Horizon 2020 Program (2014-2020)

A Framework for Pairing Circular Economy and IoT: IoT as an enabler of the Circular Economy & circularity-by-design as an enabler for IoT

D2.1: LCA and CSPDI patterns †

Abstract: LCA & CSPDI patterns – The overall aim of this deliverable is to develop a pattern specification language to specify the circular economy design Location, Condition and Availability (LCA) patterns and the IoT architectural Connectivity, Security, Privacy, Dependability and semantic Interoperability (CSPDI) patterns. It focuses on the identification and specification of concrete LCA and CSPDI machine interpretable patterns.

Contractual Date of Delivery	29/02/2020
Actual Date of Delivery	21/02/2020
Deliverable Security Class	Public
Editor	*Eftychia Lakka, FORTH*
Contributors	Ecole Nationale Des Ponts et Chaussees (ENPC) University of Cambridge (UCAM) Foundation for Research and Technology Hellas (FORTH) Deloitte Consulting & Advisory (DEL)
Quality Assurance	*Nikolaos Petroulakis, FORTH*

† *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation staff exchange programme (RISE) under the Marie Skłodowska-Curie grant agreement No 777855.*

CE-IoT Consortium

ECOLE NATIONALE DES PONTS ET CHAUSSEES (ENPC)	Coordinator	France
THE CHANCELLOR, MASTERS AND SCHOLARS OF THE UNIVERSITY OF CAMBRIDGE (UCAM)	Principal Contractor	United Kingdom
FOUNDATION FOR RESEARCH AND TECHNOLOGY HELLAS (FORTH)	Principal Contractor	Greece
CABLENET COMMUNICATION SYSTEMS LTD (CBN)	Principal Contractor	Cyprus
BLUESOFT SPOLKA Z OGRANICZONA ODPOWIEDZIALNOSCIA (BLS)	Principal Contractor	Poland
DELOITTE CONSULTING & ADVISORY (DEL)	Principal Contractor	Belgium

Document Revisions & Quality Assurance

Internal Reviewers

1. *George Alexandris, ENPC*
2. *Nikolaos Petroulakis, FORTH*
3. *Eftychia Lakka, FORTH*

Revisions

Version	Date	By	Overview
0.0	16/10/2019	Eftychia Lakka (FORTH)	Table of Contents
0.1	28/11/2019	Eftychia Lakka (FORTH)	Partner Contribution Assignment
0.2	19/12/2019	Ellie Wolmark (UCAM)	Section 2
0.3	2/1/2020	George Alexandris (ENPC)	Section 4
0.4	14/02/2020	Efthychia Lakka (FORTH)	Section 3, 5, 6, 7
0.5	16/02/2020	Nikolaos Petroulakis (FORTH)	Quality Assurance
0.6	19/02/2020	Eftychia Lakka (FORTH), George Alexandris (ENPC)	Internal Reviews
0.7	20/02/2020	Nikolaos Petroulakis (FORTH)	Finalization
1.0	21/02/2020	George Demetriou (ENPC)	Submission

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CSPDI Connectivity, Security, Privacy, Dependability and semantic Interoperability

DoW Description of Work

E2E End-to-End

EBNF Extended Backus-Naur Form

EC European Commission

EMF Eclipse Modelling Framework

EU European Union

GDPR General Data Protection Regulation

IF Information Flow

IoT Internet of Things

IPR Intellectual Property

KB Rights Knowledge Base

LCA Location, Condition and Availability

M Month

PSP Perfect Security Property

WP Work Package

UML Unified Modeling Language

✔ Status: Completed

✔ Status: Started but not completed

Executive Summary

This document is the first deliverable (D2.1) of WP2 on Patterns, circular economy business models & supply chains. The overall objective of this document is to provide the pattern specification language to specify the circular economy design LCA patterns and the IoT architectural CSPDI patterns. It focuses on the identification and specification of concrete LCA and CSPDI machine interpretable patterns. The development of this pattern language is based on [1], [2], [3] focusing on extensions that are necessary in order to support both LCA properties of intelligent assets and CSPDI properties for the different types of smart objects, network and software services and components that may exist in an IoT application compositional structures (e.g., different interaction service and component mechanisms that are required for IoT applications including message-based, event-based and data-driven interaction mechanisms interconnecting such components [4]), and the LCA and CSPDI properties identified in CE-IoT demonstrators. To achieve the above objective the following research is required and provide in the deliverable:

- Definition of the specific domains of use case scenarios in which LCA and CSPDI patterns will be applicable.
- LCA and CSPDI property requirements used to define the pattern language.
- High-level analysis of the domain use cases with respect to the fundamental circular LCA and CSPDI properties.
- Circular Economy design properties based on the described LCA and CSPDI properties.
- Pattern Language definition and semantics
- A first set of LCA-CSPDI patterns

1 Introduction

1.1 Background

Circular economy has attracted attention in recent years. It is characterised as an economy that is restorative and regenerative by design, and which aims to keep products, components and materials at their highest utility and value at all times. It is conceived as a continuous positive development cycle, reforming the current economy model of 'take-make-dispose', by preserving and enhancing natural capital, optimising resource yields and minimising system risks by managing efficiently finite stocks and renewable flows.

Circularity involves lengthening and rethinking the use and usability of assets, increasing the utilisation of resources, creating additional use cycles, rethinking supply and value chains and regenerating natural resources. IoT, on the other hand, provides valuable insights and usage information in terms of asset key enabling properties of the assets such as Location, Condition and Availability (LCA) thus transforming the physical assets to intelligent ones. In addition, the preservation of end-to-end properties of IoT applications including the Connectivity, Security, Privacy, Dependability and semantic Interoperability (CSPDI) are also required to enable circularity in IoT.

1.2 Scope and Objectives

CE-IoT aims to develop a circular-by-design, dynamically configurable, adaptive and evolutive IoT architecture to enable secure and dependable integration, runtime management and adoption of "smart" IoT objects (e.g., sensors, devices, systems, and components) into heterogeneous IoT platforms, seamlessly interconnected over virtualized and programmable software-defined networks.

The achievement of this objective is based on developing *circular economy design patterns* defining broad ways for integrating and orchestrating different types of intelligent assets that can guarantee specific LCA properties of the intelligent assets, henceforth referred to as LCA patterns. The verified patterns can support the composition of intelligent assets in ways that preserve LCA properties. The development of LCA patterns is based on assurance regarding the preservation of the location, condition and availability properties targeted by a pattern. This assurance can be based on test evidence, monitoring evidence or formal verification as it is appropriate for the type of properties and intelligent assets integrated by the pattern.

In addition, CE-IoT architecture can be based on an open, modular, *circular-by-design* approach where the networks, smart objects, and IoT platforms are integrated through *IoT architectural patterns* with proven capability to preserve end-to-end CSPDI of IoT applications. CSPDI patterns should cover both vertical composition of smart objects at different layers in the implementation stack of IoT applications – including sensors/actuators; network, infrastructure, IoT platform and IoT application components – and the horizontal composition of smart objects that appear at any of these layers, as necessary. CSPDI patterns should also cover the different and heterogeneous coordination and interaction models required for such applications, including *message-driven*, *event-driven* and *data-driven* models. To ensure its long-term applicability, the *CE-IoT* architecture will be extensible-by-design using an open approach that facilitates the integration of new smart objects, IoT mechanisms, and CSPDI patterns, in order to use them in IoT applications. It will also offer self-evolving capabilities through the analysis of operational evidence arising from the application of CSPDI patterns in different contexts and multiple application sectors, thus enhancing circular economy in the ICT industry.

Finally, the development of LCA and CSPDI patterns require the definition of a pattern language supporting the specification of all facets of patterns. The model to be developed will demonstrate how the interplay of value drivers and systems thinking from both the circular economy and IoT.

1.3 Relations to Other Deliverables

This document is the first deliverable (D2.1) of WP2 on patterns, circular economy business models & supply chains. It is the first technical deliverable introducing the LCA-CSPDI patterns that will be exploit in most of the technical deliverable of CE-IoT. The most related deliverables and the timeline regarding the duration of the project are presented below and in Figure 1.

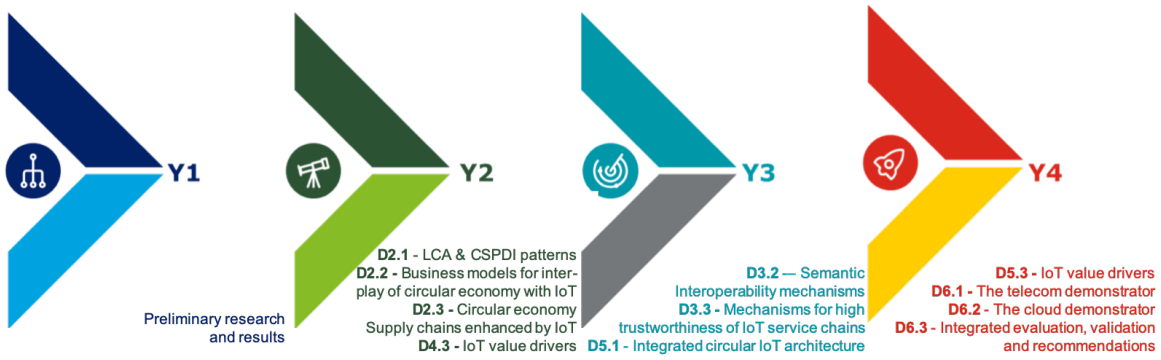

Figure 1. Deliverables of WP2 Patterns, Circular Economy Business Models & Supply Chain

D2.1 – LCA and CSPDI patterns (M20): This deliverable will define and describe the syntax and semantics of the pattern specification language and the semantic annotations for LCA and CSDPI patterns.

D2.2 – Business models for interplay of circular economy with IoT (M22): This report will document the IoT enabled circular business models in terms of design, development of product-service roadmaps and respective digital strategies.

D2.3 – Circular economy supply chains enhanced by IoT (M24): This report will document Reverse Logistics in Circular Economy and IoT, methods for maximisation of the 'retained values' of servitised products and recovery strategies management mechanisms.

D4.3 – IoT value drivers (M24)**:** This report will provide a comprehensive study of IoT technical value drivers and their potential for value creation.

D3.2 – Semantic interoperability mechanisms (M27)**:** This deliverable will define the semantic annotations for architectural patterns; it will also provide a collection of data transformation techniques and semantic interoperability mechanisms.

