Funded by the
European Union

D4.2 CLARITY CSIS Architecture

WP4 - Technology Support

Deliverable Lead: CIS

Dissemination Level: Public

Deliverable due date: 31/05/2018

Actual submission date: 07/02/2019

Version 1.0.2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730355

Document Control Page	
Title	D4.2 CLARITY CSIS Architecture
Creator	Pascal Dihé (CIS)
Description	This deliverable will describe the CLARITY CSIS Architecture. It will be updated later in the project if needed.
Publisher	CLARITY Consortium
Contributors	Pascal Dihé
Creation date	27/10/2017
Type	Text
Language	en-GB
Rights	copyright “CLARITY Consortium”
Audience	☒ Public ☐ Confidential ☐ Classified
Status	☐ In Progress ☐ For Review ☐ For Approval ☒ Approved

Disclaimer

Disclaimer

Table of Contents

List of Figures

List of Tables

CLARITY Project Overview

Urban areas and traffic infrastructures that are linking such areas are highly vulnerable to climate change. Smart use of existing climate intelligence can increase urban resilience and generate benefits for businesses and society at large. Based on the results of FP7 climate change, future internet and crisis preparedness projects (SUDPLAN, ENVIROFI, CRISMA) with an average TRL of 4-5 and following an agile and user-centred design process, end-users, purveyors and providers of climate intelligence will co-create an integrated Climate Services Information System (CSIS) to integrate resilience into urban infrastructure.

As a result, CLARITY will provide an operational eco-system of cloud-based climate services to calculate and present the expected effects of CC-induced and -amplified hazards at the level of risk, vulnerability and impact functions. CLARITY will offer what-if decision support functions to investigate the effects of adaptation measures and risk reduction options in the specific project context and allow the comparison of alternative strategies. Four Demonstration Cases will showcase CLARITY climate services in different climatic, regional, infrastructure and hazard contexts in Italy, Sweden, Austria and Spain; focusing on the planning and implementation of urban infrastructure development projects.

CLARITY will provide the practical means to include the effects of CC hazards and possible adaptation and risk management strategies into planning and implementation of such projects, focusing on increasing CC resilience. Decision makers involved in these projects will be empowered to perform climate proof and adaptive planning of adaptation and risk reduction options.

Abbreviations and Glossary

A common glossary of terms for all CLARITY deliverables, as well as a list of abbreviations, can be found in the public document "CLARITY Glossary" available at http://cat.clarityCLARITY-h2020.eu/glossary/main.

Abbreviation/Acronym	Definition
ADM	Architecture Development Method
AJAX	Asynchronous JavaScript and XML
ASE	Agile Software Engineering
BB	Building Block
BDUF	Big Design Up Front
CBS	Component-Based Software
CC	Climate Change
CCA	Climate Change Adaptation
CKAN	Comprehensive Kerbal Archive Network
CLARITY	Integrated Climate Adaptation Service Tools for Improving Resilience Measure
CRISMA	Modelling crisis management for improved action and preparedness
CS	Climate Service
CSIS	CLARITY Climate Services Information System
CSW	Catalogue Service for the Web
CTA	Constructive Technology Assessment
DC	Demonstration Case
DC	Dublin Core
DoA	Description of Action (Annex 1 to the Grant Agreement)
DRR	Disaster Risk Reduction
EC	European Commission
ERDDAP	Environmental Research Division's Data Access Program
EU-GL	Non-paper Guidelines for Project Managers: Making vulnerable investments climate resilient (Document)
EU-MACS	European Market for Climate Services
GeoJSON	geographical JavaScript Object Notation
GeoTIFF	Geographic Tagged Image File Format
GFCS	Global Framework for Climate Services
GML	Geography Markup Language
GPS	Global Positioning System
HTML5	Hypertext Markup Language, version 5

HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technologies
IPCC	Intergovernmental Panel on Climate Change
JSON	JavaScript Object Notation
KISS	Keep it simple, stupid (agile concept)
LDUF	Lean Design Up Front
MCDA	Multi-Criteria Decision Analysis
NGO	Non-Governmental Organization
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OGR	OpenGIS Simple Features Reference Implementation
PDF	Portable Document Format
PHP	PHP Hypertext Preprocessor
RDBMS	Relational Database Management System
REST	Representational State Transfer
RIA	Rich Internet Application
RM-ODP	Reference Model of Open Distributed Processing
SMS	Scenario Management System
SOA	Service Oriented Architecture
SOS	Sensor Observation Service
SPA	Single Page Application
SPS	Sensor Planning Service
SQL	Structured Query Language
SUDPLAN	Sustainable Urban Development Planner for Climate Change Adaptation
TC	Test Case
TOGAF	The Open Group Architecture Framework
TRL	Technology Readiness Level
US	User Story
WFS	Web Feature Service
WMS	Web Map Service
WMTS	Web Map Tile Service
WP	Work Package
YAGNI	You aren't going to need it (agile concept)

Executive Summary

This report is the second deliverable of WP4 “Technology Support” of the CLARITY project, funded by the EU’s Horizon 2020 Programme under Grant Agreement number 730355. WP4 intends to provide the technological backbone of the CLARITY Climate Service Information System (CSIS) by tailoring the technological background foreseen in the CLARITY work package descriptions to project needs. For this, WP4 will integrate and adapt all required and existing (background) tools and services that are necessary for realisation of the CLARITY reference scenarios (Demonstration Cases) and implementation of the EU-GL [1] into the CLARITY Climate Services.

The main aim of this deliverable is to describe the CSIS Architecture in such a concise and simple manner so that its goals and major concepts can be understood by all stakeholders (including the end users) involved in the co-creation process. It does this by communicating the most significant design decisions that shape CSIS and equips the agile development teams with "just enough" conceptual and technical knowledge to successfully implement the presented Conceptual Innovation Design.

Unlike as initially foreseen in the DoA, the CSIS Architecture follows an agile and emergent approach that aims to quickly respond to unavoidable changes imposed by the agile co-creation approach of WP1 "Co-Creation". Moreover, technology moves fast and many of the software components and technologies mentioned in the DoA are outdated or do not suit the emergent use cases and requirements introduced during the first year of the project. The impact to project plans with respect to tasks, deliverables, resources requested etc., however, are minimal and do not collide with general project objectives.

1 Introduction

The introduction chapter defines the purpose and scope of the CSIS Architecture and briefly explains the structure of the document.

1.1 Purpose of this document

The main goal of this document is to establish a shared understanding among all CLARITY stakeholders about the overall goals of the CSIS Architecture and the essential design decisions and architectural principles to realise these goals. It furthermore intends to equip the CLARITY co-creation teams with the necessary conceptual background information to successfully implement and carry out the agile development process. Is does not intend to deliver formal and heavy upfront specifications and a detailed plan that tries to consider all possibilities and cast the CSIS Architecture in stone. Instead, it explains how state-of-the art architectural design principles are applied in agile software development to create a robust *and* flexible software architecture. It then applies those principles in order to specify the overall mission, the essential concepts and realisation of the CSIS Architecture.

1.2 Intended audience

The target readers of this document are all members of the CLARITY consortium as they cover all categories of stakeholders (end users, service suppliers, developers, etc.) of the CSIS.

1.3 Structure of the document

The structure of the document and the relationships between the different chapters is as follows:

Chapter 1 (this chapter) introduces the document and explains the overall purpose of this document and its relation to other work packages and deliverables.

Chapter 2 presents CLARITY's concept towards the CSIS Architecture for using methods of traditional software architecture in agile development as well as applying agile methods in traditional software architecture and design.

Chapter 3 defines the general mission of the CLARITY CSIS in terms of goals, architectural qualities and constraints.

Chapter 4 defines the core concepts applied in the CLARITY CSIS Architecture in terms of conceptual specification of CLARITY products and services (Innovation Design) and the general principles that are used to design and implement the CSIS.

Chapter 5 briefly explains how and with help of which components the architectural concepts and principles introduced chapter 4 are applied to realise the goals formulates in chapter 3.

Chapter 6 briefly summaries the artefacts that serve for documenting the Emergent Architecture and explains where these artefacts can be found online or how and when they will be made publicly available.

Chapter 7 provides the conclusions and a summary on follow-up activities in other work packages.

2 An Architecture for Agile Software Development

CLARITY intends to follow an agile and user-centred design process, which makes sure that the inputs from other work packages as well as the stakeholders' needs as expressed in User Stories are incorporated into the CLARITY CSIS Architecture. However, there are several challenges to be faced when combing traditional software architecture and software engineering methodologies with agile approaches. Interestingly, "Architecture and agile - how much design is enough for different classes of problem?" has been considered as one of the top 10 burning research questions from software practitioners [2].

This chapter discusses therefore how established architectural methods and approaches differ from agile methods and practices and presents a concept towards the CSIS Architecture for using methods of traditional software architecture in agile development as well as applying agile methods in traditional software architecture and design.

2.1 Traditional Software Architecture and Agile Software Development

The international standard for architecture descriptions of systems and software (ISO/IEC 42010) defines architecture as the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution." [3] This somewhat implies that an architecture should try to design the whole system upfront in a rather prescriptive manner before any real implementation activities take place. In fact, highly regarded architectural frameworks like the Open Group Architecture Framework (TOGAF) and the Reference Model of Open Distributed Processing (RM-ODP) [4] *can* be applied to formally specify a complex system in its entirety (Figure 1) before handing over those specifications to the development teams.

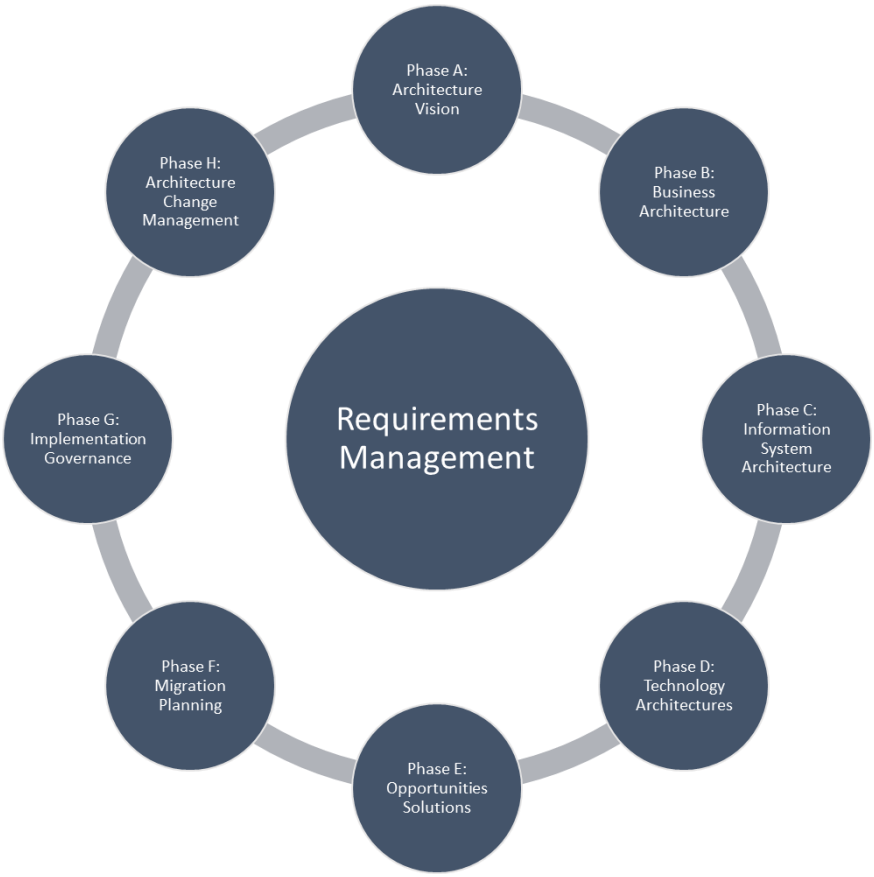

Figure 1: TOGAF Architecture Development Method (ADM) Lifecycle [5]

This often goes hand in hand with a sequential and process driven design like the waterfall model[1] that follows a linear top-down approach. Although such a "traditional" architectural design can be complemented with iterative and flexible approaches following architectural quality attributes like "design for change", it “tends to ends to embrace (software) engineering concerns too strongly and too early.” [6]

This means that such an architecture does “not only prescribe the structure of the system being developed” but that this structure also “becomes engraved in the structure of the development”. [7]. Accordingly, it tends to allow incremental changes to existing structures only and often considers any (large) change disruptive for the overall system design. Moreover, traditional architecture and software engineering, respectively, is heavily documentation-focused and requires architects as wells as developers to strictly follow established methods, rules, tools, formalisms, and notations.

In contrast to that, agile software development prefers simplicity, efficiency and continuous delivery of working software over detailed forward planning and exhaustive specification and documentation work. It attaches great importance to the dialogue between customer and developer. Thereby, it considers changing requirements, even in a late stage of the project, as opportunity to generate value for the customer instead of a disruptive factor that needs corrective action and change management. The Agile Manifesto [7], which has been declared by a group of leading software developers in 2001, defines four values (Figure 2) and twelve principles for agile software development.

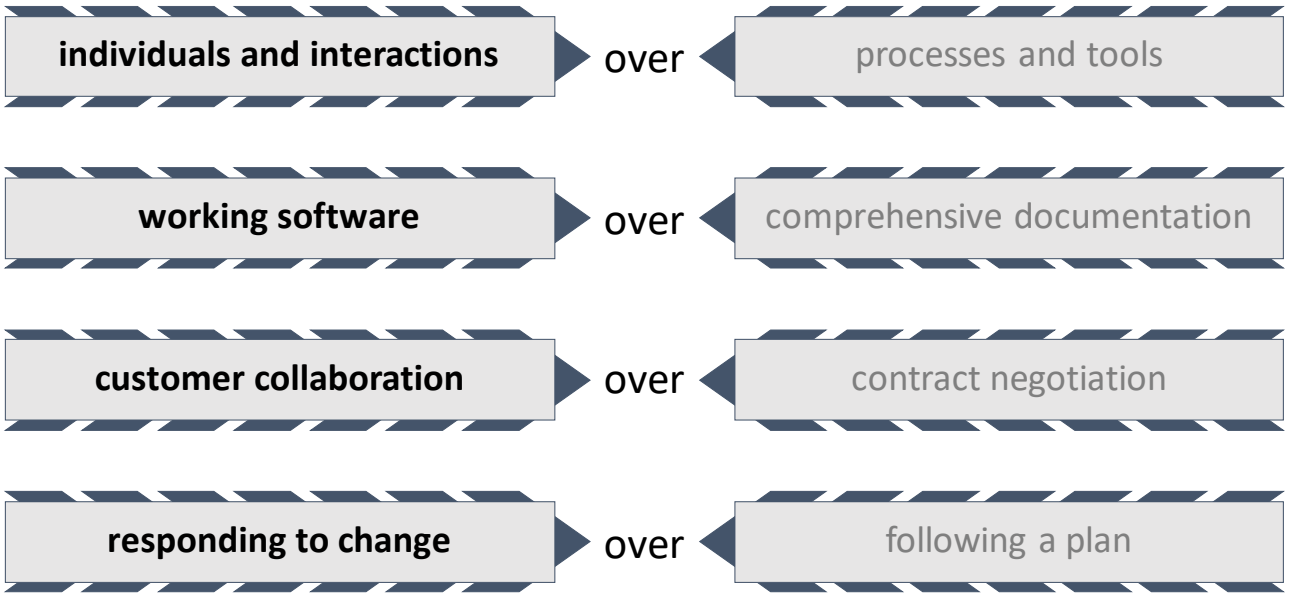

Figure 2: Values of the Manifesto for Agile Software Development [8]

Agile also embraces the ideas of lean production [9] to reduce "waste" and only doing activities or creating software that directly generates value. The main difference to traditional architecture is therefore how forward planning and upfront specification is valued: “Architecture design represents a plan for the system development, while agile development embraces change, and pays less attention to plans.” [10]

[1] The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. (https://economictimes.indiatimes.com/definition/waterfall-model)

To this effect, traditional software engineering and agile software development are likely to be seen as being mutually exclusive by proponents of either approach. Likewise, “advocates of architecture’s vital role in achieving quality goals for large software-intensive systems doubt the scalability of any development approach that doesn’t pay sufficient attention to architecture” while “proponents of agile approaches usually see little value for a system’s customers in the upfront design and evaluation of architecture.” [11]

However, when objectively comparing traditional software architecture/engineering with agile software development by considering their similarities and differences, the “two approaches are not seen to be incompatible, leading to the future possibility of an Agile Software Engineering (ASE)” [12]. In particular, traditional software engineering is not incompatible with agile values (Figure 2) and principles per se. Likewise; agile software development itself makes use of a number of traditional software engineering techniques and is not strictly against forward engineering and modelling as long as they create recognizable value. The differences are more subtle as depicted in Figure 3. “In the end it seems that there is nothing really incompatible with applying all the principles and values of agile software development, along with most (if not all) of the practices, to traditional software engineering.” [12]

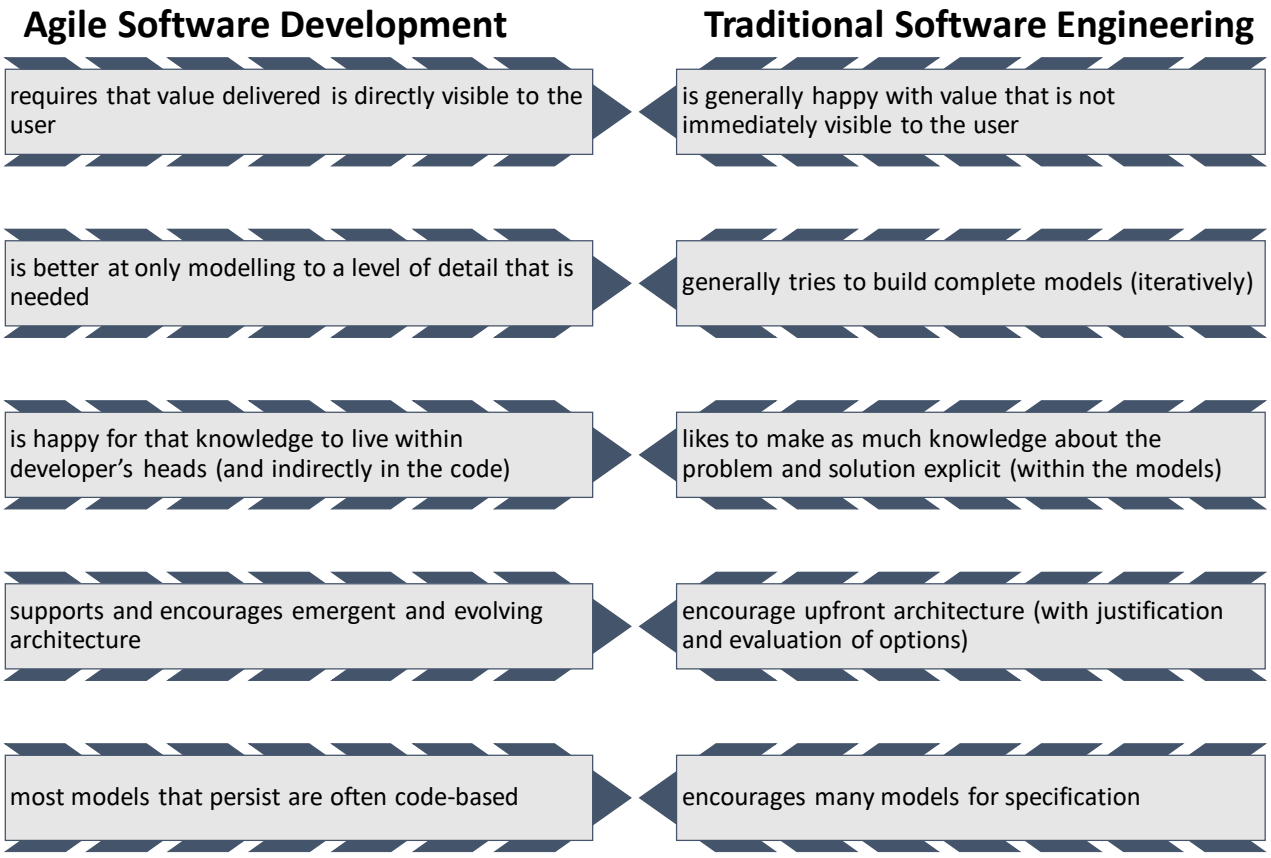

Figure 3: Comparative Analysis of Traditional Software Engineering and Agile Software Development [12]

Thus, if methodologies and patterns of traditional software engineering are applied in a lean manner, that is, to right level of detail and at the right time, they can support agile software development. However, “neither Lean nor Agile alone make architecture look easy. […] Together they illuminate architecture’s value: Lean, for how architecture can reduce waste, inconsistency, and irregular development; and Agile, for how end user engagement and feedback can drive down long-term cost.” [6].

2.2 Explicit and Emergent Architecture

Unfortunately, there is no one-size-fits-all solution, universal recipe to follow or even reference model available that helps to strike the right balance between traditional software engineering and agile methods. It is therefore necessary to take the particular problem domain and project context into account for choosing the appropriate architectural methods. In combination with the “three common sense principles of agile thinking” (Figure 4), a modern, lightweight and pragmatic approach to software architecture can then be established.

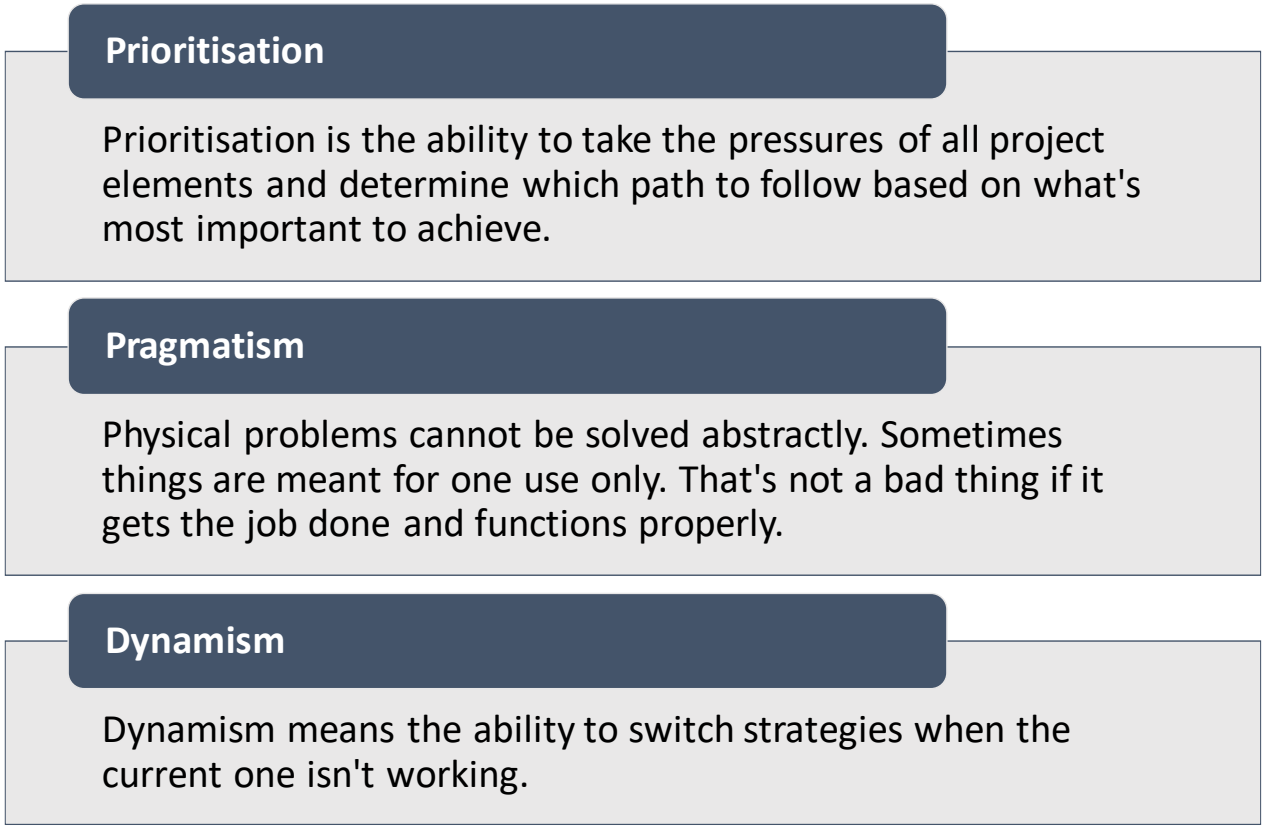

Figure 4: Three Common Sense Principles of Agile Thinking [13]

Prioritisation and pragmatism help to find the right amount of up-front design needed to stabilise the architecture and thus to reduce later rework during the dynamic and reactive development process. It is therefore crucial to identify and specify the most significant design decisions that shape the overall architecture, where “significant is measured by cost of change and by impact upon use.” [14]. Those crucial design decisions and concepts have to stay valid throughout the whole project lifecycle and are reflected in the actual product developments. In contrast to the traditional software engineering practice of doing "Big Design Up Front" (BDUF), aiming at a complete and "perfect" architecture specification, a lean and agile architectural approach has to perform "Lean Design Up Front" (LDUF). Such explicit lean upfront design avoids to make predictions about the unknown (producing waste) while at the same time focussing on those aspects that have the most impact (generating value). Thereby, the key message is that the architecture “should not over-anticipate emergent needs, delaying delivery of user value and risking development of overly complex and unneeded architectural constructs. At the same time, it should not under-anticipate future needs, risking feature development in the absence of architectural guidance and support.” [15] Thereby, it is essential that such an Explicit or “Intentional Architecture” [16] is “designed for understandability and change” [17] in order to be able to manage complexity and changeability.

Design for understandability involves mainly keeping the explicit architecture specification concise and simple so that its major concepts and design decisions can be understood by all stakeholders (including the end users) involved in the agile product development process. Design for change means to “distil the direction of change from time to time (depending on the change rate of the system environment) to optimize the design decisions.” [17]

Design for change is closely related to dynamism and demands for an architectural approach that can cope not only with evolving and changing requirements but also with varying technology choices and implementation strategies. Likewise, the Agile Manifesto states “the best architectures, requirements, and designs emerge from self-organizing teams” [8], which means that the “architecture emerges as a natural outcome of a rapid iteration cycle, implementation of prioritised value-driven user requests and a continuous re-factoring process.” [16] While this concept of "Emergent Architecture" fits perfect into the “just enough, just in time” [18] principle of incrementally evolving software systems, “as a better understanding of market needs emerges, continuously refactoring large-scale, emerging architectures becomes less practical as the size of the system grows” [18].

