EU-CIRCLE

A pan-European framework for strengthening Critical Infrastructure resilience to climate change

D5.1 CIRP detail design document v1

Contractual Delivery Date: 01/04/2016	Actual Delivery Date: 01/04/2016
Type: Report	Version: 1.0
Dissemination Level: Public Deliverable	
Statement	
This report (version 1) presents the architecture and detailed specifications of the Climate Infrastructure Resilience Platform (CIRP). More specifically the CIRP system overview along with key design considerations, design strategies and decisions, architecture and detailed description of the platform modules are presented with the aim to set the stage for the subsequent developments that will take place in the frame of Tasks 5.2 – 5.6 of WP5.	

EU-CIRCLE is a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 653824. Please see http://www.eu-circle.eu/for more information.

Preparation Slip			
	Name	Partner	Date
From	Leonidas Perlepes	STWS	20/03/2016
Reviewer	Patrick Brausewetter	IVI	30/03/2016
Reviewer	David Prior	XUV	31/03/2016
For delivery	Antonis Kostaridis	STWS	31/03/2016

Document Log			
Issue	Date	Comment	Author / Organization
V0.1	10/01/2016	TOC	L. Perlepes / STWS
V0.2	18/01/2016	Initial Text introduced	L. Perlepes / STWS
V0.3	14/02/2016	Architecture	L. Perlepes / STWS, O. Politi / STWS
V0.4	10/03/2016	Design Considerations/Strategies	O. Politi / STWS, M. Troulinos / STWS
V0.5	20/03/2016	Detailed Module Design	O. Politi / STWS, M. Troulinos / STWS
V0.6	26/03/2016	Ready for Review	L. Perlepes / STWS
V0.7	30/03/2016	Internal Review	P. Brausewetter /IVI
V0.8	31/03/2016	Revisions based on IVIs comments	L. Perlepes / STWS
V0.9	31/03/2016	Final Review	D. Prior / XUV
V1.0	31/03/2016	Final version 1.0	A. Kostaridis / STWS

Abbreviations List	
Term	Description
API	Application Programming Interface
AWT	Abstract Window Toolkit
CEF	Chameleon Enterprise Foundation
CIRP	Critical Infrastructure Resilience Platform
CI	Critical Infrastructure
DoA	Description of Action
DC	Design Consideration
DS	Design Strategy
DPT	Design Policy and Tactic
GEF	Graphical Editing Framework
GIS	Geographical Information System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
JDBC	Java Database Connectivity
JEE	Java Enterprise Edition
JMS	Java Messaging Service
JNDI	Java Naming Directory Interface
JNLP	Java Network Location Protocol
JRE	Java Runtime Environment
OGC	Open Geospatial Consortium
ORM	Object Relational Mapping
OSGi	Open Services Gateway Initiative
PC	Personal Computer
RCP	Rich Client Platform
RIA	Rich Internet Application

RMI	Remote Method Invocation
SWT	Standard Widget Toolkit
UML	Unified Modelling Language
WebDAV	Web Distributed Authoring and Versioning
XML	Extensible Markup Language

Executive Summary

EU-CIRCLE's scope is to derive an innovative framework supporting resilience of the interconnected European Critical Infrastructure to climate pressures as the increasingly dependent, interdependent and interconnected nature of CI networks exposes previously unseen risks, new vulnerabilities, and opportunities for disruption of those networks.

This document constitutes the first version of the Detailed Design of the Climate Infrastructure Resilience Platform (CIRP): an innovative modular and expandable software platform that will assess potential impacts due to climate hazards; provide monitoring through new resilience indicators, and support cost-efficient adaptation measures. In this context, CI policy-makers, decision makers and scientists have access to diverse simulation, modelling and risk assessment solutions in a homogenised environment that allows both the development of risk reduction strategies and the implementation of mitigation actions to minimize the impact of climate change on CI. This modelling approach can help improve the understanding of system interdependencies and reduce the time from gap discovery to gap closure by providing decision makers with the latest tools, based on the best scientific and engineering principles, as they emerge.

The CIRP is defined as an end-to-end collaborative modelling environment where new analyses can be added anywhere along the analysis workflow and where multiple scientific disciplines can work together to understand interdependencies, validate results, and present findings in a unified manner providing an efficient solution that integrates existing modelling tools and data into a holistic resilience model in a standardised fashion.

The CIRP detailed design has been based on key design considerations and adopted design strategies and policies in order to meet the system expectations as described in the project Description of Action and the EU-Circle Strategic Context (D1.3). These considerations and design strategies have been incorporated into the CIRP architecture and the detailed design of its components. More specifically CIRP is based on an extensible modular architecture that will be shared across multiple communities and enable users to leverage existing software analysis types and algorithms, inventory types, and fragilities while not binding the underlying platform to a particular scientific domain. This pluggable, open architecture is what will allow CIRP to support a wide variety of domain specific functionality. Domain specific functionality will be isolated in plugins and will extend to the repackaging of different functional aspects as a starting point for new applications or, through extension, to add new analytical capabilities in the future.

The CIRP is intended to be a user-friendly environment that will provide its users with the ability to analyse what-if scenarios: leveraging model selection, climate data repositories, and CI inventories in order to calculate damages for any kind of climate hazard and CI.

In this way, users will be able to understand the impact of various adaptation strategies or quantify the potential impact of a catastrophic event on society.

Contents

1 Introduction

Critical Infrastructures are essential elements for the functioning of all socioeconomic activity of industrialised nations. The functional loss of these systems due to an external event can cause severe impact on a community in many different ways. It is evident that the increasingly dependent, interdependent, and interconnected nature of European Critical Infrastructures exposes previously unseen risks, new vulnerabilities, and opportunities for disruption across the CI networks.

EU-CIRCLE's scope is to derive an innovative framework for supporting the interconnected European Infrastructure's resilience to climate pressures. As described in the project strategic context (D1.3) CI's vulnerabilities and impacts go far beyond physical damages and thus EU-CIRLCE will be concerned with an assessment of the impacts to the services provided by CIs, addressing impacts associated with the repair, and/or replacement of services but also including the externalities of the infrastructures operation, societal costs, environmental effects, and economic costs due to suspended activities.

Such assessments will be carried out on a validated Climate Infrastructure Resilience Platform (CIRP). The CIRP is a standalone and comprehensive software toolbox that will assess potential impacts due to climate hazards, provide monitoring through new resilience indicators, and support cost-efficient adaptation measures. In this context the CIRP is established as an end-to-end modelling environment where new analyses can be added anywhere along the analysis workflow, and where multiple scientific disciplines can work together to understand interdependencies, validate results, and present findings in a unified manner. The CIRP provides an efficient solution that integrates existing modelling tools and data into a holistic resilience model in a standardised fashion.

This deliverable presents the initial detailed design version of the CIRP platform from the technological perspective as part of Task 5.1 with the aim to pave the way for the subsequent system development and component integrations of WP5. This deliverable will be refined (second iteration) according to the feedback and the new design requirements and as may be identified, quantified, and prioritised by the other work packages within EU-CIRCLE including, for example, the definition of the critical infrastructure risk model for climate hazards that will delivered by the WP3.The work package structure of EU-CIRCLE and especially the separation of Tasks in WP3, 4, 5 and 6 was based on the idea that risk model development and software development are two distinct activities and that the right approach for EU-CIRCLE is the one in which scientists and engineers develop the risk model (inputs, outputs, calibration, validation) and software developers work closely with this team to build efficient and user-friendly tools that are easily extended and adapted to suit a wide range of applications. In this respect, it is mandatory from the design perspective that the software platform is able to accommodate different types of datasets (e.g. hazard, assets, interconnections, fragilities), file formats, and risk analysis algorithms and that it provides adequate user interface elements for scenario and data repository management, analysis workflows setup, and intuitive results visualisation and reporting. As described in the DoA, the CIRP should be open, modular and extensible in order to support various risk and resilience assessment analysis tools not only from the project partners themselves, but also from the scientific community as part of the outreach objective of the project.

The CIRP design is based on a list of requirements described initially by the Description of Action document (e.g. open source, web-based, multi-hazard, extensible, plug-n-play, scenario based analysis, support for different types of analysis algorithms) and enhanced by the partners' knowledge and experience in developing modular software systems; a literature review regarding assessments of existing open source software tools, and the functional and non-functional requirements according to the consortium discussions conducted during the various project plenary and technical meetings (e.g. support of data repositories, ability to execute internal and external software codes, GIS capabilities). In addition, and with the aim of improving the transfer of knowledge and innovation from the scientific community to industry and CI stakeholders, the CIRP design encompasses a friendly graphical user interface where scenarios can be built and executed not only by scientists and climate and natural hazards risk assessment experts but also by CI operators, infrastructure owners, investors and regulators.

This Deliverable is the first iteration of the CIRP detailed design. The second and final iteration will update this version at the end of Task 5.1 by incorporating adaptions and changes in the architecture and system components according to work to be conducted in WP3 and WP4 tasks and the CIRP User Interface and modules development and integration (Tasks 5.2-5.7).

The rest of the document is structured as follows: the methodological approach followed is described in the following Section 2. Section 3 presents the CIRP System Overview and Section 4 the key design considerations. In Section 5, the design strategies selected to meet the design considerations are described while in Sections 6 and 7, respectively, we present the CIRP architecture and design policies and. Finally the detailed module design is described in Section 8.

2 Methodology

The following work is based on the CIRP initial functional specifications and design requirements that are described by the Description of Action. The work has been conducted in the frame of Task 5.1 and factors for the outcomes arising from the various project meetings to date and the EU-Circle Taxonomy and Strategic Context deliverables (D1.1 and D1.3). More specifically, the following meetings shaped the key architectural decision and strategies of the CIRP detailed design:

- The project's kick-off meeting held at National Center For Scientific Research - Demokritos (NCSRD) premises, Athens Greece, 9 & 10 June2015;
- The 2nd project meeting held at the European University Cyprus (EUC) premises, Nicosia Cyprus, 26 & 27 November 2015;
- The joint EU-CIRCLE NIST-CORE workshop held in NCSRD premises, Athens Greece, 5-7 October 2015, and
- Bilateral discussions of EU-CIRCLE partners in regular Skype and telephone calls

Based on the input from the above meetings and deliverables D1.1 and D1.3, a set of Design Considerations has been compiled (Section 4). To meet these requirements a number of Architectural Strategies (Section 5) that affect the overall organization of the CIRP and its higher-level structures, as well as specific Design Policies and Tactics (Section 7), were adopted. In parallel, the state of the art in software tools and related architectures for Natural Hazard Risk Assessment have been studied [1] and suitable open source software frameworks have been selected as the basic building blocks of the CIRP. The selected strategies, policies, and frameworks provide insight into and stimulus for the selection of key abstractions and mechanisms used in the definition of the system architecture (Section 6).

Finally, the detailed design of CIRP components was developed and evaluated. This process defined the specific purpose and semantic meaning of each component: assigning primary responsibilities and/or behaviours to each component supported by relevant assumptions, limitations, or constraints.

3 System Overview

The Climate Infrastructure Resilience Platform (CIRP) will comprise a ubiquitous collaborative environment that shall create new capabilities for CI policy-makers, decision makers, and scientists by allowing them to use different and diverse modelling and risk assessment solutions, in a standardised and homogenised environment, to develop risk reduction strategies and implement mitigation actions that help minimise the impact of climate change on CIs. This approach to modelling can help improve the understanding of system interdependencies and reduce the time from gap discovery to gap closure by providing decision makers with the latest tools, based on the best scientific and engineering principles, as they emerge.

Overall, the intention is for Risk management professionals to be familiar with identifying vulnerabilities, assessing loss reduction strategies, guiding resource allocation before disasters, identifying vulnerable areas during disasters, guiding recovery efforts, and providing information to decision-makers throughout the process. The essential elements for structural damage assessment are hazard, inventory, and fragility. Hazard is considered as the descriptive parameter quantifying the possible phenomenon within a region of interest. The assets in a region exposed to hazards are defined by inventory. Finally, fragility is the sensitivity of certain types of inventory items when subjected to a given hazard. Assuring that the science and engineering principles behind the forecasting of damage probability of Critical Infrastructures (buildings, bridges, networks, pipelines, and other inventory items) from anticipated events is both pragmatic and state-of-the-art is therefore critical to minimising the impact of climate change events, reducing losses to economic resources, and the development of more stable communities.

Many risk assessment tools and platforms exist today. Most, however, lack the flexibility to easily add new algorithms or to extend their base features. This is typically due to a combination of architectural approach and closed-source licensing policies. Such software does not allow the community to actively contribute new algorithms and capabilities and, therefore, allow the software to evolve with the advancements of science. Furthermore, software-licensing fees from proprietary vendors can make such packages unaffordable for many members of the community.