D4.3 – IoT value drivers (M24)**:** This report will provide a comprehensive study of IoT technical value drivers and their potential for value creation.

D3.3 – Mechanisms for high trustworthiness of IoT service chains (M30)**:** This report will document the Security, Privacy and Dependability requirements of IoT as well as a pattern-based approach to guarantee Security, Privacy and Dependability properties across horizontal and vertical compositional structures of IoT applications.

D5.1 – Integrated circular IoT architecture (M32): This deliverable will be the first integrated working prototype of the *CE-IoT* technical architecture. It will also provide a description of this prototype; the tests performed on it, the formal analysis carried for it, and a set of basic usage instructions for it.

D6.1 – The telecom demonstrator (M42)**:** This deliverable will include the business and technical results of the demonstration of *CE-IoT* in the telecom sector (ISP).

D6.2 – The cloud demonstrator (M42)**:** This deliverable will include the business and the technical results of the demonstration of *CE-IoT* in the cloud services sector.

D6.3 – Integrated evaluation, validation and recommendations (M48)**:** This deliverable will define the details of the evaluation methodology of *CE-IoT* and an analysis of the results from D6.1 and D6.2, presenting a cross-evaluation of the *CE-IoT* approach across the different demonstrators and general final recommendation regarding the use of the *CE-IoT* approach.

1.4 Target Audience

The target audience of this document is the same of the CE-IoT activities and results. In particular, it involves the following stakeholders:

1. **Business and policy makers**, who may have interests in understanding the main concepts and implications of CE-IoT
2. **IoT end-user groups**, who may have interests in understanding the main concepts and implications of CE-IoT
3. **Research and innovation groups**, who may have interests in advancing the technical knowledge associated with CE-IoT
4. **General public**, who may have interests in understanding the main concepts and approach of CE-IoT and the implications for the sustainability of these systems and the value they bring to the economy.

1.5 Structure of the Document

The structure of the document consists of the following sections:

- **Section 2 Domain use Cases**, which describes the specific domains of use case scenarios in which LCA and CSPDI patterns will be applicable.
- **Section 3 LCA and CSPDI Property requirements,** which defines the property requirements that will used to define the pattern language.
- **Section 4 Domain Specific Requirements**, which provides a high-level analysis of the domain use cases with respect to the fundamental circular LCA and CSPDI properties.
- **Section 5 Circular Economy Design Properties**, which describes the properties for Circular Economy design based on the described LCA and CSPDI properties.
- **Section 6 Pattern Language Definition**, which constructs the pattern language for expressing/encoding dependencies between LCA-CSPDI properties at the component and at the composition/orchestration level.
- **Section 7 Pattern Rules,** which presents a first set of LCA-CSPDI patterns able to guarantee confidentiality, privacy, dependability and interoperability.
- **Section 8 Conclusions,** which provides an overview of the main concluding remarks drawn from key contributions of the document
- **References**, which provide a list of relevant references cited across the document.

2 Domain Use Cases

This section aims to describe the specific domains of use case scenarios in which LCA and CSPDI patterns will be applicable in order to improve the benefits associated with the circular economy by integrating IoT.

2.1 Exploring the capacity for IoT to allow for a system which enables electricity load control and so promotes Demand Side Response in Poland's energy sector.

In the era where resource scarcity is becoming reality, the call for replacing linear economy principles of make-take-dispose with circular economy principles is becoming ever more urgent. Poland's energy sector is one where circular economy principles may have a substantial impact. Volatility in both power generation and electricity demand engenders inefficiencies in the form of power generation asset utilisation, and costly electricity system continuous adjustments and balancing. As power supply must currently follow power demand, the power generator's assets are highly utilised in peak hours and less so in non-peak hours. Furthermore, the volatile nature of consumers' electricity usage makes it near-impossible for power providers to supply exactly the right amount of electricity to contract with power generators. Since the electricity system must be kept in balance at any given point in time, system adjustment and balancing is very expensive, costing the sector 12 billion PLN annually.

Demand Side Response (DSR) is a principle and model where power demand follows power supply instead of the other way around. The advent and development of Internet of Things (IoT) has enabled the automatic implementation of DSR in Poland's energy sector. With IoT, power providers and transmission infrastructure owners can create smart meters and smart grids which measures real-time electricity usage of consumers and automates electricity usage of the users by switching it off when there's a surge of demand and lack of supply. Along with monetary incentives, a smart protocol called IoT Load Control will allow a smart allocation of electricity load, minimising discomfort for users who relinquish full control of their electricity usage to DSR providers.

BlueSoft, one of Poland's top IT solutions and software providers, is well positioned to offer the IT architecture of such a solution to Poland's energy sector. With 15 years of experience in the IT sector, strong relationships with electricity utility companies, and its business capabilities, BlueSoft is primed to offer such a model and reap the first-mover advantage of proposing the implementation of DSR. Aside from providing new revenue streams for energy providers and replacing the expensive go-to-market operation for electricity system balancing, we also believe that DSR promotes circularity principles by curtailing excessive energy demand, promotes energy savings, and better asset utilisation over the long-term.

2.2 Case Study – Bluesoft as a Loyalty Platform Provider

Research Achievements:

The development of product-service roadmaps is explored by analysing organisational potentials in incorporating the concept of circular economy with IoT.

The benefits that circular economy can provide to organisation are explored by simulating financial statements scenarios and investment-return output.

Company Request:

The circular economy is a new concept for the company. Despite all the benefits attributed to circularity, Bluesoft finds it difficult to educate the sales team to sell this concept to the clients. Even within the company, the management could not see the benefits in going circular, hence the management is not motivated to promote it within Bluesoft. Therefore, Bluesoft would like to know how to introduce circularity to their clients and their own teams, as well as how and IT consultancy can fit in the context.

Bluesoft has a broad arrange of customers but as an IT consultancy, they don't have a lot of experience in manufacturing. They would like to know how if there is a chance to move towards this sector.

Solutions:

To start with, a case scenario of an EU fridge manufacturer is set up for further studies. The client requests a loyalty platform, which is one of the most common cases that Bluesoft has been working on daily. In this case, a three-step framework is suggested to Bluesoft.

2.3 Case Study – Focus on Smart Insurance Contracts

The purpose of this research is to explore IoT-enabled circular business models. IoT is used to describe the inter-connectivity of various devices through the Cloud. In other words, IoT is the ecosystem where a person's Smart Phone can be connected to his Smart Home and so on. Important to note is the use of sensors in the IoT ecosystem. Sensors are important because they can be used for extensive data analytics and predictive analytics. This in turn has financial benefits such as cost reductions, processing time decreases, increased efficiency and improved safety.

"Circularity" and "circular economics" is an environmentally sustainable and cost-friendly way of doing business compared to traditional linear economics. "Looking beyond the current take-make-waste extractive industrial model, a circular economy aims to redefine growth, focusing on positive society-wide benefits. It entails gradually decoupling economic activity from the consumption of finite resources and designing waste out of the system" (Ellen MacArthur Foundation, 2019).

There are numerous industries who would observe financial benefits from implementing circularity in their business. For example, some industries which would be arguably the most improved through circularity are the fashion, agriculture, and manufacturing industries. As it is impossible to perform an in-depth analysis on all industries within this paper, the focus of this research will be on how IoT-enabled circularity empowers risk management. This concept will be examined through the use case of smart insurance contracts. (Smart contracts refer to legal insurance documents which are put on the cloud or blockchain). Smart insurance contracts can in turn be applied to various industries such as manufacturing, real estate management and logistics.

2.4 Case Study – Circular economy in Industrial Construction: How IoT can Reduce Industrial Construction Waste

This case study proposes a circular economy model to transform industrial construction by leveraging the potential of IoT. Industrial construction refers to construction activities undertaken across various industries such as steel, oil and gas, cement etc. Firstly, the current landscape in industrial construction was reviewed in conjunction with the European Commission's waste management ambitions to identify opportunities. A conceptual circular economy model was proposed to prevent landfilling of excess materials generated during industrial construction. The proposed model uses IoT to address barriers to circularity.

However, using IoT exposes industrial construction to cyber-security risks that need to be acknowledged.

Following are the key insights.

- Circular economy in industrial construction is attainable. European Union and government policy and legislation can facilitate and where necessary, expedite implementation. Appropriate incentives as well as creation of relevant platforms can produce an environment conducive to circularity.
- The industrial IoT maturity model in the Industrial Construction Sector allows us to evaluate the current position of the industry and set clear objectives as to how to enact change. It serves as a path to reduce waste, giving competitive gains and efficiency.
- The adoption of an emergent maturity model that enables the industry to secure the social license to operate and thrive. Building on existing relationships within the supply chain, competitors, government and legislators as well as communities and customers. Transitioning the industry in phases from exploration to adoption and thence adaptation delivering real economic, environmental and social benefits.
- Training and awareness across all levels of organizations adopting IoT dependent strategies are critical to reducing cyber-risk.

3 LCA and CSPDI Property Requirements

The first step in the development of the pattern language for CE-IoT is the definition of the property requirements and based on these requirements we will be guided to define the pattern language. To this direction, we also have to consider the way that the defined language will be machine interpretable able to support requirements that are analyzed in the subsections below, organized per LCA - CSPDI property.

3.1 LCA Property Requirements

The composition structure of the various smart objects (devices, applications etc.) in the CE-IoT platform should include:

- The LCA properties that the CE-IoT pattern should guarantee
- Conditions that should be monitored in order to ensure those LCA properties

The adopted pattern language should be able to define the LCA properties i.e.:

- Location
- Condition
- Availability

3.1.1 Location, Condition and Availability

The interplay between Circular Economy and IoT provides a fertile ground for innovation and value creation. Circular economy value drivers include extending the useful life of finite resources and maximising the utilisation of assets, creating an emerging class of "looping assets", and regenerating natural capital for more effective and efficient use. IoT value drivers enable for a new window of opportunity to be opened on the economic cycle so that we can create a new breed of circular economics. IoT becomes a serving enabler by collating knowledge about asset locations, conditions, quality and performance in real time and over time. IoT has been catalytic already in presenting solutions to many resource related challenges that have been faced by circular economy innovators. More specifically:

- **Location:** the physical location of the component (i.e. the room where a server is installed)
- **Condition:** good or requiring maintenance, repair, refurbishment, and recycling
- **Availability:** working, ready for reuse, or broken.

