any other role that may detect changes/ changing needs.

Finally, this step describes various scenarios for maintenance and evolution of the data sharing infrastructure, such as extending the value chain configuration with new actors, including new data sources, adapting to changes in models, data and formats, as well as transforming data into new structural patterns, to adopt new or alternative ontologies. This step does not cover all possible scenarios, but gives a hint as to what will be necessary to keep the infrastructure up to date.

After concluding the previous steps, you have a robust set of technologies to create more interoperable data exchange in the Circular Economy domain. Every subsequent change is incremental. Depending on where the change happens, subsequent steps in this guide also must be reviewed. In the following paragraphs we discuss how changes to the context, requirements or certain artefacts will affect what was done in other steps.

Changes in the Circular Value Chain—Step 1
If changes appear in the context where the data sharing infrastructure is used, i.e. the value chain collaboration, then the mapping of flows may need to be updated and the subsequent steps may have to be revisited. For instance, it could involve adding new possible resource paths, involve new types of actors in the collaboration, or update the information needs and barriers.

Changes in requirements—Step 2
When a new user need is identified, this must be added to the list of requirements. Or if a change of requirements is identified based on, for instance, a change in the value chain context then the solutions and decisions made in subsequent steps will have to be revisited. Perhaps some solutions are no longer relevant.

Updating the data inventory—Step 3
When the requirements are updated, the data inventory needs to be reviewed to identify whether you can cover this requirement with your existing data inventory or you need to look for additional data sources (or data collection opportunities). On the other hand, when the set of data sources changes (e.g. a new data source is identified, or an existing data source is updated), the data inventory must be reviewed and updated to reflect that. Such a change may in some cases be an opportunity, but may in some cases instead lead to that a data source no longer matches the information need it was previously used to fulfill.

New previously unseen data may result in needs to update ontologies and queries, as well as applications, while removal of data element usually does not affect the models in the same way. However, queries can of course no longer ask for data that is not available.

Changes in data sharing setup—Step 4
When a new actor enters the network, a new data sharing web service must be set up. However, this should not affect the other actors in the network, regarding their data sharing setup.

Changes in ontologies—Step 5
As mentioned previously, ontologies may need to be updated, based on new information needs, technical requirements, new or changing data sources. However, changes could also be initiated by an update to an ontology the network relies on, potentially beyond the control of the value chanin actors. Relying on standards and frequently used ontologies is a good practice, since this increases interoperability also across value chain collaborations, but it also means that some ontologies may be outside the control of the involved actors. Updates to such ontologies may then result in the need for updating the data transformations, queries and potentially slightly modify applications using the ontologies and data.

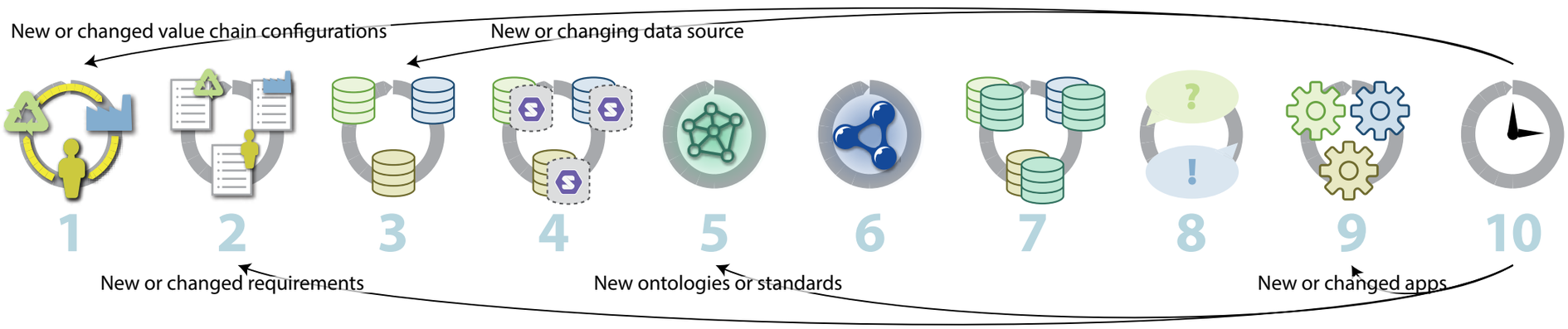
Changes in data formats—Step 6
The standards we recommend are mature and stable, and no direct impact onto or from this step is expected. In fact, one of the benefits of applying our approach is that the data formats are generic, and not specifically tailored to any industry domain or use case, which makes them particularly robust and unlikely to change.