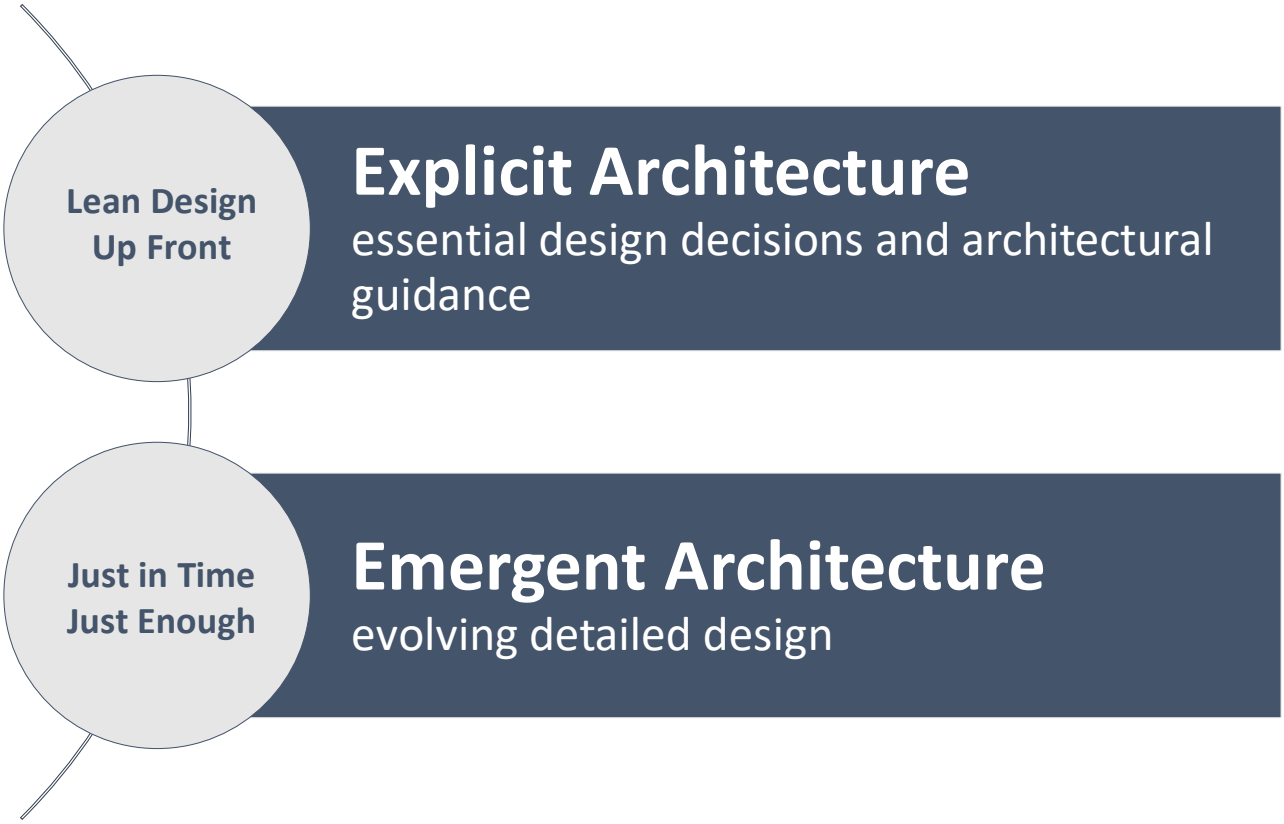

Figure 5: Explicit and Emergent Architecture

To achieve architectural agility without neglecting necessary architectural design decisions it is therefore required, “to design the essence of the system explicitly and let the rest evolve using emergent architecture - potentially guided by some architectural constraints to avoid duplicate solution designs.” [17]. This leads to the conclusion, that for the agile development of an enterprise class system like the CLARITY CIS, besides an Emergent Architecture that serves the purpose of documenting and communicating the evolving detailed design, an Explicit Architecture which captures the essential design decisions and provides architectural guidance is needed (Figure 5).

Table **1** provides a comparison of the main properties of Explicit and Emergent Architecture that have gathered from various literature sources.

Table 1: Comparison of Explicit and Emergent Architecture

Explicit Architecture	Emergent Architecture
represents the highest level decomposition of the system	represents the detailed design of the system
defines and explains essential design decision	defers decisions until the last responsible moment
Lean Design Up Front: setting essential architectural design at the starting phase of the project	just enough, just in time: considering only essential features needed for the current iteration
high cost of change	no or low cost of change
traditional architectural methods are applied in a lean and agile fashion	agile software development is guided by explicit governance
is defined explicitly during initial stakeholder workshops	emerges implicitly during iterative development
co-created by all stakeholders and conceptualised by system architects	created and conceptualised by self-organized product implementation teams
architectural information provided as static layer diagrams and lightweight and understandable documentation	architectural information provided as Mock-Ups, in test cases specifications, issue tracking system or embedded in code
validated by the successful product implementation and thus the emergent architecture	validated by stakeholders by means of unit tests, acceptance tests, etc.
minimises project risk by offering a shared understanding of the high-level system design and the domain context	minimises up-front architecture design cost by avoiding unnecessary complexity and "overengineering"
focus on non-functional requirements	focus on functional requirements

The next two sections of this chapter explain how the aforementioned concepts are applied to the CSIS Architecture.

2.3 Modelling and documenting the CSIS Architecture

In agile development architects and other stakeholders often encounter difficulties related to very long, complex, and not self-explanatory architecture specification documents that "requires significant effort to review and maintain throughout the development lifecycle" [19]. In CLARITY, we follow therefore an approach towards a lean and self-explanatory architectural documentation that is easier to review, update, and communicate. A major concept of this approach is the separation of the CSIS Architecture into an Explicit Architecture and an Emergent Architecture as described in section 2.2.

In the Explicit Architecture we document those architecturally significant design decisions that bear the most impact and cost of change and thus have been made early before the actual product implementation. To guide the feature implementing process in each agile iteration, we introduce common architectural patterns, design constraints and general implementation technologies.

We present a high-level solution design that facilitates common understanding and collaboration among all stakeholders by connecting business and domain models with a shared "Product Vision". Such high-level abstractions will not only increase understandability of the overall system but also support changeability.

The Explicit Architecture will explain “how the system is divided into components and how the components interact through interfaces.” [20]. Thereby, it will only include those logical components (Building Blocks) that are understood by all stakeholders, leaving their concretisation and technical and implementation details to the Emergent Architecture. This logical view on the CSIS Architecture will be presented as simple layer diagram, which organises the different Building Blocks of the CSIS into logical, abstract groups (layers). However, instead of trying to make predictions about the unknown, which may lead to overspecification or compromised design, we defer all non-critical design decision and technology choices. Thus, we can reduce time and effort for specifying and communicating architecture by concentrating on "essential complexity"[2] avoiding "accidental complexity"[3] [21] or "dynamic complexity"[4] [22] in the first place with help of lean and agile principles like KISS ("Keep it simple, stupid") and YAGNI ("You aren't going to need it"). In CLARITY, the Explicit Architecture is represented by this deliverable D4.2 "CSIS Architecture" and corresponds to the shared understanding of the high-level system design and the domain context.

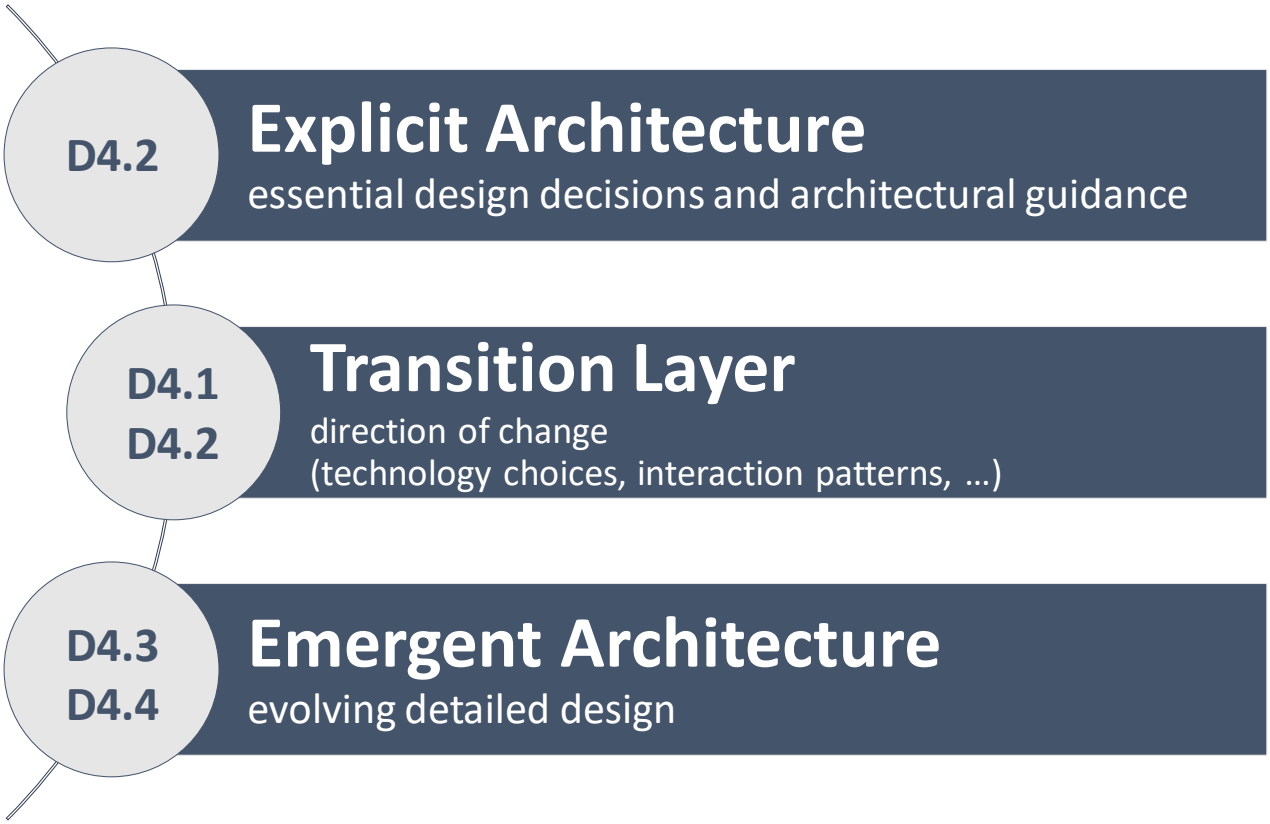

Figure 6: Transition between Explicit and Emergent Architecture

[2] “Essential complexity is the part of your software development that is essential and inherent to the problem and which can’t be reduced. Essential complexity is often the business and customer value of your problem“ [37]

[3] “Accidental complexity is determined by external factors from your environment not inherent in the problem. It is often driven by existing features, requirements and regulations and how they have been implemented.” [37]

[4] “Dynamic complexity is something that is produced (often at a moment we do not expect). It is formed through interactions, interdependencies, feedbacks, locks, conflicts, conventions, prioritisations, enforcements, etc.” [41]

To anticipate non-disruptive changes and corrections in the shared vision of the CSIS and its interacting logical components and to bridge the gap between the conceptual and implementation levels of the overall co-development process, we introduce a Transition Layer between both architectural perspectives (Figure 6).

In the Emergent Architecture, the detailed design as well as the related documentation emerges implicitly during iterative development while we consider only essential features needed for the current iteration and defer decisions until “the last responsible moment” [23]. Thereby, we reduce the amount of explicit architectural work and “address the documentation problem by shifting from high-overhead artefacts such as comprehensive UML documents to zero-overhead documentation such as API (Application Programing Interface) specifications” [6]. For the documentation of both implementation-level design decisions as well as user-interface-level and service-level contracts we re-use artefacts that emerge during the co-development process, including Test Case Specifications, Mock-Ups, OpenAPI Specifications (OAS)[5], source code documentation and so forth. This "zero overhead" approach requires no additional explicit architectural work when properly carried out by development teams.

We use lightweight architecture diagrams to document and communicate the evolving software architecture of the CSIS. Thereby, we apply the “Software architecture as code” paradigm [24]. This approach allows us to use the continuous delivery platform introduced in D1.1 "Initial Workshops and the CLARITY Development Environment" [25] not only for the development of industrial-quality code but also to for the "development" of the software architecture.

2.4 Towards the CSIS Architecture

D4.1 – "Technology Support Plan" [26] described in chapter 2 "CSIS Architecture, approach and results" the initial common and concerted methodological approach pursued by WP1 "Co-Creation" and WP4 "Technology Supports" towards the CSIS Architecture, which is still valid in the main. This approach identified and described key artefacts (formally named "concepts") and their relation to the CLARITY product development phases and the co-creation processes. D4.1 "Technology Support Plan" [26] furthermore suggested continuously improving and adapting the technology support plan to meet new demands arising not only from stakeholder requirements but also from business and market conditions.

With the introduction of the Explicit- and the Emergent Architecture, and, in particular the Transition Layer between these architectural perspectives, CLARITY architecture team has found an elegant way for anticipating expected changes (knowing the “direction of change” [17]) while preserving the invariant “essence of the system” [17]. To ensure conceptual consistency and foster understanding among stakeholders, D4.2 furthermore provides an update of the initial methodological approach in the context of the architectural perspectives. Part of this update is also an alignment of the key artefacts’ definitions to establish a "ubiquitous language"[6] for communicating the architecture.

[5] “The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for REST APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic.” [36]

[6] "ubiquitous language" is a key concept of "Domain Driven Design" [43]. Put simple, it strives to facilitate stakeholder communication (end users as well as developers) in a given (business) domain by establishing a commonly agreed terminology.

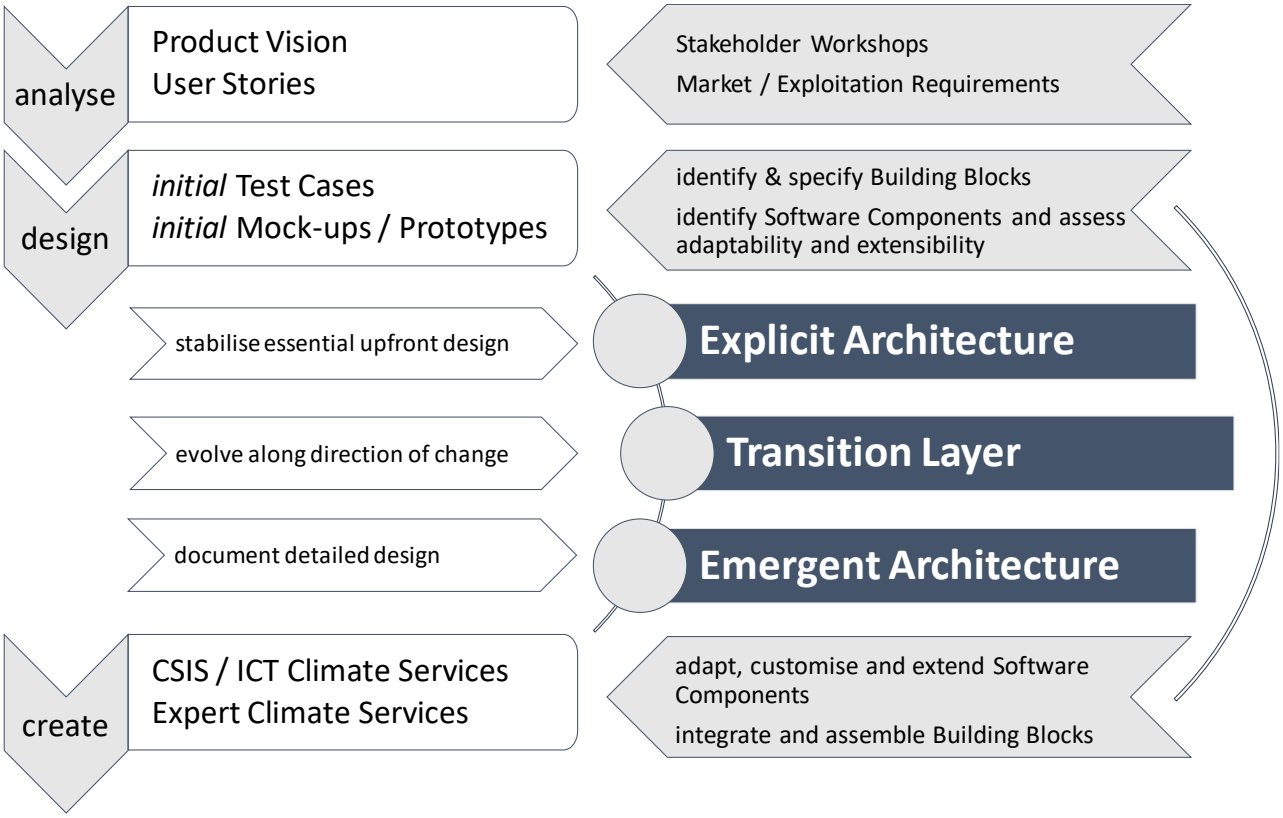

Figure 7: CLARITY Product Development Phases and Relation to CSIS Architecture

Figure 7 shows how the different product development phases relate to the architectural perspectives and the key artefacts of the CSIS Architecture. After an initial analysis and design phase, CLARITY gathered enough insights to stabilise the essential upfront design ("highest impact and cost of change" [14]) which corresponds to the Explicit Architecture of the CSIS. The main input for developing Explicit Architecture comes thereby from

a) The results of the project-internal discussions in the first six project months and the resulting common "Product Vision" and "User Stories" of the four CLARITY Demonstration Cases that were developed in this period and documented in D1.1 "Initial Workshops and the CLARITY Development Environment" [25] and D1.2 "Database of Initial CLARITY CSIS User Stories and Test Cases" [27];

b) the methodology of the "Non-paper Guidelines for Project Managers: Making vulnerable investments climate resilient" [1] (EU-GL), that has been refined and improved according to the IPCC-AR5 framework [27] in the D3.1 "Science Support Plan and Concept" [28];

c) the baseline requirements elicitation and the assessment process of available Test Cases and Exploitation Requirements (D5.1 "Exploitation Requirements and Innovation Design" [28]) that have yielded to functional requirements on Buildings Blocks documented in D4.1 "Technology Support Plan" [26];

d) the initial Mock-Up activities that translated functional requirements into user interface design studies for discussion and evaluation with end users; and

e) the evaluation of "spike solutions"[7] based on the technological offer (technological possibilities and the related open-source frontend and backend software components) presented in D4.1 "Technology Support Plan" [26] and the DoA.

Interestingly, the key artefacts Mock-Ups and Test Cases produced during the initial analyse and design phase represent the first iteration of the Emergent Architecture (Figure 8). The actual product development iterations of the subsequent create and design phases that consist of the provision of technological (IT) support and the actual (co-)creation of Expert and ICT Climate Services will then contribute to the evolvement of the Emergent Architecture. The Transition Layer ensures thereby that certain high-level abstractions of the Explicit Architecture like the system context model and the component interacting model can evolve in the same manner without interfering with essential assumptions made in the Explicit Architecture. The role of the Transition Layer can also be seen as keeping the shared understudying of all stakeholders aligned with the rather technical and near-to-development viewpoint of the Emergent Architecture.

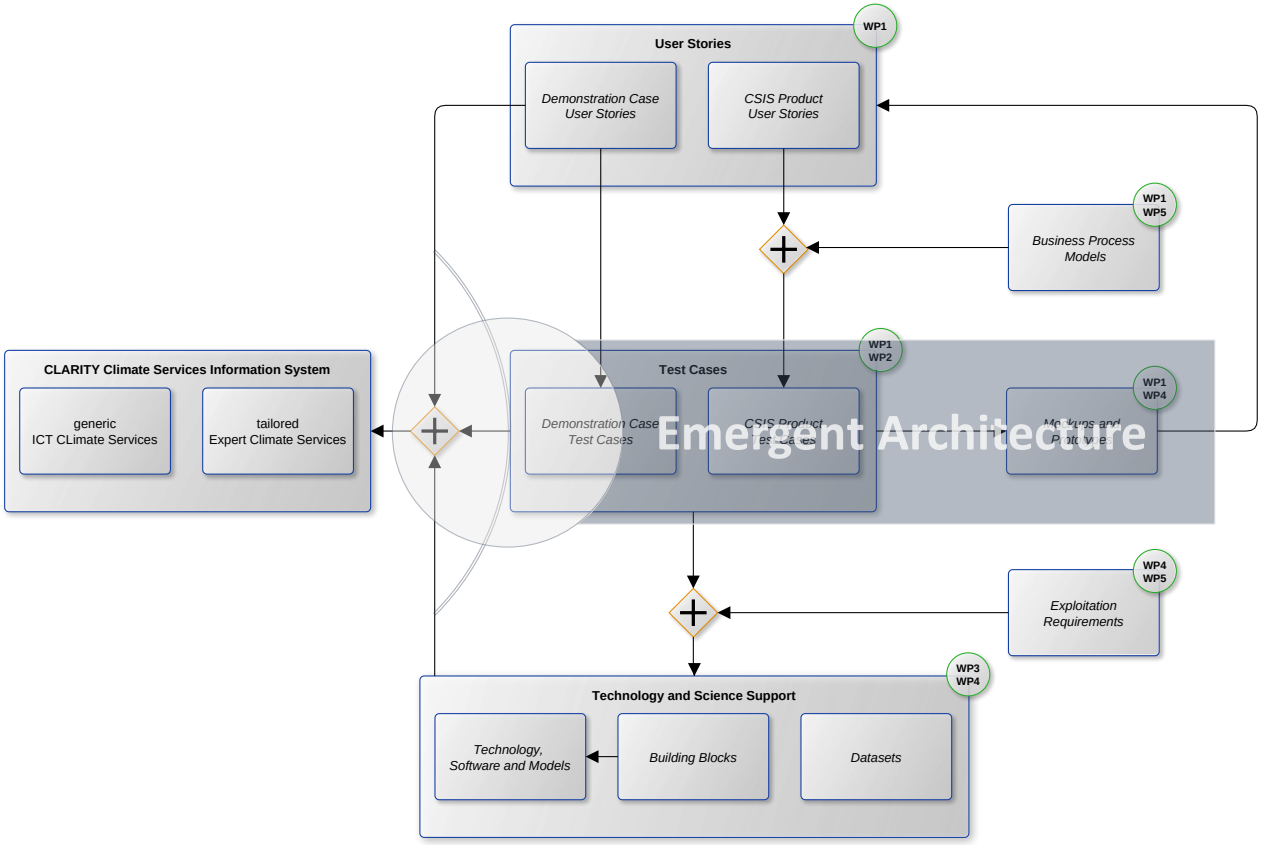

Figure 8: Relationships between Key Artefacts, Work Packages and Emergent Architecture

Figure 8 gives an overview on the key artefacts that constitute to large part the "ubiquitous language" of the architecture. Each of the concepts is represented as distinct item in the CLARITY coordination platform (http://cat.clarity-h2020.eu/) or CLARITY's internal OwnCloud repository.

[7] "A spike solution is a simple program to figure out answers to tough technical or design problems. It only addresses the problem under examination and ignores all other concerns." [18]

They are briefly explained in the following:

"Product Vision"
The common CLARITY "Product Vision" that is presented in D1.1 "Initial Workshops and the CLARITY Development Environment" [25] served as the basis for the initial architecture outlined in chapter 2 of the Technology Support Plan. In D4.1 "Technology Support Plan" [26], it addressed mainly the core business processes and functionalities of the CSIS. D4.2 makes this vision explicit by taking a step forward and performing a conceptualisation into an Explicit Architecture that represents the shared understanding of all stakeholders involved in the product development process. The most significant design decision of the "Product Vision" that devolved into the Explicit Architecture is the methodological concept of ICT- and Expert Climate Services.

Business Processes Models
Business Processes Models were used to elicit core business processes of the CSIS related to co-creation, dissemination and exploitation of tailored Expert Climate Services with help of generic ICT Climate Services in the scope of the EU-GL methodology. Their most notable influence on the Explicit Architecture was to clarify the role of ICT- and Expert Climate Services regarding their value proposition in a general ecosystem of Climate Services. These draft models were developed further into the joint CLARITY business approach presented in D5.3 "Exploitation and Business Plan (v1)" [29].

User Stories
The CLARITY "User Stories", introduced in D1.1 "Initial Workshops and the CLARITY Development Environment" [25] and further refined and completed in D1.2 "Database of Initial CLARITY CSIS User Stories and Test Cases" [27], represent informal descriptions of the key (user) requirements on (mainly Expert) Climate Services expressed from the viewpoint from users that intend to perform site-specific climate change adaptation assessments. For their largest part, they are specific to the implementation of the four CLARITY Demonstration Cases in WP2 "Demonstration & Validation". They demand for site-specific (Expert) Climate Services in different climatic, regional, infrastructure and hazard contexts, focusing on the planning and implementation of urban infrastructure development projects. User Stories served as basis for specification of an initial set of Test Cases, which represent a more formal description of the user's needs.

Test Cases
Test Cases are the counterpart to the Business Processes Models and User Stories. They specify possibilities for implementing the business processes or resolving the User Stories and linking the business and user requirements with the data, models and components (Building Blocks) that shall actually be produced or used in the project. While not a direct concept of Agile Software Development, the initial Test Cases of D1.2 "Database of Initial CLARITY CSIS User Stories and Test Cases" [27] were used to derive functional requirements on Building Blocks and thus to make possible technological choices for Software Components in D4.1 "Technology Support Plan" [26]. In the Emergent Architecture, Test Cases can still be useful for documentation and validating purposes.

Prototypes and Mock-Ups
Prototypes and Mock-Ups are a powerful agile instrument for collecting early feedback from end users by offering a visual preview of the envisaged products and services. Moreover, Mock-Ups can serve as a blueprint for user interface design and help agile software teams to recognize further functional requirements (functionality to be provided by Building Blocks) as well as non-functional requirements (quality attributes of the overall system) that haven't been considered in User Stories and Test Cases. In the Explicit Architecture, the initial Mock-Ups were used to validate and stabilise the essential upfront design, in the Emergent Architecture they are used to select and prioritize the features to be developed during each agile iteration and for documentation purposes.

Exploitation Requirements
An Exploitation Requirement is a requirement that must be met to allow for a successful exploitation of the project's results. A technology-focused assessment of Exploitation Requirements has led to a set of functional and technical requirements on Building Blocks that must be considered during product design and implementation. Besides the jointly developed "Product Vision", the Exploitation Requirements are the main driving force for the Explicit Architecture and its business objectives and quality attributes.

Building Blocks
A Building Block is a generic, composable, adaptable as well as domain- and location-independent unit of functionality (component) that meets the identified business and user requirements by implementing a set of related functional requirements. Products and services are a composition of interacting Building Blocks. The interdependency- and interaction patterns, that is, how Building Blocks interact together across all horizontal layers of the component-based architecture are in general part of the Explicit Architecture and repressed by different component diagrams. Since these patterns as well as technical details of the participating components are subject to change during iterative development, the Explicit Architecture considers them as "black boxes" (no assumptions about the internal logic and structure). In the Translation Layer, those black boxes are then transformed into "white boxes", providing more information on internal details, e.g. on the usage of concrete Software Components.

Software Component
A Software Component is a concrete IT service, tool, system or model that is suitable for the realisation of a Building Block. It can be adapted, customised or configured to provide the functionality defined by a Building Block. The Technology Support Plan in D4.1 "Technology Support Plan" [26] provided an assessment of different Software Components and recommendations for their usage within the architecture. Concrete choices will be made in the Translation Layer of the architecture ("white box" diagrams) by the evaluation of further "spike solutions" and implicitly during the co-development of the CSIS.

Dataset
A Dataset description provides information on used and produced data according to the requirements of the CLARITY Data Management Plan [30].