Especially since the Availability is a requirement that belongs as a part of both Security and Dependability accordingly, a detailed description is presented in the next Sections (3.2.2, 3.2.4).

The feedback-rich nature of circular economy models might conversely make them particularly suitable to help extract value from the large amount of data generated by IoT. Therefore, an even more prominent range of opportunities emerges when these value drivers are paired and their congruency is being extensively explored.

IoT, on the other hand, provides valuable insights and usage information in terms of asset key enabling properties of the assets such as LCA thus transforming the physical assets to intelligent ones.

Recent IoT technologies enable products or machines to continuously create value even after they have left the supply chain. Through intelligent assets delivering information concerning

their location, condition or availability, companies and end users can capture value in new ways throughout an asset's use cycle. Additionally, the manufacturer could use the information generated by the asset to further improve the product design. Finally, connecting people and things via mobile devices, enables the sharing or leasing of assets, which becomes a significant economic opportunity for businesses and individuals across multiple sectors.

3.2 CSPDI Property Requirements

Moreover, the development of the pattern specification language aims not only to specify the circular economy design LCA patterns but also the IoT architectural CSPDI patterns. Specifically, the CE-IoT platform should include:

- The **CSPDI** properties that the CE-IoT pattern should guarantee
- Conditions that should be monitored in order to ensure those CSPDI properties

The adopted pattern language should be able to define the CSPDI properties i.e.:

- Connectivity
- Security
- Privacy
- Dependability
- Interoperability

3.2.1 Connectivity

Apart from the non-functional properties that are described in the next subsections, there are also some functional properties related to the QoS and the SPDI properties that affect the design of IoT systems such as connectivity and scalability.

Connectivity is the property that given a network, a path between end nodes can be determined. Moreover, a network is connected if every pair of nodes are connected through a path. Connectivity is required due to the fast-growing number of interconnected users, smart objects and applications. At the network layer, the vastly increased demands require highly efficient programmable connectivity.

Together with connectivity, scalability is the property that enables the ability of a network to change in size or scale. This includes the capability of seamless discovery and bootstrapping of additional components, as well as highly efficient orchestration, event processing, analytics and platform integration.

3.2.2 Security

Traditionally, Security consists of the three separate properties; confidentiality, integrity, and availability (see Figure 2), sometimes also abbreviated as CIA. Particularly,

- Confidentiality: the disclosure of information happens only in an authorised manner; i.e. non-authorised access to information should not be possible.
- Integrity: maintenance and assurance of the accuracy and consistency of data.
- Availability: the invocation of an operation to access some information or use a resource leads to a correct response to the request.

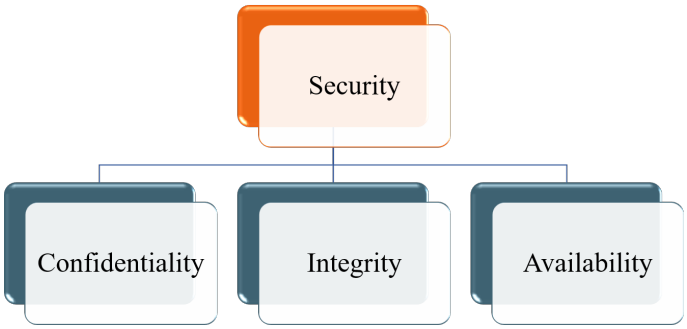

Figure 2. Security Properties

Consequently, in order to develop a pattern specification language to specify the circular economy design LCA patterns and the IoT architectural CSPDI patterns for CE-IoT, we will also implement patterns which cover these three aspects.

Moreover, all smart object/component/activity must provide the above required properties, through standardized APIs for security functions, in order to achieve end-to-end (E2E) security, i.e. applications or components must not use their own cryptography libraries.

Besides the above, monitored conditions about pattern components are demanded to ensure above-mentioned E2E properties. The monitor for the **Confidentiality** property depends on the state of the data. This means that for the data transmission, at least one of the endpoints needs to be monitored. In that case, if there is a standard system-wide API for cryptography functions, the behavioural monitoring is a proposed solution. The main characteristic of this procedure is the encryption of data, before their transformation, i.e. a call to a sufficiently strong encryption function must be observed.

- Data at rest: Similarly, to data in transit, behaviour monitoring can be used to determine whether confidentiality of data is ensured using sufficiently strong encryption. Before data is written to a file, there must be a corresponding call to an encryption function. After data is read from a file, there must be a corresponding call to a decryption function.

- Data in processing: For data processing, data at rest must be decrypted. During processing, it must be monitored that there are no unexpected network connections by the data processing process(es).

An example for the description of the monitor for the **Availability** property would be simple to devise, for the reason that a component is considered to be available if it can be reached via the network and is able to perform specified services. Non-availability can be due, e.g., to loss of network connectivity or the hardware running a network component failing. Finally, for **Integrity**, in case of data in transit, many network protocols provide integrity protection. Thus, if data integrity is required, it must be monitored that protocols meeting this requirement are used. Moreover, in case of data at rest, data at rest is usually integrity protected at the hardware level and/or at the file system level.

3.2.3 Privacy

Until now, many efforts have been made to define privacy, but so far no universal definition can be established. Although the requirement for privacy is universal, its specific form varies according to the current era and context (technical landscape) [5].

In any case, IoT/IIoT devices generate, process, exchange and store huge amounts of security-critical data which include privacy-sensitive information. For that reason, not only ethical but also regulatory handling is required where appropriate (e.g. medical data).

Another important issue about privacy, is the fact that information which is collected in a system becomes personal if identity (direct or indirect) can be correlated with an activity [6]. The term of identifier is mentioned for a name, an identification number, location data or an online identifier (such as IP address). Except that, it may be specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person. Therefore, data protection law does not apply to anonymous data (i.e., data in which the data subjects are no longer identifiable). Although, for a reasonably high risk of identification, the information should be regarded as personal data [7] and based on the literature, the risks in these cases, may be quite high [8].

Moreover, the compliance with regulations (such as the General Data Protection Regulation (GDPR) of European Union – Regulation (EC) 2016/679 (European Parliament 2016))[1] and several standards, like the ISO/IEC standards 27018 (ISO/IEC 2014)[2] and 29100 (ISO/IEC 2011)[3] are included in the issue of privacy (Figure 3).

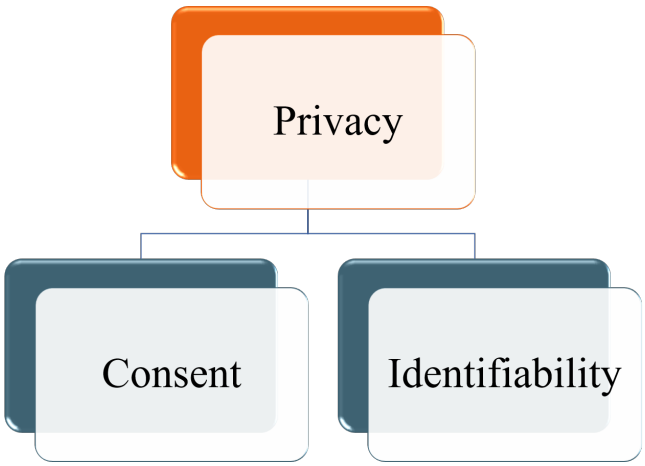

Figure 3. Privacy Properties

3.2.4 Dependability

In general, dependability is the ability of a system to deliver its intended level of service to its users [9]. The main features which constitute dependability are reliability, availability, safety and maintainability. Reliable systems impose the need for greater fault and intrusion resistance. Satisfying these properties can prevent threats such as faults, errors and failures offering fault prevention, fault tolerance and fault detection. In particular, dependability in CE-IoT focuses on three key attributes reliability, availability and fault tolerance as follows (see Figure 4):

- **Reliability** is the ability of a system to perform a required function under stated conditions for a specified period of time [10]. It is one among the characteristics of system dependability and is related to availability. For hardware components, the property is usually provided by the manufacturer. Its calculation is supported by the

[1] E. Union, "Regulation 2016/679 of the European parliament and the Council of the European Union," Off. J. Eur. Communities, vol. 2014, p. 1–88, 1995.

[2] ISO/IEC 27018, 2014. [Online]. Available: http://www.iso27001security.com/html/27018.html.

[3] ISO/IEC 29100:2011, 1996. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:29100:ed-1:v1:en.

complexity and the age of the component. Reliability will be classified into two basic classes, deterministic models and probabilistic models.

- **Availability** guarantees that data is accessible once it's required. The lack of availability in network transmissions features a severe influence on each security and therefore the dependability of network. Particularly, network availability is the ability of a system to be operational and accessible once needed to be used. Moreover, availability in networks is the chance of no-error packet reception [11]. Other factors that have an effect on the availability of a link are the transmission range of the signal strength, noise, fading effects, interference, modulation method, and frequency.
- **Fault Tolerance** is the ability of a system or component to continue normal operation in spite of the presence of hardware or software faults [10]. In recent years there has been considerable interest for network fault tolerance. The most common solutions to guarantee fault tolerance and avoid single point of failure, include the replication of paths forwarding traffic in parallel, the use of redundant paths and the ability to switch in case of failure (failover) and traffic diversity.

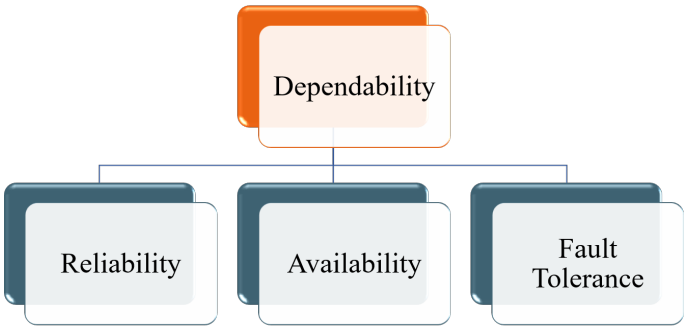

Figure 4. Dependability Properties

Hence, dependability analysis of an IoT system includes whether non-functional requirements such as availability, reliability, safety and maintainability are preserved. The conditions depend on the respective dependability property that the system guarantees. The satisfiability of a property can be defined by a Boolean value (i.e. true, false), an arithmetic measure (i.e. delay) or probability measure (i.e. reliability/uptime availability).