Changes in data transformation—Step 7
Changes in the mappings of data to the ontologies will not appear by themselves, but are always consequences of other changes. For instance, when the data inventory is updated or ontologies change, the data transformations must be reviewed to identify if there are user needs that are no longer met, if new user needs can be met, and assess the need of changes in the mappings or whether it is merely enough to update the mappings to a new ontology vocabulary.

Changes in queries—Step 8
Similarly to step 6 the standards we work with are mature and stable, and no direct impact onto or from this step are expected. While still consequences of other changes may have to be reflected in the queries, e.g. using new vocabulary in the queries if the ontologies are changed, or querying over new data sources if such become available.

Changes in applications—Step 9
The use of standard languages, and by separating the models and queries from the applications, make applications built on top of this kind of

Figure: **Maintenance and evolution means to go back to previous steps in the process.**



infrastructure surprisingly robust. While changes in data availability and requirements may certainly result in new or modified data visualisations, all the formats and data access components remain the same. Hence, no major changes in applications will most likely be required based on changes in any of the previous steps.

Below, we give an overview of how the steps are dependent on each other. Note how Steps 4, 6, and 9 are deemed to be independent of changes happening in the other steps.

Change in scenario A—Adding a new actor to the value network

In the context of our scenario 1, using recycling material, as time passes new recycling methods will appear, and new actors become interested in the recycled feedstock. In such a marketplace scenario, there will constantly be new suppliers of recycled material, and new actors interested in purchasing batches of materials, hence the concrete set of actors will change, and new material flows (step 1) will be added (and others removed).

In this case, the business case is still selling and buying recycled materials as feedstock for new products, hence, requirements (step 2) of the value network remain more or less the same. While the data inventory (step 3) is constantly changing, including data of new suppliers, and removing actors that no longer provide any materials for the marketplace. Each new actor that desires to participate in the material, and data, exchange needs to set up their data sharing service (step 4), e.g. contract a Solid pod service.

In case the new flows require new data points about the recycled materials, the ontologies may need to be specialised (step 5), however, given that the requirements remain more or less the same no extensions to the core ontologies would be needed. Next, the data source of the new actors need to be mapped to these ontologies (step 7), but if the new data is described by a specialisation of the previous ontologies, then it may even be possible to use the previous queries unchanged, simply adding new sources to the source index used as starting point for the query. And thus the marketplace application, if built based on the core ontologies, should remain more or less unchanged.

Change in scenario B—Adding a new data source to the infrastructure

In the context of our scenario 2, reuse of building elements, we may consider the addition of a BIM system, providing a digital twin view of the building, by the building owner. Such a system provides a new source of information about the used building elements, i.e. also including usage and repair data about the elements, in addition to the manufacturing data, product retail data etc.

Adding such a new data source does not change the flows, or requirements, nor the way the building owner will share data with others. However, the data inventory (step 3) is updated, by adding the new data source. Then this new data source is compared with the current ontologies, to see if they already cover all the needed data points. In this case, it may be the case that extensions are needed, if usage data for instance was not previously covered by the ontologies. Thus, an ontology extension is needed, where external ontology models for BIM can be reused, through adding an alignment (step 5c) to the CEON core modules.

Subsequently, a mapping of the data source (step 7) is needed, to these new ontology extensions, and formulation of new queries (step 8) that include these data, to complement previous reuse queries. If the applications (step 9) for mediating and marketing reused building elements are not able to take into account usage data aspects in their presentation of the available elements, they may need to be updated. For instance, consider the case of adding a way to search and filter used doors based on the type of usage of the rooms they have been mounted in, or based on the usage frequency.