Building Blocks and Software Components that can be assembled into to concrete applications are also one of the core artefacts of the Framework Architecture of the CRISMA FP7 Project (http://www.crismaproject.eu/) as specified in CRISMA deliverable D32.2 - ICMS Architecture Document V2 [31]. The CRISMA Architecture is based largely on The Open Group Architecture Framework's (TOGAF) Architecture Development Method (ADM) [32] and can be considered a Reference Architecture that follows a lightweight and pragmatic reference model approach (architecture to build architectures). That means it provides the conceptual and methodological framework consisting of architectural concepts, rules and guidelines to actually specify an architecture of a concrete software system ("application") like the CLARITY CSIS. Thereby, it builds upon the concepts of an implementation independent and technology-agnostic Functional Architecture (Building Blocks) and a technology and solutions focused Implementation Architecture (Software Components). A concrete instance of an architecture design according to the respective concepts, rules and guidelines is called Application Architecture (Figure 9). Admittedly, this traditional architectural approach is in the first instance suitable for "Big Design Upfront" (BDUF) as explained in 2.1.

However, if these methods of traditional software architecture are applied in a lean way, they allow the CLARITY architecture and technology support team to follow an agile and solutions oriented approach by focussing on the implementation of concrete products rather than spending effort on fundamental concept development or exhaustive upfront specifications. Thereby, CLARITY can benefit from exhaustive and sound theoretical foundations that have been validated by means of several CRISMA Reference Applications. In this sense, the CSIS Architecture can be considered an Application Architecture that adheres to CRISMA's conceptual and methodological architectural framework.

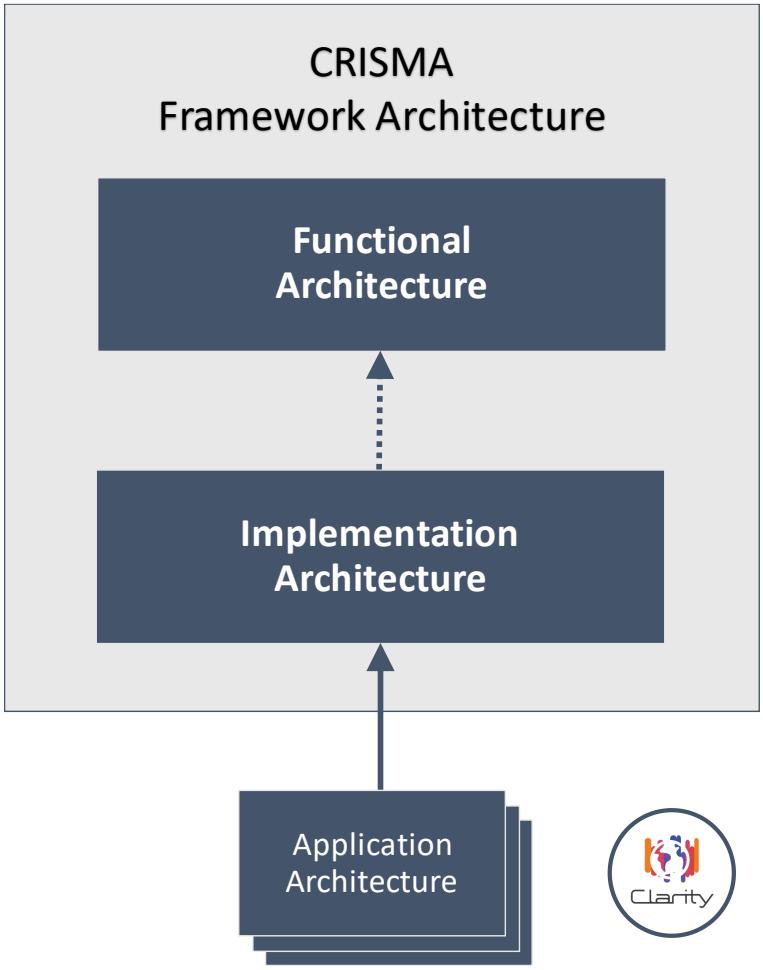

Figure 9: CRISMA Framework Architecture applied to CLARITY [31]

Following the methodology of the CRISMA Framework Architecture, the architecture documentation can be structured according to the MCRI - Mission, Concepts, Realisation and Implementation scheme (Figure 10) originally introduced in [32].

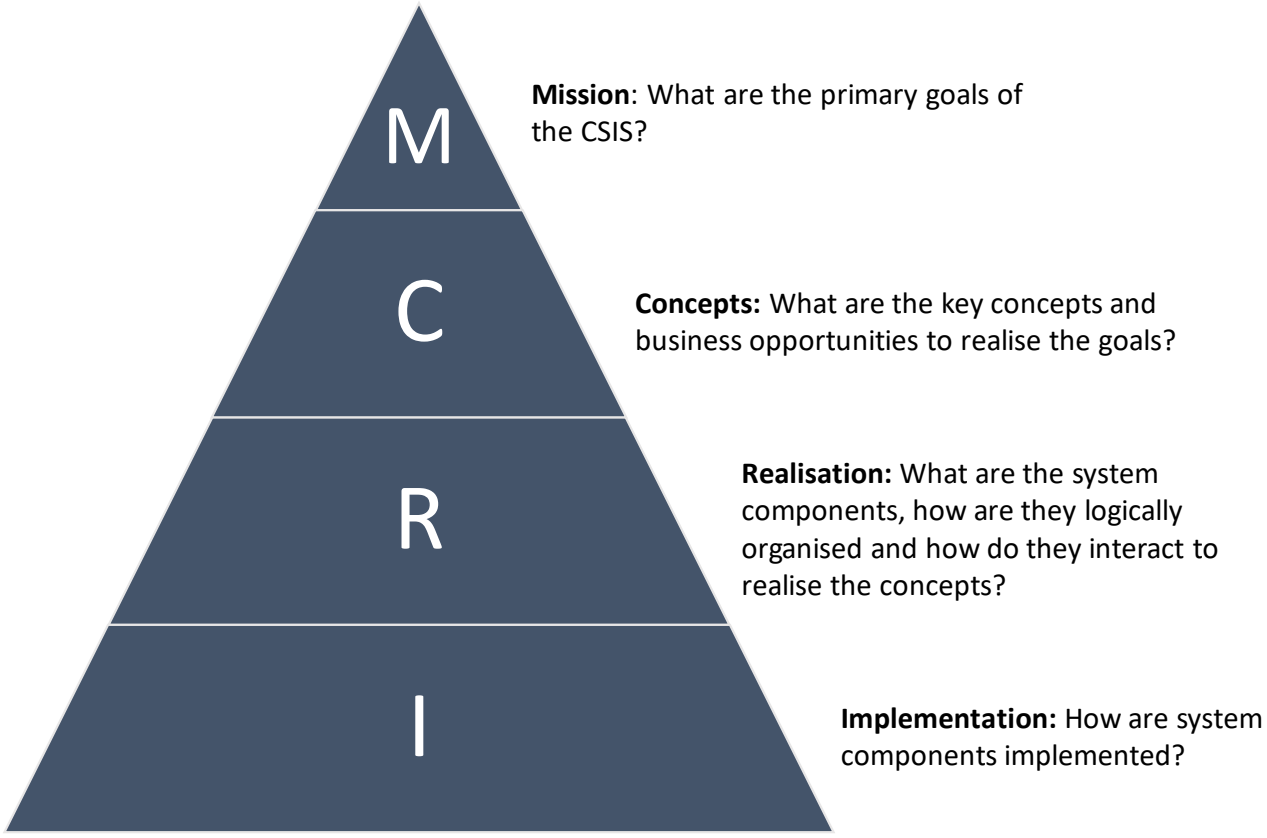

Figure 10: CLARITY MCRI Pyramid

3 Mission

This chapter defines the general mission of the CLARITY CSIS in terms of goals, architectural qualities and -constraints, which have been derived from the project objectives, the elicitation and evaluation of Exploitation Requirements (D5.1 "Exploitation Requirements and Innovation Design" [28]) and during stakeholder workshops (D1.1 "Initial Workshops and the CLARITY Development Environment" [25]).

Figure 11: Architectural Perspective of the CSIS Mission

The Mission of the CSIS clearly belongs to the Explicit Architecture, as it has been concretized explicitly during initial stakeholder workshops and bears the highest cost of change. It is the basis for all significant design decisions described in the Concepts (4).

3.1 Goals

The main goal of the CLARITY CSIS, as initially envisaged in the Description of the Actions (Annex 1 to the Grant Agreement) and more precisely formulated in [26] is to “exploit the added value of Climate Services by providing a climate change adaptation platform based on a coherent methodology integrating a marketplace and a community for Climate Services”. This coherent methodology is based on the "Non-paper Guidelines for Project Managers: Making vulnerable investments climate resilient" [1] (EU-GL) and has been refined and improved according to the IPCC-AR5 framework [33] in the D3.1 "Science Support Plan and Concept" [34]. By implementing this methodology, the CSIS shall allow users to perform a standardised adaptation planning process that is supported by products and services available from the CLARITY marketplace.

This goal is closely related to CLARITY's key exploitation objective, which is the marketing of operational and sustainable products and services. Together with the marketplace and the related CLARITY Community (https://myclimateservices.eu/), the CSIS has to play a vital role in an ecosystem where actors of the supply- and demand-side of Climate Service can connect and collaborate. The different categories of those stakeholders and their roles are described in D5.3 "Exploitation and Business Plan (v1)" [27]. In short, the CSIS shall enable customers to identify the Climate Services that are most relevant to their needs (demand-side) and thereby offer suppliers a platform for disseminating and co-creating commercial services tailored to these user needs (supply-side).

Case studies shall serve to illustrate how the CSIS and related Climate Services provide benefit for the end-users from in different climatic, regional, infrastructure and hazard contexts. The cases studies are represented by four CLARITY Demonstration Cases that are described in detail in D2.1 "Demonstration and Validation Methodology" [35]. The individual business requirements of these Demonstration Cases are translated into marketable products and services that will be integrated into the CSIS, and advertised via the marketplace.

More general business requirements regarding the CSIS are represented by Exploitation Requirements that have been elicited during the first period of the project based on the methodology outlined in chapter 2 "Concept and approach" of D5.1 "Exploitation Requirements and Innovation Design" [26]. Table 2 lists the Exploitation Requirements of D5.1 categorised according to the thematic clusters “Business objectives”, “Communication, community building” and “Quality and novelty”:

Table 2: Exploitation Requirements and Thematic Clusters

Thematic Cluster	Exploitation Requirements
Business objectives	1. Develop a viable business ecosystem, business model and secure access to funding 2. Offer free basic Climate Services based on free and open data
Communication and community building	3. Demonstrate and communicate the (co-)benefits of Climate Services 4. Establish trust in Climate Services and their providers 5. Co-design Climate Services engaging a community of users, providers, purveyors and researchers 6. Follow a multi-sectoral approach that crosses the boundary of climate sciences
Quality and novelty	7. Offer commercial fit-for-purpose tailored Climate Services targeting specific sectors and user groups 8. Consider the role of new regulatory frameworks in stimulating the emergence of Climate Services 9. Provide a user-friendly, intuitive and context-aware discovery and communication infrastructure for Climate Services 10. Use, define and promote open standards for data and services

While these requirements partially translate into functional requirements as described in D4.1 "Technology Support Plan" [26], the Explicit Architecture is mainly concerned with their implications on architectural constraints (chapter 0) and the design decision of the overall architectural concept (chapter 4). A detailed assessment of Exploitation Requirements regarding their impact and concrete technical implications on the CSIS are given in Annex 1 of D5.1 "Exploitation Requirements and Innovation Design" [28].

3.2 Qualities

Architectural quality attributes, sometimes also referred to as architectural properties or architectural principles are high-level non-functional requirements that do not only drive and shape the overall architectural design but also have major impact on the on implementation. While some of those requirements are specific to the CSIS, most are common non-functional requirements of distributed software systems.

3.2.1.1 Software Quality Attributes

ISO/IEC9126 [3] defines and exhaustive list of software quality attributes and provides the respective definitions. In accordance to the aforementioned goals (chapter 3.1), the following qualities are considered architecturally significant for the CSIS:

Reliability: Maturity
“The capability of software to maintain its level of performance under stated conditions for a stated period of time.”

Usability: Understandability
"The effort needed for use and on the individual assessment of such use by a stated or implied set of users."

Efficiency: Time Behaviour
"The relationship between the level of performance of the software and the amount of resources used, under stated conditions."

Maintainability: Changeability, Stability, and Testability
"The effort needed to make specified modifications."

Portability: Adaptability and Installability
"The ability of software to be transferred from one environment to another."

Functionality: Suitability, Accuracy, Interoperability, Security, and Functionality Compliance
"The existence of a set of functions and their specified properties. The functions satisfy stated or implied needs."

3.2.1.2 Application Architectures Quality Attributes

As an instance of the CRISMA Framework Architecture (Figure 9), the CSIS Architecture has to consider also the following qualities of Application Architectures:

Clean and structured system design
The CSIS Architecture shall be based upon well adopted and commonly used architectural design principles. The architecture has to support structured specifications and documentation.

Use of concepts and standards
The CSIS shall make use of proven concepts and standards in order to decrease dependency on vendor-specific solutions and help ensure the openness of the CSIS and support its evolutionary development process.

Loosely coupled components
Components involved in the CSIS shall be loosely coupled, where loose coupling implies the use of mediation to permit existing components (background technologies and software) to be integrated and interconnected with other components.

Extensibility and flexibility
The CSIS shall not be a "closed" system with a fixed set of functionalities. It must be possible to easily integrate new Climate Services into the CSIS.

Security and confidentiality
The CSIS shall be designed to allow state of the art security mechanisms to be incorporated. These mechanisms shall include user management (authentication, authorisation), as well as control of access to data, services and tools.

While some qualities like "use of concepts and standards" and "structured system design" have a direct influence on the CSIS Architecture as such, most are technical in nature and are used to achieve and measure quality and technological readiness (TRL) of software. Similar to Exploitation Requirements, these non-functional requirements can therefore be "implemented as functional requirements" [17] in the Emergent Architecture.

3.3 Constraints

Constraints, as quality attributes, influence the architecture as well as the implementation. Some of them are imposed from the outside, e.g. from the contracting customer or authority of the project. Such contractual constraints encompass typically time, budget and resources constraints. Often they also include constraints to ensure interoperability with the customer's existing infrastructures, for example the usage of predetermined technologies and standards. Contractual constraints related to timing, budget and resources imposed on the CSIS are defined in the CLARITY Grant Agreement.

Other types of constrains indirectly arise from predetermined constraints or boundary conditions like the system and domain context. In CLARITY, the system and domain context are the Climate Change (Adaptation) domain and the EU-GL methodology. Furthermore, as Innovation Action project, CLARITY has to deliver tangible exploitable results in form of new or improved products or services instead of inventing new approaches and concepts or developing of proof-of-concept prototypes. This imposes not only architectural attributes (e.g. maturity and stability) but also the constraint to maximise the reuse of existing components and minimise individual development effort.

As part of the architectural design process, a balance between partially overlapping, partially conflicting conditions, constraints and requirements (Figure 12) has to be found. In CLARITY, this was done early in the project during the initial stakeholders workshops and resulted in a "Production Vision" that adheres to common sense and agile principles (Figure 4).

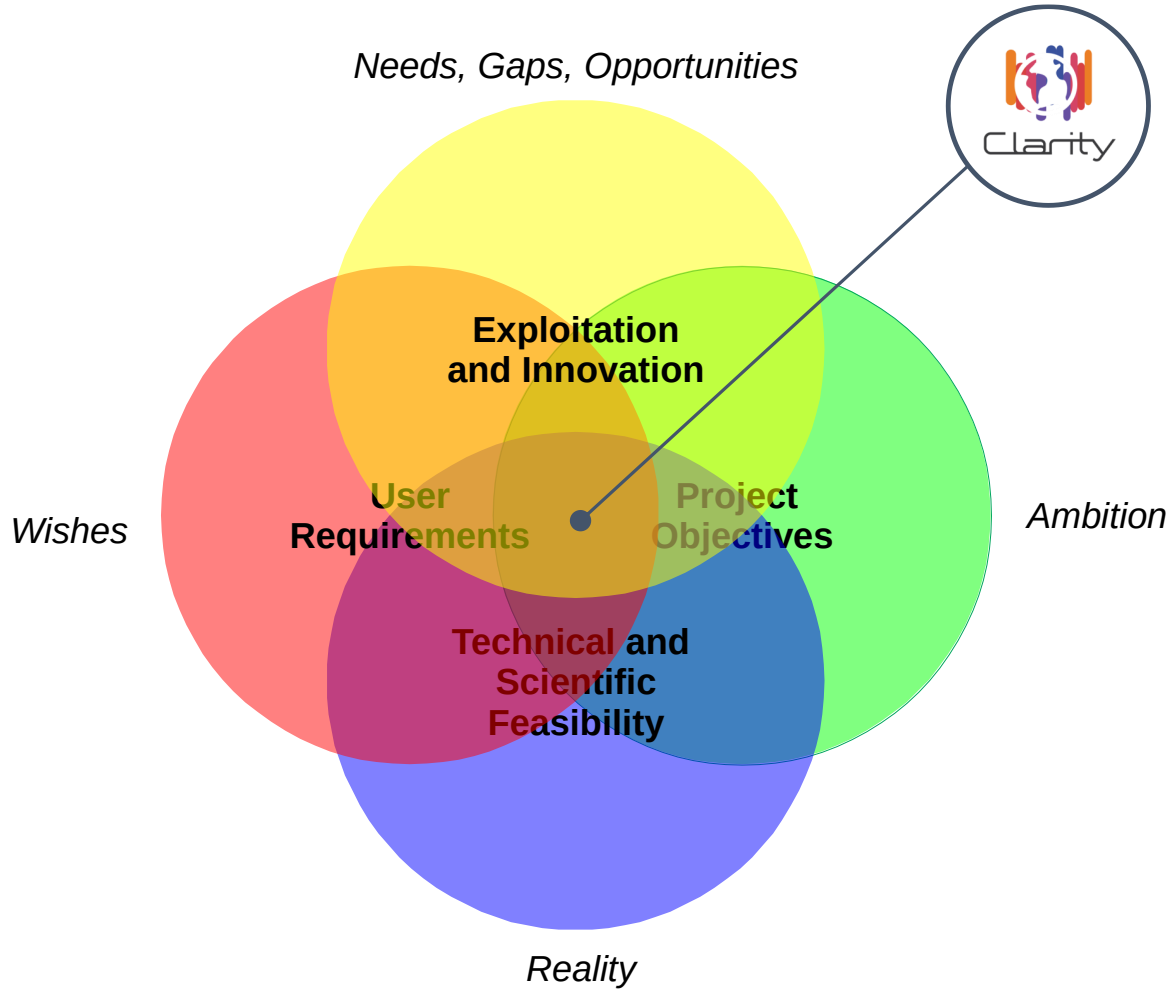

Figure 12: Constraints and Challenges

Since constraints are also helpful to “limit the options that can be used to build the solution” [36], they can reduce both essential and accidental complexity (see chapter 2.3) of the architecture. This is in particular important for the design of products and services that need to become successful in a new market which is characterised by “complexity and lack of maturity” [29].

CLARITY stakeholders therefore constrained the "Production Vision" to pragmatic and realistic options and set clear boundaries of what the CSIS is *not*:

- an all-purpose, one size fits all, off-the-shelf product that generates a tailor made climate change adaptation strategy at the click of a button
- a complex project planning tool that tries to cast the whole EU-GL into software
- a complicated scenario management system oriented towards a specialist and academic audience
- another climate change data infrastructure
- another adaptation platform consisting of a collection of documents that provide conceptual and practical guidance
- a detailed conceptual or theoretical framework and four site-specific prototypes

4 Concepts

This chapter defines the core concepts applied in the CLARITY CSIS Architecture in terms of the conceptual specification of CLARITY products and services (Innovation Design) and the general principles that are used to design and implement the CSIS.

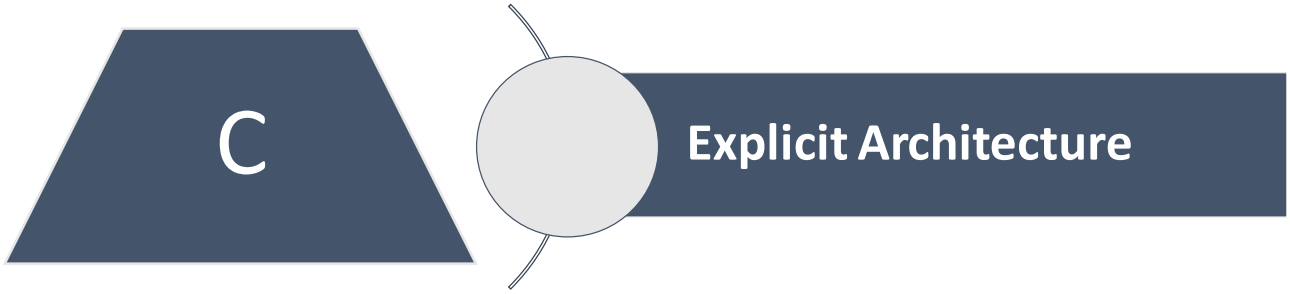

Figure 13: Architectural Perspective of the CSIS Concepts

The Concepts of the CSIS belong to the Explicit Architecture and represent most significant design decisions taken by the architecture team. They do not only influence the CSIS development as such (Realisation and Implementation), but, in case of the Conceptual Innovation Design, also the whole project.

4.1 Conceptual Innovation Design

D5.1 "Exploitation Requirements and Innovation Design" [28] initially defined Innovation Design as “an activity that is incorporated in the architectural design and product development process to support the project in the creation of high impact novelties (products and services) on the basis of existing background (technologies, concepts, prototypes, products and services), while anticipating and addressing the involved risks.”

In the context of the CSIS Architecture, Innovation Design is understood as the conceptualisation of the "Product Vision" introduced in D1.1 "Initial Workshops and the CLARITY Development Environment" [25] into a high-level specification of conceptual product and services types as well as the related integration and dissemination platforms. Thus, the main aim of Conceptual Innovation Design is to make all stakeholders aware of what general types of products and services CLARITY intends to deliver and how they are integrated and disseminated. It furthermore enables internal and external stakeholders representing users from both the demand (customers) and supply (providers) to understand

a) what the general benefits of using the CSIS and related Climate Services are, and
b) what business opportunities of offering Climate Services in the CSIS they can expect.

It furthermore equips agile development teams with the right amount of conceptual background information to understand the high-level domain context (D3.1 "Science Support Plan and Concept" [34]) and general business objectives (D5.3 "Exploitation and Business Plan (v1)" [27]). While CLARITY Conceptual Innovation Design certainly incorporates the most significant design decision of the Explicit Architecture, it is not a roadmap or blueprint for the implementation of the CSIS and related Climate Services. According to the “just enough, just in time” design principle [18] introduced in chapter 2.2, it prepares instead the stage for an agile co-creation process that will result in concrete product and service innovations and the Emergent Architecture (chapter 5 and 6), respectively.

CLARITY’s Innovation Design addresses the core business processes that relate to the 7 modules of the IPCC-AR5/EU-GL-based CLARITY methodology (Figure 14) described in D3.1 "Science Support Plan and Concept" [28]. Accordingly, D5.3 "Exploitation and Business Plan (v1)" [27] recognized EU-GL workflow as “a general process model for decision-driven projects with multiple stakeholders context”.

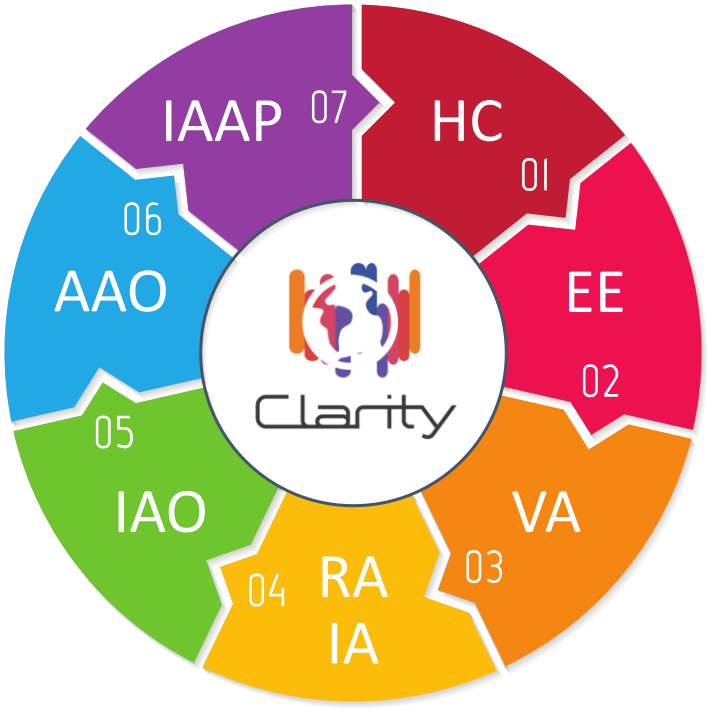

Figure 14: 7 Modules of the CLARITY methodology [34]

The methodology is designed as incremental process that supports iterative improvements and re-assessments but also "exit points" like pre-feasibility assessments that “are rapid screening exercises undertaken early in the project development cycle” [1]. For this purpose, EU-GL initially defined "high-level" versions of the Hazard Characterisation (HC), Evaluation of Exposure (EE), Vulnerability Analysis (VA) and Risk Assessment / Impact Scenario Analysis (RA/IO) modules that can be applied in a lean manner to perform a simple climate change risk screening in an early phase of project development (e.g. the design stage). The result of such a screening can then serve as basis for further detailed climate change adaptation studies by incrementally applying the detailed versions of all seven modules. CLARITY’s updated methodology additionally introduces a "high-level" version of module 5 - Identification of Adaptation Options (IAO), allowing project managers to pre-assess possible options for adapting their infrastructure investments to climate change induced risks.

This modular and incremental approach offers interesting possibilities of value proposition for different categories of stakeholders (see D5.3 "Exploitation and Business Plan (v1)" [27]) whereby CLARITY DoA already identified the following main lines of commercial products and services:

(1) low- or even zero-price/high volume automated pre-assessment;
(2) customization and extensions of the data and services; and
(3) consulting and project-specific modelling of the key adaptation options for projects.

In this regard, Conceptual Innovation Design defines the following architectural artefacts:

- ICT- and Expert Climate Services implementing one or more EU-GL modules at different levels of detail;
- the CSIS as platform of integrated Climate Services for supporting standardised adaptation planning following the scientifically sound CLARITY methodology; and
- a Marketplace as outbound portal and entry point for using and ordering these Climate Services; and
- Demonstration Cases as means for showcasing the benefits of (CLARITY) Climate Services and as "incubator" for tailored products.

4.1.1 CLARITY Climate Services

According to EU-GL and CLARITY objectives, main types of direct end users of CLARITY Climate Services and the CSIS, respectively, are project managers, city planners and climate resilience mangers. Stakeholder workshops and market analysis confirmed this statement and more concretely identified “enterprises or organisations who promote concrete infrastructure projects, e.g. real estate developers (e.g. port authorities, municipalities), their investors (e.g. banks, insurances, funds, public authorities) and suppliers (e.g. construction companies, planners, (landscape) architects)” [27] as direct customers of CLARITY Climate Services.