3.2.5 Interoperability

Desired interoperability characteristic imposes special requirements on the designed CE-IoT framework. Interoperability gives an ability to a system to interact with other systems. In other words, to take advantage of the full potential of the IoT vision, we need standards to enable the horizontal and vertical communication, operation, and programming across devices and platforms, regardless of their model or manufacturer.

The following types of interoperability can be distinguished and will be covered by CE-IoT (see Figure 5):

- **Technological interoperability** – enables seamless operation and cooperation on heterogeneous devices that utilize different communication protocols
- **Syntactic interoperability** – establishes clearly defined formats for data, interfaces and encoding
- **Semantic interoperability** – settles commonly agreed information models and ontologies for the used terms that are processed by the interfaces or are included in exchanged data

- **Organizational interoperability** – cross-domain service integration and orchestration through common semantic and programming interfaces.

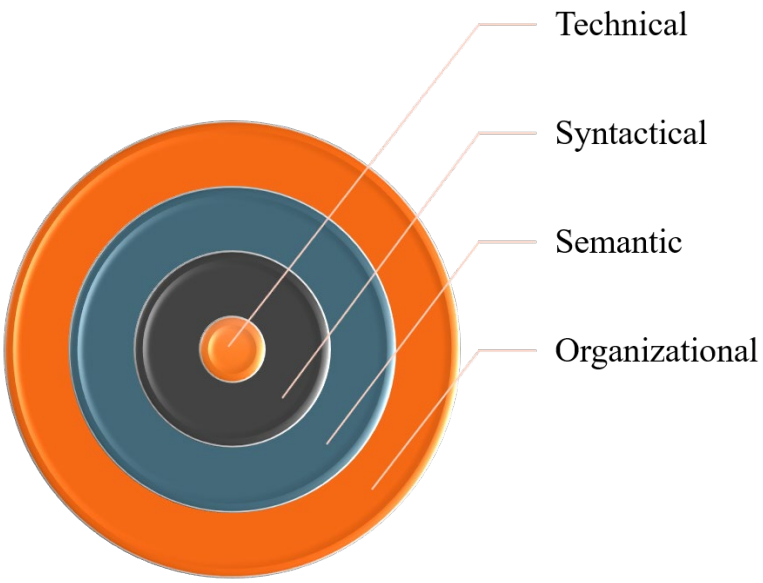

Figure 5. Different levels of Interoperability

4 Domain Specific Requirements and Properties

This section will provide a high-level analysis of the domain use cases described in the previous section with respect to the fundamental circular properties *Location*, *Condition* and *Availability*. Further, the basic properties for smart networked devices, *Connectivity*, *Security*, *Privacy*, *Dependability*, *Interoperability* are considered.

4.1 Use Case 1: Electricity Load Control

As mentioned in the previous section, Demand Side Response is the notion of demand adapting to supply instead of the opposite; this is made possible through providing incentives to end-users to e.g. lower their power consumption during supply shortages. The crucial element for materializing the Demand Side Response system is the IoT Load Control device; located at the consumer-end, this device implements the aforementioned IoT Load Control protocol by adapting the household's or facility's power consumption to the current power supply. It needs to satisfy the properties and requirements listed in the table below:

Property	**Key Requirements**
Location	Geographical location of all load control devices should be known to the operator. Location needs to be tied with device ownership for accounting purposes.
Condition	Load control devices need to be aware of the entire current load used in the household or industrial grid, as well as what it is utilized for. The latter is important to determine the criticality of the used energy, so as to curtail energy use depending on global grid fluctuations.
Availability	Load control device should calculate and report power consumption in real-time.
Connectivity	Permanent online capability with all power devices and to grid is required.
Security	Control of devices and related automation should be highly secure.
Privacy	Power demand and consumption patterns should not be disclosed to unauthorized parties.
Dependability	Device should be able to operate continuously and accurately under all conditions. Asset is fixed, therefore less complex network infrastructure for reliability is required.
Interoperability	Device should be able to communicate with different power devices, and should be compatible with one (or more) power grids.

4.2 Use Case 2: Loyalty Platform

In the context of this use case, a general-purpose platform is conceived which acts as a general purpose instrument to provide loyalty services. For the particular scenario, the platform will operate with a fixed home appliance, e.g. a refrigerator. The home appliance will function as a bi-directional information node, through which the loyalty platform will perform maintenance, but also communicate with the owner (e.g. for promotions). The properties and requirements to be satisfied are listed in the table below:

Property	**Key Requirements**

Location	Geographical location of asset (required for reverse logistics, customer service)
Condition	Awareness of asset condition with respect to its operational capabilities and health. Required for predictive maintenance.
Availability	Periodic reporting of asset condition. Ability for the platform to send push notifications and/or control commands to the asset.
Connectivity	Asset should periodically send health and usage reports to platform backend and receive information (e.g. notifications).
Security	Asset control should be highly secure and confined to the loyalty platform backend.
Privacy	Asset usage information should not be disclosed to unauthorized parties.
Dependability	Device should be able to operate continuously and accurately under all conditions. Asset operation is mainly confined to customer premises, therefore less complex network infrastructure for reliability is required.
Interoperability	Asset should be able to communicate with the loyalty platform. Additional compatibility with other platforms (e.g. for reuse, recycling etc.) can be considered.

4.3 Use Case 3: Smart Insurance

The underlying concept of smart insurance is the monitoring of insured assets by the insurer, so that information asymmetry (which leads to higher premiums) can be mitigated. In order to accomplish this, the assets need to be fitted with the relevant sensors and IoT-enabled, and communicate with the insurance IT platform. The properties and requirements to be satisfied are listed in the table below:

Property	**Key Requirements**
Location	Geographical location of asset to be insured. For mobile assets (e.g. trucks, cars) additional information is required e.g. speed, heading, acceleration.
Condition	Deep knowledge of an asset's health condition. Required primarily to adjust premiums. Can also be used for predictive maintenance.
Availability	Continuous reporting of asset condition required.
Connectivity	Asset needs to be constantly monitored. Decrease in asset visibility will affect premium.
Security	Communication with the insurance IT platform should be secured and any unauthorized access to the asset should be prevented.
Privacy	Asset usage information should not be disclosed to unauthorized parties.
Dependability	Asset should be able to operate continuously and accurately under all conditions. Especially for mobile assets, care should be taken to ensure ubiquitous connectivity.

Interoperability	Asset should be able to communicate with the insurance IT platform. A cross-company IT platform standard can also be envisaged, but would require coordination of different insurers.

4.4 Use Case 4: Construction

Within this use case, the main objective is to promote circularity by reduction of industrial construction waste through smart redistribution of unused building materials. This is achieved by fitting the building materials with smart IoT devices comprising of sensors and RDIF tags, and monitoring the building materials throughout the entire supply chain via a common data sharing platform. The properties and requirements to be satisfied are listed in the table below:

Property	**Use Case Relevance**
Location	Geographical location of construction material. Traceability with respect to construction site and ownership.
Condition	Knowledge of a material's condition with respect to its building qualities. Awareness of state of material packaging and/or container.
Availability	Frequent reporting of material condition is desired in order to build trust between parties. Reporting is mandatory when ownership is changed.
Connectivity	Periodic and ad-hoc monitoring required depending on building material substance and storage conditions.
Security	Communication with the data sharing platform should be secured. Appropriate safeguards should be in place for material ownership changes.
Privacy	Usage information and material location should not be disclosed to unauthorized parties.
Dependability	Material location, condition and availability reporting should be able to operate continuously and ubiquitously. Special provisions should be made for reliable outdoor storage reporting irrespective of weather conditions. Provisions should be made to cater for tracking items belonging to groups or batches.
Interoperability	Material should be able to communicate with the data sharing platform; common data interface should be in place for all stakeholders (manufacturers, consumers, procurers, forwarders, regulators, recycling entities etc.)

5 Circular Economy Design Properties

5.1 Overview

The implementation of a usable CE-IoT-enabled ecosystem, as shown in Figure 6 (also depicting the complementarity of Circular Economy and IoT that provides added value), must be based on a **requirements' analysis** focusing on:

5. circularity-enabling properties of smart assets, namely **location, condition and availability (LCA)**,
6. key enabling IoT technologies, namely connectivity, security, privacy, dependability and interoperability (CSPDI), and
7. **application-specific business and technical needs** for each application domain (e.g. telecom, smart energy, e-health).

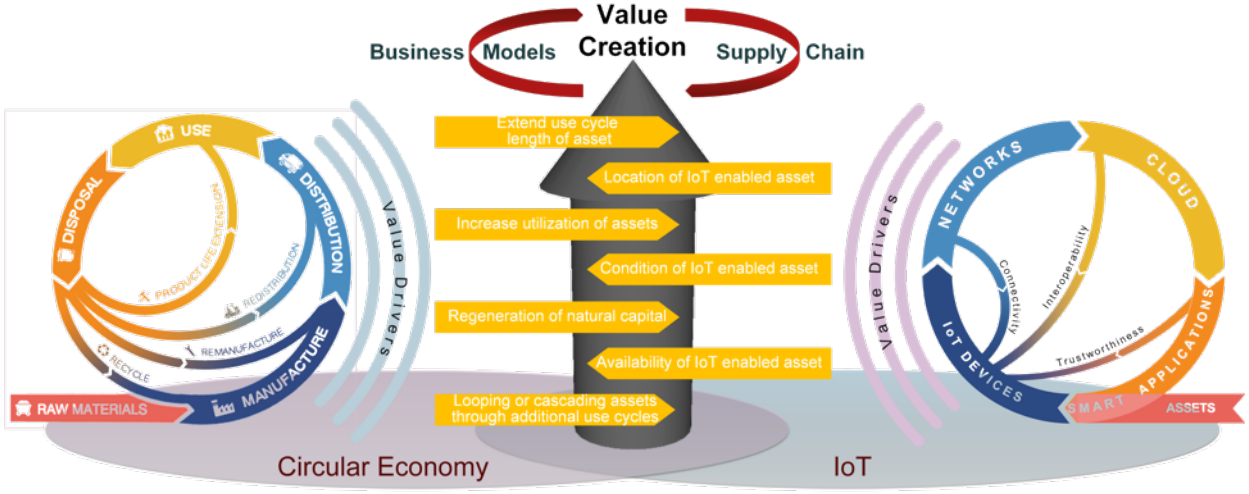

Figure 6: CE-IoT Concept

The goal is, but not limited, to design a product-services-systems supply chain framework with effective and secure reverse logistics capabilities. The essential LCA and CSPDI properties of the CE-IoT deployment in each of the application domains will have to be considered. Moreover, concrete business models, IoT enhanced supply chains, as well as robust networking, interoperability, monitoring and adaptation capabilities will need to be encompassed to address the needs of each specific domain. This will have to be carried out in two phases: in the first phase, an initial set of architectural CSPDI patterns has to be developed, along with the CE and IoT value drivers. In the second phase, the outcomes of the first phase will be integrated into the overall CE-IoT architecture and adapted to the specific application/vertical domain.