Changes across scenarios—Modifying an ontology

We have previously described the flexibility of the graph data model in adapting to different situations that may occur in the real world (especially, in a complex domain such as that of CE value chains) that is to be modelled. Yet, even if ontological data schemas are less rigorous than most other schema types, sometimes we need to reshape the data in such a way that even the ontology itself needs to be modified. This may hold even though the world itself has not changed. The motivation for such a change would typically be the need for greater conciseness of the data graph, but sometimes also, a bit opposite, putting some (previously tiny or implicit) parts of the graph into the spotlight. The ontology should then undergo what we label as *transformation*.

A simple example of ontology transformation is de-reification. Remember the material content modeling problem: in the reification style of modeling, CEON contains a MatterComposition class, connected with relations possibly called “item”, “matter”, “value” and “unit” . Then we find out that reaching from the item (such as “tile”) to the matter (such as “calcium sulfate”) is cumbersome for some applications, because of the intermediate hops in the graph. For example: a data diagram in a visualization tool would be too cluttered, or some large-scale data analytics tasks would have to search in an unnecessarily large space.

An ontology transformation tool, such as PatOMat2 developed in the Onto-DESIDE project (see further information on our website), can then help us identify the possible “shortcut” (leading directly from the item to the matter), and introduce it into the ontology. Moreover, it can suggest, through a call to a pre-trained language model, a possible label for the new “shortcut” relation, for example, “contains matter”.

Step 9 and 10—Examples for different circular strategies

Steps 9 and 10 bring the infrastructure to end users and keep it adaptable. Step 9 develops data access applications—dashboards, portals or extensions to existing tools—that run SPARQL queries over decentralised sources, presenting integrated, access-controlled data in task-oriented views without exposing technical complexity. Step 10 plans maintenance and evolution: governance for catalogues and roles, management of data-sharing agreements, and monitoring of changing needs, sources, standards or ontologies. Because data formats, ontologies, mappings and queries are separated from applications, most updates can be made by adjusting these layers rather than rebuilding systems, keeping the setup robust over time.

(A) Beginning-of-life: using recycled input

What: Cross-sector recycling of apparel waste into feedstock for floor tiles.

Step 9–Develop Data Access Applications: The developers of the materials marketplace edit their dashboard to browse available batches of recycled rubber based on the new terms of the extended **CEON** vocabulary. This app connects to the **OCP**, consisting of the published Solid pods, and runs the predefined SPARQL queries to retrieve batch composition, recycled content, and certification status. Sensitive data, such as exact toxic substance levels, is only shown to specific authorized users. The interface allows filtering by material type and compliance status, supporting informed procurement decisions and enabling traceable, circular sourcing.

Step 10–Plan Maintenance and Evolution: As new suppliers join the marketplace or regulations change, the data steward and the developers update the data inventory, ontology, YARRRML mapping files and SPARQL queries. As the system was designed to accommodate new actors and data sources this does not disrupt existing workflows, and long-term adaptability and compliance in the circular value chain is ensured.

(B) Middle-of-life: repair

What: Repair of audio system through access to reliable spare parts & instructions.

Step 9–Develop Data Access Applications: A building owner can now use a repair portal to check the options for repair or replacement of the speaker unit. The app connects to the Solid pods of the **OCP** and runs SPARQL queries to display repair and replacement options, including details on recycled content, and compliance certificates. Sensitive data is restricted to authorized users. The interface allows verification of environmental compliance, supports informed decision-making, and ensures traceability of repaired components.

Step 10–Plan Maintenance and Evolution: If the OEM updates its repair process or introduces new materials into their spare parts, the ontology can be extended to reflect these changes. New data mappings are then created, and SPARQL queries are adjusted to include updated terms. Thus, such new data can then be displayed by apps built on top of the infrastructure, without substantial changes to their implementation. The system supports ongoing evolution, allowing new actors, data sources, and regulatory requirements to be integrated without disrupting existing workflows.

(C) End-of-life: reuse

What: Reuse and resale of a door for use in other building projects.