A primary objective for the CIRP is that it be engineered as a pluggable and extensible platform that will enable the Risk Management community to bring new data and modelling capabilities into practice. From the CIRP policy and decision maker perspective, the platform capabilities will be offered as a toolbox that consists of a collection of diverse analyses of Risk and Resilience of Critical Infrastructures that are exposed to the direct and indirect effects of climate change.

CIRP users will be able to create and store scenarios by means of selection of a chain of analysis tools. Each analysis tool is associated with input and output parameters and relevant datasets that conform to the platform supported types. It will be possible to chain analysis tools to form analysis workflows and each individual analysis in the workflow will be monitored as provided for by the analysis type, the geographical span of the scenario, and the number of CI elements analysed. An analysis will be able to be executed in seconds, minutes or even hours. The design of the CIRP provides a uniform user experience for the user input of values and selection of input datasets. Each analysis tool within the CIRP is described in the Extensible Markup Language (XML) and transformed at runtime into suitable widgets and user interface controls.

The following Use Case UML diagram presents the main use cases of the CIRP. They are organised in the following five (5) packages:

- Administration
- Scenario Setup
- Scenario Processing
- Post-Processing

- Collaboration

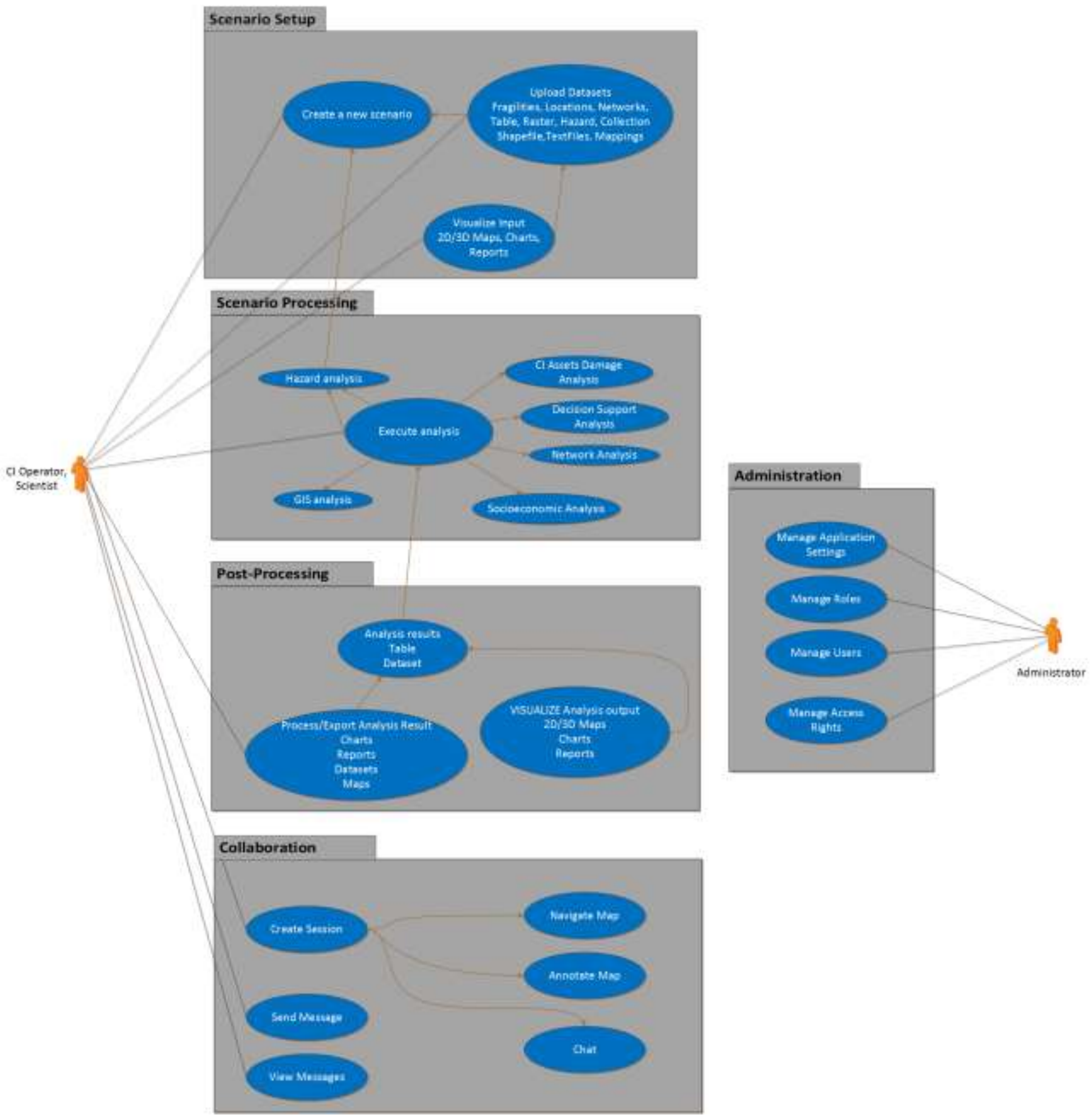


Figure 1: CIRP Use Cases Diagram

The administration package encompasses user requirements from the point of view of CIRP enterprise application administration. The scenario setup, processing, and post-processing packages encompass all of the different requirements and interactions of the CI Operators and Scientists in the context of preparing, executing, and post-processing risk analysis workflows. Finally, the collaboration package consists of the sequence of actions for creating collaborative sessions in which users will be able to navigate into simulation areas, observe the results, annotate the map, and exchange messages in real time.

4 Design Considerations

The Climate Infrastructure Resilience Platform (CIRP) framework is designed as an end-to-end modelling environment where new analyses can be added anywhere along the analysis workflow enabling scientists to create new end-to-end analyses or to enhance existing analyses. Through this framework, multiple scientific disciplines can work together to understand interdependencies, validate results, and present findings in a unified manner. As a result, the CIRP framework provides an efficient "Best of Breeds" solution integrating existing modelling tools and data into a holistic resilience model in a standardised fashion.

The CIRP framework is the project's core component. As a result, the provided functionalities must be in accordance to the project's objectives; such as the analysis of the resilience of multiple interconnected CI for multiple climate hazards assessing multi-tier impacts and the evaluation of alternative solutions for mitigation – adaptation activities.

The criteria that were taken into account in order to design the CIRP framework are presented in the following table:

Table 1. CIRP Design Considerations

Code	Design Consideration	Description
DC.1	Platform Independence	Many factors influence a software package ease of use and among them are the Operating System and coding language. Even if Linux is currently the most common operating system for supercomputers, most basic users have experience in Windows operating systems.
DC.2	Flexible and Intuitive GUI	The GUI is an important factor in the design of the CIRP because it determines its usability. Few users have the technical skills that allow them to execute models using command lines alone. For non-experts, grappling with risk assessment concepts is usually quite difficult; attempting to come to grips with what is being modelled using a new software package makes things even harder. Thus, simple software that allows a user to point, click, and then understand is best for a non-expert. As a result, the CIRP framework provides an easy-to-use GUI as well as hazard, exposure, and vulnerability analyses, so that users have more control over their analysis activities.
DC.3	User Orientation and Assistance	The software application should also be user-oriented, with separate documentation available for those wishing to modify or extend the tools and leverage any available APIs (application programming interfaces), and with tutorials, sample data, and expected results available for training and testing model installation. User assistance includes those tools that work collectively to introduce the user to the software, to guide the user through tasks, and to help the user find more information about the software or specific component.
DC.4	Performance	The CIRP framework design is focused on the maximisation of accuracy with the minimal

		computational effort. All the components can be run on a standard PC (2.5GHz processor with 4GB of RAM and a 500GB hard drive). However, computationally expensive algorithms and GIS-based components may require more computing power. Ideally, users determine whether the software algorithms are reasonable for their computational purposes. The actual physical computation is generally not computationally demanding but, where memory is insufficient, the large volumes of data (providing exposure or hazard event sets) can cause problems. For deterministic use in post-disaster studies, all of the components can be run in a reasonable time (assuming the region is not extremely large, such as the region of the city with its surroundings, and the data are available). Rapid response data can be problematic, however, if data sets are not publicly available for reuse. In contrast, computing power plays a much more central role in stochastic or probabilistic modelling—i.e. in the simulation of 10,000+ years of hazard events analyzed against exposure data sets of varying sizes.
DC.5	Modularity and Extensibility	The CIRP design needs to be fully modular, allowing users to perform specific analyses using only specific components of the software, but also fully extensible by project and third-party partners in order to introduce new analyses and new data types.
DC.6	Multi User Collaborative Application	The CIRP will be accessible over the web to multiple, simultaneous, registered users providing the capability for collaboration sessions in which users can chat with other users, mark up areas on the terrain, and toggle scenario layers. In this way, users interact with the platform risk assessment tools and at the same time collaborate and discuss the results.
DC.7	Web based	The platform ideally should be accessible via the Web (either as a pure Web based or browser downloadable rich client application)
	Design Considerations specific to Risk Assessment	
DC.8	Integrated GIS engine	Many Risk modelling software are based on commercial license based Geographical Information System packages that can be prohibitively expensive for many users. A key CIRP design consideration is to be able to integrate with an open source GIS engine that supports standardised formats for input and output, spatial analysis utilities, and a map view component.
DC.9	Exposure	A critical factor for any risk assessment is exposure data. Thus, the CIRP framework will provide tools for managing exposure data. Exposure is defined as the amount of human activity located in the zones of hazard as defined by the stock of infrastructure in that location. Depending

		upon the vulnerability functions, exposure information can be restricted to structural features, or it can be extended to nonstructural features such as CI building contents and to infrastructure services such as lifelines and emergency response facilities. Exposure is a function of the population, remote sensing, building use and other building inventory data used. CIRP will classify the various exposure elements using construction and occupancy information associated with location information. This information should be compatible with any vulnerability function. CIRP will allow advanced users to provide additional classification types, new risk indicators, and supplemental socioeconomic parameters once relevant checks have been made to the applicability of the vulnerability, hazard, and loss modules.
DC.10	Vulnerability	One of the fundamental factors influencing a risk assessment is vulnerability of the exposed assets. The availability of data for input, calibration, and validation governs the quality of the vulnerability module. The vulnerability functions should be computationally simple to allow for rapid response as well as consistent with observations of historical damage. CIRP will be able to accept additional exposure types and able to simulate not only physical vulnerabilities but also socioeconomic vulnerabilities.
DC.11	Hazards	The CIRP should accommodate both different built-in Hazard analyses (the ones that are influenced directly or indirectly by the climate change) and Hazard results calculated in specialised external software packages.
DC.12	Risk	Risk can be quantified in a variety of ways. However, in order to maximize effectiveness of the platform, CIRP is designed according to a specific risk calculation. Losses here may be calculated via a damage-loss conversion. The economic losses generally account for direct loss while estimates of indirect loss are less common. It is generally common to use the mean damage ratio (repair to replacement cost) and the variability from a vulnerability function to derive an economic loss. In addition, using a model for land-use planning and/or cost-benefit analysis may be relevant but this capability is highly dependent on the resolution of the model
DC.13	Visualization of Results	Model results are the most important output of any risk analysis. CIRP should be able to visualise the results (in terms of hazard, exposure, and vulnerability) in multiple 2D/3D GIS windows as well in multiple tabular views with chart based statistic options as well as comprehensive reports. In this way users will be able to analyse the results in their preferred format and to compare them for different input data.

DC.14	What-if scenario support	The ability to introduce and compare "What-if scenarios", translating into the selection of models, data, interconnected CI description, multi-level damage assessment, and adaptation / mitigation strategies, as may be required to investigate the policy objective
DC.15	Post event scenario support	The ability to support post-event scenarios where speed, simplicity of use, and quick access to information are critical. The CIRP framework is designed according to the need to generate products that would complement post-event response, recovery, and reconstruction efforts. GIS capabilities can also be important for post-event analysis.
DC.16	Comprehensible results	Scenario results will be provided to users of the system and the EU-CIRCLE stakeholders through the use of common 'language' that is fully understood by all parties.
DC.17	Interconnected Analysis Workflows	The ability to graphically interconnect/chain various analysis tools based on the output and input datasets for independent or interdependent analysis types (e.g. Interdependent Network Analysis).
DC.18	Development Simplicity	Provide the means for scientists/developers to define specific analysis inputs and outputs (e.g. file result name) in XML descriptor documents and, thereafter, automatically generate the required User Interface forms to support those input and output requirements. In this way, scientists will not be required to produce user interface specific code and platform end users are presented with a unified experience across the platform.
DC.19	Efficient Data and Metadata Management	A semantic content library is required to both manage the data and metadata imported into the CIRP and to provide the ability to search, tag, and annotate the imported data.