5.2 Circular Economy Business Models

The development of the *CE-IoT* approach will be driven by the analysis of: (1) key business-driven requirements for intelligent (IoT enabled) assets (location, condition and availability - LCA), (2) the technical circularity requirements of the key enabling IoT technologies (connectivity, security, privacy dependability and interoperability - CSPDI), and (3) the vertical, business and technical requirements arising from the two demonstrators (telecom and cloud services). These requirements will lead to the definition of the value drivers both for the circular economy and IoT and to the definition of the integrated *CE-IoT* framework and the integrated value creation. The overall *CE-IoT* framework will include the exact form of the

essential LCA and CSPDI properties of the demonstrators that should be addressed by patterns. They will also indicate concrete business models and supply chains enhanced by IoT as well as concrete networking, interoperability, monitoring, and adaptation capabilities that would need to be tackled by the IoT in order to address the needs of the demonstrators.

5.3 Description of Circular Business Model LCA and CSPDI Concept

The key element in our approach that will enable the *CE-IoT* to fulfil this role is the use of **patterns**. These patterns define generic ways of composing (i.e., establishing the connectivity between) and configuring the different, heterogeneous smart objects and software components that may exist at all layers of the IoT applications implementation stack, including: sensors and actuators; smart devices; software components at the network, cloud, IoT platforms and/or other middleware layers; as well as software components at the IoT application layer. To do so, patterns encode abstract and generic smart object interaction and orchestration protocols, enhanced, if necessary, by transformations to ensure the semantic compatibility of data. Furthermore, and more importantly, the smart object interaction and orchestration protocols encoded by the patterns must have an evidenced ability (i.e., an ability proven through formal verification or demonstrated through testing and/or operational evidence) to achieve not only a semantically viable interoperability between the smart objects that they compose but also specific CSPDI properties, or LCA properties which may be required of compositions. The compositions defined by patterns are both vertical and horizontal, i.e., they can involve smart objects at the same (horizontal) or different (vertical) layers of the IoT implementation stack.

6 Pattern-Language Definition

6.1 Overview

This section defines the Pattern Language. Overall, this language:

- provides constructs for expressing/encoding dependencies between LCA-CSPDI properties at the component and at the composition/orchestration level.
- is structural; It does not prescribe exactly how the functions should be executed nor, e.g., how the ports ensure communication.
- supports the static and dynamic verification of LCA-CSPDI properties.
- is automatically processable by the CE-IoT framework so that IoT applications can be adapted at runtime.

The key element in our approach that will enable the *CE-IoT* to fulfil this role is the use of **patterns**. These patterns define generic ways of composing (i.e., establishing the connectivity between) and configuring the different, heterogeneous smart objects and software components that may exist at all layers of the IoT applications implementation stack, including: sensors and actuators; smart devices; software components at the network, cloud, IoT platforms and/or other middleware layers; as well as software components at the IoT application layer. To do so, patterns encode abstract and generic smart object interaction and orchestration protocols, enhanced, if necessary, by transformations to ensure the semantic compatibility of data. Furthermore, and more importantly, the smart object interaction and orchestration protocols encoded by the patterns must have an evidenced ability (i.e., an ability proven through formal verification or demonstrated through testing and/or operational evidence) to achieve not only a semantically viable interoperability between the smart objects that they compose but also specific CSPDI properties, or LCA properties which may be required of compositions. The compositions defined by patterns are both vertical and horizontal; i.e., they can involve smart objects at the same (horizontal) or different (vertical) layers of the IoT implementation stack.

6.2 IoT application architecture and orchestration modelling

The overall objective of CE-IoT is to develop a framework that will be capable of managing the IoT applications based on circular economy patterns. Therefore, it is necessary to develop a language for demonstrating how the interplay of value drivers and systems thinking from both the circular economy and IoT. A model with such characteristics will effectively serve as a general “architecture and workflow model” of the IoT application. Once defined, this model will be used in conjunction with patterns to enable the reasoning required for determining the applicability of particular LCA-CSPDI patterns in specific IoT applications.

The main constructs for defining an IoT application model in CE-IoT is highlighted in the following Figure 7. It describes the basic modelling constructs of the language and their relations in the form of a Unified Modeling Language (UML) diagram.

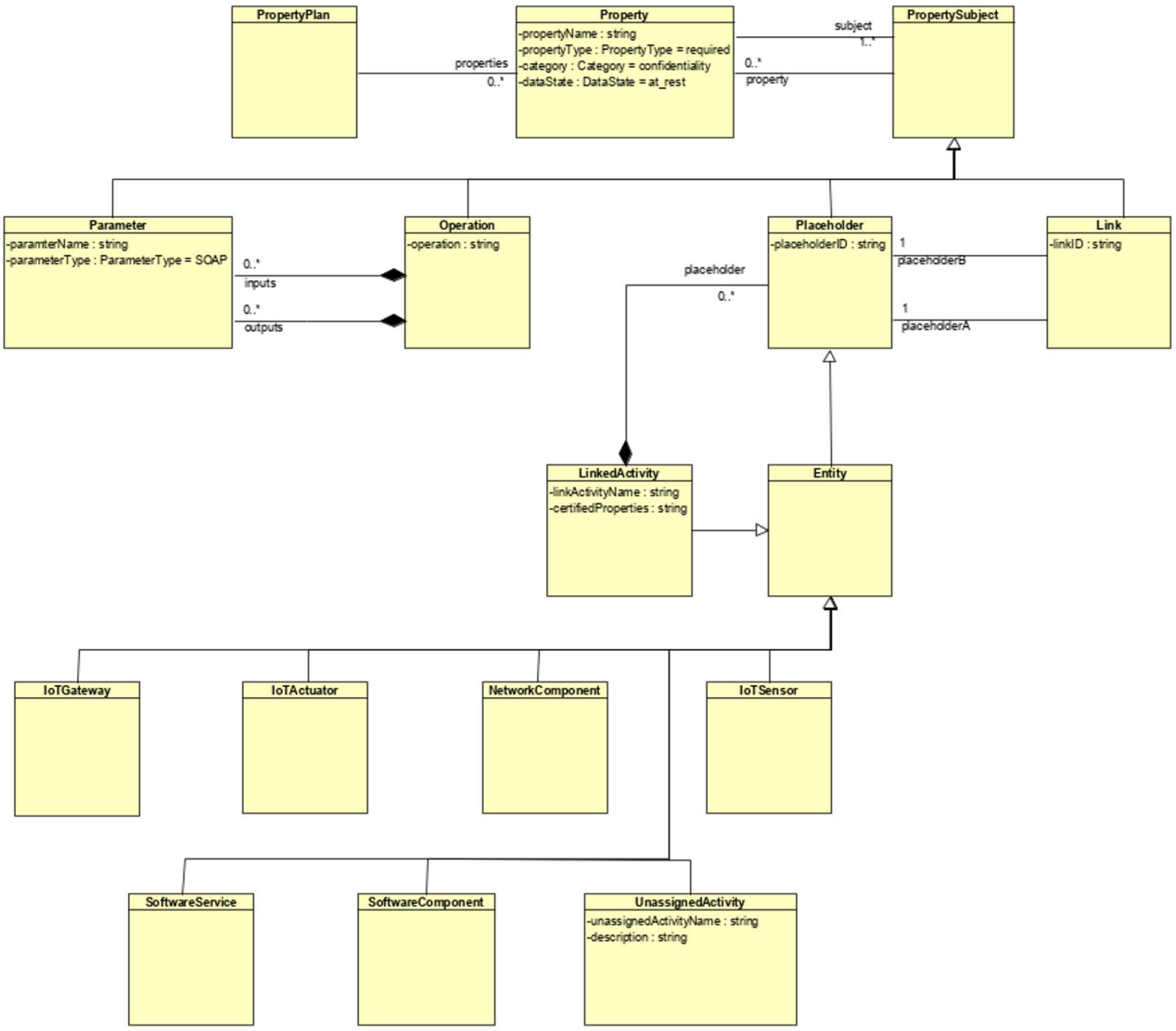

Figure 7. CE-IoT Orchestration System Model

More details about the main constructors of IoT application model in CE-IoT, which is presented in the Figure 7, are described in the Table 1.

Table 1. Description of basic CE-IoT orchestration system model constructs

Constructs	**Description**	**Interactions**
Placeholder	It may also be characterised by their LCA-CSPDI and properties.	A property of a placeholder is specified according to the class Property.
Property	Property has a name, a type, a verification, a category and a dataState. The attribute type refers to the state of the property,	A required property is a property that a placeholder must hold in order to be included (considered for) the orchestration. For example, if the required property of an orchestration defining a secure logging process is Confidentiality, then all

	which can be required or confirmed.	placeholder activities involved in the orchestration and the links between them may be required to have the Confidentiality property. On the other hand, a confirmed property is a property that is verified at runtime, through a specific means as defined in the Verification.
Entity	It involves some specific activities. The Entity class is extended by LinkedActivity and UnassignedActivity classes.	The implementation of an activity in an IoT application orchestration may be provided by: (i) A *software component*, i.e., a software module with an available and modifiable implementation that encapsulates a set of functions and data and makes them available through a programmatic interface. (ii) A *software service*, i.e., a software module that encapsulates a set of functions and data and makes them available through a programmatic interface, accessible remotely over a network, whose implementation is neither available to the owner nor modifiable. (iii) A *network component*, such as software defined network controllers, software switches/vSwitches, and potentially legacy networking components. (iv) An *IoT sensor*, i.e., a device that collects data from the environment or object under measurement and turns it into useful data. (v) An *IoT actuator*, i.e., a device that takes electrical input and transforms the input into tangible action. (vi) An *IoT gateway*, i.e., is a physical device or software program.
LinkedActivity	It is referring to activities whose implementation is known, defines the specification of the LCA-CSDPI properties of the involved activity.	At any instance of time, activities (see Entity interaction) may have a known implementation or a not known implementation. In the former case, the activity will be a linked activity. In the latter, the activity will be an unassigned activity (see class UnassignedActivity).
UnassignedActivity	It has an attribute Name, as LinkedActivity, in order to be identified unambiguously.	In contrast with LinkedActivity class, it is referring to a not known implementation.