Step 9–Develop Data Access Applications: The intermediary hosting a marketplace application, for sale of reused construction elements, extends their application from manual data input to using SPARQL queries over the **OCP** to retrieve data from door passports shared via decentralised Solid pods. This data is processed to present items that match the needs of the logged in users. Interested buyers can request read access to sensitive data such as price or exact location. This information is rendered only after this access is granted to the logged in user.

Step 10–Plan Maintenance and Evolution: When new fire ratings, new component attributes or new certification types appear, data inventories, ontologies, mappings and queries are updated. As the graph-based structure of RDF allows for seamless integration of new properties into the datasets, without the need to adapt all other datasets, the data on the decentralized Solid pods remains backwards compatible and reuse transactions and queries continue reliably.

D) End-of-life: remanufacturing

What: Take-back of the floor tiles by the manufacturer for remanufacturing.

Step 9–Develop Data Access Applications: An intermediary builds a digital portal to support take-back: owners view eligibility status by zone; contractors see removal/packaging instructions; OEMs pre-book pickups, generate manifests, and issue compensation offers; logistics receives route plans. The app runs predefined SPARQL queries to retrieve the data needed in the portal. Sensitive pricing information and location data appears only when an authorized party logs in.

Step 10–Plan Maintenance and Evolution: As adhesives, testing methods, or take-back programs evolve, the data stewards and developers update the data inventories, ontology modules, mappings, and queries. When new building owners enter the network, OEMs grant them read access to selected parts of their data, and the intermediary expands the sources of the SPARQL queries feeding the digital portal with the data of the new building owners, ensuring OEMs can reliably retrieve end-of-life tiles as future feedstock.

Closing words

To scale the Circular Economy, we need to scale the information flows that underlie a well-functioning value network. Luckily most IT infrastructure and standards we need are already in place, in some cases since many years - it is just about using them in a new way, and developing the necessary models and tools that guide the users specifically when developing CE solutions.

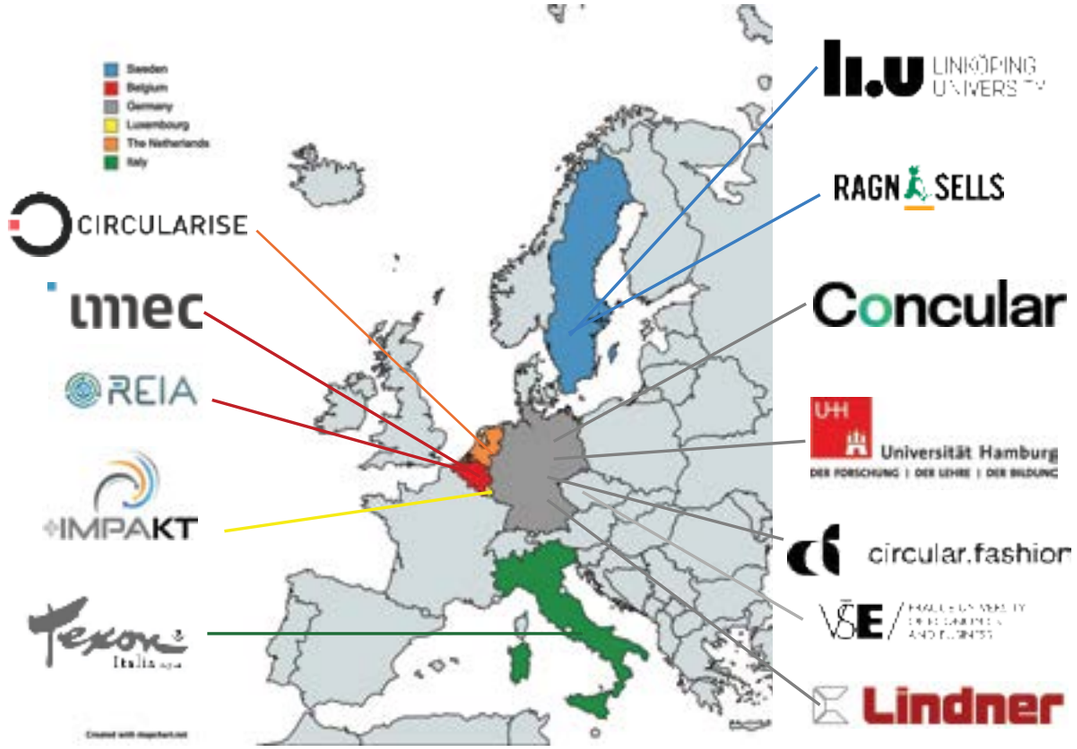
It is most likely the case that you as an organisation already have most of the basic prerequisites in place. For instance most organisations have a web server, either as part of their in-house IT infrastructure or as a hosted service, with security and access control as part of its basic setup. URI lookup and linking are also things already supported by our web application and browsers. Hence, the step is not too far, to extend this to linking also our data!