5 Architectural Strategies

In the section we describe the main decisions and/or strategies that affect the overall organization of the CIRP and its higher-level structures. These strategies provide direction for the key abstractions and mechanisms used in the system architecture (Section 6).

The following Table summarizes the decisions and strategies. The reasoning leading to each outcome is described by reference to the stated goals and principles of the previous Section 4. Additionally, any design goals or priorities that were balanced or traded-off in the selection process are also detailed.

Table 2. Architectural Decisions and Strategies

Code	Decision / Strategy	Description	Design Considerations Ref. Codes
ADS.1	Java Coding Language	The Java Language is one of the most popular programming languages in use; particularly for client-server web applications. This choice was informed by both platform independence and the modularity and extensibility offered by the JVM based Eclipse Rich Client Platform and Open Services Gateway Initiative technologies (see ADS.5 and ADS.6). In this decision we traded-off the potentially enhanced performance of other languages against the flexibility, modularity and extensibility offered by Java/OSGi.	DC.1 – "Platform Independence", DC.4 – "Performance"
ADS.2	Java Enterprise Edition	The JEE is a widely used enterprise computing platform developed under the Java Community Process. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications.	DC.1– "Platform Independence", DC.6 – "Multi User Collaborative Application"
ADS.3	Object Relational Mapping	ORM enables developers to more easily write applications whose data outlives the application process. An Object/Relational Mapping (ORM) framework is concerned with data persistence as it applies to relational databases (via JDBC).	DC.18 – "Development Simplicity"
ADS.4	Web Start based Rich Client application	The Java Web Start is a framework developed by Sun Microsystems (now Oracle) that allows users to start application software for the Java Platform directly from the Internet using a web browser. Some key benefits of this technology include seamless version updating for globally distributed applications and greater control of memory allocation to the Java virtual machine.	DC.6– "Multi User Collaborative Application", DC.7 – "Web based"
ADS.5	Service Platform	The OSGi Service Platform is the de-facto standard for modularised Java Language. It is a framework that provides a dynamic environment for the deployment of services and modules (referred as bundles in OSGi	DC.5 – "Modularity and Extensibility"

		terminology).	
ADS.6	Eclipse RCP	RCP which provides the architecture and framework to build rich client application. RCP's close integration with OSGi makes it one of the only UI technologies to leverage modularity from the ground up.	DC.2 – "Flexible and Intuitive GUI", DC.5– "Modularity and Extensibility"
ADS.7	Development frameworks – Reuse of existing software components	The CIRP will be based on two distinct frameworks both based on the OSGi. The first one is the CEF (Chameleon Enterprise Foundation), an Enterprise Application Framework developed by Satways Ltd. The second one is the ERGO-CORE platform, an Open Source Risk Assessment desktop software.	DC.5 - DC.19 "Modularity and Extensibility", "Multi User Collaborative Application", "Web based", "Integrated GIS engine", "Exposure", "Vulnerability", "Hazards", "Risk", "Visualization of Results", "What-if scenario support", "Post event scenario support", "Comprehensible results", "Interconnected Analysis Workflows", "Development Simplicity", "Efficient Data and Metadata Management"
ADS.8	Data repositories for Data and Metadata Management	CIRP will provide efficient management and synchronization of different data repositories, either of public domain, of cached locally (local repository). Execution of analysis based on cached/local data provide the necessary speedup and tolerance of low speed networks while the synchronization with public repositories provide the means for scenario input and output results dissemination to other platform members. A semantic content library will track the provenance of data so that users can determine information such as which algorithms were used, the date it was created, the author, which machine was used, etc.	DC.19-" Efficient Data and Metadata Management"
ADS.9	Context Sensitive Help	In the context of CIRP, user assistance will be provided by Developer and User manuals as well as context sensitive help support, where a user can summon help	DC.3 – "User Orientation and Assistance"

		for a particular element in the UI (e.g. by pressing F1 key).	

In the following subsections each architectural strategy is presented in more detail.

5.1 Open Services Gateway Initiative

The OSGi technology is a set of specifications that define a dynamic component system for Java. These specifications enable a development model where applications are (dynamically) composed of many different (reusable) components. The OSGi specifications enable components to hide their implementations from other components while communicating through services, which are objects that are specifically shared between components.

A practical advantage of OSGi is that every software component can define its API via a set of exported Java packages and that every component can specify its required dependencies. The components and services can be dynamically installed, activated, de-activated, updated, and de-installed. The OSGi specification defines a bundle as the smallest unit of modularization, i.e., in OSGi, a software component is a bundle.

The OSGi has a layered model that is depicted in the following figure.

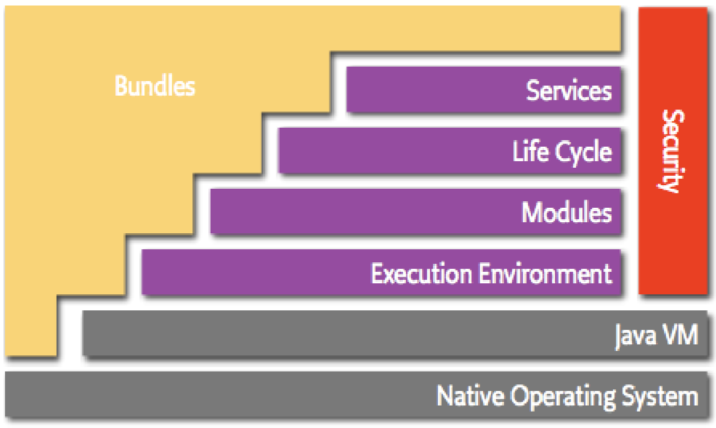


Figure 2: OSGi layered model

The following list contains a short definition of the terms:

- Bundles – Bundles are the OSGi components made by the developers.
- Services – The services layer connects bundles in a dynamic way by offering a publish-find-bind model for plain old Java objects.
- Life-Cycle – The API to install, start, stop, update, and uninstall bundles.
- Modules – The layer that defines how a bundle can import and export code.
- Security – The layer that handles the security aspects.
- Execution Environment – Defines what methods and classes are available in a specific platform.

5.1.1 Modules

The fundamental concept that enables such a system is modularity. Modularity, simply put, is about assuming less. Modularity is about keeping things local and not sharing. Therefore, modularity is at the core of the OSGi specifications and embodied in the bundle concept. Though the code hiding and explicit sharing provides many benefits (for example, allowing multiple versions of the same library being used in a single Virtual Machine), the code sharing exists only to support the OSGi services model. The services model is concerned with bundles that collaborate.

5.1.2 Deployment

Bundles are deployed on an OSGi framework: the bundle runtime environment. It is a collaborative environment within which bundles run in the same VM and can actually share code. The framework uses the explicit imports and exports exposed by each bundle to wire up the bundles so they do not have to concern themselves with class loading. Moreover, the management of the framework is standardised. A simple API allows bundles to install, start, stop, and update other bundles, as well as enumerating the bundles and their service usage. Many management agents have used this API methodology to control OSGi frameworks.

5.2 Java Enterprise Edition

JEE (Java Enterprise Edition) is a Java platform designed for the mainframe-scale computing typical of large enterprises. Sun Microsystems (together with industry partners such as IBM) designed JEE to simplify application development in a thin-client, tiered environment. Java Enterprise Edition, Java EE, or JEE is a widely used enterprise-computing platform developed under the Java Community Process. The platform provides an API and runtime environment for the development and execution of enterprise software; including network and web services as well as other large-scale, multi-tiered, scalable, reliable, and secure network applications. JEE simplifies application development and decreases the need for programming and programmer training by creating standardised, reusable, modular components and by enabling the enterprise tier to handle many aspects of programming automatically. Java EE includes several API specifications, such as RMI, e-mail, JMS, web services, XML, etc., and defines how these should be co-ordinated. Java EE also features some unique component specifications. These include Enterprise JavaBeans, connectors, servlets, Java Server Pages (JSP), and several web service technologies. This allows developers to create enterprise applications that are portable and scalable and that integrate with legacy technologies. A Java EE application server can handle transactions, security, scalability, concurrency and management of the components deployed within it. This enables developers to concentrate more on the business logic of the components rather than on infrastructure and integration tasks.

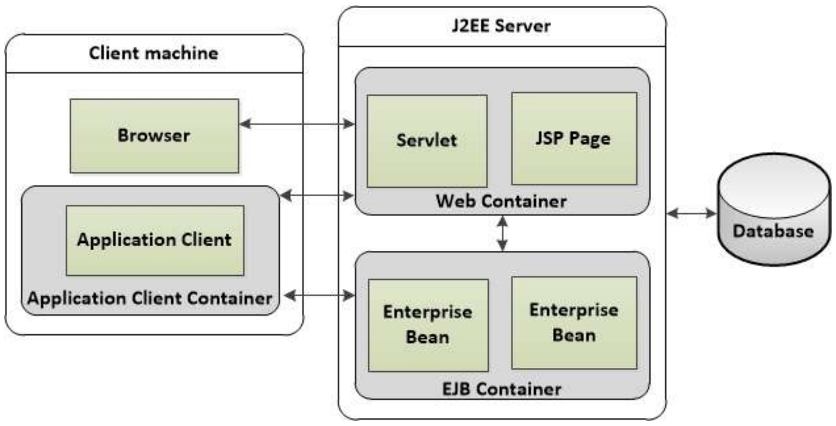


Figure 3: JEE Architecture

Normally, thin-client multi-tiered applications are hard to write because they involve many lines of intricate code to handle transaction and state management, multithreading, resource pooling, and other complex low-level details. The component-based and platform-independent JEE architecture makes JEE applications easy to write because business logic is organised into reusable components. In addition, the JEE server provides underlying services in the form of a container for every component type. Because developers are not developing these services themselves, they are free to concentrate on solving the business problem at hand.

5.2.1 Containers

Containers are the interface between a component and the low-level platform-specific functionality that supports the component. Before a Web, enterprise bean, or application client component can be executed, it must be assembled into a JEE application and deployed into its container.

The assembly process involves specifying container settings for each component in the JEE application and for the JEE application itself. Container settings customize the underlying support provided by the JEE server, which includes services such as security, transaction management, Java Naming and Directory Interface (JNDI) lookups, and remote connectivity. Here are some of the highlights:

- The JEE security model lets you configure a Web component or enterprise bean so that system resources are accessed only by authorised users.
- The JEE transaction model lets you specify relationships among methods that make up a single transaction so that all methods in one transaction are treated as a single unit.
- JNDI lookup services provide a unified interface to multiple naming and directory services in the enterprise so that application components can access naming and directory services.
- The JEE remote connectivity model manages low-level communications between clients and enterprise beans. After an enterprise bean is created, a client invokes methods on it as if it were in the same virtual machine.

The fact that the JEE architecture provides configurable services means that application components within the same JEE application can behave differently based on where they are deployed. For example, an enterprise bean can have security settings that allow it a certain level of access to database data in one production environment and another level of database access in another production environment.

The container also manages non-configurable services such as enterprise bean and servlet life cycles, database connection resource pooling, data persistence, and access to the JEE platform APIs.

5.3 Object-relational Mapping

Object-relational mapping (ORM, O/RM, and O/R mapping) in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages. Object-relational mapping (ORM) is a mechanism that makes it possible to address, access and manipulate objects without having to consider how those objects relate to their data sources. ORM lets programmers maintain a consistent view of objects over time, even as the sources that deliver them, the sinks that receive them, and the applications that access them change. Based on abstraction, ORM manages the mapping details between a set of objects and underlying relational databases, XML repositories, or other data sources and sinks while simultaneously hiding the more volatile details of related interfaces from developers and the code they create. ORM encapsulates and abstracts change in the data source itself, so that when data sources or their APIs change, only ORM needs to change to keep up—not the applications that use ORM to insulate themselves from this kind of effort. This capacity lets developers take advantage of new classes as they become available and also makes it easy to extend ORM-based applications. In many cases, ORM changes can incorporate new technology and capability without requiring changes to the code for related applications.

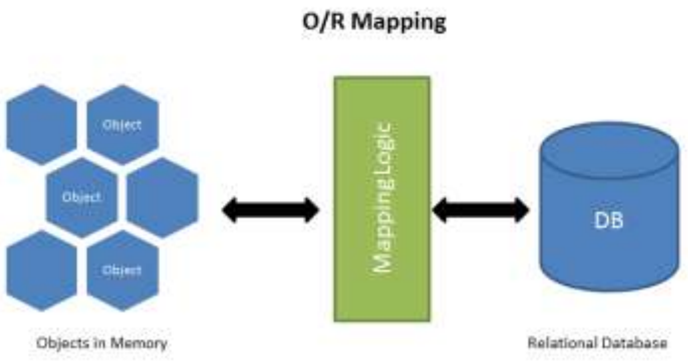


Figure 4: Object-relational Mapping

Many ORM frameworks exist and Hibernate is the framework of choice. In addition to its own "native" API, Hibernate is also an implementation of the Java Persistence API (JPA) specification. As such, it can be easily used in any environment supporting JPA including Java SE applications, Java EE application servers, Enterprise OSGi containers, etc.