6.3 Implementation aspects

An important requirement for implementing the LCA-CSPDI pattern-driven management in CE-IoT is to support the automated processing of developed patterns. To achieve this, the CE-IoT LCA-CSPDI patterns is expressed as Drools[4] business production rules, and the associated rule engine, by applying and extending the Rete algorithm [12]. It is an efficient pattern-matching algorithm known to scale well for large numbers of rules and data sets of facts, thus allowing for an efficient implementation of the pattern-based reasoning process.

In particular, the generic structure of Drools production rule is presenting below:

```
rule name <attributes>*
when <conditional element>* then <action>* end
```

The ***when*** part of the rule specifies a set of conditions and the ***then*** part of the rule a list of actions. When a rule is applied, the Drools rule engine checks whether the rule conditions (defined within the <conditional element> above) match with the facts in the Drools Knowledge Base (KB) and if they do, it executes the actions (i.e. "<action>") of the rule. Rule actions are typically used to modify the KB by inserting, retracting or updating the objects (facts) in it, through the standard Drools actions "insert", "retract" and "update", respectively. The conditions of a rule are expressed as patterns of objects that encode the facts in the Drools KB. These patterns define object types and constraints for the data encoded in objects which may be atomic or complex. Complex Drool object constraints are defined through logical operators (e.g. and, or, not, exists, forall, contains). The full grammar of the current version of the Drools rule language (version 7.16.0 as of writing this deliverable) can be found online. An overview of the major specification constructs is presented in the following Table.

Table 2. High level DROOLS rules specification constructs

Type	Construct	Description
Conditional element	`and-CE | or-CE | not-CE | exists-CE | forall-CE | contains-CE | from-CE | collect-CE | accumulate-CE | eval-CE`	Conditional elements are used to specify conditions in the *when* part of a rule and in constraint expressions (see Pattern construct below). Conditional elements realise basic logical operators (e.g. *and*, *or*, *not*); quantified logic operators (*contains*, *forall* and *exists*); and object collection operators (e.g. *collect*, *accumulate*).
Pattern	`Top level syntax:` `Pattern: <pattern-Binding ":" > PatternType "(" Constraints ")"`	Patterns are matched with elements in the working memory. The pattern binding is typically a variable and the pattern type refers to declared object types that could be matched with the pattern. Constraints are specified by logical expressions. Such expressions can be constructed by logic conditional elements (see above); object collection elements; unification operators; relational; arithmetic; property/list access operators; data accumulation functions; regular expression matching operators, and; temporal operators.
Action	`Modify | Update | Insert | Retract`	Pattern-related actions include *Modify* to modify the contents of a fact, Update a face, *Insert* to insert new fact in the KB and *Retract* to delete a fact.

[4] https://www.drools.org/

7 Pattern Rules

This section presents the first set of CE-IoT pattern rules, using the language and associated constructs defined in the previous section. The Security properties of Confidentiality, Integrity and Availability are analysed separately in the corresponding subsections below, as different types of property reasoning and monitoring conditions need to be defined for each one of them.

7.1 Security

7.1.1 Confidentiality

7.1.1.1 Pattern Definition

The achievement of Confidentiality requires that the disclosure of information can be only in an authorised manner. Formal definitions of Confidentiality are typically based on the concept of Information Flow (IF) [13], separating users in classes with different access rights to the system's information and distinguishing the information flows within the system according the user classes they should be accessible to. Taking to account this method, the Perfect Security Property (PSP) [14] requires low-level users (i.e. a user with restricted access, in contrast to high-level users having full access). The said users are only allowed to view public information, should not be able to determine anything concerning high-level (confidential) information.

Let us consider a sequential orchestration **P** with two activity placeholders, **A** and **B**, whereby B is executed after A. For each *x* in *{P, A, B}* the following hold:

- IN^x and OUT^x are the sets of inputs and outputs of x, and $E^x = IN^x \cup OUT^x$;
- V^x and C^X are two disjoint subsets of E^x, portioning into public parts and confidential parts respectively.
- The inputs of A are the inputs of the workflow P
- The inputs of B are the outputs of A
- The outputs of the orchestration P are the outputs of B

Based on the above, the pattern model for preserving PSP (i.e. confidentiality) on the service orchestration P can be defined as follows:

i. NP:
  a. $PSP(A,\ V^A,\ C^A)\ and\ V^A \subseteq V^P\ and\ C^A \cap V^P = \emptyset$
  b. $PSP(B,\ V^B,\ C^B)\ and\ V^B \subseteq V^P\ and\ C^B \cap V^P = \emptyset$

ii. OP:
  a. $SecReq^P = PSP(P,\ V^P,\ C^P)$

PSP then holds on the orchestration ***P*** if, for all activity placeholders *x* in *{A, B}*, the following are true:

a) $V^X \subseteq V^P$; i.e. the actions of *x* that reveal public information are part of the actions of P that reveal public information are part of the actions of P that reveal public information, and
b) $C^X \cap V^P = \emptyset$; i.e. the actions of *x* that reveal confidential information do not include any action of P that reveal public information.

7.1.1.2 Pattern specification rule

The confidentiality (PSP) pattern can be represented in Drools as shown below:

```
1. rule "PSP on Cascade"
2. when
3.  $A: Placeholder($input : operation.inputs,
4.   $intData : parameters.outputs)
```

```
5.  $B: Placeholder(parameters.inputs == $intData,
6.    $output : parameters.outputs)
7.  $ORCH: Sequence(parameters.inputs == $inputs,
8.    parameters.outputs == $outputs,
9.    firstActivity == $A, secondActivity == $B)
10. $OP: Property( propertyName == "PSP",
11.   subject == $ORCH, satisfied == false)
12. $SP: PropertyPlan (properties contains $OP)
13.then
14. PropertyPlan newPropertyPlan = new newPropertyPlan ($SP);
15. newPropertyPlan.removeProperty($OP);
16. Set V_P = $OP.getAttributesMap().get("V");
17. Property NP_A = new Property($OP, "PSP", $A);
18. NP_A.getAttributesMap().put("V", new Operation("subset", V_P));
19. NP_A.getAttributesMap().put("C", new Operation("subset", new  Operation("complement",V_P)));
20. newPropertyPlan.getProperty().add(NP_A);
21. insert(NP_A);
22. Property NP_B = new Property($OP, "PSP", $B);
23. NP_B.getAttributesMap().put("V", new Operation("subset", V_P));
24. NP_B.getAttributesMap().put("C", new Operation("subset", new   Operation("complement",V_P)));
25. newPropertyPlan.getProperties().add(NP_B);
26. insert(NP_B);
27. insert(newPropertyPlan);
28.end
```

7.1.2 Integrity

7.1.2.1 Pattern definition

Data Integrity refers to the maintenance and assurance of the accuracy and consistency of data.

Let us consider a sequential orchestration **P** with two activity placeholders, **A** and **B**, whereby B is executed after A. For each *x* in *{P, A, B}* the following hold:

- IN^x and OUT^x are the sets of inputs and outputs of x
- Dx(i) the data of x at the given time t
- Hash(i) are the cryptographic hash function result applied to data i
- The inputs of A are the inputs of the orchestration P
- The inputs of B are the outputs of A
- The outputs of the orchestration P are the outputs of B

Based on the above specification, a generic pattern for integrity can be defined at data at rest as the following:

$$\mathrm{Hash}(D^x(i))=\mathrm{Hash}(D^x(i-1))$$

7.1.2.2 Pattern specification rule

The integrity pattern can be represented in Drools as shown below:

```
1.  rule "Integrity"
2.  when
3.    $A: Placeholder($input : operation.inputs,
4.      $intData : parameters.outputs)
5.    $B: Placeholder(parameters.inputs == $intData,
6.      $output : parameters.outputs)
7.    $ORCH: Link(firstActivity == $A, secondActivity == $B)
8.    $OP: Req( propertyName == "Integrity",
9.      subject == $ORCH, satisfied == false)
10.   $SP: PropertyPlan (properties contains $OP)
```

```
then
  PropertyPlan newPropertyPlan = new PropertyPlan($SP);
  newPropertyPlan.removeRequirement($OP);
  Req Hash1 = new Req($OP, "equality",sha512($A.input),sha512(operation.input));
  newPropertyPlan.getProperties().add(Hash1);
  insert(Hash1);
  Req Hash2 = new Req($OP, "equality",sha512($A.output),sha512($B.inputs));
  newPropertyPlan.getProperties().add(Hash2);
  insert(Hash2);
  Req Hash3 = new Req($OP, "equality",sha512($B.output),sha512(operation.inputs));
  newPropertyPlan.getProperties().add(Hash3);
  insert(Hash3);
  insert(newPropertyPlan);
end
```

7.1.3 Availability

7.1.3.1 Pattern definition

The Availability is defined as “readiness for correct system service”; a service is deemed to be correct if it implements the specified system function. Readiness of a system in this definition means that if some agent invokes an operation to access some information or use a resource, it will eventually receive a correct response to the request.