The identification of these target groups exerted major influence on the formulation of the overall goals of the architecture (chapter 3.1) and general constraints (chapter 0) on the CSIS. In consequence, CLARITY has to offer new and innovative Downstream Climate Services on basis of available Upstream Climate Services (climate data services). While such Downstream Climate Services must not be oriented towards a specialist and academic audience like the climate science community, they nevertheless have to be robust, credible and scientifically sound. Addressing usability requirements of the relevant target groups requires essential complexity (see chapter 2.3) to be hidden from the user. However, since essential complexity cannot be eliminated or even reduced, trying to develop a fully automated “push-button” solution for climate change adaptation plans is neither scientifically nor technically feasible (Figure 12).

CLARITY Innovation Design introduces therefore the concepts of generic and focused ICT Climate Services and integrated and customised Expert Climate Services. Interestingly, this concept, although developed independently, fits seamlessly into the Climate Service implementation scenario definitions of the EU-MACS[8] (EUropean MArket for Climate Services) project. EU-MACS intends to “clarify how the market for climate services could abound by improving the matching of supply of and demand for climate services” [37]. The Constructive Technology Assessment (CTA) workshops identified in EU-MACS deliverable D1.4 "A multi-layer exploration on innovations for climate services markets" [37] four scenarios for implementing climate services into institutional/organizational context in different socio-technical formats (Figure 15).

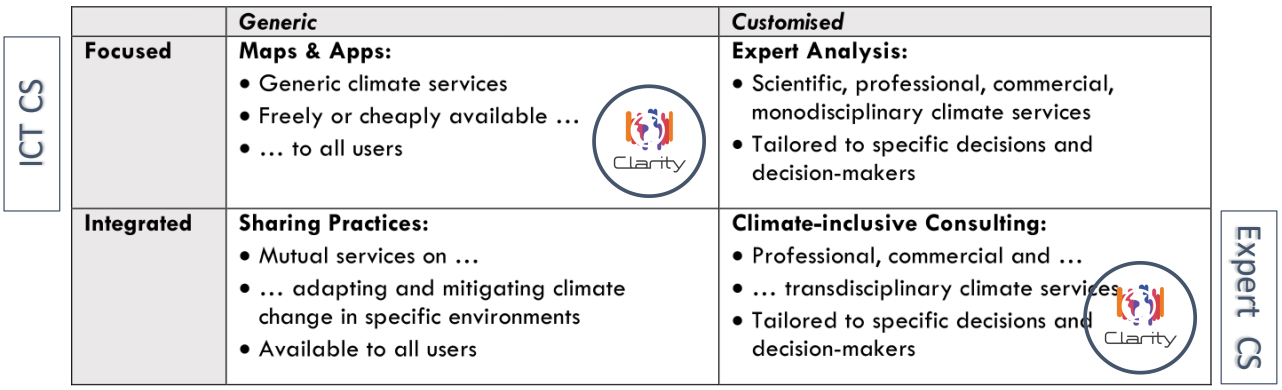

Figure 15: CTA Scenario core Characteristics applied to CLARITY [38]

Thereby, the customisation dimension distinguishes between tailored or generic services and the integration dimension between specialized services or services integrated in a broader package like Disaster Risk Reduction (DRR).

[8] Project ID: 730500. Funded under: H2020-EU.3.5.1.

Table 3: Comparison of CTA Scenarios relevant for CLARITY Climate Services [37]

ICT Climate Service "Maps & Apps" Scenario	Expert Climate Services "Climate-inclusive consulting" Scenario
users themselves incorporate climate data into their decision making	a customised climate service integrated in a broader consulting service, for instance aimed at financial risk management, urban planning, or regional development
all users have in principle the same climate data available, typically in the form of digitalized dynamic maps	users would pay for accurate data and a highly contextualized interpretation of the consequences of climate change
value creation depends on good user interfaces and users that are knowledgeable enough to handle the information they get	value would be created through specified user questions being answered by specialized and professionalized climate service providers
the data infrastructure must be unified and preferably global to enable these applications	data infrastructure would remain heterogeneous, with a variety of measuring grids, adapted to the local situation
although the models are generic, sufficient accuracy on local situations is required for climate change informed decision making	users making decisions based on expert analysis of the effects of climate change for their specific location and problem
generic services are not sufficient for the complex decision making situations; not all users may be competent to interpret the data (and the uncertainty in the models)	potential tensions would arise when climate expert analysis leads to biased analysis and suboptimal solutions for complex problems, not taking into account expertise from adjacent disciplines

Table 3 provides a comparison of relevant properties of the "Maps & Apps" and "Climate-inclusive consulting" Scenarios in relation to CLARITY ICT- and Expert Climate Services.

4.1.1.1 ICT Climate Services

In the context of Conceptual Innovation Design, the general support of the EU-GL methodology is realised by the CSIS itself and a set ICT Climate Services which are generic in their nature and can be considered according to EU-MAC's "maps & apps climate service scenario" (Figure 15) as products offered by CLARITY. Their main properties are listed in Figure 16.

ICT Climate Services are (partially) free, simple, ready to use, online ICT tools (software). Thereby, most complexity is hidden from the end-user to provide an easy-to-use product, for which no or just minimal knowledge of climate change science or technical skills is needed.

The basic implementation scenario for ICT Climate Services is within the scope of a pre-feasibility analysis as anticipated by the EU-GL methodology (high-level application of EU-GL Modules). Such basic services should not depend on costly site-specific modelling, high performance computing or expensive local high-resolution climate data. Instead, they can "compromise temporal and spatial resolution" [39] and rely on freely available data and model outputs. They will mainly cover the first step in developing an overall adaptation strategy based on commercial and tailored Expert Climate Services (consultancy, advisory, modelling and development).

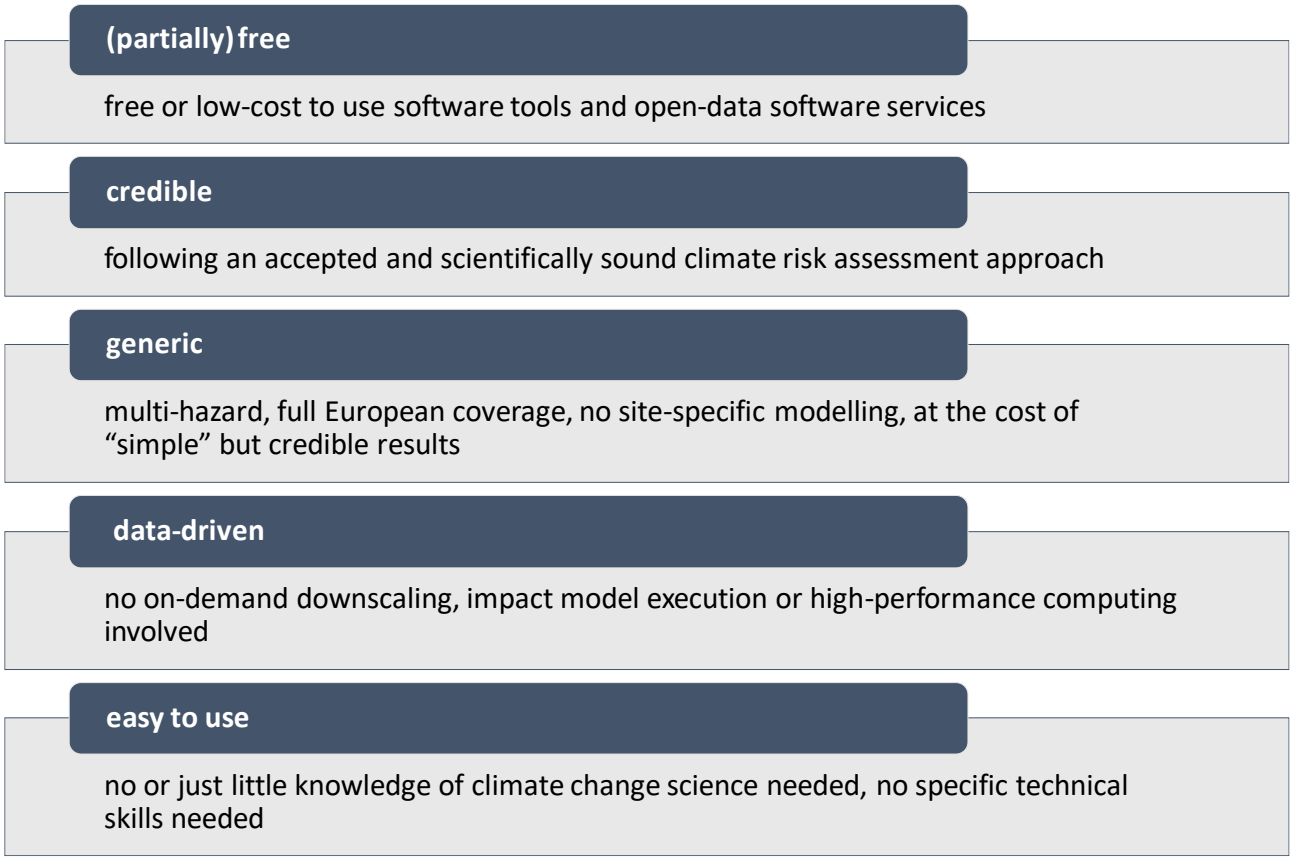

Figure 16: Main Properties of ICT Climate Services

Further advanced implementation scenarios include for example paid features (freemium model) such as

- usage of pre-calculated high-resolution climate or exposure data based on "Data Packages" (see chapter 4.2.4)
- customisation of vulnerability functions of element at risk's vulnerability classes
- advanced analysis, comparison and decision support related to standardised out of impact scenarios (e.g. Multi Criteria Decision Support Analysis)
- sector specific products related to the four CLARITY Demonstration Cases (refer to D5.3 "Exploitation and Business Plan (v1)" [29] for a list of potential products)

However, even advanced implementation scenarios of ICT Climate Services are not meant as complete replacement for a detailed climate risk assessment and adaptation study according to the EU-GL methodology. For such a detailed study, additional Expert Climate Services are needed.

4.1.1.2 Expert Climate Services

In the context of Conceptual Innovation Design, the assessment of highly customised and site-specific adaptation scenarios following the EU-GL methodology is realised by Expert Climate Services which are tailored in their nature and can be considered according to EU-MAC's "climate inclusive consulting climate service scenario" (Figure 15) as services offered by CLARITY. Their main properties are listed in Figure 17 below:

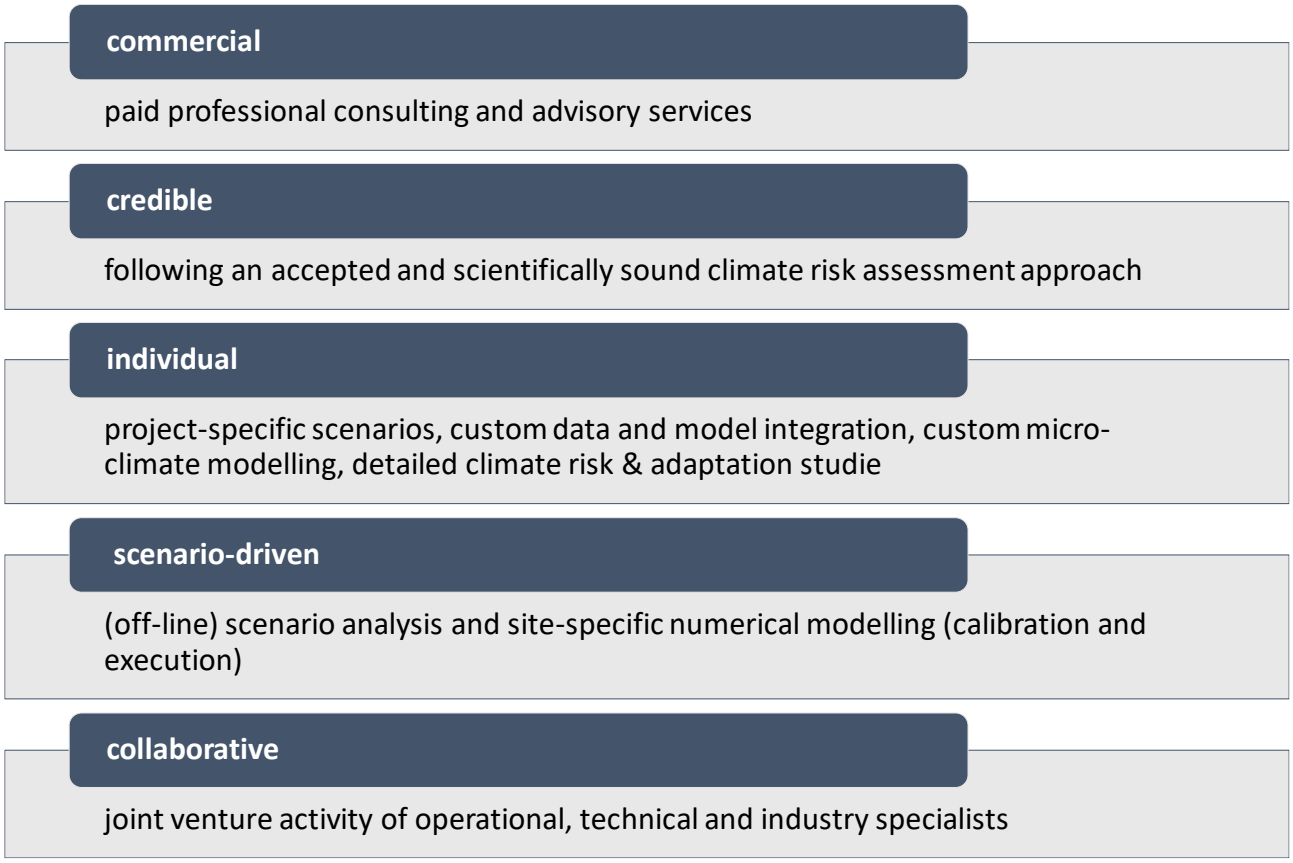

Figure 17: Main Properties of Expert Climate Services

An Expert Climate Service is an individual and professional consulting and advisory service that can be provided as joint venture activity of operational, technical and industry specialists. It can be considered a tailored and fit-for purpose "micro" Climate Service that is co-created according to individual user needs and thus a commercial service that users are willing to pay for. Consequently, it may involve highly customised activities such as sector-, project- and site-specific analysis, custom data and model integration, site-specific numerical modelling and so on. If disseminated via the CSIS, such Expert Climate Services must be provided according to specific rules and guidelines that are set out for the one part in the CLARITY Modelling Methodology and for the other part in technical specifications that allow integration with the CSIS.

Thereby the CSIS offers a technical platform that acts as mediation and integration layer between Expert- and ICT Climate Services: Customers will be supported in requesting Expert Services via the Marketplace and experts will be supported in uploading their results to the user's workspace within the CSIS. Backend software like local models that is needed by an expert to perform such service does not need to be (technically) integrated in CSIS nor accessible through it. Instead, the CLARITY's data-driven approach (4.2.4) requires experts to deliver their interoperable data formats that are compatible with ICT Climate Services.

4.1.2 The CLARITY Climate Service Information System

The overall role of the CSIS in the CLARITY business approach is "to bridge the gap from supply driven (Upstream) Climate Services to demand driven (Downstream) Climate Services by offering (partially) free basic and generic ICT Climate Services and to help end users to identify and discover their need for fit-for-purpose commercial Expert Climate Services" [3].

The CSIS a co-creation environment that allows Climate Service Customers and Climate Service Suppliers to create fit-for-purpose and tailored Expert Climate Services under the umbrella of a scientifically sound conceptual methodology (EU-GL) for Climate Change Adaptation Assessment.

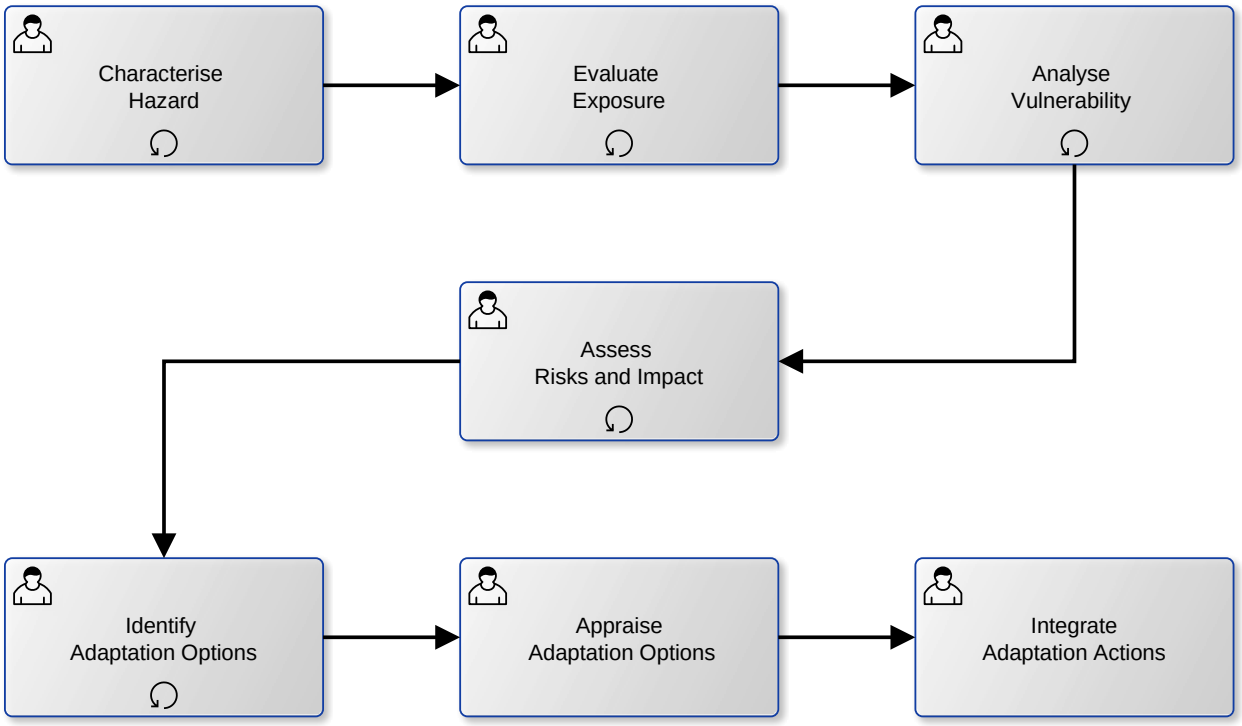

Figure 18: EU-GL Workflow

The CSIS supports and enforces the standardised workflow of the EU-GL for each of the distinct modules (Figure 18) and provides respective user interfaces that guide the user through the process. The end user is presented with the list of recommended and required steps for performing a complete Climate Adaptation Study for the respective (infrastructure) project under assessment and is asked to provide the information that is needed to complete the current module and advance to the next module. Thereby, the starting point is a simple baseline (without taking adaption measures into account) pre-feasibility assessment to current and future climate conditions that is performed with help of ICT Climate Service on basis of general climate hazard and exposure/vulnerability data (refer to D3.1 "Science Support Plan and Concept" [28] for more details).

The general idea of the underlying interaction concept is thereby to prevent a potential climate service customer from being overtaxed by too much information and too many choices or questions. Instead, the process starts intentionally simple and then gradually increases complexity while presenting condensed information about the current module step and the underlying (climate) data and models. If available, contextual links to detailed background information (meta-data catalogues) or relevant services (from the Marketplace) are integrated. During the whole process, the CSIS collects information that is relevant for later Expert Service provision.

To this end, the user will not only obtain a simple but credible preliminary assessment but creates in parallel an initial specification for a tailored assessment. This specification can then be handed over to Climate Service Suppliers (via the Marketplace) to prepare an offer for tailor-made Climate Services (made available in the CSIS). Such a tailored service specifically addresses local conditions and additional user requirements that cannot be covered by generic and automated pre-feasibility assessment.

The possibility for customer and suppliers to collaboratively perform a Climate Adaptation Study following the distinct steps of the EU-GL methodology opens for various value streams with dedicated business models (D5.3 "Exploitation and Business Plan (v1)" [29]).

The overall idea is roughly depicted in Figure 19 and will be detailed in the Emergent Architecture in form of Mock-Ups (6.2).

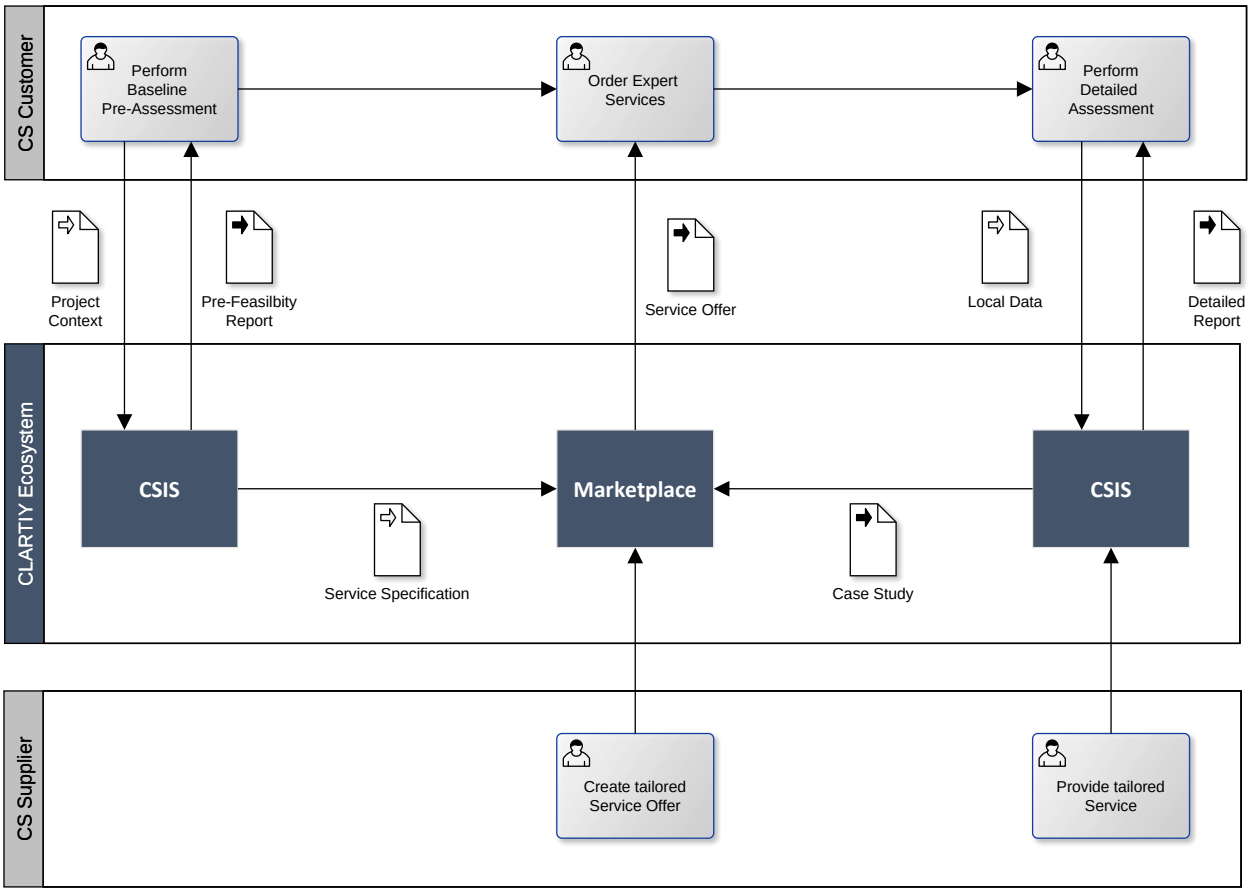

Figure 19: CS Customer / Supplier Interaction Scenario

When put in context of CLARITY's "sister project" EU-MACS [39], the CSIS fits best into the Communication Infrastructure and Service Infrastructure dimensions of Climate Services as shown Figure 20.

However, the aim of the CLARITY CSIS is not to establish a universal and all-encompassing Climate Service infrastructure. As EU-MACS deliverable D1.3 "Analysis of existing data infrastructures for climate services" [39] pointed out "Since the degree of data organisation in climate services and neighbouring areas are far from being fully established, an enormous effort is required before it is entirely fit for purpose by specific users" and "…it might be hard to develop a universal yet single solution covering all infrastructure dimensions that serves the user community effectively. Striving for this single solution may in fact lead to an overly complex structure making the interface at the end even less user-friendly."

The CLARITY project does not intend to develop a general information system for any type of Climate Services, as for example the Global Framework for Climate Services (GFCS) aims for with its Climate Information System. To be more precise, the CLARITY CSIS is not really just an information system for the collection, organization, storage and communication of arbitrary climate-change related information. Instead, the CLARITY CSIS represents a platform (see chapter 4.2.6) that unites under a common user interface Climate Services that support climate change risk/impact assessments targeted at mitigation/adaptation options priorities identification following the EU-GL-based CLARITY modelling methodology defined in D3.1 "Science Support". By implementing the EU-GL's general process model for decision-driven projects it imposes CLARITY's modelling methodology on risk/impact assessments and mitigation/adaptation planning studies that are carried out via the CSIS. This is the main difference and most important innovation in comparison to the existing Climate Change Adaptation Platforms that provide conceptual and practical guidance but not the technical means to ensure compliance to underlying theoretical framework.

Therefore, the CSIS puts the co-creation and marketplace aspects upfront and tries to stimulate the creation and uptake of tailored Expert Climate Services instead of providing a technical infrastructure for hosting or integrating any kind of (ICT) Climate Services.