5.4 Eclipse Rich Client Platform

The Eclipse RCP is not a single framework, but a collection of lower-level frameworks. Most technologies rely on other lower-level technologies, and RCP is no different. Specifically, RCP leverages:

OSGi – At the lowest level, all RCP applications run on top of an OSGi framework.

SWT – The Standard Widget Toolkit provides the primitive widgets (text controls, radio buttons, etc.) used to create platform independent RCP user interfaces. It functions much like AWT in a traditional Swing application.

JFace – Building on top of SWT, JFace provides a variety of advanced UI functionality including wizards, preference pages, data binding and much more.

Other Eclipse APIs – RCP relies on a host of other Eclipse frameworks that are not strictly part of RCP itself. These include the Jobs API for managing concurrency and the Commands API that provides menu support.

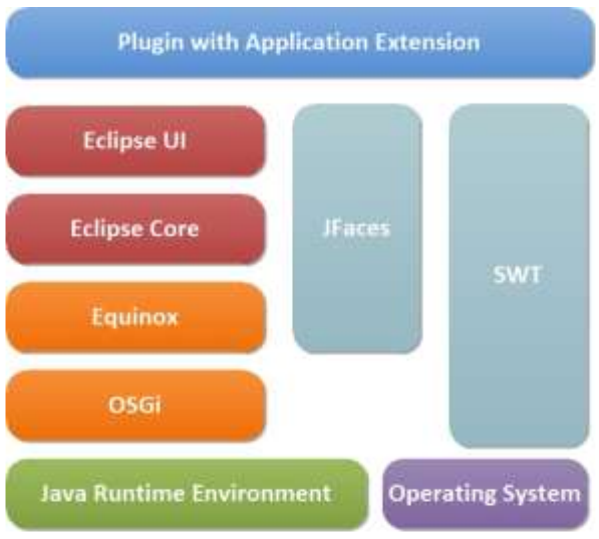


Figure 5: The Eclipse RCP layers

What RCP does is to integrate these frameworks to provide a workbench into which developers can contribute content. This workbench has a specific, though highly-customisable, structure which defines the places where content can be added. A workbench defines where we can contribute menus, wizards, preferences, help content, and much more. A workbench also contains perspectives, which themselves can contain editors and views.

As the Chameleon Enterprise Framework and ERGO-CORE bundles both are based on the Eclipse RCP plugin system their integration and further extension with Risk Assessment and User Collaboration plugins is straightforward.

5.5 Java Web Start

The JNLP is a protocol that is used interchangeably with the term "Web Start". The JNLP protocol, defined with an XML schema, specifies how to launch Java Web Start applications. JNLP consists of a set of rules defining how exactly to implement the launching mechanism. JNLP files include information such as the location of the jar package file and the name of the main class for the application, in addition to any other parameters for the program. A properly configured browser passes JNLP files to a Java Runtime Environment (JRE), which in turn downloads the application onto the user's machine and starts executing it.

5.6 The CEF Platform

The Chameleon Enterprise Foundation (CEF) is a software platform, designed and developed by Satways Ltd, for building geospatial multi-user Enterprise Applications. It is based on the RCP, OSGi and JEE technologies and provides a rich collection of OSGi bundles for organization and user management, logging, UI widgets, ORM persistence, etc. These bundles interact with each other via the RCP extensions and extension points in the form of an Application Programming Interface (API). The API is defined as a set of classes and methods that can be used from other bundles. The CEF provides the basic set of services for a RIA enterprise application

The basic set of functionalities offered by the CEF is the following:

- Intuitive Graphical User Interface
- Event driven architecture (EDA)
- Multiple monitor/screen support
- Organization, User and Role Management
- User authentication
- User Collaboration
- JEE Integration (JNDI, EJB, JMS)
- Support of Multiple Mapping/GIS engines
- Extensibility Framework (contribution interfaces and contributions)
- Database Connection Pooling and Object relational mapping (ORM)
- Common User interface framework
- Common Logging framework
- Various developer Utilities
- Automatic Fail-over and load balancing
- Extensibility (new plugins can be created by 3rd parties)
- Java Web start support

Each of the above bundles interacts with a number of Enterprise Java Beans (EJBs) components on the server side and a Messaging System for real time notifications. On the back end, a PostgreSQL database with the PostGIS extension or an Oracle Enterprise Edition database is deployed to provide the geospatial capabilities at the database layer.

The CEF platform provides a bundle extensibility framework according to the OSGi specification. This means that a contributing bundle can provide contribution to existing CEF interfaces and can provide contribution interfaces for other bundles. Any resulting application is therefore highly modular and, accordingly, extensible. In addition, use of a 'lazy' loading bundle mechanism provides greater loading speed and memory efficiency as bundles are loaded only when needed.

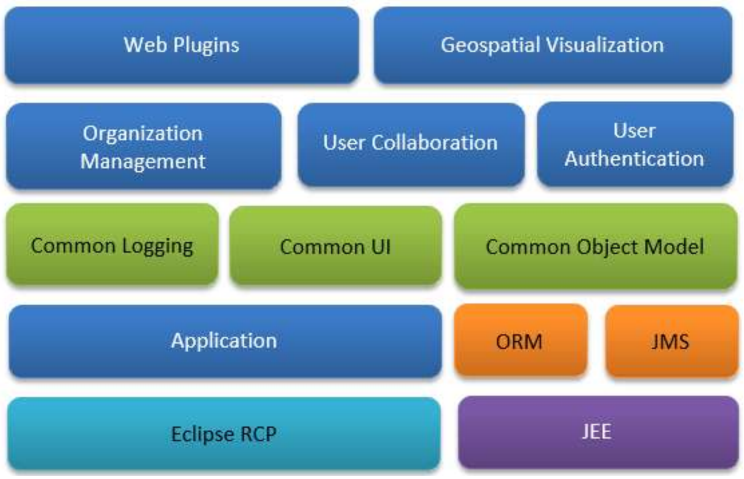


Figure 6: The CEF platform components

5.7 The ERGO-CORE Platform

The ERGO-CORE is the base IT infrastructure of the ERGO [2] an open-source project that was originally developed under the name Maeviz to perform seismic risk assessment. ERGO's objective is to reduce the time-from-discovery gap that exists between researchers, practitioners, and decision makers by integrating the latest research findings, most accurate data, and new methodologies into a single software product.

The ERGO-CORE is based on RCP technology and consists of a set of OSGi bundles that provide baseline data/metadata management, visualization, modelling, analysis, and user interface functions.

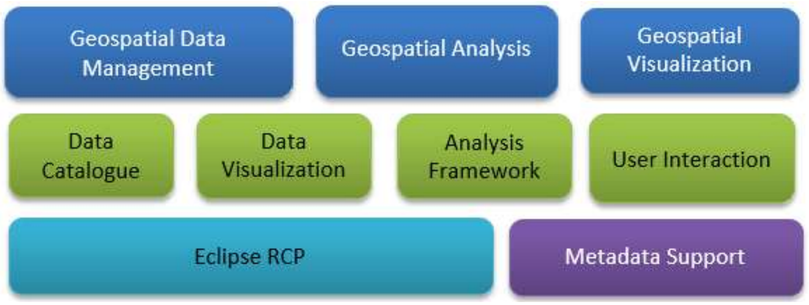


Figure 7: The ERGO-CORE components

ERGO-CORE provides map-based visualizations, tables, charts, graphs, and printable reports for result data. It is designed to be quickly and easily extensible. When new scientific knowledge, source data, or methodologies are discovered, these can be added to the system by developers or end users through a standard plug-in system.

The ERGO-CORE framework implements Consequence-Based Risk Management (CRM) using a visual, menu-driven system to generate damage estimates from scientific and engineering principles and data. ERGO-CORE leverages other open source software, particularly GeoTools, VTK, JasperReports, JFreeChart, Ktable, and iText.

6 System Architecture

The CIRP platform is based on a set of tools and components capable of providing resilience management functionality arising from a dynamic climate risk approach to critical infrastructure. This section provides a high level overview of how the CIRP functionalities and responsibilities of the system are partitioned and then assigned to subsystems and components.

In architectural terms, the CIRP is designed as a pluggable, multi-user, and collaborative N-tier software system that will be accessible to end users either as a Client-Server installation or as a Web start-able rich client application. The first type of installation addresses the EU-CIRCLE scientific partners that will develop, in close collaboration with the software engineering partners, new dataset types and analysis plugins and thus need to have direct access to the client part (set of plugins) of the CIRP. The second type of installation addresses the policy and decision makers and CI owners that need to access the system from a browser, operate in diverse locations, and receive automatic software updates as these become available from the consortium.

The high level logical architecture in terms of modules (collection of OSGi bundles) is depicted in the following Figure. This is a layered software approach comprised of multiple and discrete “shells” around the core of the system which is an OSGi specification implementation module (Equinox) and a Widget framework (SWT/JFace). Each shell expands and provides additional capabilities to the inner shells. As depicted in the outer blue shell the CIRP platform functionality will be based on the collaboration and expansion of two frameworks: the CEF (client & server) and the ERGO-CORE (client only). Both frameworks are as described in previous sections.

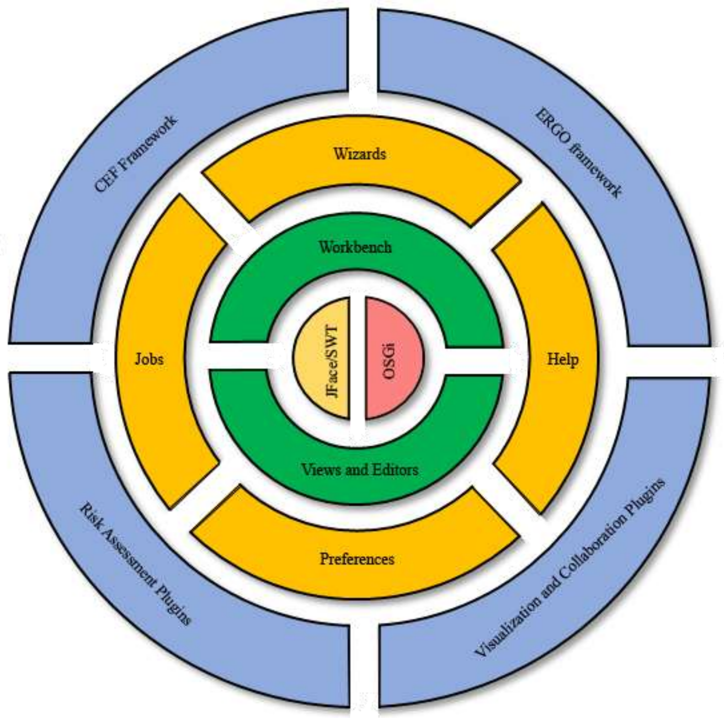


Figure 8: The CIRP modular software layers

Each of the two core frameworks provides a set of discrete functionalities that may be exploited independently or in a collaborative manner. The ERGO-CORE framework provides the functionality related to inventory, data and metadata management, and the ability to wrap new analysis types and execute them on the workflow engine. The CEF framework provides funcionality including the CEF Core module, the User Management & Roles and Access Rights modules, and the 3D GIS modules.

The CEF Core Module is the basic module that will interconnect the ERGO-CORE framework and its functionalities with the rest of the CEF modules.

The CEF framework, the Climate Change Risk Assessment and the Collaboration modules will also provide a set of modules able to:

- Support new types of infrastructures and links to societal functions;
- Support risk and resilience assessment models for multiple hazards;
- Support analysis and modeling of inter-dependent physical systems and non-technical systems that are essential for the recovery of a regional area (e.g. financial, social, healthcare, public safety, education etc.);
- Link to external software for climate hazards (e.g. flood simulators) and infrastructure operation models, and
- Support the collaborative and interactive exchange of risk analysis information and related scenarios

The envisaged logical architecture is depicted in the following Figure.