7.1.3.2 Pattern specification rule

The availability pattern can be represented in Drools as shown below:

```
rule "Availability"
when
  $A: Placeholder($input : operation.inputs,
   output : parameters.outputs)
  $T: Timer(time.Interval(“Default time interval”))
  $ORCH: Check($A,$T)
  $OP: Req( propertyName == "Availability", subject == $ORCH, satisfied == false)
  $SP: PropertyPlan (properties contains $OP)
then
  PropertyPlan newPropertyPlan = new PropertyPlan($SP);
  newPropertyPlan.removeRequirement($OP);
  Req Hash1 = new Req($OP,"ResponseTime",$A, ”Default response time”);
  newPropertyPlan.getProperties().add(Hash1);
  insert(Hash1);
  insert(newPropertyPlan);
end
```

7.2 Privacy

7.2.1 Consent

7.2.1.1 Pattern definition

Due to GDPR constrains, patterns should be developed in order for CE-IoT to be GDPR compliant. One of the constraints that need to be considered is for the user to give her consent on their data to be used. Let us consider a simple service composition with following conditions:

- IN^A and OUT^A are the sets of inputs and outputs of A
- Dx Are the data which belong to owner X
- C is a set of users who have agreed their data can be processed and stored

Then in order to be able to able to create every service composition the following pattern should be applied

$$IN^P = D^A \text{ where } A \subseteq C$$

7.2.1.2 Pattern specification rule

The consent pattern can be represented in Drools as shown below:

```
rule "Consent"
when
  $A: Placeholder($input : operation.inputs, $output:operation.output)
  $ORCH: Single(parameters.inputs == $input,
   parameters.outputs == $output)
  $OP: Property( propertyName == "UserConsenus",
   subject == $ORCH, satisfied == false)
  $SP: PropertyPlan(properties contains $OP)
then
 PropertyPlan newPropertyPlan = new PropertyPlan ($SP);
 newPropertyPlan.removeProperty($OP);
 insert(newPropertyPlan);
end
```

7.2.2 Identifiability

7.2.2.1 Pattern definition

In order to guarantee privacy not only components that form the service should be checked for privacy but also their composition. At each layer of composition, the data union that the layer produces should be evaluated. Let us consider the composition of a service of two components, that for each *x* in *{A, B, C}*.

- OUT^X are the sets of outputs of x
- IN^X are the sets of inputs of x
- $E^X = IN^X \cup OUT^X$
- V^X and C^X are two disjoint subsets of E^X which partition it into public parts V^X and confidential parts C^X
- L is a corpus of sets that are pre-defined that expose privacy

For the privacy of the composition, the following conditions should be satisfied:

- $V^A \cap L = \varnothing$
- $V^B \cap L = \varnothing$
- $V^C \cap L = \varnothing$

7.2.2.2 Pattern specification rule

The identifiability pattern can be represented in Drools as shown below:

```
rule "Identifiability"
when
  $A: Placeholder($output_A: Activity.output)
  $B: Placeholder($output_B: Activity.output)
  $ORCH: Merge($A, $B)
  $OP: Property( propertyName == "Identifiability",
   subject == $ORCH, satisfied == false)
  $SP: PropertyPlan(propeties contains $OP)
then
 PropertyPlan newPropertyPlan = new PropertyPlan($SP);
 newPropertyPlan.removeProperty($OP);
 Property NP_A = new Property($OP, "Identifiability", $A);
 Property NP_B = new Property($OP, "Identifiability", $B);
  insert(NP_A)
 insert(NP_B)
 insert(newPropertyPlan);
end
```

7.3 Dependability

7.3.1 Reliability

7.3.1.1 Pattern definition

Dependability typically refers to the provision of expected service, towards task accomplishment in a reliable and trustworthy manner, and it entails reliability, safety, availability and security[15]. The Security concept has already covered in the Section 7.1. Therefore, in the context of this work, Dependability properties will mainly focus on reliability, fault tolerance and safety aspects.

One of the most important issues for a system designer is to validate system dependability of components as a critical condition for the design of complex network infrastructures and identify the weakest components in order to replace, redesign and find alternative solutions. System dependability properties such as reliability and availability depend on component's arrangements. Stepwise decomposition can be used to recursively build network topologies using forward or de-orchestrations using backward chaining respectively. The two basic arrangements which we are focused on are components in series and in parallel.

Definition 1. *Let C={C1,C2,...Cn} be a number of components in series and R1, R2,...,Rn be the reliability of each component, then the component composition C will have reliability r equal to:*

$$R=\prod_{k=1}^{n}(Rk)$$

Definition 2. *Let C = {C1,C2,...Cn} be a number of components in parallel and R = {R1,R2,⋯,Rn} be the reliability of each component, then the parallel component composition C will have reliability R:*

$$R=1-\prod_{k=1}^{n}(1-Rk)$$

In case of arithmetic models such as latency for availability, the following approaches can be used:

- For components in series (sequential): $A = \sum_{k=1}^{n} A_k$
- For components in parallel (multi-choice): $A = min\{A_1, A_2, \dots, A_n\}$
- For components in parallel (parallel split): $A = max\ \{A_1, A_2, \dots, A_n\}$

7.3.1.2 Pattern specification rule

Reliability pattern can be expressed as rules in Drools production rules. They encode orchestrations in Drools corresponding to the structure of the logical reliability arrangements. It also specifies rules that dictate the properties that the constituent components must have.

$$\mathrm{R(t)} = \mathrm{Prob}(\mathit{Comp}\ \text{is fully functioning in [0,t]})$$

metric to measure the Reliability of the composition.

The verification of sequential reliability can be represented in Drools as shown below:

```
1. rule "Serial Reliable Composition"
2. when
3.  $A: Placeholder($input : operation.inputs, $intData: parameters.outputs,
4.            $r1:= reliabilityValue)
5.  $B: Placeholder(parameters.inputs == $intData, $output: parameters.outputs,
6.            $r2:= reliabilityValue)
7.  $ORCH: Sequence(parameters.inputs:= $input, parameters.outputs == $output,
8.          firstActivity == $A, secondActivity == $B)
9.  $OP: Property(subject:= $ORCH, propertyName== "Reliability",
```

```
      $rel:= propertyValue, $rel<= $r1*$r2, satisfied == false)
  $SP: PropertyPlan(property contains $OP)
then
  PropertyPlan newPropertyPlan = new PropertyPlan($SP);
  newPropertyPlan.removeProperty($OP);
  Property NP_A = new Property($OP, "Reliability", $A);
  newPropertyPlan.getProperty().add(NP_A);
  insert(NP_A);
  Property NP_B = new Property($OP, "Reliability", $B);
  newPropertyPlan.getProperties().add(NP_B);
  insert(NP_B);
  insert(newPropertyPlan);
  modify($OP){satisfied=true};
end
```

7.4 Interoperability

7.4.1 Technical Interoperability

7.4.1.1 Pattern definition:

Technical Interoperability is about enabling the communication between systems and platforms at a protocol level and the infrastructure needed for those protocols to operate. Within the context of the work presented herein, the associated pattern rule aims to cover and address the technological issues that may arise from the interaction among heterogeneous devices, with different technical specifications and supported communication means on the transmission layer (e.g., wireless motes communicating via ZigBee, other motes via 802.15.4, and more powerful infrastructure devices communicating over WiFi or Ethernet), as is often the case in IoT environments [16].

Let us consider:

- C := the set of all instantiated components
- TA := A set of technical attributes
- $C_1, C_2 \subseteq C$, where $C_1 \neq C_2$
- $C_{i\_TA} \subseteq TA$:= technical attributes of C_i
- TMD := Technical Mediator (mediator which connects to components with various technical attributes; e.g., a sensor gateway that acts as a bridge between 802.15.4 radio and wired network infrastructures using 6LBR [17]).

Then, we can define the following:

Lemma 1: *If C1, C2 are at the same domain and C1_TA ∩ C2_TA ≠ ∅ then C1 and C2 are directly technically interoperable.*

Lemma 2: *If C1, C2 are on different domain but are both directly technically interoperable with TMD then C1, C2 are indirectly technically interoperable.*

Lemma 3: *If C1, C2 are directly or indirectly technical interoperable, then C1, C2 are technically interoperable.*

7.4.1.2 Pattern specification rule:

Based on the above detailed pattern, the workflow-based definition of Technical Interoperability is described below, for the fundamental scenario of two IoT components communicating with each other:

24. WF "technical-interoperability"
25. Placeholder (A1, (PlaceholderActivity, PlaceholderDescription))
26. Placeholder (A2, (PlaceholderActivity, PlaceholderDescription))
27. Placeholder (TMD, (PlaceholderActivity, "technical mediator"))
28. Link (L1, A1, A2)
29. Link (L2, A1, TMD)
30. Link (L3, A2, TMD)
31. Property (conn1, L1, required, (pattern-based, pattern),"_technical-interoperability", in_transit)
32. Property (conn2, L2, required, (pattern-based, pattern),"_technical-interoperability", in_transit)
33. Property (conn3, L3, required, (pattern-based, pattern),"_technical-interoperability", in_transit)
34. Property (conn4, "_technical-interoperability", required, (pattern-based, PR1), "_technical-interoperability", end_to_end)
35. **Pattern rule: (PR1: conn1 || (conn2, conn3) ➔ conn4)**

7.4.2 Syntactic Interoperability

7.4.2.1 Pattern definition:

Syntactic Interoperability establishes clearly defined and agreed formats for data, interfaces, and encodings[16]. This is especially challenging in the IoT domain where, while manufacturers typically try to adopt standardised messaging protocols, the plethora of such established protocols with different intrinsic characteristics (e.g., RESTful HTTP, CoAP, XMP, MQTT, DPWS) and a variety of data formats (e.g., XML, JSON), which leads to a fragmented landscape.

Let us consider:

- C := the set of all instantiated ingredients/activities in an IoT orchestration
- PR := A set of protocols
- $C_1, C_2 \subseteq C$, where $C_1 \neq C_2$
- $C_{i\_PR} \subseteq PR :=$ protocols supported by Ci
- SyMD := Syntactic Mediator (component which connects to components with various protocols, and translates between them, such as SeMIBIoT [18])

Then, we can define the following:

Lemma 1: *If C_1, C_2 are technically interoperable and $C_{1\ PR} \cap C_{2\ PR} \neq \varnothing$ then C_1 and C_2 are directly syntactically interoperable.*

Lemma 2: *If C_1, C_2 are technically interoperable and are both directly syntactical interoperable with SyMD then C_1, C_2 are indirectly syntactically interoperable*.

Lemma 3: *If C_1, C_2 are directly or indirectly syntactically interoperable, then C_1, C_2 are syntactically interoperable.*

7.4.2.2 Pattern specification rule:

Based on the above detailed pattern, the workflow-based definition of Syntactic Interoperability is described below, between two IoT activities A1, A2 interacting with each other.