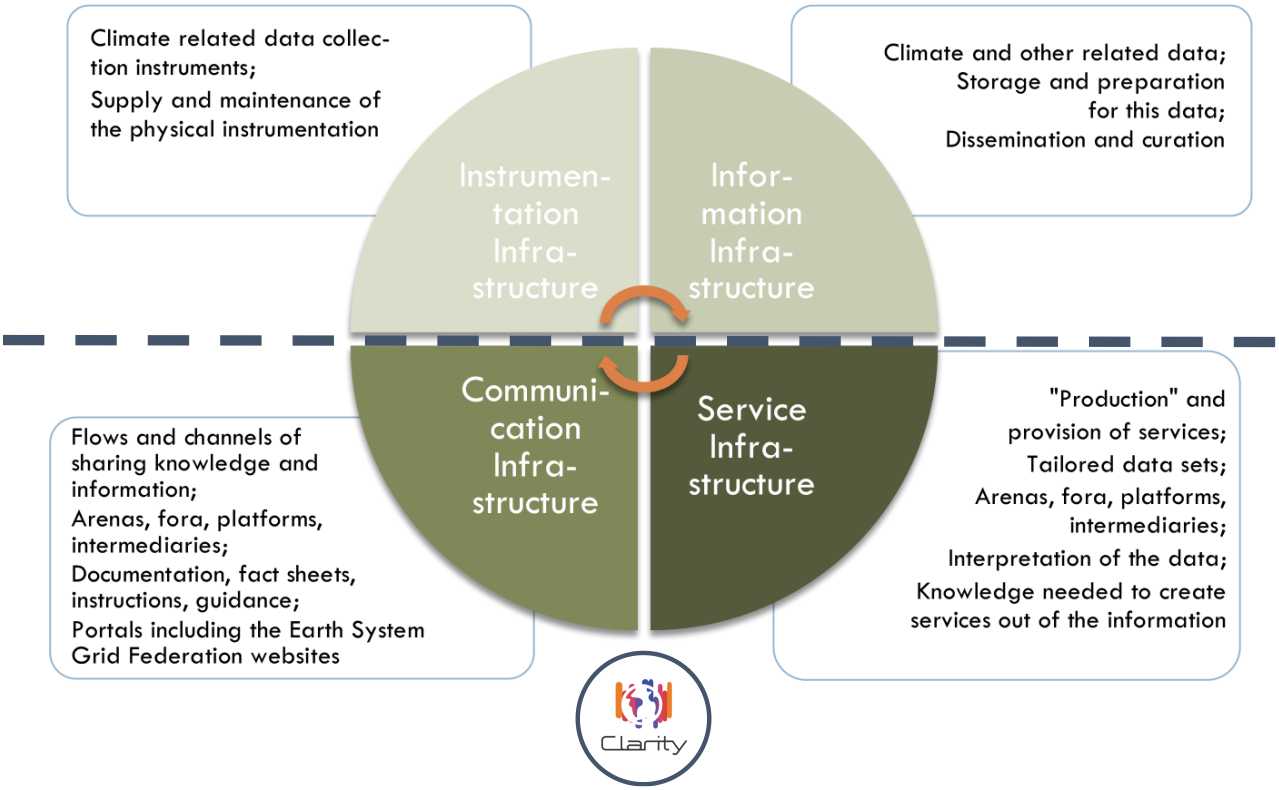

Figure 20: Climate Service Infrastructure Dimensions applied to CLARITY [39]

4.1.3 The CLARITY Marketplace

The CLARITY Marketplace as entry point for using and ordering Climate Services and tools and the related outbound portal “MyClimateService.eu” is detailed in Deliverable D6.2 “Communication and dissemination plan and report” [40], D5.3 "Exploitation and Business Plan (v1)" [28] and D4.1 "Technology Support Plan" [26]. Besides the general supplier / service catalogue and procurement functionalities summarised in Figure 21, the marketplace aims at building up and service a vivid community interested in climate change adaptation. Together with the CSIS, the Marketplace forms the CLARITY ecosystem where Climate Service suppliers and customers can connect. It supports customers in discovering suitable Expert Climate Services and relevant climate data and tools to minimize the climate change impact on their infrastructure projects. It supports suppliers in disseminating and advertising (e.g. by means of case studies) their core products and services as well as connecting them with other supplier to create new innovative products and services tailored to user needs.

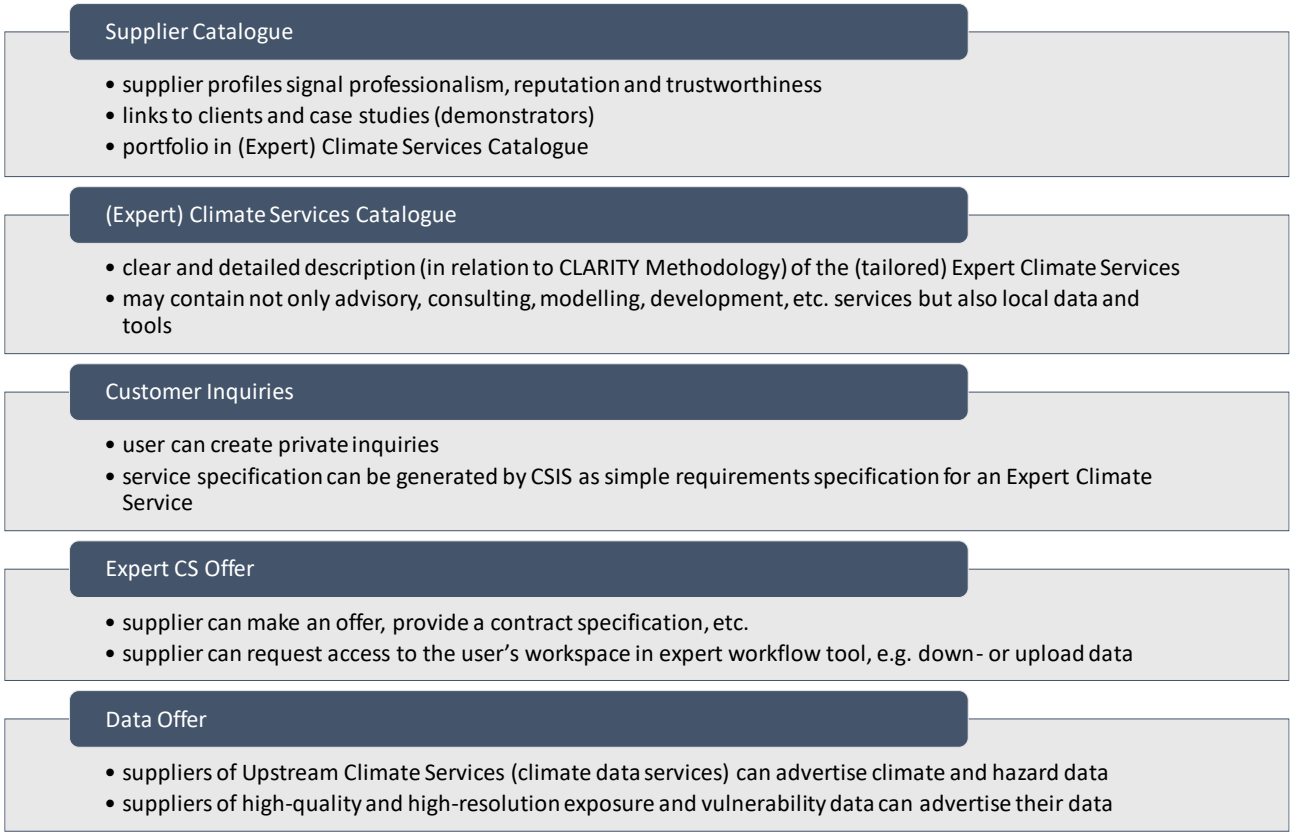

Figure 21: Main Functionalities of the CLARITY Marketplace [26]

4.1.4 CLARITY Demonstration Cases and extended use cases

According to the DoA and the goals of the CSIS Architecture (3.1), the four Demonstration Cases intend to "showcase CLARITY Climate Services in different climatic, regional, infrastructure and hazard contexts in Italy, Sweden, Austria and Spain; focusing on the planning and implementation of urban infrastructure development projects". Their relation to CSIS and the demonstration of benefits of CLARITY Climate Services is thereby manifold, as they demonstrate how:

(1) the basic functionally in terms of simple pre-feasible assessment offered by ICT Climate Services can help to get a better understanding of climate-change related issues the possibilities for adapting to them;
(2) in collaboration with experts from different disciples a tailor-made adaptation strategy to concrete climate change impacts can offer additional social and economic benefits; and
(3) such tailored Climate Services can be integrated into existing planning processes and thus represent an additional exploitable products on their own (D5.3 "Exploitation and Business Plan (v1)" [28]).

As stated in chapter 4.1.2, the system isn't going to be universal. For the Demonstration Cases, this means that even within the bounded context [41] of the CLARITY Modelling Methodology, it would not be feasible to strive to develop a solution that is entirely fit for purpose for any particular use case within this context for obvious technical reasons as explained in Annex 2 of the Technology Support Plan [26]. This holds not only true for the CLARITY Demonstration Cases, which represent four more or less disparate use cases of climate change adaptation planning, but also yet unknown extended use cases from other infrastructure project types and domains. Although the Demonstration Cases related to urban planning are similar in their nature, they still don't represent exactly the same use case.

While the system is not universal in the sense that it would provide a “one size fits all” solution that covers all needs of the Demonstration Cases or use cases beyond the scope of CLARITY, it is however both architecturally and technically universal in the sense that it represents the common denominator for use cases that intend to follow the EU-GL process model and the CLARITY's modelling methodology, respectively.

The CSIS and its parts (Building Blocks) itself are generic beyond the scope of CLARITY and the Demonstration Cases in that they will comprise a generic core of ICT Climate Services supporting climate change risk/impact assessments. The starting point for the development of the CSIS are the location and use case independent "high-level" versions of the EU-GL modules resulting in a system for climate change risk screening as part of a pre-feasibility analysis. Such a general but nevertheless credible and scientifically sound screening exercise is also part of each Demonstration Case and represents the first step towards a detailed and tailored expert study for climate proofing.

To support the CLARITY Demonstration Cases and in particular additional infrastructure projects in performing more detailed, more use-case specific and ultimately tailor-made assessments, the CSIS is designed to evolve along two customisation dimensions: The data dimension and the feature dimension. Those dimension define the degree of tailoring needed to support a particular use case within the CSIS platform and thus also the boundaries and characteristics of the Expert Climate Services (see chapter 4.1.1.2).

The data dimension covers among others local hazard-, exposure- and vulnerability data as well as impact and adaptation scenarios and the related derived performance indicators for comparing and ranking different (adaptation) scenarios. For the CSIS as an information technology system to "stay in a realm of limited, manageable complexity levels" [41], a data-driven approach (see chapter 4.2.4) is followed that defers those data integration and processing tasks which cannot be solved generically to Expert Climate Services. Requirements on such Expert Services emerge from T2.2 “Demonstrator-specific data collection“ and are jointly addressed in T1.3 “Climate Services Co-creation”, T2.3 “Demonstration” and WP4 “Technology Support”. In general, evolvement along the data dimension does not lead to the need to introduce new components (Building Blocks / ICT Climate Services) into the CSIS as the core features of the basic EU-GL workflow (Figure 18) like visualisation, analysis, report generation, etc. are generic as long as they can rely on data being provided in the format by CLARITY's Data Package Specification.

The feature dimension covers functionalities that are not considered in the basic EU-GL workflow and relate to use-case specific feature requests expressed in the Demonstration Case User Stories. Thus, the extension of the CSIS' basic feature set requires the provision and integration of additional components. To support such extended use cases, the CSIS offers a User Interface Integration Platform (see chapter 4.2.5) able to integrate third-party components (“CLARITY Apps”). The requirements on the related ICT Climate Services representing either generic or tailored “Apps” emerge from T1.1 “Climate Service Requirements “ and are jointly addressed in T1.3 “Climate Services Co-creation”, T2.3 “Demonstration” and WP4 “Technology Support”.

4.2 Principles

The CSIS architecture adopts a number of common architectural principles and approaches to ensure consistency in how the CSIS and related Climate Services are realised and implemented. This chapter intends to give a brief overview on the most important principles of the Explicit Architecture.

4.2.1 Component-based Architecture

Architectural constraints, especially those requiring to deliver innovative solutions under consideration of the resources and time available demand for an “implementation strategy that must be based mainly on combination, integration and adaptation of existing background technologies and software rather than entirely new developments” [26].

The CSIS adopts therefore a component-based approach that is additionally able to address the quality attributes usability, extensibility, performance and maintenance. "In Component-Based Software (CBS) development, the designer designs systems by using readily available (possibly third party) software components without needing the source code for the components." [42] The components of the CSIS are the Building Blocks and their respective Software Components (see chapter 2.4).

The relation between the Artefacts of CLARITY's component-based Architecture is shown in Figure 22. A Building Blocks is a realisation of (one or more) Software Components. Both CSIS and ICT Climate Services are composed of several interacting Building Blocks, while the CSIS additionally integrates ICT Climate Services.

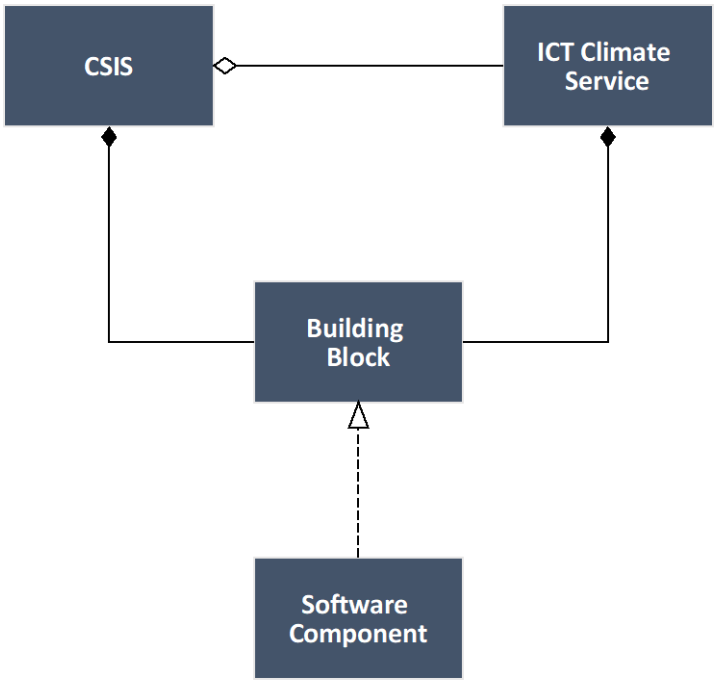

Figure 22: Artefacts of CLARITY's component-based Architecture

4.2.2 Service Oriented Architecture

The Organization for the Advancement of Structured Information Standards (OASIS) defines Service Oriented Architecture (SOA) as "a paradigm for exchange of value between independently acting participants; participants (and stakeholders in general) have legitimate claims to ownership of resources that are made available within the SOA ecosystem; and the behaviour and performance of the participants are subject to rules of engagement which are captured in a series of policies and contracts." [43] The CSIS Architecture adopts this paradigm and selects or develops components that either expose or consume RESTful Web Services Interfaces (5.2.4.1) and that can be independently deployed in the CLARITY infrastructure as described in Annex 3 of D1.1 "Initial Workshops and the CLARITY Development Environment" [25]. To guarantee interoperability between independently developed components, well-defined service contracts (see chapter 2.3) and standards-based service interfaces and data formats (e.g. from the Open Geospatial Consortium) are used.

4.2.3 Layered Architecture

For communicating the architecture, especially the Realisation part (see chapter 5), the Explicit Architecture provides a view of the CSIS that organises Building Blocks into logical, abstract groups (layers). "These layers help identify, describe, and differentiate the kinds of tasks that those artefacts perform" [18].

Thereby it is important to understand, that this separation in to layers is a means to increase the shared understanding of the high-level system design (see chapter 2.2). It does not impose any constraints on implementation or usage of components. During an agile iteration, developers concentrate mainly on vertical feature development that is development of a particular feature across all layers.

4.2.4 Data-driven Approach

The CSIS and integrated ICT Climate Services follow a data-driven approach that builds upon standard data formats like Data Packages (5.2.8.1) and OGC Geo Packages (5.2.8.2). Data-driven means here, that complex local model execution, like downscaling or urban climate modelling, is not performed within the runtime context of the CSIS but "offline" within the scope of an Expert Climate Services. The required integration and harmonisation tasks like data transformation, model calibration, post-processing of results, etc. can be performed as join-venture activity of it-specialists and model experts. Expert Climate Services or datasets disseminated via the CLARITY Marketplace have to adhere to the Data Package specification defined by CLARITY. This includes also the provision of an appropriate set of meta-information (e.g. related to uncertainty and data provenance) to support discoverability and transparency.

The CLARITY Data Package specification intends to establish/define a set of minimum requirements for datasets in order to facilitate compatibility and interoperability among systems and stakeholders involved in their creation and consumption. Such pre-compiled data packages contain all (or several of) the datasets necessary to carry out climate proofing studies following the CLARITY Modelling Methodology. Quality and appropriateness of these datasets may depend on the origin of the data (e.g., data owner, climate expert who made the analysis, etc.), formats supported by the destination system, as well as spatial and temporal resolutions and level of uncertainty contained in the data itself required for performing the project climate proofing assessment. All datasets included in the Data Package enclose the corresponding metadata records so that parties using it are aware of who, when, how (including information about the uncertainty) and for what purpose the data was produced. Among the various datasets that can be included in a CLARITY Data Package we can find the following:

- Hazard Maps of the various hazards affecting the area of study
- Exposure Maps related to the previous hazards in the area of study
- Vulnerability Maps related to the vulnerable elements in the area affected by the hazards
- Impact scenarios
- Lists of Adaptation Options applicable to the different elements according to the hazards affecting

This not only improves interoperability among ICT Climate Services but creates enormous possibilities to widen and improve the current Climate Services Market, allowing new business opportunities for Climate and Disaster Risk experts, data owners and ICT developers and integrators among others by offering Expert Climate Service for the production of tailored Data Packages.

4.2.5 User Interface Integration Approach

The main intention of a component-based Architecture is maximise the re-use of existing components. Since those components may potentially use different technologies such as pure client-side HTML5/AJAX technologies (5.2.2), server-side technologies or even a mixture of both, there is a need for a User Interface Integration approach. The CSIS Architecture introduces therefore the UI Integration Platform Building Block (5.1.1.1) that combines different independently developed interactive JavaScript/HTML5 applications into a common user interface that acts and looks more like one integrated Rich Internet Application (RIA) rather than a portal or website. Thereby, this UI Integration Platform provides also some basic functionality like user management, per-user customisation, user workspace, etc. More technical details, especially related to the realisation to the Emergent Architecture, can be found in chapter 7.5 of D4.1 "Technology Support Plan".

Thereby, the aim of the User Interface Integration Approach is not only to facilitate the implementation of the basic EU-GL workflow, but also to facilitate the evolvement of the CSIS along the feature dimension (see 4.1.4). Correspondingly, the task of developing and integrating tailored “CLARITY Apps” that addresses use case specific requirements (including but not limited to those of the CLARITY Demonstration Cases) can be offered as commercial Expert Service.

4.2.6 Platform Architecture

A Framework offers a set of composable and generic Building Blocks, which can be integrated and assembled together with local and heterogeneous data and models (external components) into complex applications (Figure 23). This approach has for example been followed in the SUDPLAN and CRISMA (Figure 9) projects. In case of CLARITY this would mean, four separate applications (prototypes) would have to be developed - one for each Demonstration Case (see chapter 4.1.4). This approach is suitable when the use cases to be implemented as applications are rather heterogeneous, don’t follow a common and prescriptive methodology, don’t intend to share a common user interface, etc. The main transferable and exploitable results are then the Building Blocks and the Framework as such. The prototypes serve mainly as proof that the distinct Building Blocks of the Framework can be used for the development of independent and fit-for-purpose applications. A related exploitation model is therefore the development of custom applications (“Expert Service”).

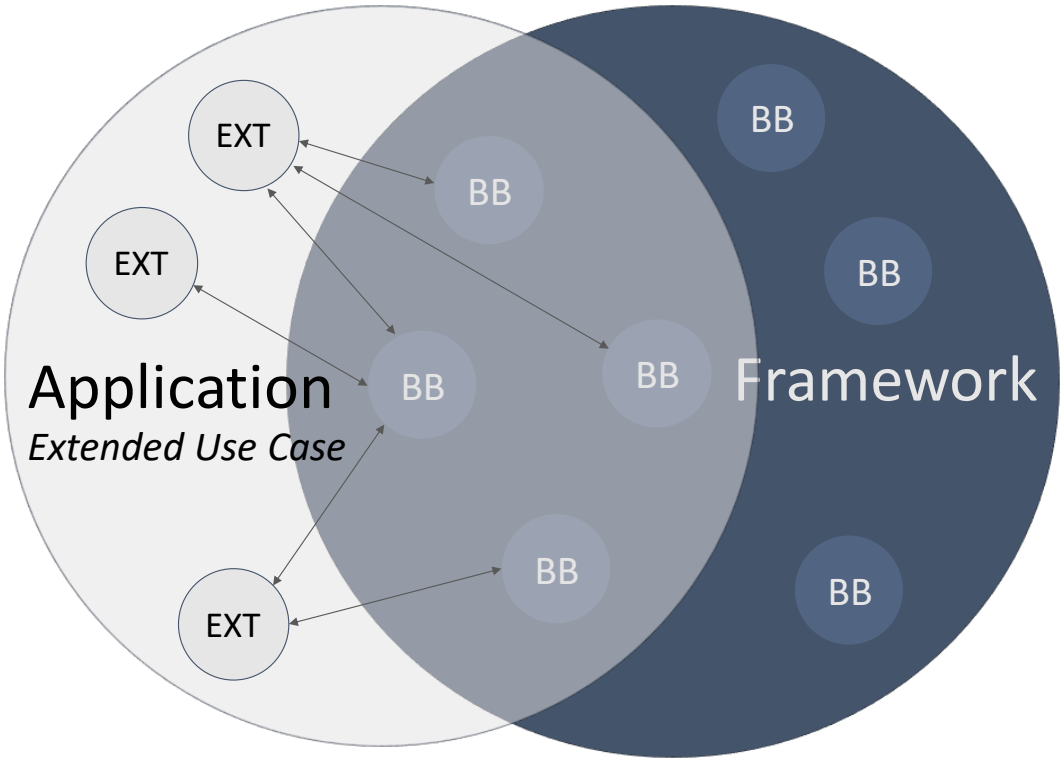

Figure 23: Framework Architecture

A Platform on the other hand offers one central and uniform entry point and user interface for both common and extended use cases (Figure 24). The role of Building Blocks here is mainly to serve for the development of the platform but not for individual applications. In case of CLARITY this means, that the four Demonstration Cases use the CSIS as the platform for carrying out their climate proofing studies. The CSIS as platform supports the core EU-GL process with an initial offer of Data Packages at European-level suitable for simple screening studies. Additional data and feature requirements of the Demonstration Cases lead to the development of tailored Data Packages and "Apps" (tailored or generic ICT Climate Services) that can be integrated into the platform without the need to develop new and separate applications for each new use case. Thereby, the platform is designed to evolve along a data and feature dimension (see chapter 4.1.4).

The main exploitable results are the platform and the individual extension which can be distributed through the CLARITY Marketplace. Related exploitation models are therefore a subscription model for platform usage, selling (generic) feature extensions and transferable Data Packages as well as the development of custom feature extensions and tailored Data Packages ("Expert Services").

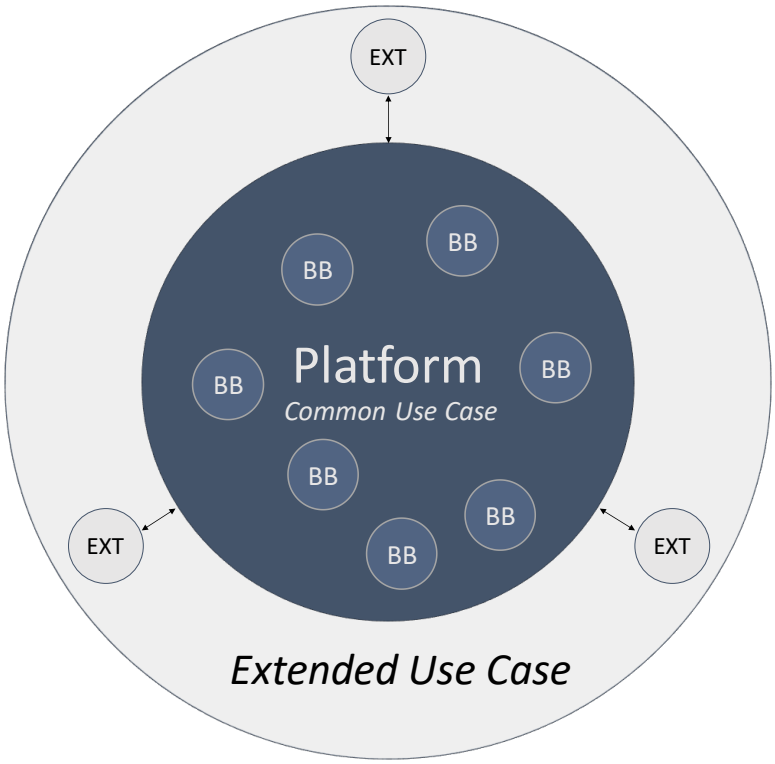

Figure 24: Platform Architecture

5 Realisation

This chapter briefly explains how the architectural concepts and principles introduced chapter 4 are applied to realise the goals formulates in chapter 3. In particular, it presents the Building Blocks of the CSIS that interact in a layered and component based architecture.

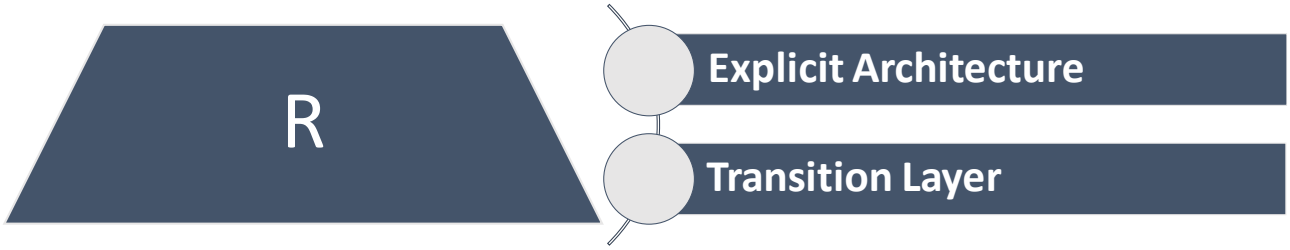

Figure 25: Architectural Perspective of the CSIS Mission

The absence of detailed product specifications, which is inherent to the agile development approach followed by CLARITY (see chapter 2.3), requires some degree of flexibility in the planning process and may involve some contingencies. Although the Realisation of the CSIS belongs to the Explicit Architecture, some details can therefore not be decided upfront and are deferred to the Transition Layer.

5.1 Component-based layered Architecture

In accordance to the concepts of the component-based (4.2.1) and layered (4.2.3) Architecture, the CSIS Architecture is be separated into the following logical layers (Figure 26):

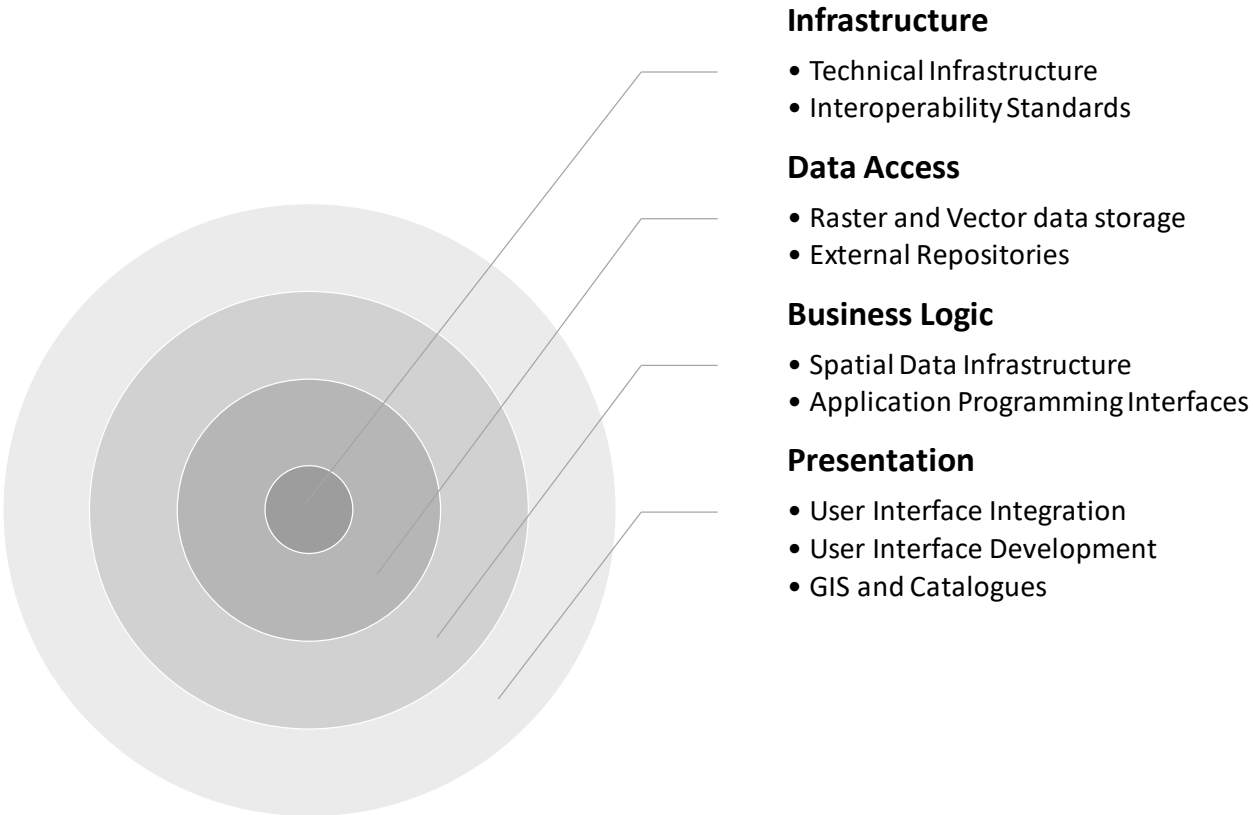

Figure 26: CSIS Architectural Layers

Main purpose of organising Building Blocks into these abstract groups is to differentiate the kinds of tasks that those Building Blocks perform in the CSIS Architecture. Although some Building Blocks are implemented as vertical components and thus span different layers, the intention of the diagram in Figure 27 is to show their logical responsibilities within the CSIS. While the Marketplace Building Block (5.1.2.2) for example also offers graphical user interfaces (Presentation Layer), in the context of the CSIS its role is mainly to offer to APIs (Business Layer) for integrating marketplace functionality (to leverage uptake of Expert Climate Services) into the CSIS.