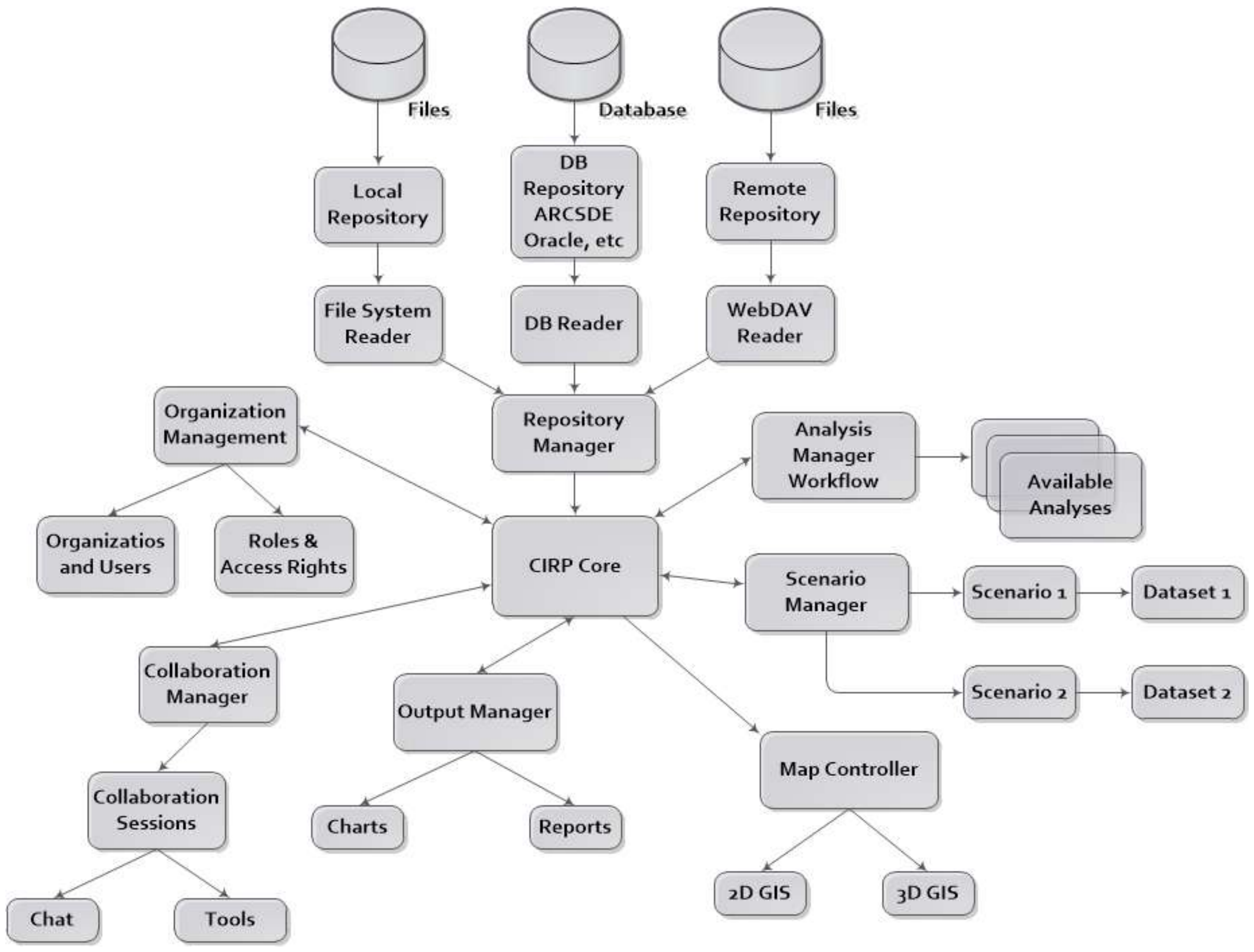


Figure 9: CIRP Logical Module decomposition

The following Deployment Diagram shows the physical layout of the various hardware components (nodes) that compose a system as well as the distribution of executable environments and software components on that hardware. The diagram depicts the actual devices (workstations, servers), along with the connections they have to each other, and provides an effective system topology. In that topology, as illustrated below, the location of executable components and objects illustrates where the software units are deployed and on which nodes they are executed.

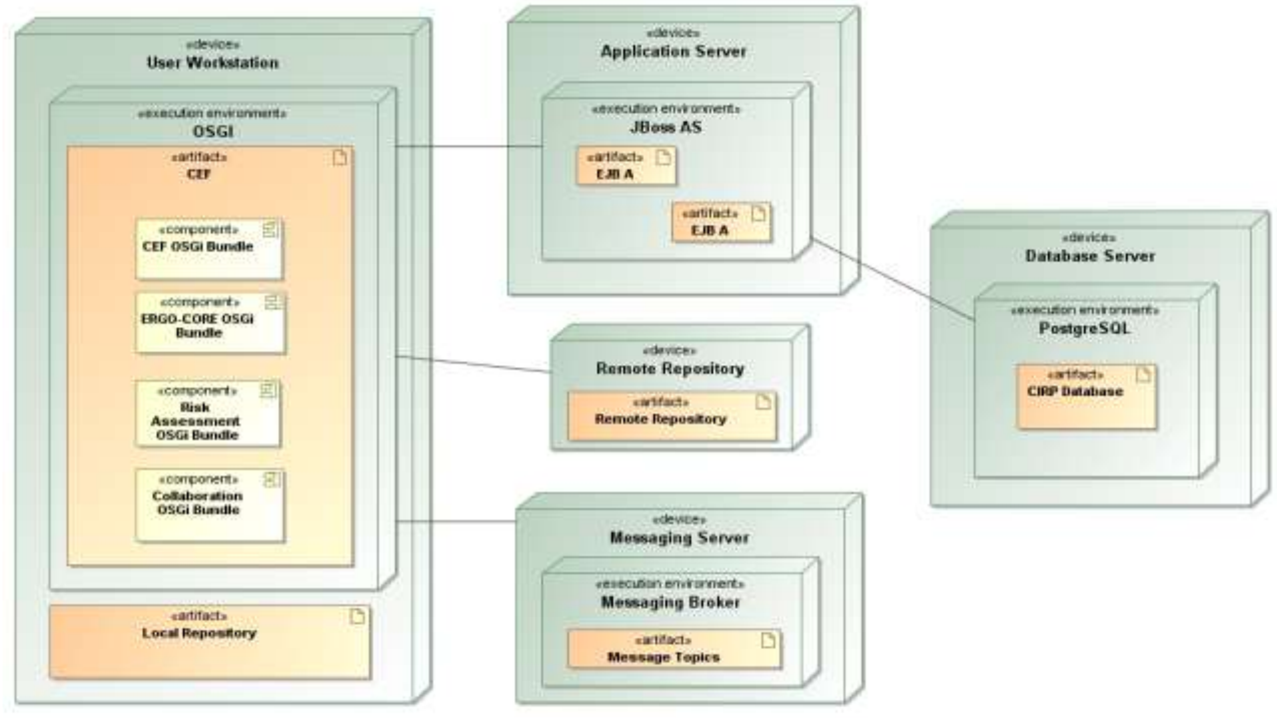


Figure 10: UML deployment diagram of the CIRP software system

The deployment diagram illustrates:

- The Application Server:
  - The core of the system running all server side Business Logic. The JBoss Application Server is the chosen execution environment. It stands between workstations and the Database, handling requests and storing and retrieving data and performing all necessary validations and actions. Communication with the Operator Workstations uses Enterprise Java Beans remote method invocations (RMI) technology.
- The Messaging Server:
  - Hosts the Message Broker that will support the user collaboration sessions.
- The Database Server:
  - Stores all configuration and runtime data for the system. PostgreSQL is the chosen Relational Database System. This is extended with PostGIS to support geographical data structures and spatial queries.
- The User Workstation:
  - The host device for the CIRP software. The latter will be a multi-screen Rich Internet Application. The workstation operating system will be the Microsoft Windows (XP, Vista, 7, 8, 10) while the RIA will run on top of the Java and OSGi framework, which allows the application to be fast, efficient, extensible, scalable and adaptable to the user needs.

7 Design Policies and Tactics

In this section we present the design polices and tactics that do not have sweeping architectural implications and thus do not significantly affect the overall organization of the system and its high-level structures. Nonetheless, these considerations do affect the details of the interface and implementation of various aspects of the system. Consequently, these design policies and tactics are described in the following Table.

Table 3. Design Policies and Tactics

Code	Decision / Strategy	Description	Architectural Design Strategies Ref. Codes
DPT.1	RCP version	Despite not being the latest version, RCP version 3.7 has been selected. From version 3.7 and onwards (known as the e4 platform), RCP contains a modified Application Programming Interface and internal architecture and, given that the frameworks of the design strategy ADS.7 are based on the 3.7 version, the choice for this design policy is obvious.	ADS.6
DPT.2	Desktop GIS	GeoTools is an open source Java GIS toolkit providing reference implementations of many Open Geospatial Consortium (OGC) specifications. Geotools supports many OGC standards (such as Grid Coverage, Styled Layer Descriptor, and Filter Encoding), different coordinate reference systems and transformations, as well as graphs and networks. In addition, it incorporates the Java Topology Suite with support for the OGC Simple Features Specification - used as the geometry model for vector features.	ADS.7
DPT.3	Analysis Codes	The CIRP will support both analysis software in the Java language as well as software written in another language and have built as executables (.exe). Special Java wrappers will wrap such external processes and monitor their execution. Initially only local analysis software will be supported. The second version of this Deliverable will describe any provisions for additional supported types (e.g. remote services).	ADS.1, ADS.7
DPT.4	Graphical User Interface	The end user graphical user interface will be modular and expandable and will provide easy to use Wizards, a Graphical Editing framework, Drag n Drop capabilities,and context sensitive help.	ADS.6, ADS.7
DPT.5	Unit testing	Unit testing will be performed with the use of the JUnit library.	ADS.7

8 Detailed Module Design

In this section we elaborate on the modules and components presented in Section 6 with details of definition, functions, attributes, responsibilities, and constraints.

8.1 The CEF Core Component

The CEF Core component will be the main OSGi bundle of the CIRP platform. It will function as a bridge between the different plugins that are composed to provide CIRP platform functionality. In order to provide this functionality, the CEF Core component provides the concept of extensions to contribute functionality to a certain type of API by one or more plug-ins. The type of API is defined by another plug-in as an extension point.

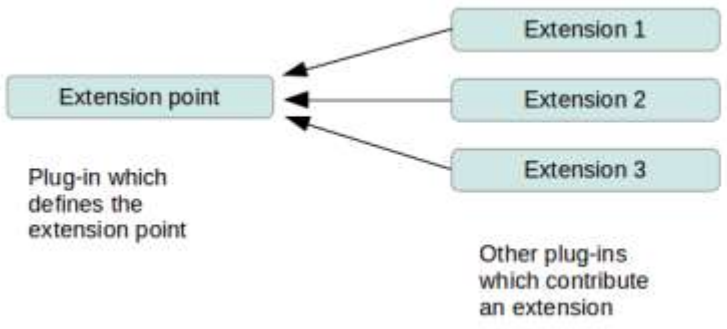

Extension 1
Extension point
Extension 2
Extension 3
Plug-in which defines the extension point
Other plug-ins which contribute an extension

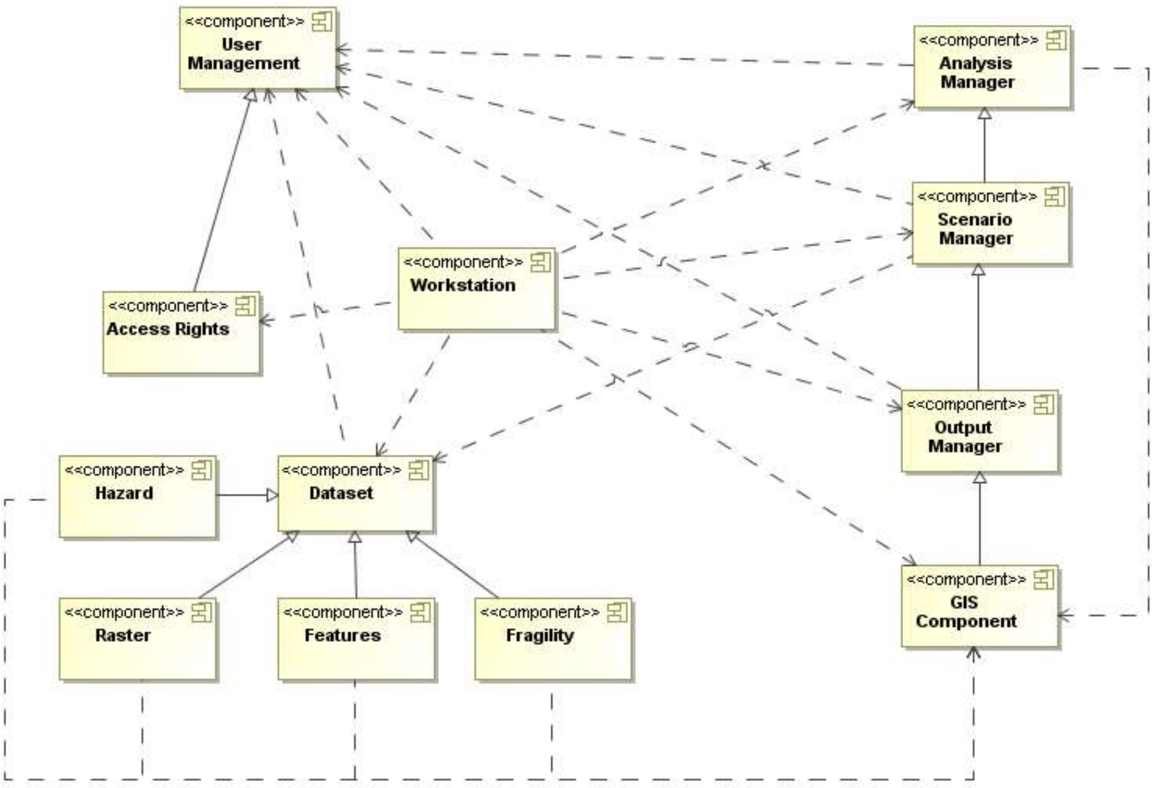
Figure 11: The concept of extension points.