The challenge is to accept and embrace diversity instead of attempting to create (yet another) standard or data template to fit all scenarios. And to avoid lock-in through commercial APIs and formats, but rather focus on open standards and shared agreements. Technologies, such as ontologies, can be used to manage this diversity, create data descriptions and mappings, and allow for navigating and making use of the diverse data landscape.

On behalf of all Onto-DESIDE contributors,

Prof Dr Eva Blomqvist
Project coordinator

ONTO-DESIDE



References

1 For example, resource extraction has already more than tripled since 1970 and is projected to rise another 60 % by 2060, accounting for over 60 % of global greenhouse gas emissions and 40 % of pollution-linked health impacts (UNEP, 2024). Such scale places enormous pressure on ecosystems and communities.
2 Raworth, Kate. Doughnut Economics: Seven Ways to Think like a 21st-Century Economist. Random House Business Books, 2017.
3 Circle Economy. Circularity Gap Report 2025. 2025. https://www.circularity-gap.world/2025.
4 European Commission. Study on the Critical Raw Materials for the EU 2023: Final Report. Directorate General for Internal Market, Industry, Entrepreneurship and SMEs. Publications Office, 2023. https://data.europa.eu/doi/10.2873/725585.
5 European Commission. Circular Economy Action Plan: For a Cleaner and More Competitive Europe. Directorate General for Communication. Publications Office, 2020. https://data.europa.eu/doi/10.2779/05068.
6 Circle Economy. Circularity Gap Report Finance. 2025. https://finance.circularity-gap.world/.
7 https://github.com/CommunitySolidServer/CommunitySolidServer
8 https://www.w3.org/TR/2012/REC-owl2-overview-20121211/
9 https://www.w3.org/TR/shacl/
10 https://www.w3.org/TR/rdf11-concepts/
11 Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., & García-Castro, R. (2022). LOT: An industrial oriented ontology engineering framework. Engineering Applications of Artificial Intelligence, 111, 104755. https://doi.org/10.1016/j.engappai.2022.104755
12 Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., & García-Castro, R. (2022). LOT: An industrial oriented ontology engineering framework. Engineering Applications of Artificial Intelligence, 111, 104755. https://doi.org/10.1016/j.engappai.2022.104755
13 Blomqvist, E., Hammar, K., & Presutti, V. (2016). Engineering ontologies with patterns–the eXtreme design methodology. In Ontology Engineering with Ontology Design Patterns (pp. 23-50). IOS Press
14 Noy, N., & McGuinness, D. L. (2001). Ontology development 101. Knowledge Systems Laboratory, Stanford University.
15 Euzenat, Jérôme, and Pavel Shvaiko. Ontology matching. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007
16 https://www.dbpedia.org/resources/archivo/
17 https://bioportal.bioontology.org/
18 https://lov.linkeddata.es/dataset/lov/
19 https://github.com/LiUSemWeb/Circular-Economy-Ontology-Catalogue
20 https://github.com/ernestojimenezruiz/logmap-matcher
21 https://github.com/liseda-lab/Matcha-DL
22 https://protege.stanford.edu/
23 https://robot.obolibrary.org/
24 https://github.com/mapping-commons/sssom
25 http://w3id.org/rml/portal/
26 https://www.w3.org/TR/r2rml/
27 https://www.w3.org/TR/vc-overview/
28 https://rml.io/yarrrml/spec/

www.ontodeside.eu