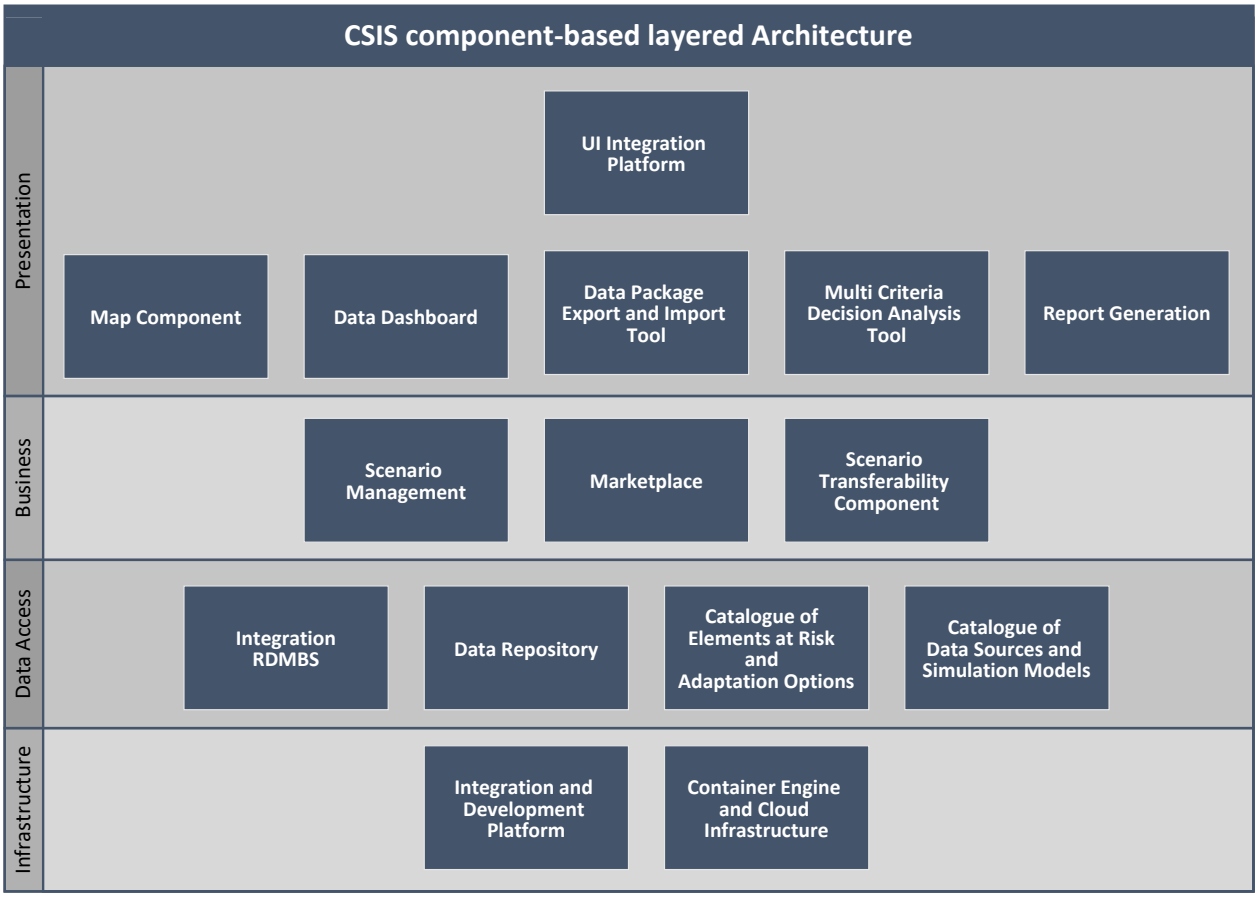

Figure 27: CSIS component-based layered Architecture

This chapter provides thereby a synthetic overview on the Building Blocks that have been defined to realise the goals of the CSIS identified chapter 3. A detailed description of each Building Block is given in D4.1 "Technology Support Plan" [25]. Among others, this description includes also a list of functional requirements that yielded form baseline requirements elicitation and the assessment of presently available Test Cases. Since this information is mainly of interest for developers participating in the agile co-creation process, it is omitted in this document. Instead, this chapter provides "just enough" information for all stakeholders of the CSIS Architecture.

5.1.1 Presentation Layer

The Presentation Layer of the CSIS Architecture contains user interface and user interaction components for Climate Service Customers and -Suppliers. These Building Blocks make use of the related Building Blocks in the Business Layer, either by embedding a user interface or by calling a service interface (API).

5.1.1.1 UI Integration Platform

CLARITY's common User Interface Integration Platform is the unified entry point to the CLARITY ecosystem. It integrates the different frontends (user interfaces) of CLARITY Building Blocks and ICT Climate Services, respectively, with the CLARITY Marketplace and the CSIS.

5.1.1.2 Map Component

The Map Component is understood as a reusable, flexible and highly configurable Building Block meant to be used throughout CSIS. It is envisioned as an embeddable component that can be easily adapted to different parts of the common CSIS UI. The core functionalities of this component must be a clear and easy visualization of different maps and layers. It is also a key feature of the map component to allow for a degree of interactivity meant to enable users to better define locations, elements at risk, hazards, scenario results, etc.

5.1.1.3 Data Dashboard

The Data Dashboard Building Block provides an overview of all the different datasets that are used, produced, ordered, collected, requested, exchanged etc. by an end user (e.g. project planner or climate resilience manager) during an assessment (planning session). Datasets are organised (e.g. in a folder-like structure) according to their relation to the modules of the EU-GL (e.g. hazard maps, impact scenario results, elements of risk inventory).

5.1.1.4 Data Package Export and Import Tool

The Data Package Export and Import Building Block is a tool that can used at any stage of the adaptation planning process to export (download) any data that is directly available in the CSIS in standardised format, the CLARITY Data Package (4.2.4). It can furthermore be used to import additional Data Packages prepared by experts and available from the Marketplace.

5.1.1.5 Multi Criteria Decision Analysis Tool

The Multi Criteria Decision Analysis Tool supports the analysis and comparison of (adaptation) scenarios regarding performance indicators that can be defined by the end user and thus leverages what-if decision support to investigate the effects of adaptation measures and risk reduction options in the specific project context, and allows the comparison of alternative strategies. Thereby the tool provides multi-criteria ranking functions to compare and rank different scenarios and corresponding adaptation plans according to different criteria and their relative weight and level of importance.

5.1.1.6 Report Generation

The result of a climate adaptation study is a report that could be (semi-)automatically generated. Report Generation should enable the user to easily access and download draft and final reports packages at the end of the project assessment process. Such report packages should include automatically generated documentation (with embedded supporting tables, graphs and maps of the area under study).

5.1.2 Business Logic Layer

The Business Logic Layer contains Building Blocks that offer public service interfaces (APIs) or embeddable user interface components which can be used by Building Blocks in the Presentation Layer. The related components implement most of the (server-side) business logic of the CSIS.

5.1.2.1 Scenario Management

The Scenario Management Building Block supports and enforces first and foremost the standardised workflows of the EU-GL [1] for each of the distinct planning steps and provides respective user interfaces that guide the user through the process of co-creating a Climate Adaptation Study. Basically, the end user is presented with the list of recommended and required steps for performing a complete Climate Adaptation Study for the respective (infrastructure) project under assessment and is asked to provide the information that is needed to complete the current step and advance to the next step.

5.1.2.2 Marketplace

The Marketplace Building Block represents a collaborative web platform where urban infrastructure or transport projects could check their climate-proof capabilities and get valuable information for decision making by the Climate Services available in the Marketplace. Users can register into the Marketplace as data or service (human/software) suppliers by describing each service or data set metadata within a personal services portfolio. A 'matchmaking' functionality available in the Marketplace enables customers to find suitable Expert Climate Services (e.g. tailored advisory services) and relevant climate data and ICT Climate Services (e.g. software) available in the Marketplace to be used as inputs on the EU-GL compliant workflow provided by CSIS in order to find available adaptation measures that could minimise the impact on the infrastructures under study.

5.1.2.3 Scenario Transferability Component

The Scenario Transferability Component can be used for discovery and matchmaking of related entities like scenarios, projects, elements at risk, adaptation options, etc. For example, applied to the Catalogue of Elements at Risk and Adaptation Options (5.1.3.3), infrastructure projects being assessed by end users can be matched to other projects that share the same elements at risk (covering a variety of sectors). It can also be used for (visual) scenario analysis and comparison. Thereby, it allows the side-by-side comparison not only of different climate scenarios (Climate Twins Concepts), but also the comparison of alternate adaptation scenarios resulting from Impact Scenario Analysis as described in EU-GL Module 4 "Assess Risks and Impact" and in chapter 3.3 "Risk Assessment and Impact Scenario Analysis" of D3.1 [34].

5.1.3 Data Access Layer

The Data Access Layer is concerned with the storage and management of data and meta-data (e.g. catalogue data).

5.1.3.1 Integration RDMBS

The Integration RDMBS is the central relational database management system for management and integration of common and shared information stored as relations (in tabular form). It stores, among others, the individual infrastructure project configurations and the associated assessment and adaptation planning information created by end users. Thereby, it is important to highlight, that the actual datasets generated during the EU-GL/CLARITY adaptation planning process (hazards maps, model outputs, etc.) are not stored in this Integration RDMBS but in general in a separate Data Repository (5.1.3.2).

5.1.3.2 Data Repository

This Building Block represents a set of generic data repositories that can be used to store, manage, and retrieve different types of (file-based) vector and raster datasets. Among others, this Building Block is used to facilitate the sharing of datasets between users and providers of Climate Services.

5.1.3.3 Catalogue of Elements at Risk and Adaptation Options

The Catalogue of Elements at Risk and Adaptation Options is strongly linked to the EU-GL modules/steps "Characterise Hazard", "Evaluate Exposure", "Vulnerability Analysis", "Assess Risks and Impact" and "Identify Adaptation Options" (Figure 18: EU-GL Workflow) as the actions to be carried out in these steps (except for "Characterise Hazard") are based on the respective elements at risk types or inventories of elements at risk. The catalogue is capable of handling geo-data (e.g. points, lines, grids, political areas, etc.) which is especially relevant for the handling of the elements at risk (e.g. points for building locations, lines for roads/transport networks, grids for population densities, etc.

5.1.3.4 Catalogue of Data Sources and Simulation Models

The Catalogue of Data Sources and Simulation Models is a meta-data catalogue that makes climate-related information accessible by providing functionalities to streamline publishing, sharing, finding and using data and models. The catalogue can be used for data discovery and meta-data storage by different Climate Services and Building Blocks, respectively.

5.1.4 Infrastructure Layer

The Infrastructure Layer of the CSIS is described in detail in Annex 3 of D1.1 “Initial workshops and the CLARITY development environment” [24]. It covers the complete technical infrastructure needed to develop and operate the CSIS.

5.1.4.1 Integration and Development Platform

The purpose of this Building Bock is to provide a continuous integration platform allowing every consortium partner to be equipped with the tools and measures for best practices in software engineering. One of the most important factors on a successful IT project is to develop high quality software. Thereby, an appropriate development infrastructure and best practices are crucial in development in a distributed environment.

5.1.4.2 Container Engine and Cloud Infrastructure

The CSIS is envisioned to be composed of a set of (micro) services that can independently be deployed as isolated containers either on a self-hosted physical server that provides its own container engine or in a virtualized environment offered by a cloud-hosting provider. For this purpose, a Container Engine and Cloud Infrastructure Building Block must be part of the CSIS Architecture.

5.2 Software Components and Key Technologies

In the course of the preparation of the Technology Support Plan, “the CLARITY technology support team first performed a critical assessment of the background technologies inherited from former Research & Development projects regarding their principal suitability for the implementation of the envisaged innovative products and services. To fill the gaps of essential Building Blocks not considered in the initial CLARITY work plan and to supplement initially foreseen background that does not offer a sufficient level of technological readiness or fitness for purpose, the team selected market-ready technologies and software components for the implementation of the respective Building Blocks.” [26]

The Software Components and Key Technologies foreseen for the realisation of the CSIS are briefly presented in this chapter. However, it lies within the nature of the agile approach followed in CLARITY that this selection as well as the actual choices (see chapter 6.1) cannot be cast in stone and are therefore part of the Transition Layer.

5.2.1 User Interface Integration

Key technologies and products for realisation of the User Interface Integration Concept (4.2.5).

5.2.1.1 Rich Internet Applications (RIA)

“A Rich Internet Applications (RIA) is a web application, which uses data that can be processed both by the server and the client. Furthermore, the data exchange takes place in an asynchronous way so that the client stays responsive while continuously recalculating or updating parts of the user interface. On the client, RIAs provide a similar look-and-feel as desktop applications and the word “rich” means particularly the difference to the earlier generation of web applications.” [23]

5.2.1.2 Drupal

“Drupal is an open-source (free) content-management framework. Drupal has great standard features, like easy content authoring, reliable performance, and excellent security. But what sets it apart is its flexibility; modularity is one of its core principles. Its tools help you build the versatile, structured content that dynamic web experiences need. It's also a great choice for creating integrated digital frameworks. You can extend it with any one, or many, of thousands of add-ons. Modules expand Drupal's functionality. Themes let you customize your content's presentation.”
https://www.drupal.org/about

5.2.2 User Interface Development

Key technologies for implementing interactive user interfaces that can be seamlessly embedded into the UI Integration Building Block (5.1.1.1).

5.2.2.1 Drupal Modules

“A Drupal module is a collection of files containing some functionality and is written in PHP. Because the module code executes within the context of the site, it can use all the functions and access all variables and structures of Drupal core. In fact, a module is no different from a regular PHP file that can be independently created and tested and then used to drive multiple functionalities. This approach allows Drupal core to call at specific places certain functions defined in modules and enhance the functionality of core.”
https://www.drupal.org/docs/user_guide/en/understanding-modules.html

5.2.2.2 React

“React is a JavaScript library for building user interfaces. Declarative views make your code more predictable, simpler to understand, and easier to debug. Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM. React is flexible and can be used in a variety of projects. React can also render on the server using Node and power mobile apps using React Native.” https://github.com/facebook/react/

5.2.2.3 Angular

“Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.”
https://angular.io/docs

5.2.2.4 eCharts

"eCharts is a free, powerful charting and visualization library offering an easy way of adding intuitive, interactive, and highly customizable charts to your commercial products. It is written in pure JavaScript and based on zrender, which is a completely new lightweight canvas library."
https://github.com/ecomfe/echarts

5.2.2.5 Flamingo 4

"Flamingo 4 is an open source geo content management solution. It allows non-technical administrators to manage the way geospatial data is published in the Flamingo 4 web-viewer. The web based viewer can be configured by dragging and dropping components in a layout."
https://github.com/flamingo-geocms/flamingo/wiki

5.2.3 GIS and Catalogues

Products and development tools for the realisation of geospatial information systems and (meta-) data catalogues.

5.2.3.1 Mapbox GL

"Mapbox GL is a suite of open-source libraries for embedding highly customizable and responsive client-side maps in Web, mobile, and desktop applications. Maps render at a super high framerate. You can use custom styles designed in Mapbox Studio. You can also manipulate every aspect of the style's appearance on the fly, because Mapbox GL renders vector tiles. The abbreviation "GL" comes from OpenGL, the industry-standard Open Graphics Library."
https://www.mapbox.com/help/define-mapbox-gl/

5.2.3.2 Leaflet

"Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps. Weighing just about 38 KB of JS, it has all the mapping features most developers ever need. Leaflet is designed with simplicity, performance and usability in mind. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to."
https://leafletjs.com/

5.2.3.3 CKAN

"CKAN is a tool for making open data websites. It helps you manage and publish collections of data. It is used by national and local governments, research institutions, and other organizations who collect a lot of data. Once your data is published, users can use its faceted search features to browse and find the data they need, and preview it using maps, graphs and tables - whether they are developers, journalists, researchers, NGOs or citizens." http://docs.ckan.org/en/latest/user-guide.html

5.2.3.4 ckanext-geoview CKAN extension

"ckanext-geoview is a CKAN (see chapter 5.2.3.3) extension that contains view plugins to display geospatial files and services in CKAN. It contains an OpenLayers based viewer originally developed by Philippe Duchesne and other view plugins like Leaflet.js that used to be part of ckanext-spatial (see chapter 5.2.3.5)."
https://github.com/ckan/ckanext-geoview

5.2.3.5 ckanext-spatial CKAN extension

"The ckanext-spatial CKAN extension contains plugins that add geospatial capabilities to CKAN, including aspatial field on the default CKAN dataset schema (uses PostGIS as the backend), harvesters to import geospatial metadata into CKAN from other sources in ISO 19139 format and others and commands to support the CSW standard using pyCSW."
https://github.com/ckan/ckanext-spatial

5.2.4 API Development

Key technologies and products for the development of service interfaces offered by Building Blocks of the Business Logic Layer (5.1.2) of the CSIS.

5.2.4.1 RESTful web services

"Representational state transfer (REST) or RESTful web services are a way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations."
https://en.wikipedia.org/wiki/Representational_state_transfer

5.2.4.2 Asynchronous JavaScript and XML

"AJAX is an acronym for Asynchronous JavaScript and XML, which already contains two technologies and a technique for loading information. AJAX is not a single new technology, but a combination of standard technologies including HTML, CSS, JavaScript, XML and DOM which together with the XMLHttpRequest object achieve web application richness. AJAX applications work unconditionally in browsers without the need to install any plug-ins." [23]

5.2.4.3 Drupal RESTful Web Services API

"The RESTful Web Services API is part of the core functionality in Drupal 8. It builds on top of Drupal 8's Serialization module to provide a customizable, extensible RESTful API of data managed by Drupal. Out of the box, it allows you to interact with any content entity (nodes, users, comments …) or config entity (vocabularies, user roles…) as well as watchdog database log entries."
https://www.drupal.org/docs/8/api/restful-web-services-api/

5.2.4.4 Drupal Form API

"For an interactive, data-driven website such as one built with Drupal, collecting and processing user submitted data will be exceptionally important. Most of this data can be captured using a web based form, i.e. An HTML structure with text fields and widgets for selecting different options. Getting a form on to a webpage is easy, getting the user's responses is easy too, getting them securely is much, much harder. Drupal provides a standard, easy to use, easy to extend and secure way of adding forms to your Drupal website: Form API or FAPI for short."
https://www.drupal.org/docs/user_guide/en/understanding-modules.html

5.2.5 Spatial Data Infrastructure

Products for the implementation of a spatial data infrastructure that supports standards based access to and server-side visualisation of climate- and exposure data.

5.2.5.1 GeoServer

"GeoServer is an OGC compliant implementation of a number of open standards such as Web Feature Service (WFS), Web Map Service (WMS), and Web Coverage Service (WCS).Additional formats and publication options are available including Web Map Tile Service (WMTS) and extensions for Catalogue Service (CSW) and Web Processing Service (WPS)."
https://geoserver.org/

5.2.5.2 MapServer

"MapServer is an Open Source platform for publishing spatial data and interactive mapping applications to the web. Originally developed in the mid-1990's at the University of Minnesota, MapServer is released under an MIT-style license, and runs on all major platforms (Windows, Linux, Mac OS X). MapServer is not a full-featured GIS system, nor does it aspire to be."
https://mapserver.org/

5.2.6 Raster and Vector Data Storage

Products for the storage of all kinds of data needed by the Building Blocks of the Business Logic (5.1.2) and Presentation Layer (5.1.1).

5.2.6.1 AIT EMIKAT

EMIKAT is a client/server application for collecting heterogeneous datasets for a specific project to then manage this data and perform model calculations in a scenario context. Datasets are typically defined in a spatial context including position and a geometric description. For data which is changing over time, EMIKAT manages the historical development in a documented way. EMIKAT has been developed by CLARITY partner AIT.

5.2.6.2 cids Integration Base

"The cids Integration Base is a distributed meta database which consists of a generic meta data model placed in a relational Data Base Management System (RDBMS). It is the basis for a concrete information system and is able to describe arbitrary objects (real-world objects, services, models, geographical features, other information systems, etc.), their attributes (e.g. geographical location) and relationships by means of so-called meta classes and objects." https://www.cismet.de/cidsReadme.html

5.2.6.3 PostgreSQL

"PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, and exceptional documentation."
https://www.postgresql.org/about/

5.2.6.4 PostGIS

"PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL. In addition to basic location awareness, PostGIS offers many features rarely found in other competing spatial databases such as Oracle Locator/Spatial and

SQL Server. “
https://postgis.net/

5.2.6.5 ERDDAP

“ERDDAP is a data server that gives you a simple, consistent way to download subsets of gridded and tabular scientific datasets in common file formats and make graphs and maps.”
http://coastwatch.pfeg.noaa.gov/erddap/information.html

5.2.7 Technical Infrastructure

Products that are solely used in the Infrastructure Layer (5.1.4).

5.2.7.1 Docker

“Docker is a software technology providing operating-system-level virtualization also known as containers, promoted by the company Docker, Inc. Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux. Docker uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines (VMs).”
https://en.wikipedia.org/wiki/Docker_(software)

5.2.7.2 Kubernetes

“Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available. Google open-sourced the Kubernetes project in 2014.”
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

5.2.7.3 Apache Maven

“Apache Maven is an Open Source management and build tool for Java projects based on XML configuration files. It uses a Project Object Model (POM) to describe the software project, its dependencies, modules and external components. It downloads modules dynamically from repositories and it is capable of upload artefacts to the final repository after building.”
https://maven.apache.org/

5.2.7.4 Nexus

“Sonatype Nexus is a repository manager for software "artefacts" required for development. It collects and manages software dependencies making easy to distribute software components for a collaborative software development environment.”
https://www.sonatype.com/nexus-repository-oss

5.2.7.5 SonarQube

“SonarQube is an open source platform for continuous code quality inspection, to perform automatic reviews and code. It manages rules, exclusions, alerts, thresholds and allows combining different metrics. It covers main topics of code quality like duplications, unit tests, complexity, potential bugs, coding rules, etc.”
https://www.sonarqube.org/

5.2.7.6 Selenium-Grid

"Selenium-Grid is a software-testing framework for web applications. It allows running parallel tests on different machines and even different web browsers at once. All this features make it appropriate for distributed environment tests execution."
https://github.com/

5.2.7.7 Gulp

"Gulp is an Open Source JavaScript build system working over NodeJS to automatize common development tasks like code minifying, web browser reload, source code validating and image compression among others."

https://gulpjs.com/

5.2.8 Interoperability Standards

Interoperability standards selected for the relation of CLARITY's data-driven approach (4.2.4).

5.2.8.1 Data Package

"Data Package is a simple container format used to describe and package a collection of data. The format provides a simple contract for data interoperability that supports frictionless delivery, installation and management of data. Data Packages can be used to package any kind of data. At the same time, for specific common data types such as tabular data it has support for providing important additional descriptive metadata - for example, describing the columns and data types in a CSV."
https://frictionlessdata.io/data-packages/

5.2.8.2 OGC GeoPackage

"A GeoPackage is an open, standards-based, platform-independent, portable, self-describing, compact format for transferring geospatial information. It is a platform-independent SQLite database file that contains the GeoPackage data and metadata tables."
http://www.geopackage.org/spec/

5.2.8.3 JavaScript Object Notation (JSON)

"JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language."
https://www.json.org/

6 Implementation

This chapter briefly summaries the artefacts that serve for documenting the Emergent Architecture and explains where these artefacts can be found online or how and when they will be made publicly available. The deliverables D4.3 & D4.4 "Technology Support Report" will provide an update of this documentation.

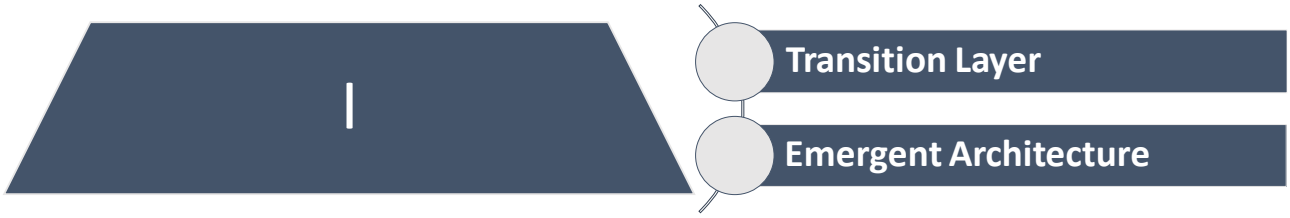

Figure 28: Architectural Perspective of the CSIS Implementation

While most of these artefacts like Mock-Ups or source code emerge during the agile co-creation process, some of them are the result of preliminary upfront planning and thus are part of the Transition Layer. This encompasses mainly the technology choices for the implementation of Building Blocks.

6.1 Technology Choices

The preliminary technology choices in terms of software components and key technologies (5.2) for the realisation of Building Blocks (5.1) have been documented in D4.1 "Technology Support Plan" [26]. To foster a common understanding of the CSIS Architecture, the CSIS Architecture document gives a brief overview with help of simple "white box" diagrams (Figure 29).