Using this concept, the CEF Core component is responsible for the declaration of a set of interfaces. These interfaces are the ones that will be used by other CIRP plugins in order to “contact” each other and exchange details of their functionalities and data. These interfaces are known are extension points. In implementing this approach, each of the other plugins that wants to provide the functionality of a specific extension point must implement the equivalent interface.

Table 4. Extensions and extension points table

Term	Description
Plug-in defines extension point	The plug-in defines a contract (API) with the definition of an extension point. This allows other plug-ins to add contributions (extensions) to the extension point. The plug-in which defines the extension point is also responsible for evaluating the extensions. Therefore, it typically contains the necessary code to do that.
A plug-in provides an extension	This plug-in provides an extension (contribution) based on the contract defined by an existing extension point. Contributions can be code or data.

The following figure visualizes the connections between the different components of the CIRP framework. Each connection corresponds to an extension point that is defined on the core component and provides the desired functionality between each component. In this diagram, connections are shown as arrows. The source component of each arrow plays the role of the contributor to the specific extension point, whereas the end of each arrow indicates extension point consumers. It is obvious that each component is able to play either the role of extension point contributor or the role of extension point consumer, or both of these roles (for different extension points). Two types of arrows are used. Dashed arrows represent a “use” relationship, where each component that is located on the end of a dashed arrow uses the extension point provided by the component located on the source of the arrow. Normal arrows represent a dependency relationship. A dependency relationship is a relationship in which the functionality of the component located at the end of the arrow is depended on the functionality (extension point) provided by component at the source the arrow.



<<component>> User Management
<<component>> Analysis Manager
<<component>> Scenario Manager
<<component>> Workstation
<<component>> Access Rights
<<component>> Output Manager
<<component>> Hazard
<<component>> Dataset
<<component>> GIS Component
<<component>> Raster
<<component>> Features
<<component>> Fragility

Figure 12: CIRP Risk Assessment Component Diagram

8.1.1 Conceptual Design

The Core component is based on OSGi (Open Services Gateway Initiative) technology that determines the logic components (bundles) as well as on an Extension Registry. The Extension Registry is a sub- component of the Core component. It is the place where predefined extension points are combined with extensions. An extension point is a statement made by a bundle to indicate that it is open to extension with new functionality in a specific manner. This statement is declared using the XML / XML Schema language. The life cycle of any given bundle is specified by the OSGi technology.

The concept of extension points works as follows:

- The core component declares the available extension points.
- Other components contribute the prescribed information in the form of extensions. These provide data and/or identify classes to run and locations to access.
- The Extension Registry discovers both extensions and extensions points and links them together according to their contributing components' lifecycle.
- Components that act as extension point consumers are free to access and use contributed extensions.

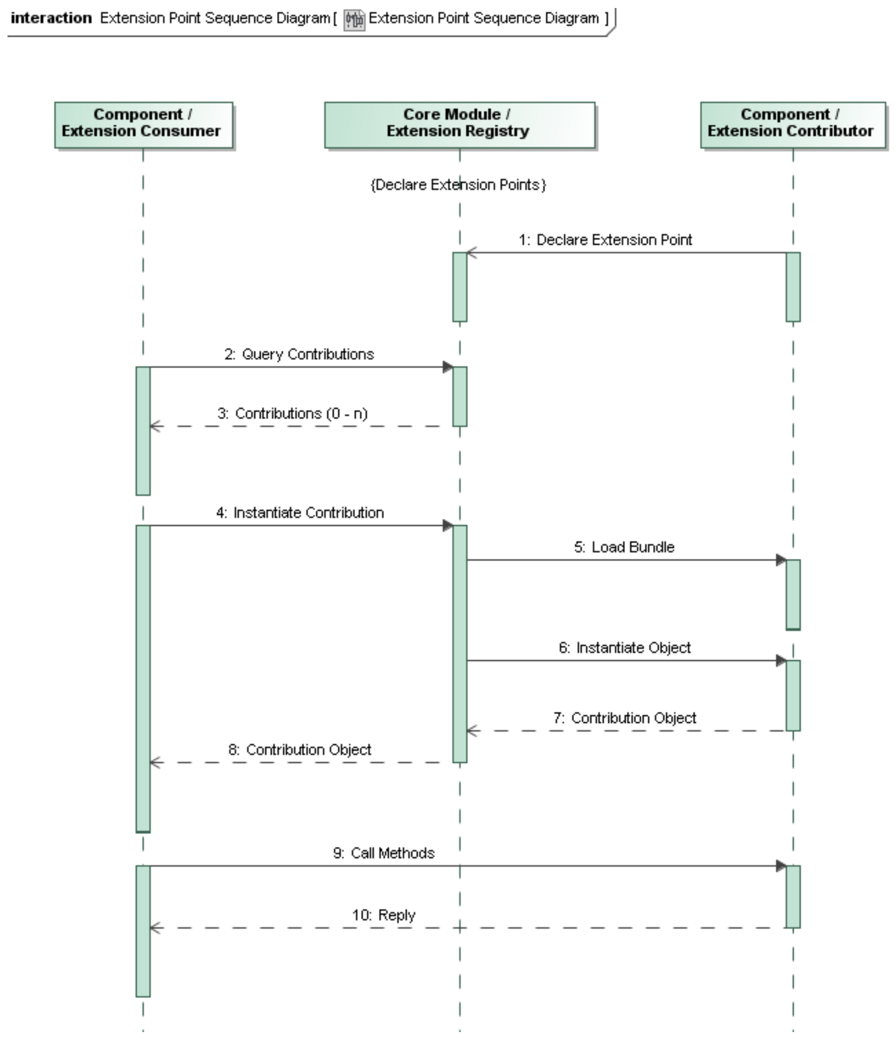

interaction Extension Point Sequence Diagram [Extension Point Sequence Diagram]
Component / Extension Consumer
Core Module / Extension Registry
Component / Extension Contributor
{Declare Extension Points}
1: Declare Extension Point
2: Query Contributions
3: Contributions (0 - n)
4: Instantiate Contribution
5: Load Bundle
6: Instantiate Object
7: Contribution Object
8: Contribution Object
9: Call Methods
10: Reply

Figure 13: Extension point sequence diagram

8.1.2 Extensibility

The extension point concept that is used by the Core component, and consequently by the CIRP framework, provides the desired modularity and extensibility. Through this concept, CIRP components are independent of each other and their exchanged functionalities are executed through the provided extension points. As a result, the CIRP framework is open by design and easily extended to new components that will be available in the future. Such extensions would include new analysis components which implement new analysis algorithms. The connection between new analysis components and the CIRP framework will be enabled through a specific extension point, the AnalysisExtensionPoint.

8.2 User Management Component

The User Management component applies hierarchical management to the users of the CIRP framework and offers the following functions:

- Hierarchical management of users according to their organisation and their responsibilities.
- Role Management
- User Groups Management
- Management of functionalities access rights
- Display of interconnected users in real time

- Exchange of synchronous and asynchronous short text messages between users

8.2.1 Hierarchical management of organisations

The User Management component supports the organisation's hierarchical structure and its operating structures. The user with the corresponding access rights has the ability to create, modify, and delete services in a hierarchical manner (organization tree). The organisation tree plays an important role since it defines the access rights to the information and associated services for each user. Each user, depending on their position at the hierarchy, can execute the privileged scenarios and monitor the respective assets and scenario results.

The hierarchical user management structure is displayed in a tree structure in the administration perspective (see Section 8.8) but also to those users / roles that have access to the organisational structure via the corresponding access right.

Each user belongs to an organisation and can only access the tree structure from his organisational department and below. This means that each user has access only to information that corresponds to his level and below. Similarily, the user can import and process data relevant to the access rights he has. CIRP User Management component allows the management of an unlimited number of user groups.

8.2.2 Roles and Access Rights

The Role and Access Rights component of the CIRP framework allow users to define and manage an unlimited number of roles. The role management includes the configuration of the layout of the different Views and Perspectives (see section 8.8) of the CIRP user interface that each role can access, the ordering of the available perspectives in accordance to the number of monitors (multiple monitor configurations), and the available functionalities that are offered per role type.

The component provides a set of permissions that permit accessing of framework's capabilities and functionalities. Access rights are assigned by both role level and user level for greater flexibility. The framework is able to adapt to the user permissions according to the organisation's hierarchical structure and its operating structures. Each user, depending on their role and its position in the organisation's hierarchy, can monitor information relevant to their role and context.

CIRP framework will offer the following base set of access rights:

- Management of organisation services
- User Management
- Usage of specific GIS tools
- Scenario Management (creation , editing, execution)
- Execute specific analysis types
- Initiate collaboration sessions

8.3 Repository Manager Component

All data used and created in the CIRP framework is stored in Data Repositories. There are three kinds of repositories: Local Repositories (folders on the local machine drive), Database Repositories, and Remote Repositories. The CIRP framework transparently exposes each repository type: creating a local repository on the user's system that contains caches of other repositories.

The component that is responsible for the management of repository types and that implements the connection between the other CIRP framework components with the repositories is the Repository Manager component. This component has a connection with the CEF Core component. This connection is achieved through the implementation of the corresponding extension point that is provided by the CEF

Core. Through this extension point, each CIRP component able to access the datasets that are stored and managed by the Repository Manager.

The Repository Manager is extended with a number of adapters. These adapters implement the management of the datasets that are provided by each of the repositories. Using these adapters, the Repository Manager, and consequently the CIRP framework, is agnostic to the type of the repository that is used (local, remote, database).

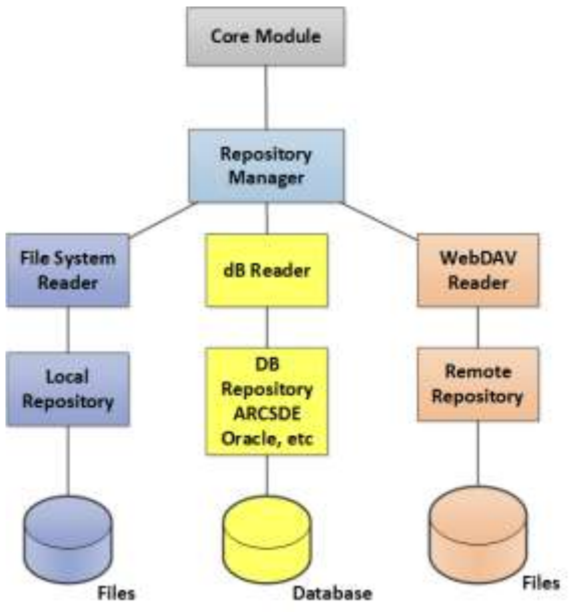


Figure 14: Repository Manager

8.3.1 Local Repository

The Repository Manager uses a predefined method to access local repositories. As mentioned previously, local repositories are located on the local machine: formatted specifically for the manager to read them correctly. A user can create a new local repository through the CIRP framework, but the structure and initialization details of that repository are left for the manager. This ensures the specific repository formatting expected by the framework.

The Repository Manager implements a dataset cache on the local filesystem in order to support the agnostic use of repositories. This local cache will be organised according to a specific hierarchical structure. It will contain the following sub-folders (organised based on dataset schema ids).

- cache – this folder contains the datasets and properties folders:
  - datasets – this folder contains the actual data files for each dataset, grouped in folders by dataset schema ids
  - properties – this folder contains an xml file in a custom format that defines meta-data about each dataset, including, but not limited to
- managers – this folder contains the following subfolders:
  - scenario – contains all the scenarios created by users in CIRP framework
  - default report templates and default sets
  - unit conversions, as well as some configuration files for updates and preferences.