1. WF "syntactic-interoperability"
2. Placeholder (A1, (PlaceholderActivity, PlaceholderDescription))
3. Placeholder (A2, (PlaceholderActivity, PlaceholderDescription))
4. Placeholder (SyMD, (PlaceholderActivity,"syntactic mediator"))
5. Link (L1, A1, A2)
6. Link (L2, A1, SyMD)
7. Link (L3, A2, SyMD)
8. Property (conn01, L1, required, (pattern-based, pattern),"_technical-interoperability", in_transit)
9. Property (conn02, L2, required, (pattern-based, pattern),"_technical-interoperability", in_transit)
10. Property (conn03, L3, required, (pattern-based, pattern),"_technical-interoperability", in_transit)
11. Property (conn1, L1, required, (pattern-based, pattern),"_syntactic-interoperability", in_transit_ ∨ in_processing)
12. Property (conn2, L2, required, (pattern-based, pattern),"_syntactic-interoperability", in_transit_ ∨ in_processing)
13. Property (conn3, L3, required, (pattern-based, pattern),"_syntactic-interoperability", in_transit_ ∨ in_processing)
14. Property (conn4, "_syntactic-interoperability", required, (pattern-based, PR1), "_syntactic-interoperability", end_to_end)
15. **Pattern rule: (PR1: (conn01,conn1) || (conn02,conn2,con03,conn3) ➔ conn4)**

7.4.3 Semantic Interoperability

7.4.3.1 Pattern definition:

Semantic Interoperability settles commonly agreed information models and ontologies for the used terms that are processed by the interfaces or are included in the exchanged data [16]. For example, temperature units can be Fahrenheit, Celsius or Kelvin, but they express the same information which can be obtained after proper instance transformation.

Let us consider:

- C := the set of all instantiated components
- MDL := A set of semantic models
- $C_1, C_2 \subseteq C$, where $C_1 \neq C_2$
- $C_{i\_MDL} \subseteq$ MDL := semantic models used by Ci
- SeMD := Semantic Mediator; e.g., a Semantic Mediator.

Then, we can define the following:

Lemma 1: *If C_1, C_2 are syntactically interoperable and $C_{1\_MDL} \cap C_{2\_MDL} \neq \varnothing$ then C_1 and C_2 are directly semantically interoperable*

Lemma 2: *If C_1, C_2 are syntactically interoperable and are both directly semantically interoperable with SeMD, then C_1, C_2 are indirectly semantically interoperable*

Lemma 3: *If C_1, C_2 are directly or indirectly semantically interoperable, then C_1, C_2 are semantically interoperable.*

7.4.3.2 Pattern specification rule:

Based on the above detailed pattern, the workflow-based definition of Semantic Interoperability is described below, between two IoT activities A1, A2 interacting with each other.

1. WF "semantic-interoperability"
2. Placeholder (A1, (PlaceholderActivity, PlaceholderDescription))
3. Placeholder (A2, (PlaceholderActivity, PlaceholderDescription))
4. Placeholder (SeMD, (PlaceholderActivity,"Semantic Broker"))
5. Link (L1, A1, A2)
6. Link (L2, A1, SeMD)
7. Link (L3, A2, SeMD)
8. Property (conn01, L1, required, (pattern-based, pattern),"_syntactic-interoperability" , in_transit_ ∨ in_processing)
9. Property (conn02, L2, required, (pattern-based, pattern),"_syntactic-interoperability" , in_transit_ ∨ in_processing)
10. Property (conn03, L3, required, (pattern-based, pattern),"_syntactic-interoperability" , in_transit_ ∨ in_processing)
11. Property (conn1, L1, required, (pattern-based, pattern),"_semantic-interoperability" , in_processing)
12. Property (conn2, L2, required, (pattern-based, pattern),"_semantic -interoperability" , in_processing)
13. Property (conn3, L3, required, (pattern-based, pattern),"_semantic -interoperability" , in_processing)
14. Property (conn4, "_semantic-interoperability", required, (pattern-based, PR1)," "_semantic-interoperability", end_to_end)
15. **Pattern rule: (PR1: (conn01,conn1) || (conn02,conn2,conn03,conn3) ➔ conn4)**

7.4.4 Organisational Interoperability

7.4.4.1 Pattern definition:

The Organisational Interoperability supports cross-domain service integration and orchestration through common semantics and programming interfaces [16].

Let us consider:

– C := the set of all instantiated IoT platform deployments
– CSPI := A set of common semantic and programming interfaces
– C_1,$C_2 \subseteq C$, where $C_1 \neq C_2$
– $C_{i\_CSPI} \subseteq$ CSPI := common semantic and programming interfaces supported by Ci
– IP := Integration Proxy; i.e., a proxy, broker or middleware.

Then, we can define the following:

Lemma 1: *If C_1, C_2 are semantically interoperable and $C_{1\_CSPI} \cap C_{2\_CSPI} \neq \varnothing$ then C_1 and C_2 are directly organisationally interoperable*

Lemma 2: *If C_1, C_2 are semantically interoperable and are both directly organisationally interoperable with IP, then C_1, C_2 are indirectly organisationally interoperable*

Lemma 3: *If C_1, C_2 are directly or indirectly organisationally interoperable, then C_1, C_2 are organisationally interoperable.*

7.4.4.2 Pattern specification rule:

Based on the above detailed pattern, the workflow-based definition of Semantic Interoperability is described below, between two IoT platform deployments A1, A2 interacting with each other.

1. WF "organisational-interoperability"
2. Placeholder (A1, (PlaceholderActivity, PlaceholderDescription))
3. Placeholder (A2, (PlaceholderActivity, PlaceholderDescription))
4. Placeholder (IP, (PlaceholderActivity,"Integration Proxy"))
5. Link (L1, A1, A2)
6. Link (L2, A1, IP)
7. Link (L3, A2, IP)
8. Property (conn01, L1, required, (pattern-based, pattern),"_semantic -interoperability" , in_processing)
9. Property (conn02, L2, required, (pattern-based, pattern),"_semantic -interoperability" , in_processing)
10. Property (conn03, L3, required, (pattern-based, pattern),"_semantic -interoperability" , in_processing)
11. Property (conn1, L1, required, (pattern-based, pattern),"_organisational-interoperability" , in_transit_ ∨ in_processing)
12. Property (conn2, L2, required, (pattern-based, pattern),"_organisational-interoperability" , in_transit_ ∨ in_processing)
13. Property (conn3, L3, required, (pattern-based, pattern),"_organisational -interoperability" , in_transit_ ∨ in_processing)
14. Property (conn4, "_organisational-interoperability", required, (pattern-based, PR1)," semantic-interoperability", end_to_end)
15. **Pattern rule: (PR1: (conn01,conn1) || (conn02,conn2,conn03,conn3) ➔ conn4)**

8 Conclusions

This document defines LCA-CSPDI requirements and patterns for circular economy business models & supply chains for the CE-IoT project. In particular, this document:

- Provides domains that patterns can be applied based on case studies as identified during the CE-IoT project
- Describes the LCA-CSPDI property requirements that are required by the respective patterns
- Identifies a number of domain specific requirements based on use cases in which the different properties are preserved
- Definition of the pattern language including the semantics in order to satisfy the LCA-CSPDI properties
- A first set of CSPDI patterns is also provided, defining them from both a formal, workflow-based perspective, as well as their machine-processable representation in Drools.

References

[1] N. E. Petroulakis, G. Spanoudakis, and I. G. Askoxylakis, "Patterns for the design of secure and dependable software defined networks," *Comput. Networks*, vol. 109, pp. 39–49, Nov. 2016.

[2] K. M. L. Pino, G. Spanoudakis, S. Gürgens, A. Fuchs, "D02.2 - ASSERT aware service orchestration patterns. www.assert4soa.eu.".

[3] G. S. Pino L., Spanoudakis G., Fuchs A., "Generating Secure Service Compositions, In Communications in Computer and Information Sciences," *Springer*, 2015.

[4] V. Issarny *et al.*, "Service-oriented middleware for the Future Internet: State of the art and research directions," *J. Internet Serv. Appl.*, vol. 2, no. 1, pp. 23–45, 2011.

[5] A. Lukacs, "What is Privacy? The History and Definition of Privacy," *Keresztes, Gábor Tavaszi Szél 2016 Tanulmánykötet I., Budapest, Doktoranduszok Országos Szövetsége*.

[6] M. F. Dennedy, J. Fox, and T. R. Finneran, *The privacy engineer's manifesto: Getting from policy to code to QA to value*. Apress Media LLC, 2014.

[7] I. C. Office, "Anonymisation: Managing Data Protection Risk, Code of Practice." Inf. Comm. Off., p. 106, 2012.

[8] P. Ohm, "Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization," *UCLA Law Rev.*, vol. 57, pp. 1701–1777, 2010.

[9] J.-C. Laprie, "Dependable Computing and Fault Tolerance," Digest of Papers FTCS-15, 1987.

[10] A. Geraci, F. Katki, L. McMonegal, B. Meyer, and H. Porteous, *IEEE Standard Computer Dictionary. A Compilation of IEEE Standard Computer Glossaries*. 1991.

[11] P. Park, P. Di Marco, C. Fischione, and K. H. Johansson, "Modeling and optimization of the IEEE 802.15.4 protocol for reliable and timely communications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 550–564, 2013.

[12] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artif. Intell.*, vol. 19, no. 1, pp. 17–37, Sep. 1982.

[13] D. E. Denning, "A lattice model of secure information flow," *Commun. ACM*, vol. 19, no. 5, pp. 236–243, May 1976.

[14] A. Zakinthinos and E. S. Lee, "General theory of security properties," in *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 1997.

[15] J. C. Laprie, "Dependability: Basic Concepts and Terminology," Springer, Vienna, 1992, pp. 3–245.

[16] G. Hatzivasilis *et al.*, "The Interoperability of Things: Interoperable solutions as an enabler for IoT and Web 3.0," in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, 2018, vol. 2018-Septe.

[17] "CETIC 6LBR, 6LoWPAN/RPL Border Router solution." [Online]. Available: https://github.com/cetic/6lbr/wiki. [Accessed: 14-Feb-2020].

[18] E. Palavras, K. Fysarakis, I. Papaefstathiou, and I. Askoxylakis, "SeMIBIoT: Secure multi-protocol integration bridge for the IoT," in *IEEE International Conference on Communications*, 2018, vol. 2018-May.