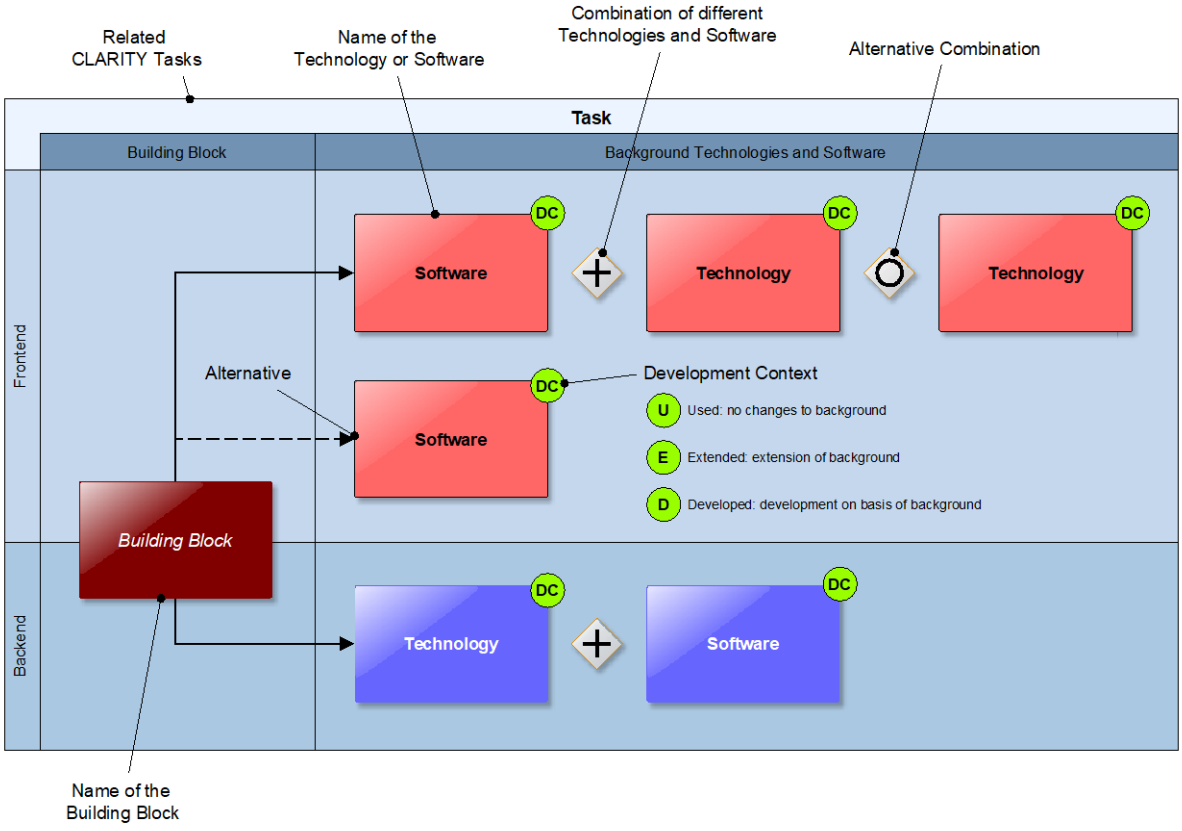

Figure 29: Technology Support Plan Overview Diagram

The Technology Support Plan provides detailed and technical explanations of the different options. CLARITY stakeholders directly involved in the agile development process may therefore refer to D4.1 "Technology Support Plan" [26]. As the preliminary plan is based on technology assessment, best practices, experience and the evaluation of spike solutions (see chapter 2.4), it is part of Transition Layer between Explicit and Emergent Architecture. Once been validated and possibly changed in the course of the agile development process, it will become part of the Emergent Architecture and documented in the Technology Support Reports (D4.3 and D4.4).

6.1.1 Presentation Layer

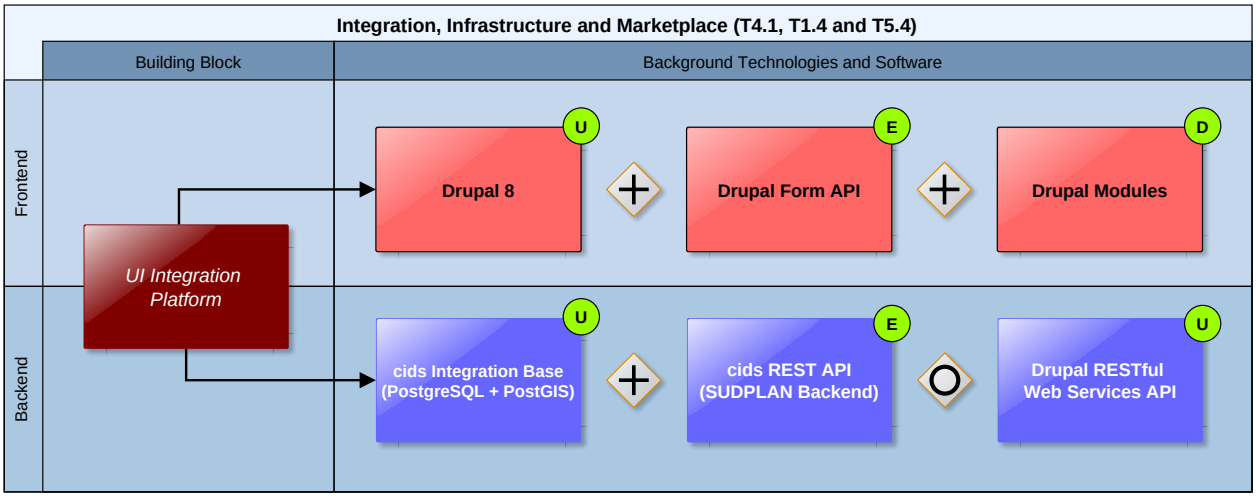

Figure 30: UI Integration Platform Technology Support

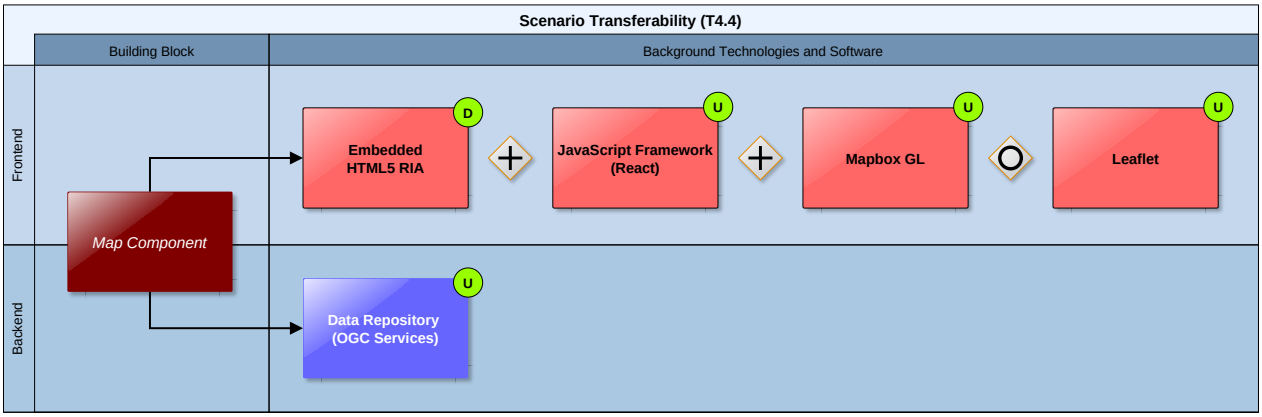

Figure 31: Map Component Technology Support

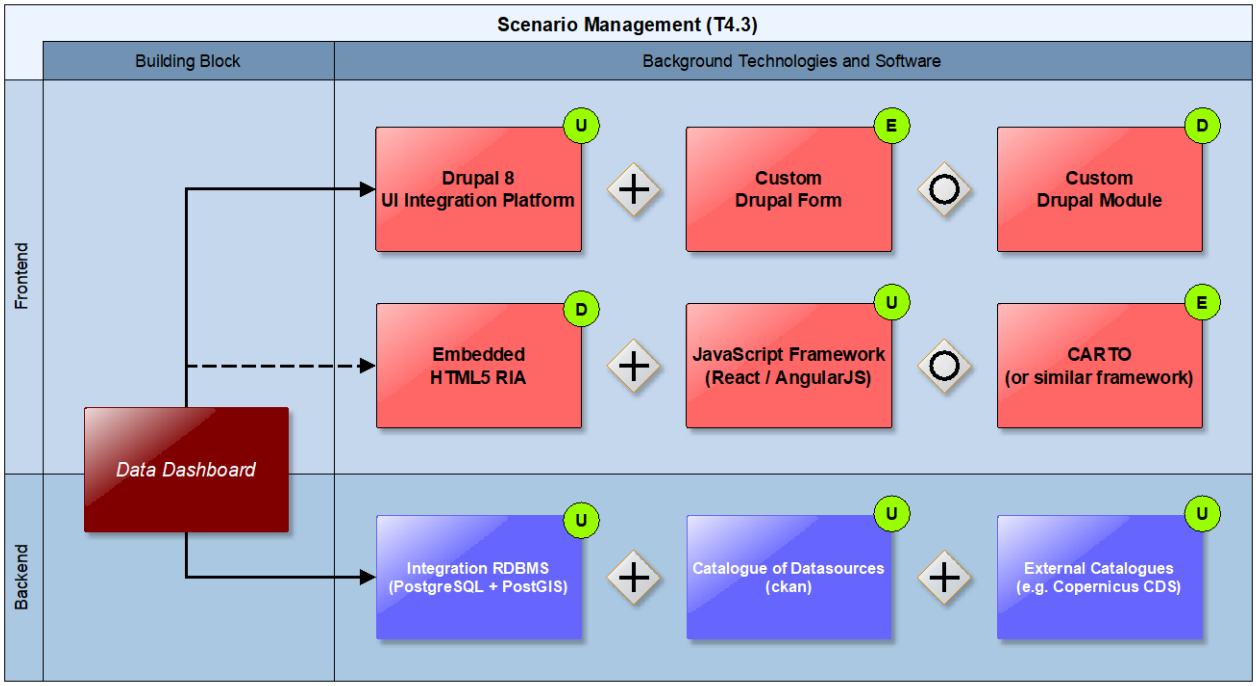

Figure 32: Data Dashboard Technology Support

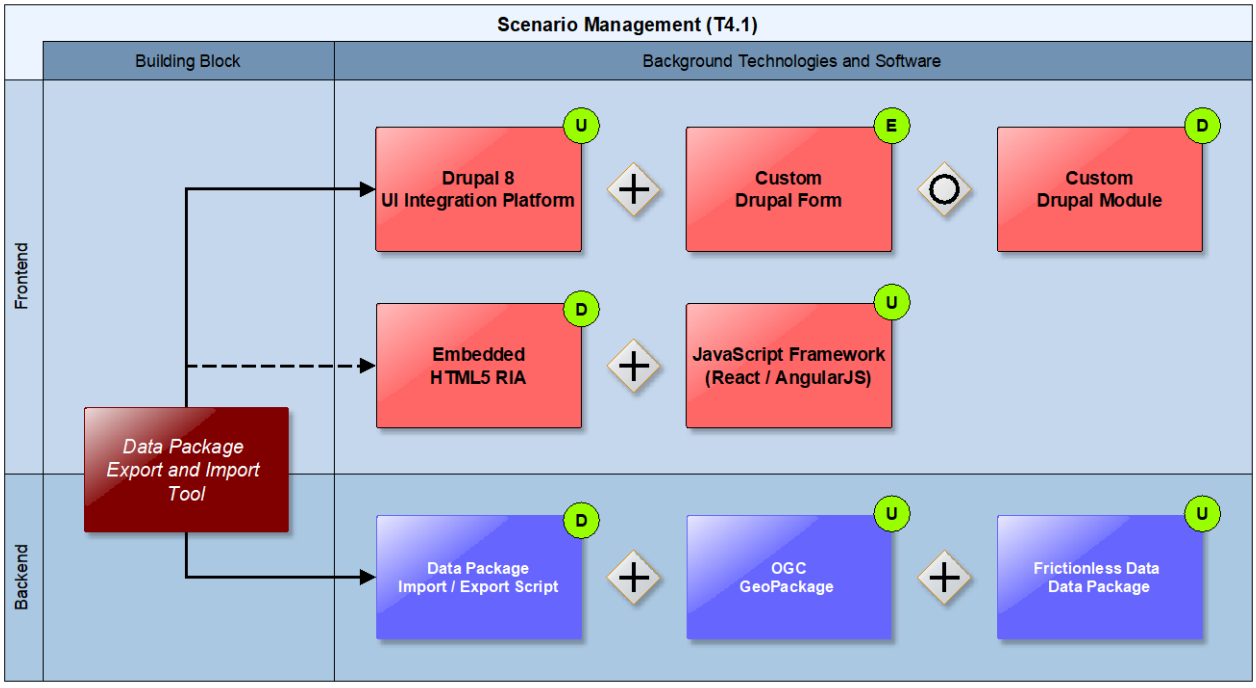

Figure 33: Data Package Export and Import Tool Technology Support

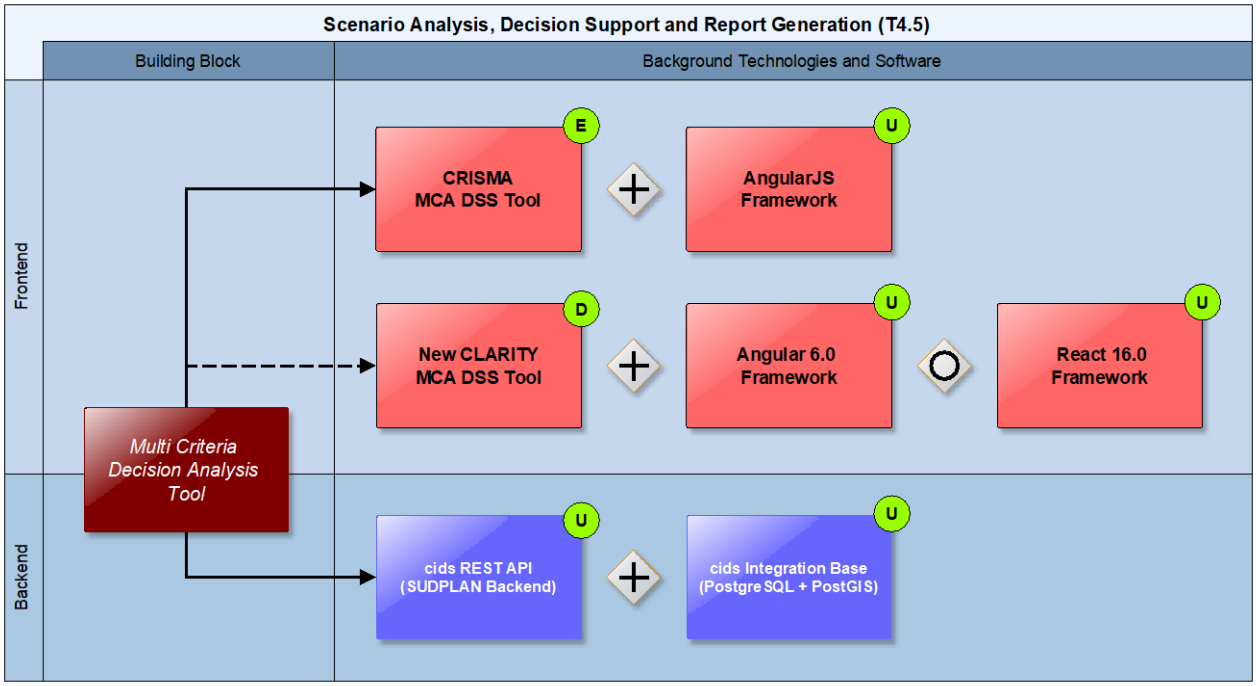

Figure 34: Multi Criteria Decision Analysis Tool Technology Support

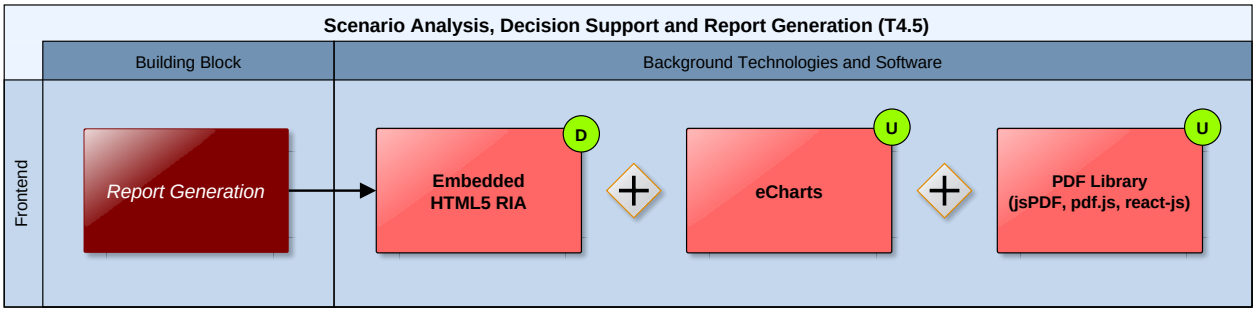

Figure 35: Report Generation Technology Support

6.1.2 Business Logic Layer

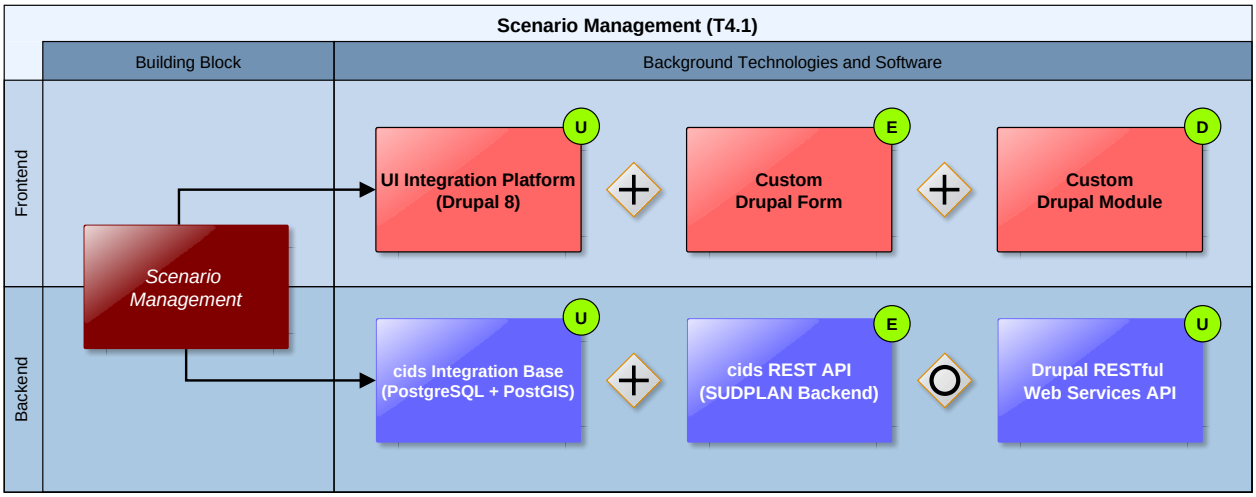

Figure 36: Scenario Management Technology Support

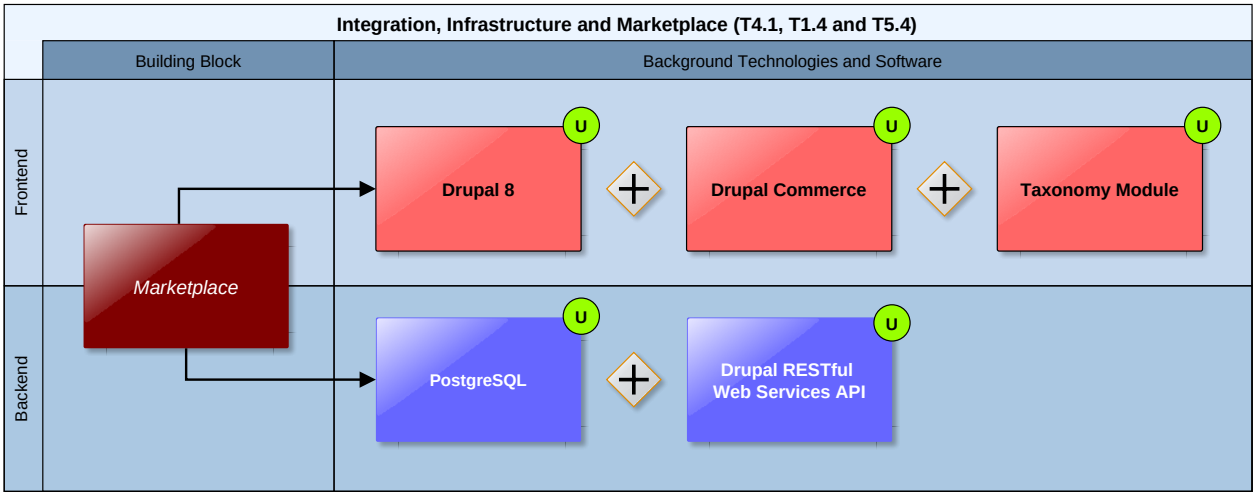

Figure 37: Marketplace Technology Support

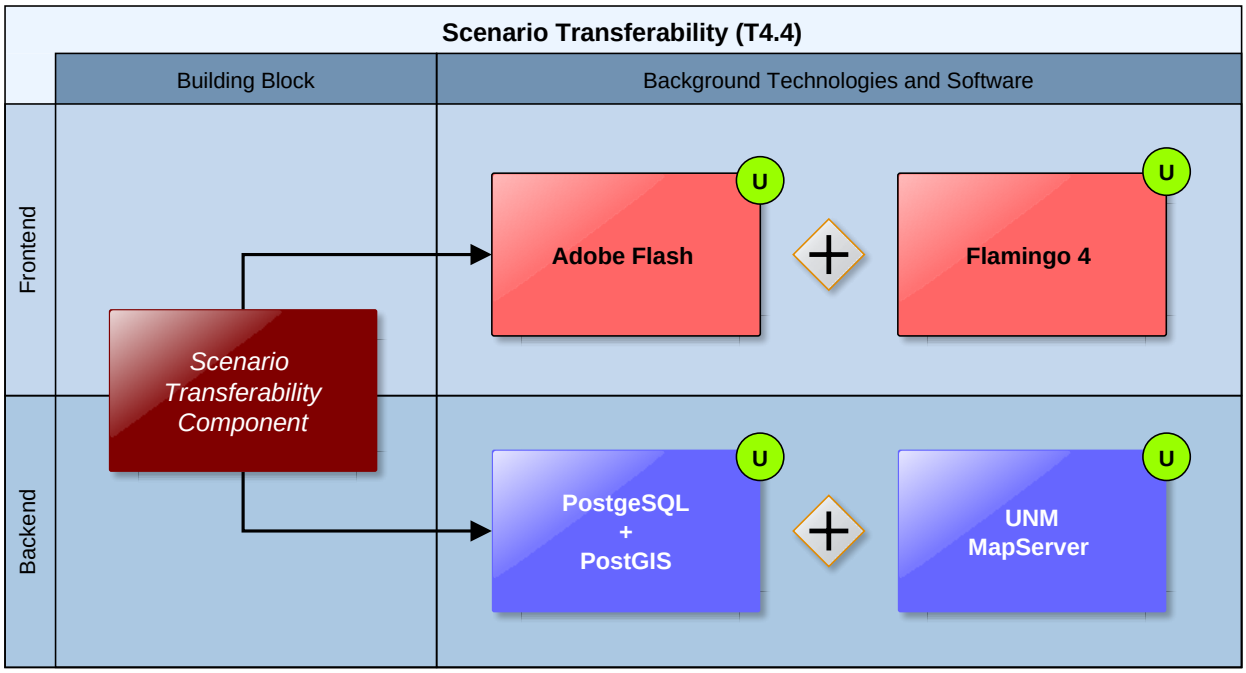

Figure 38: Scenario Transferability Component Technology Support

6.1.3 Data Access Layer

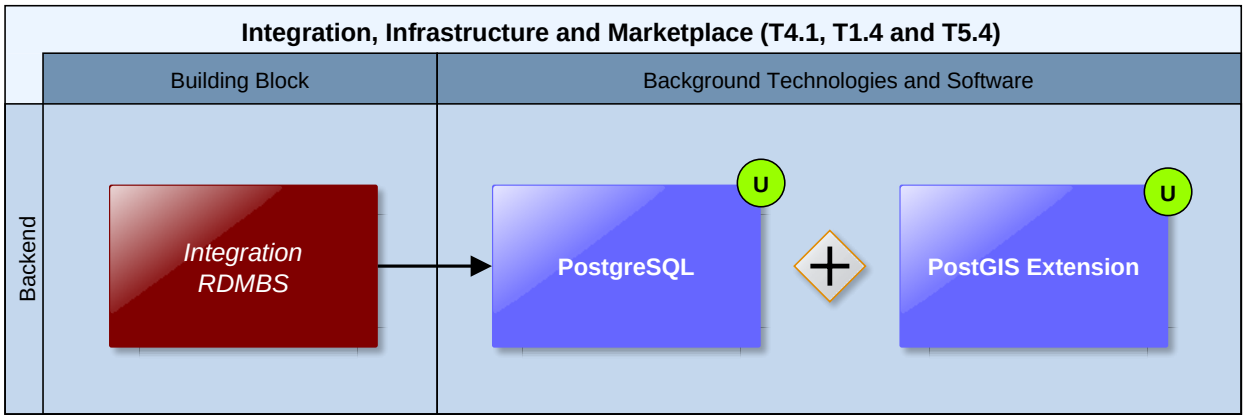

Figure 39: Integration RDMBS Technology Support

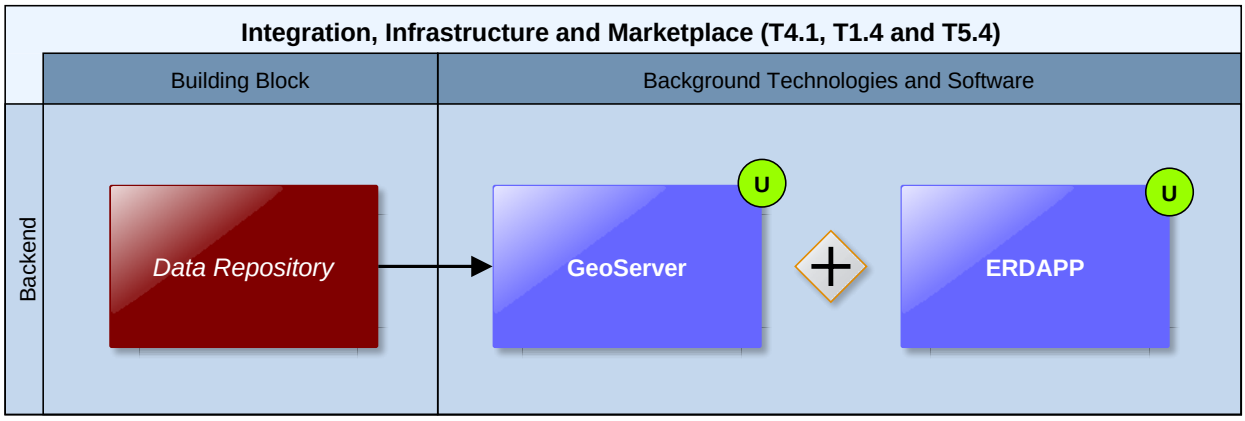

Figure 40: Data Repository Technology Support

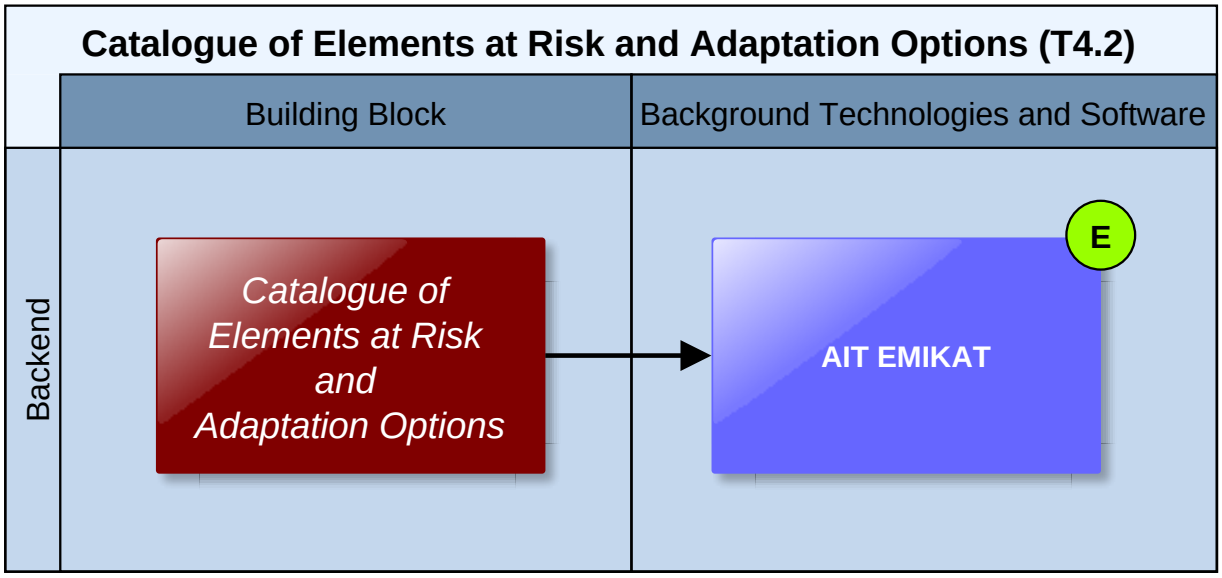

Figure 41: Catalogue of Elements at Risk and Adaptation Options Technology Support