8.3.2 Remote Repository

A Remote Repository is used for data sharing and publishing between CIRP users. Such repositories are typically servers that can support multiple connections from different CIRP instances. Through the CIRP framework, users are able to synchronise remote repositories with their local repositories. The remote repositories functionality is provided through two types of repositories, the WebDAV and the database repository.

WebDAV repository

The most common type of remote repository is the WebDAV (Web Distributed Authoring and Versioning) repository. It is an extension of the Hypertext Transfer Protocol (HTTP) that allows clients to perform remote Web content authoring operations. The WebDAV protocol provides a framework for users to create, change, and move documents on a server (typically a web server or web share). The most important features of the WebDAV protocol include the maintenance of properties about an author or modification date, namespace management, collections, and overwrite protection. Maintenance of properties includes such things as the creation, removal, and querying of file information. In the CIRP framework, WebDAV is combined with a web server in order to handle WebDAV requests directly.

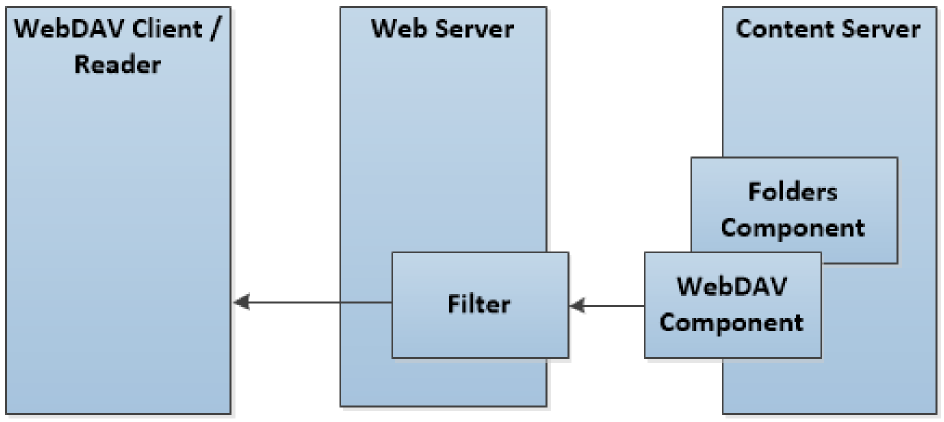


Figure 15: WebDAV architecture

The CIRP framework will support connections to any standard WebDAV server as a remote repository. In order that such remote connections can be secured, a password-protected mechanism is provided.

Database Repository

The CIRP framework will also provide a database repository. The core of this repository is a database server - which, by definition, is a remote repository - able to store and handle geospatial data. An example of this type of repository is the PostgreSQL database enhanced with the PostGIS extension. This type of repository is a combination of local and remote repository, as the database system may be installed either on the local or on a remote system. In either case, the database itself is 'remote' to the CIRP client. In the case of a remotely installed database server, multiple users are able to access it as a centralised data repository.

8.3.3 Data Formats

The Repositories will be able to store and manage datasets that are needed by the other components of the CIRP framework. These datasets are imported to the system through a set of data files that use specific data formats. The following section presents the envisaged file types.

Shapefile (.shp)

Shapefile is a common vector data format used by GIS programs for storing geospatial information. The format is developed and regulated by the Environmental Systems Research Institute (ESRI), the vendor of widely used GIS software called ArcGIS. A shapefile stores the geometry of spatial features as shapes comprising sets of vector coordinates, in addition to attribute information regarding the features. GIS datasets stored as shapefiles can support the following features: point, polyline, and polygon. The data type of a shapefile is defined at its creation and a shapefile can only contain that single type of objects [11]. A shapefile is a group of several files in the same folder, three of which are mandatory files consisting of the essential information to make up a shapefile. [12]

Mandatory files are:

- .shp: Shape file featuring the geometry.
- .shx: Positional shape index of the feature geometry allowing quick forwards and backwards searches.
- .dbf: Database file containing the attribute tables for each shape.

Optional files are:

- .prj: A text file containing the coordinate system and projection information.
- .sbn / .sbx: Spatial index of the features.
- .fbn / .fbx: Spatial index of the read-only features for shapefiles.
- .ain / .aih: An index of the active fields in the attribute table.
- .ixs / .mxs: Geocoding index for shapefiles.
- .atx: An attribute index for the .dbf file (for ArcGIS 8 and later).
- .shp.xml: Metadata file in XML format.
- .cpg: A file for the .dbf file specifying the code page to identify the character encoding.

This data format will be used by the CIRP framework in order to import and export vector GIS data, such as locations and features of assets, a water network, etc.

A sample water network consisting of shapefiles is given in figure below. The figure is displayed as a result of overlaying three shapefiles. In the figure, red, green, and white objects representing network facilities are in point data format; black objects representing water pipes are in polyline data format; and the grey area representing an administrative unit is in polygon data format.

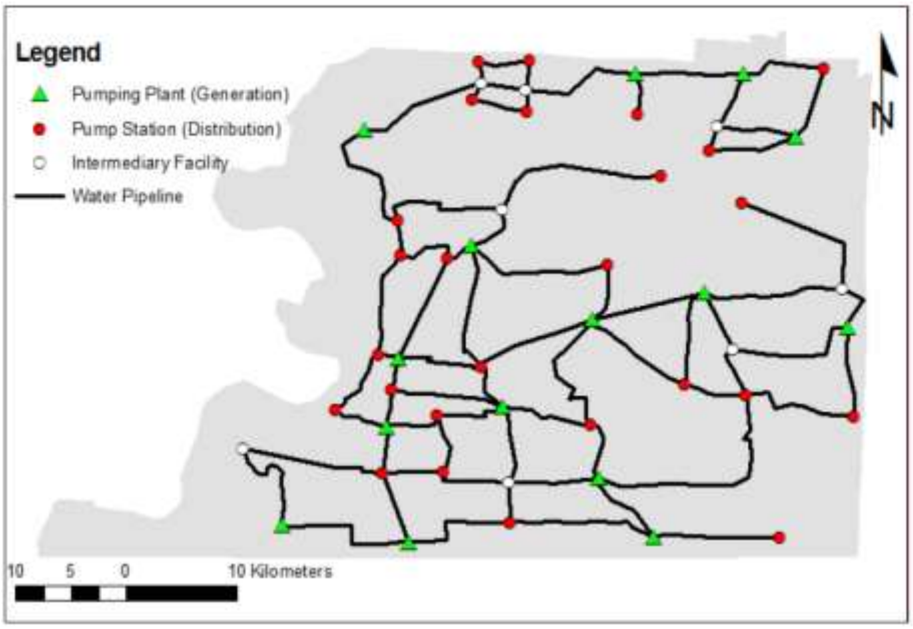
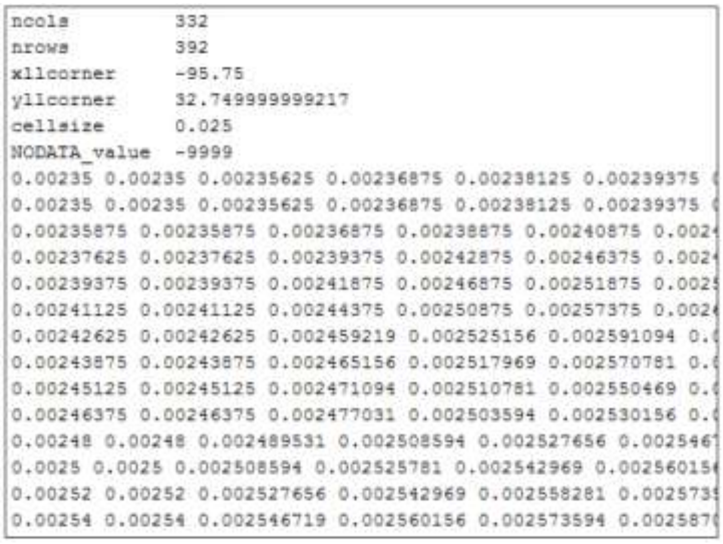


Figure 16: Sample shapefile data for point, polyline, and polygon formats. [12]

ASCII raster (.asc)

Raster data is used in field-based computational models where space is divided into regular units with fixed locations, most commonly into square grids. Field values can be gathered via either remote sensing or map algebra in which the grid units contain spatial variables for the locations they fall on.

ASCII raster is a GIS raster format to represent data in a grid structure defining the geographic space as equally sized square cells arranged in rows and columns. Each cell stores a numeric value representing an attribute related to the geographic space of that cell referenced with a pair of X and Y coordinates.



```
ncols         332
nrows         392
xllcorner     -95.75
yllcorner     32.749999999217
cellsize      0.025
NODATA_value  -9999
0.00235 0.00235 0.00235625 0.00236875 0.00238125 0.00239375
0.00235 0.00235 0.00235625 0.00236875 0.00238125 0.00239375
0.00235875 0.00235875 0.00236875 0.00238875 0.00240875 0.002
0.00237625 0.00237625 0.00239375 0.00242875 0.00246375 0.002
0.00239375 0.00239375 0.00241875 0.00246875 0.00251875 0.002
0.00241125 0.00241125 0.00244375 0.00250875 0.00257375 0.002
0.00242625 0.00242625 0.002459219 0.002525156 0.002591094 0.
0.00243875 0.00243875 0.002465156 0.002517969 0.002570781 0.
0.00245125 0.00245125 0.002471094 0.002510781 0.002550469 0.
0.00246375 0.00246375 0.002477031 0.002503594 0.002530156 0.
0.00248 0.00248 0.002489531 0.002508594 0.002527656 0.002546
0.0025 0.0025 0.002508594 0.002525781 0.002542969 0.00256015
0.00252 0.00252 0.002527656 0.002542969 0.002558281 0.002573
0.00254 0.00254 0.002546719 0.002560156 0.002573594 0.002587
```

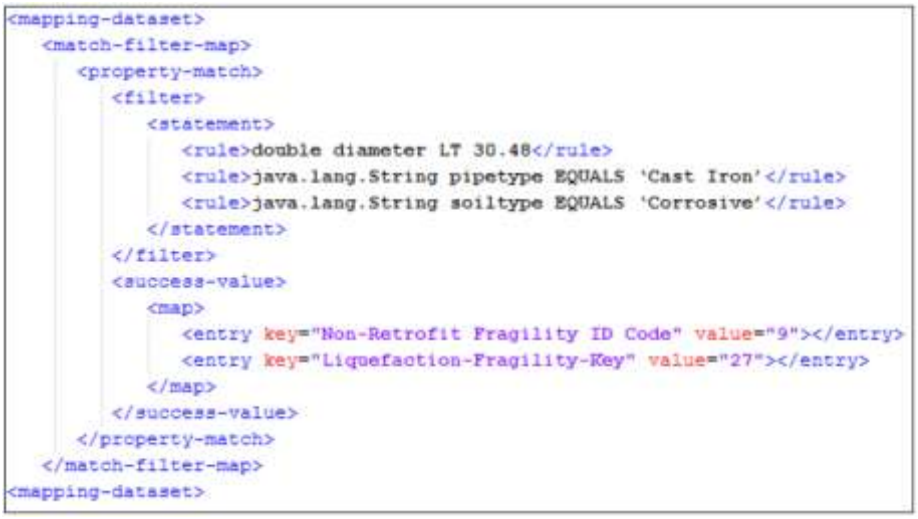
Figure 17: Sample structure of an ASCII raster dataset.

This type of data format will be used by the CIRP framework in order to import and export data related to the spread of a specific event in space, such as the values of maximum water depth in the case of a flood simulation.

Extensible Markup Language (.xml)

The eXtensible Markup Language (XML) is often used to represent data so that it can be shared among different kinds of applications and operating systems. XML provides an efficient and effective way to store, retrieve, and exchange information between peers. The XML format allows users to define their own tag set to describe data. This flexibility enables users to develop their own standard tags particular to their area of interest and to therefore make data available to other applications. Applications refer to the tag set in order to extract information from the XML document [12].