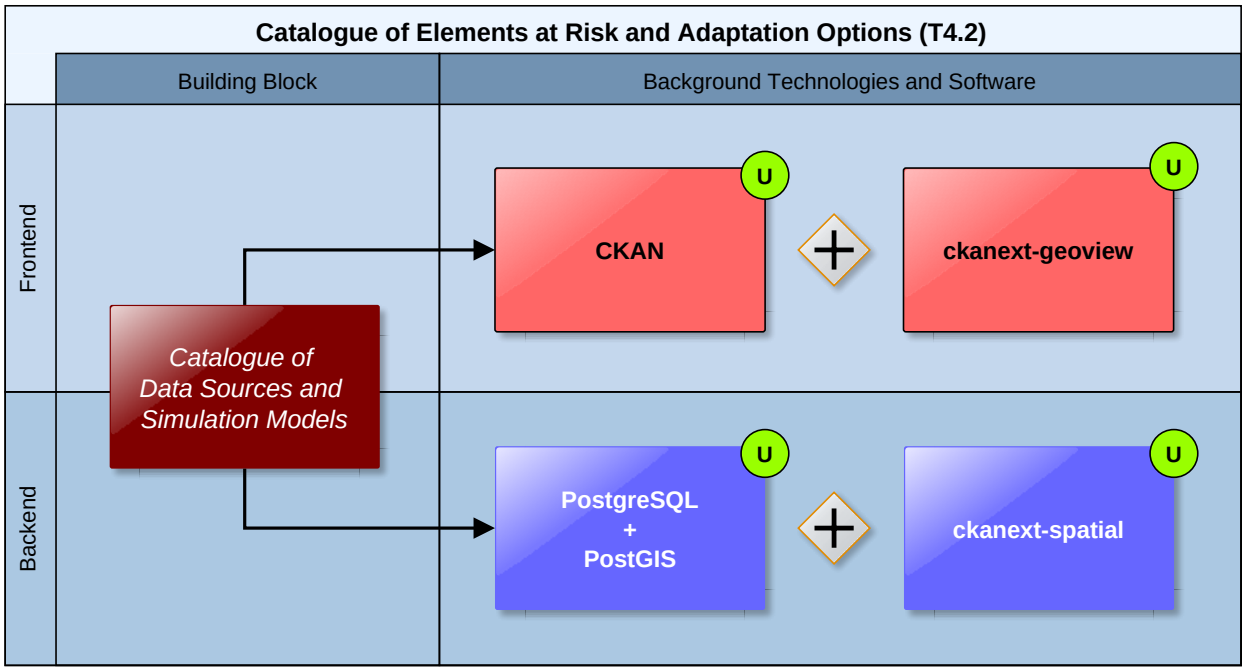

Figure 42: Catalogue of Data Sources and Simulation Models Technology Support

6.1.4 Infrastructure Layer

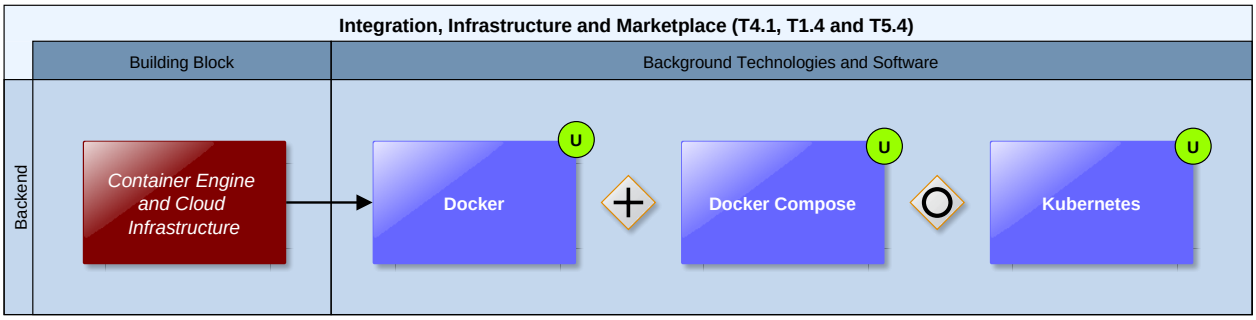

Figure 43: Container Engine and Cloud Infrastructure Technology Support

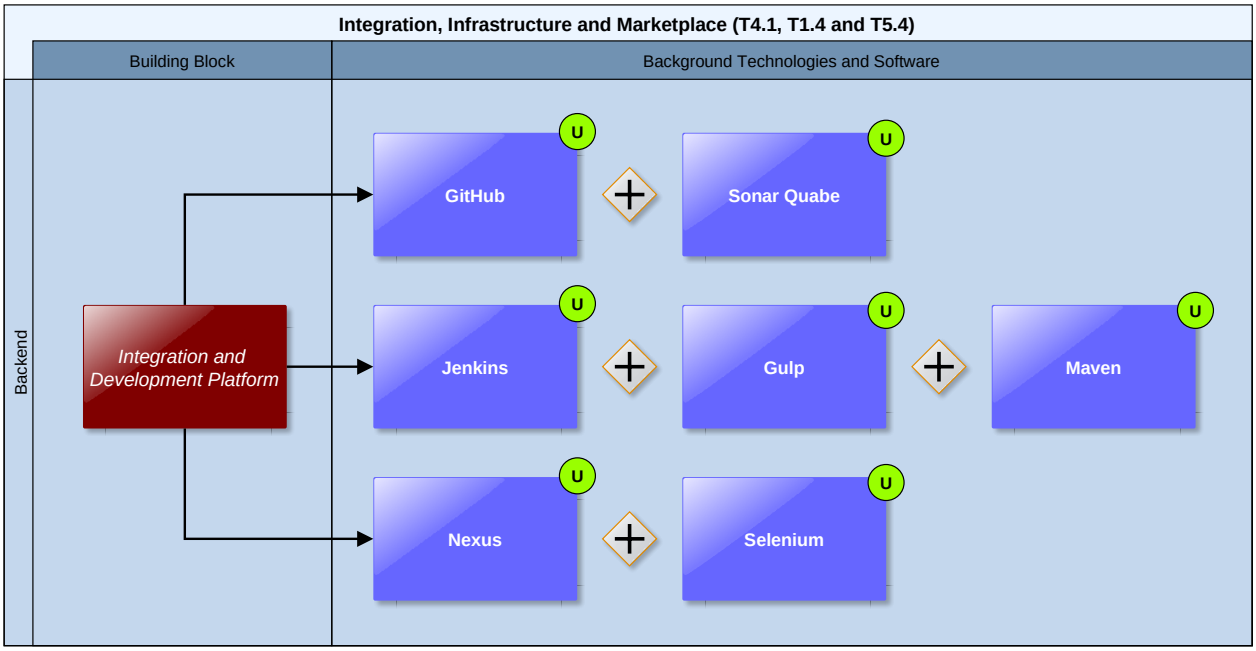

Figure 44: Integration and Development Platform Technology Support

6.2 Mock-Ups

As explained in chapter 2.3, Mock-Ups offer a visual preview of the envisaged products and services. Thereby, they are not only helpful for early feedback from end users but serve also developers to select and prioritize the features to be developed during each agile iteration. Thus, Mock-Ups contribute also to the documentation of the Emergent Architecture. They are currently stored in the internal filed-based CLARITY OwnCloud repository and will eventually be made available on the CLARITY coordination platform (http://cat.clarityCLARITY-h2020.eu/).

Figure 45 shows an example of a Mock-Up for the definition of the geospatial area of an infrastructure project (urban planning) under assessment. As software for creating Mock-Ups, Baslamiq[9] is used. As "low-fidelity wireframing tool", Baslamiq fits perfectly into the iterative, lean and agile approach followed in the CSIS Architecture: Instead of creating exhaustive detailed specifications of products that will never be developed, Baslamiq lets end users and developers together "wireframe the key screens, implement them, see how they feel and go back to the wireframes to tweak them if needed" (https://balsamiq.com/products/).

[9] "Balsamiq Mock-Ups is a quick, low-fidelity wireframing tool which can be used to wireframe any kind of software interface, be it for the desktop, web, mobile, kiosks, etc. We intentionally offer "just enough" prototyping capabilities, but not more. We believe that wireframing + fast iterations with real code is much better than prototyping in the vast majority of cases." https://support.balsamiq.com/sales/howtochoose/

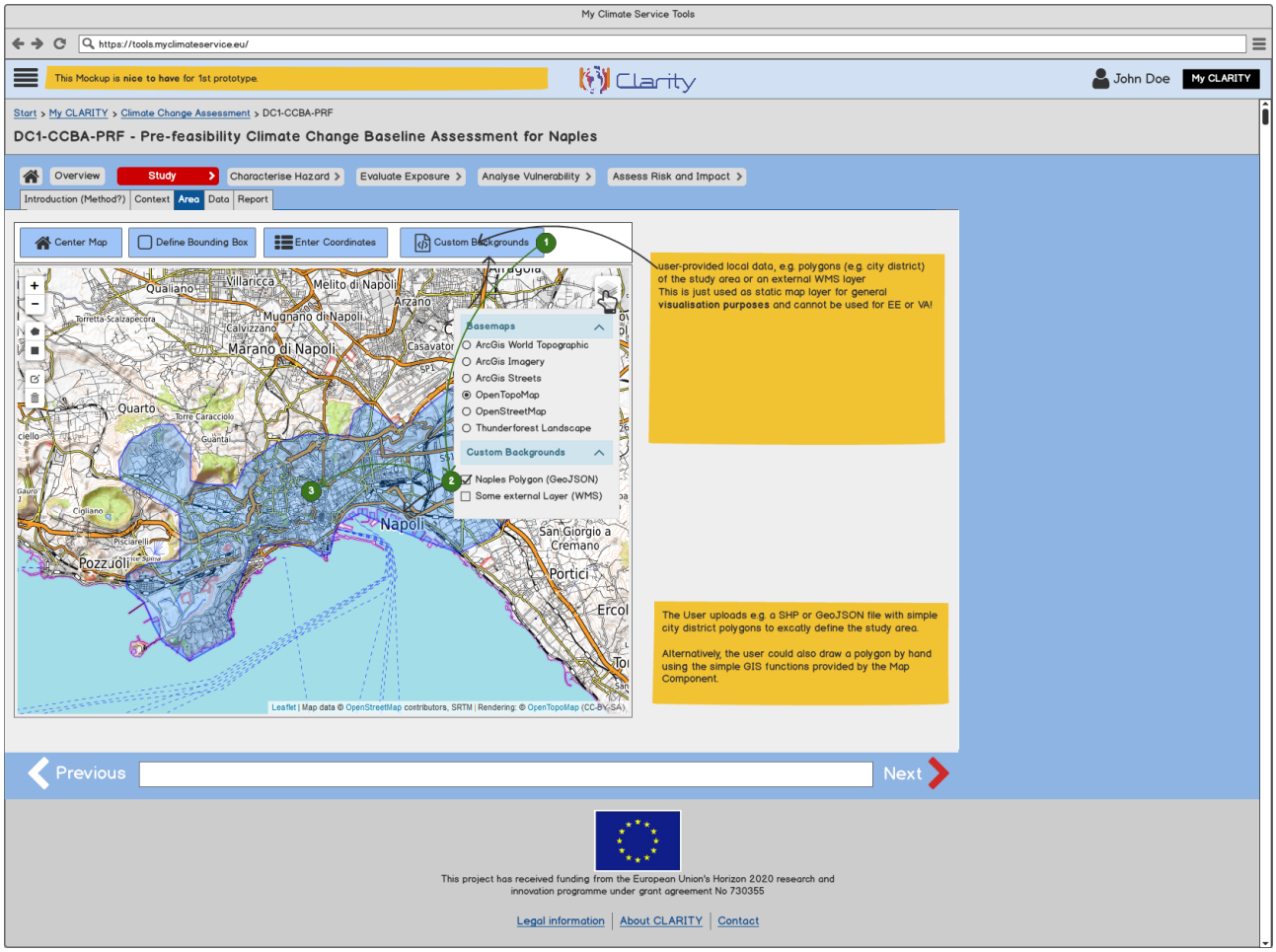

Figure 45: Mock-Up Example

6.3 Test Cases

As explained in chapter 2.3, Test Cases are not a direct concept of Agile Software Development. The initial Test Cases of D1.2 "Database of Initial CLARITY CSIS User Stories and Test Cases" [27] however, were useful to derive functional requirements on the Building Blocks described in chapter 5.1. In the Emergent Architecture, they can be further maintained serving documentation and validation purposes. Test Cases are documented in the CLARITY coordination platform (http://cat.clarityCLARITY-h2020.eu/).

6.4 Source Code

Source code of adapted newly developed components is available in CLARITY's source code repository on GitHub[10] at https://github.com/clarity-h2020.

[10] "GitHub is a web based version control repository. It provides bug tracking, features request, wiki, issue tracking, task management, etc. It is easily integrated with other tools like Jenkins among others making it a good choice for CID (Continuous Integration & Delivery)." https://github.com/

6.5 Others

Other artefacts that emerge from agile development are for example software releases, snapshot builds, API documentation, unit and integration test specifications and related test results, bug reports, etc. As they closely relate to the integrated development environment (5.1.4.1) the documentation and project management facilities of CLARITY's source code repository (6.4) is used as entry point to these artefacts.

7 Conclusion

Deciding how much effort for architectural description is needed in agile development is a challenge. In CLARITY, we follow therefore an approach towards a lean and self-explanatory architectural documentation that is easier to review, update and communicate. For this purpose, we separate the CSIS Architecture into an Explicit Architecture and an Emergent Architecture. In the Explicit Architecture, we present a high-level solution design that facilitates common understanding and collaboration among all stakeholders by connecting business and domain models with a shared "Product Vision". We defer all non-critical design decisions and technology choices to the Emergent Architecture that iteratively evolves during the agile co-creation process. We furthermore introduce a Transition Layer between these architectural perspectives that anticipates expected changes as opportunity to generate value while preserving the invariant essence of the system.

We structure the architecture documentation according to the MCRI (Mission, Concept, Realisation, and Implementation) principle and define

- the general mission of the CLARITY CSIS in terms of goals, architectural qualities and -constraints, which have been derived from the project objectives, the elicitation and evaluation of Exploitation Requirements and during stakeholder workshops;
- the core concepts applied in the CLARITY CSIS Architecture in terms of the conceptual specification of CLARITY products and services (Innovation Design) and the general principles that are used to design and implement the CSIS;
- the realisation of the goals by means of Building Blocks that interact in a layered and component based architecture; and
- the implementation as summary of those artefacts that serve for documenting the Emergent Architecture.

This document fosters the shared understanding among all CLARITY stakeholders about the CSIS Architecture and equips the CLARITY co-creation teams with the necessary conceptual background information to successfully implement and carry out the agile development process. The deliverables D4.3 & D4.4 "Technology Support Report" will provide update of the Emergent Architecture and thus report on the results of the implementation and integration process carried out in WP4 "Technology Support".

References

[1] Directorate-General Climate Action, “Non-paper Guidelines for Project Managers: Making vulnerable investments climate resilient,” European Comission, 16 April 2013. [Online]. Available: http://climate-adapt.eea.europa.eu/metadata/guidances/non-paper-guidelines-for-project-managers-making-vulnerable-investments-climate-resilient/guidelines-for-project-managers.pdf. [Accessed 6 November 2017].

[2] S. Freudenberg and H. Sharp, “The Top 10 Burning Research Questions from Practitioners,” *IEEE Software,* pp. 8-9, 2010.

[3] ISO/IEC/IEEE 42010, Systems and software engineering - Architecture description, 2011.

[4] ISO/IEC 10746, Reference Model of Open Distributed Processing, 1996.

[5] V. Temnenco, “Software estimation, enterprise-wide,” IBM developerWorks, 20017 July 15. [Online]. Available: https://www.ibm.com/developerworks/rational/library/jul07/temnenco/temnenco-pdf.pdf. [Accessed 19 April 2018].

[6] J. Coplien and G. Bjørnvig, Lean Architecture for Agile Software Development, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom: John Wiley & Sons Ltd, 2010.

[7] S. Stuurman, A. Bijlsma, B. Heeren and E. Roubtsova, “Introduction to Software Architecture,” in *Software Architecture*, O. U. i. t. Netherlands, Ed., Heerlen, 2014, p. 10.

[8] K. Beck, M. Beedle, A. V. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. K. J. Jeffries, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland and D. Thomas, “Manifesto for Agile Software Development,” 2001. [Online]. Available: http://www.agilemanifesto.org. [Accessed 19 April 2018].

[9] J. P. Womack, D. T. Jones and D. Roos, The machine that changed the world: The story of lean production, New York: Harper Perennial, 1991.

[10] P. Avgeriou, C. Yang and P. Liang, “A Systematic Mapping Study on the Combination of Software Architecture and Agile Development,” *Journal of Systems and Software,* January 2016.

[11] P. Abrahamsson, M. Ali Babar and P. Kruchten, “Agility and Architecture: Can They Coexist?,” *IEEE SOFTWARE,* March/April 2010.

[12] A. Aitken and V. Ilango, “A Comparative Analysis of Traditional Software Engineering and Agile Software Development,” in *46th Hawaii International Conference on System Sciences*, Hawaii, 2013.

[13] Moczar and Lajos, “Why Agile Isn't Working: Bringing Common Sense to Agile Principles,” *CIO,* 4 June 2013.

[14] G. Booch, “Handbook of Software Architecture,” 2017. [Online]. Available: https://handbookofsoftwarearchitecture.com/?p=63. [Accessed 19 April 2018].

[15] N. Brown, R. Nord and I. Ozkaya, “Enabling Agility Through Architecture,” *CrossTalk,* pp. 12-17, Nov/Dec 2010.

[16] D. Leffingwell, R. Martens and M. Zamora, “Scaling Software Agility,” 1 July 2008. [Online]. Available: https://scalingsoftwareagility.files.wordpress.com/2008/08/principles_agile_architecture.pdf. [Accessed 2018 15 May].

[17] U. Friedrichsen, “Opportunities, Threats, and Limitations of Emergent Architecture,” in *Agile Software Architecture: Aligning Agile Processes and Software Architectures*, Burlington, Massachusetts, Morgan Kaufmann Publishers, 2014, pp. 335-355.

[18] M. Vincent, “Emergent Architecture – Just Enough Just in Time,” 15 July 2016. [Online]. Available: https://www.agilealliance.org/resources/sessions/emergent-architecture-just-enough-just-in-time/. [Accessed 16 May 2018].

[19] I. Hadar, S. Sherman, E. Hadar and J. J. Harrison, “Less is more: Architecture documentation for agile development,” in *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, San Francisco, CA, USA, 2013.

[20] M. Fowler, “Who Needs an Architect?,” *IEEE SOFTWARE,* 2003.

[21] F. P. Brooks, “No Silver Bullet – Essence and Accident in Software Engineering,” *Computer, Volume: 20, Issue: 4,* pp. 10-19, April 1987.

[22] N. Abu el Ata and M. J. Perks, Solving the Dynamic Complexity Dilemma, Berlin, Heidelberg: Springer-Verlag, 2014.

[23] M. Busch and N. Koch, “Rich Internet Applications - State-of-the-Art,” in *Ludwig-Maximilians-Universität*, München, 2009.

[24] S. Brown, Software Architecture for Developers - Volume 2: Visualise, document and explore your software architecture, Leanpub, 2018.

[25] R. Duro and D. Havlik, “D1.1 Initial workshops and the CLARITY development environment,” Deliverable D1.1 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 5 January 2018. [Online].

[26] P. Dihé, “D4.1 Technology Support Plan,” Deliverable D4.1 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 2018. [Online].

[27] M. Ángel Esbrí, M. Núñez, D. Havilk, R. Duro, P. Dihé, M. Leone, M. Zuvela-Aloise, A. Jorge, L. Strömbäck, I. Torres, L. Torres, Á. Rivera, R. Cortinat and L. Parra, “D1.2 Database of Initial CLARITY CSIS User Stories and Test Cases,” Deliverable D1.2 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), March 2018. [Online].

[28] P. Dihé, “D5.1 Exploitation Requirements and Innovation Design,” Deliverable D5.1 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 2017. [Online].

[29] J. Alonso, J. Lopez and A. Geyer-Scholz, “D5.3 Exploitation and Business Plan (v1),” Deliverable D5.3 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 2008. [Online].

[30] P. Dihé, J. H. Amorim and G. Schimak, “D7.8 Data Management Plan V1,” Deliverable D7.8 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 27 November 2017. [Online].

[31] P. Dihé, M. Scholl, S. Schlobinsk, T. Hell, S. Frysinger, P. Kutschera, W. Manuel, D. Havlik, A. DeGroof, Y. Vandeloise, O. Deri, K. Rannat, J. Yliaho, A. Kosonen, M. Sommer and W. Engelbach, "D32.2 - ICMS Architecture Document V2," Deliverable D32.2 of the European Project FP7-284552 Modelling crisis management for improved action and preparedness (CRISMA), 2 February 2014. [Online]. Available: http://www.crismaproject.eu/deliverables/CRISMA_D322_public.pdf. [Accessed 30 January 2018].

[32] E. Rome, U. Beyer, S. Cohnitz, J. Stachowiak, A. Usov, C. Beyel and J. Börding, "Deliverable D4.1b Final Architectural Design," DIESIS - Design of an Interoperable European Federated Simulation network for critical InfraStructures, , Collaborative Project, Specifically Targeted Research Project (STReP), FP7 RI Grant Agreement N° 212830, 2010, 2010.

[33] Intergovernmental Panel on Climate Change, "Climate Change 2014: Impacts, Adaptation, and Vulnerability. Part B: Regional Aspects. Contribution of Working Group II to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change," Cambridge University Press, Cambridge, United Kingdom and New York, USA, 2014.

[34] M. Zuvela-Aloise, A. Kainz, C. Hahn, M. Leone, G. Zuccharo, D. Del Cogliano, M. Iorio and S. Schlobinski, "D3.1 Science Support Plan and Concept," Deliverable D3.1 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 2018. [Online].

[35] G. Zuccaro, M. Leone, D. De Gregorio, F. Gallinella, M. Zuvela-Aloise, A. Kainz, W. Loibl, T. Tötzer, L. Strömbäck, Y. Hundecha, J. H. Amorim, L. T. Michelena, A. R. Campos and I. Torres, "D2.1 Demonstration and Validation Methodology," Deliverable D2.1 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 2018. [Online].

[36] S. Brown, Software Architecture for Developers - Volume 1: Technical leadership and the balance with agility, Leanpub, 2018.

[37] F. Larosa and A. Perrels, "D1.2 Existing Resourcing And Quality Assurance of Current Climate Services," EU-MACS - European Market for Climate Services, 14 July 2017. [Online]. Available: http://eu-macs.eu/wp-content/uploads/2017/07/EUMACS_D12_EXISTING-RESOURCING-AND-QUALITY-ASSURANCE-OF-CURRENT-CLIMATE-SERVICES.pdf. [Accessed 27 October 2017].

[38] P. Stegmaier and K. Visscher, "D1.4 A multi-layer exploration on innovations for climate services markets," Deliverable D1.4 of the European Project H2020-730500 European Market for Climate Services (EU-MACS), [Online]. Available: http://eu-macs.eu/wp-content/uploads/2016/12/EU-MACS_D14_submitted_31102017-corrected-171113-JAK.pdf. [Accessed 26 January 2018].

[39] R. Hamaker, E. Jiménez-Alonso, A. Rycerz, A. Baglee and P. Stegmaier, "D1.3 Analysis of existing data infrastructures for climate services," 14 July 2017. [Online]. Available: http://www.acclimatise.uk.com/wp-content/uploads/2017/08/EU-MACS_Analysing_Data_Infrastructures_For_Climate_Services.pdf. [Accessed 29 November 2017].

[40] A. Geyer-Scholz and J. Alonso, "D6.2 Communication and dissemination plan and report (v2)," Deliverable D6.2 of the European Project H2020-730355 Integrated Climate Adaptation Service Tools for Improving Resilience Measure Efficiency (CLARITY), 2018. [Online].

[41] E. Eric, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2003.

[42] R. Seker, A. Van der Merwe, P. Kotzé, M. Tanik and R. Paul, “Assessment Of Coupling And Cohesion For Component-Based Software By Using Shannon Languages,” *Journal of Integrated Design & Process Science,* pp. 33-43, 2004.

[43] P. Brown, J. A. Estefan, K. Laskey, F. G. McCabe and D. Thornton, “Reference Architecture Foundation for Service Oriented Architecture Version 1.0,” December 4 2012. [Online]. Available: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html. [Accessed April 8 2018].

[44] OpenAPI Initiative, “OpenAPI Specification (OAS),” OpenAPI Initiative, 12 April 2018. [Online]. Available: https://github.com/OAI/OpenAPI-Specification. [Accessed 17 May 2018].

[45] S. Schmidt, “On the Nature of Complexity in Software Development,” *Medium,* 14 Oct 2014.