An XML document is structured in a nested manner that implies the hierarchy between tags. To illustrate the data structure of an XML document, an example of the ERGO-CORE fragility mapping file is given in the following figure.



```
<mapping-dataset>
   <match-filter-map>
      <property-match>
         <filter>
            <statement>
               <rule>double diameter LT 30.48</rule>
               <rule>java.lang.String pipetype EQUALS 'Cast Iron'</rule>
               <rule>java.lang.String soiltype EQUALS 'Corrosive'</rule>
            </statement>
         </filter>
         <success-value>
            <map>
               <entry key="Non-Retrofit Fragility ID Code" value="9"></entry>
               <entry key="Liquefaction-Fragility-Key" value="27"></entry>
            </map>
         </success-value>
      </property-match>
   </match-filter-map>
<mapping-dataset>
```

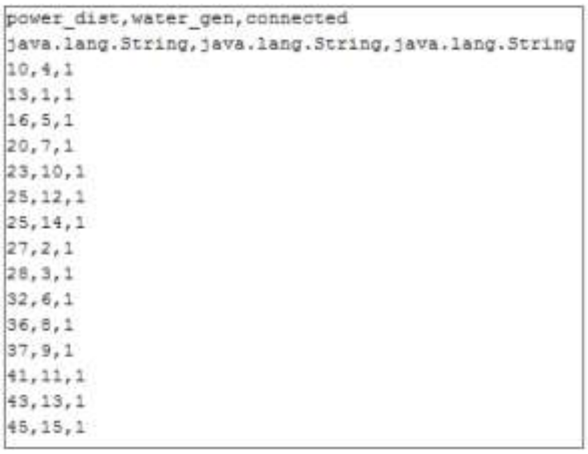
Figure 18: Sample structure of a fragility mapping XML document. [12]

The <mapping-dataset>, and <match-filter-map> tags in the XML document contain information about all matching rules for the seismic damage analysis of a buried pipeline. The <property-match> tag contains the rules for one individual match within <filter> and <success-value> tags, which provides the selection filters and matching fragility IDs for the filtered elements, respectively. The tag <statement> sets the filter statements by defining rules such as pipe diameter, pipe type, pipe joint, or corrosivity conditions. The filtered out pipes that meet the conditions must be matched with a specific fragility or damage function. The mapping entries, which are provided in <entry> tags under <map>, define the assigned fragility IDs to the selection.

Similar mapping files will be defined by the project partners and will map the fragilities of the various CIRP analyses with the inventory and asset attributes.

Comma-separated values (.csv)

Comma separated value (CSV) format is typically used to exchange and convert data between spreadsheet or database applications. While numerous implementations are present for the format, there is no formal specification, resulting in a wide variety of interpretations. This causes considerable differences among implementations. Generally each record is located on a separate line, delimited by a line break. An optional header may be provided as the first line with the same format as the record lines. The header must contain names corresponding to the individual fields in one record and must contain the same number of fields with the records. Each line must contain the same number of fields throughout the file and the last field in each record should not be followed by a comma [13]. An example for a CSV document is given with an interdependency table shown in the following figure [12].



```
power_dist,water_gen,connected
java.lang.String,java.lang.String,java.lang.String
10,4,1
13,1,1
16,5,1
20,7,1
23,10,1
25,12,1
25,14,1
27,2,1
28,3,1
32,6,1
36,8,1
37,9,1
41,11,1
43,13,1
45,15,1
```

Figure 19: Sample structure of interdependencies in a CSV document [12]

8.4 GIS Engine Component

The CIRP framework will offer advanced geospatial functionality through the set of Open Geospatial Consortium compliant open source libraries known as GeoTools. By building on top of the open source (OS) libraries, such as the Java Topology Suit (Vivid Solutions), the Open Geospatial Consortium's (OGC) libraries, and the GeoAPI, GeoTools leverages the best of these lower level libraries to provide a mid-level library of functions that simplifies the construction of complex spatial data processing applications, while hiding the complexities of data sources, feature models, and projections from the end user.

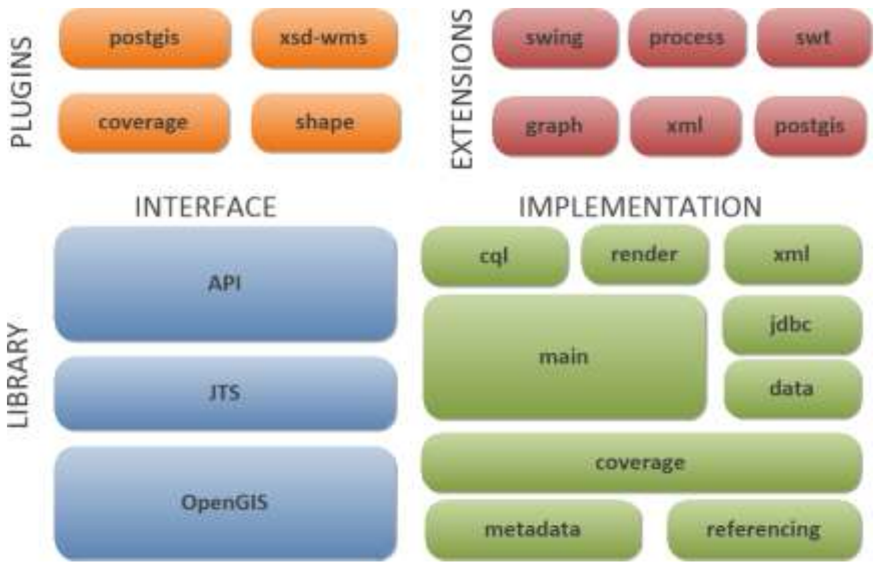


Figure 20: GeoTools components

The GeoTools libraries form a software "stack" within which each layer of functionality builds on the ideas and concepts defined in the previous one. The general architecture is depicted in the figure above. The GeoTools library consists of a set of sub-components able to provide functionalities related to the supporting of additional data formats, different coordinate reference system authorities, and much more.

The GeoTools library provides support for many GIS vector formats including ESRI Shapefile, GML, WFS, PostGIS, Oracle Spatial, ArcSDE, MySQL, GeoMedia, Tiger, VPF, and MIF. It also gives support for several raster formats including ESRI ArcInfo ASCII Grid Format, GRASS ASCII Grid Format, geo-referenced image format, and OGC WMS.

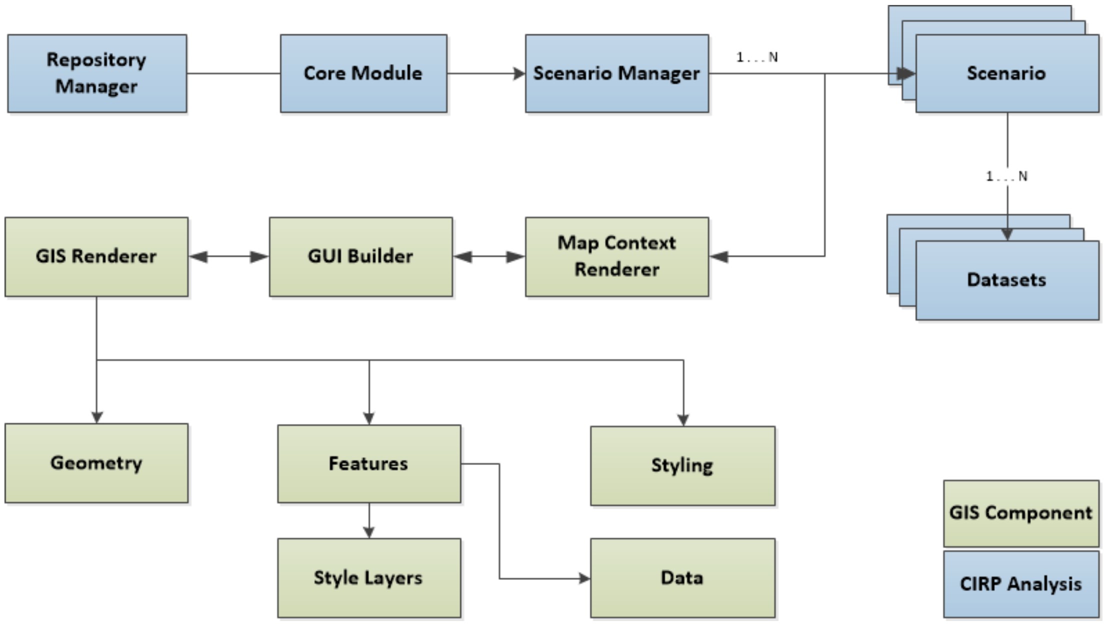


Figure 21: GIS component.

The CIRP framework will utilize the GeoTools functionalities in order to fulfill the following requirements:

1. The support for GIS format files, such as the raster and ascii grid format files. The support of GIS format files, such as ascii grid and raster files, is needed in order to create the datasets. These files may contain information that is required by analysis procedures and may include data such as building assets, networks, hazards, etc.
2. The provision of GIS functionalities, in order to execute the desired analysis procedures. Many Analysis components will base their analysis procedures on GIS functionalities, such as coverage and topology computations.
3. The provision of 2D Map rendering functionality, in order to visualize the analysis inputs and outputs. The Output component uses the 2D rendering functionalities provided by the map component and users will be able to select 2D GIS preferences, such as styling, data aggregation, layer ordering, attribute filtering, grid overlays, and coordinate transformation.

8.4.1 Conceptual Design

Currently GeoTools consists of three major modules:

- Main – provides the key functionality of the library, including the feature model, spatial referencing, data handling, and rendering packages.
- Plugins – provides interchangeable parts of the library, which can be added or subtracted as needed by end user programs. This includes all of the actual data source packages and implementations of spatial referencing databases.
- Extensions – provides add-ons to the library that give additional functionality not needed for simple programs, but which might be useful for some users. For example, the Colorbrewer tool and graph functions are contained in extensions [16].

Core Functionality

The core functionality of the GeoTools library is defined in the main module. This is where the internal model of GeoTools is established. It is here that the feature model is defined in a way that allows all parts of the program to agree about what a feature is and how it should be represented. This section of the library also defines how to read and write data, how to create filters that can be applied to data streams in order to select specific features, and how to style a feature in order to render it [16].

🛈 Feature Model

The feature model lies at the heart of the GeoTools library. A feature is a representation of a geographic object (e.g., a road, river, or house). In GeoTools a feature is stored as an array of Java objects that represent the feature attributes. One attribute is always the geometry (i.e., its representation in space) of the feature. The other attributes can be any type that is required by the feature being modeled. A feature is defined by a Feature-Type object, which represents the schema of the feature. This allows a program to determine what type of object an attribute should be decoded or encoded as when loading or unloading data from features. The feature class provides helper methods that provide the user with a convenient way to work with the attributes. Since an attribute can be any geographic object, features can even contain information about other features, allowing containment to be modeled by the system.

🛈 Data Stores

GeoTools provides an abstract datastore concept that allows an application developer to access data from a variety of data sources without having to worry about the actual implementation details of each data source. For example, data stored in an ESRI shapefile or a PostGIS database is accessed in exactly the same way once the datastore has been opened. The use of a factory design pattern allows developers to pass a request for a new object to the factory class, rather than needing to hardcode the reference to the class. The factory can then look up a specific implementation and return a concrete object that has been determined at runtime to the program. By providing a factory system to access plug-in datastores, GeoTools allows programs to be completely agnostic to data sources, provided that a specific data store has been implemented for the data source required.

🛈 Filters

GeoTools implements the OGC filter specification [15], which defines three groups of filters:

- Spatial filters, which involve a geometry element and a spatial relationship.
- Comparison filters, which compare attributes of features with other expressions or attributes.
- Logical operators, which allow the joining of other types of filter.

🛈 Styling and Rendering

Styling and rendering are the means of converting the abstract representation of a geographic feature into an actual drawing of a map on the screen or on paper. The GeoTools styling system is based on the OGC Styled Layer Description (SLD) language [15]. The GeoTools styling package allows programs to control the following characteristics:

- color of lines and fills
- size and type of symbol for points
- text properties (e.g., position, font, color) for labels, and
- rules for all above based on feature attributes.

🛈 Spatial Referencing

GeoTools supports almost any type of known geographic projection. It provides a collection of predefined projections as defined by the European Petroleum Spatial Group (EPSG). It contains functions that are capable to convert geometry objects from one projection to another. Additionally, GeoTools provides query projection methods such as valid geographic area, ellipsoid, and so on.

8.5 ERGO-CORE Components

The ERGO-CORE framework provides an extensible set of components for workflow based geospatial risk assessment analysis, scenario management, and tabular and geospatial data and metadata management. The following subsection presents the main design concepts. As the ERGO-CORE has been initially developed for seismic analysis, the CIRP will re-use and extend the generic functionalities in order to be used as the building blocks in the climate change risk assessment domain.

8.5.1 Conceptual Design

The methodology that is used by the ERGO-CORE framework can be divided in two main parts: structural model and topological model. Inventory, hazard, and fragility definitions are within the scope of the structural model. The topological model is the aspect within which dependency and post-event serviceability analyses are carried out through topological network connectivity and flow models.

Performance assessment of interdependent networks requires utilization of two separate models used consecutively: a structural model for damage estimation and a topological model for connectivity and flow analyses based on the output of the structural damage assessment (figure below). The inventory datasets must be provided in compliance with the requirements of both models since the output of the structural assessment is used as an input in interdependent performance assessment.

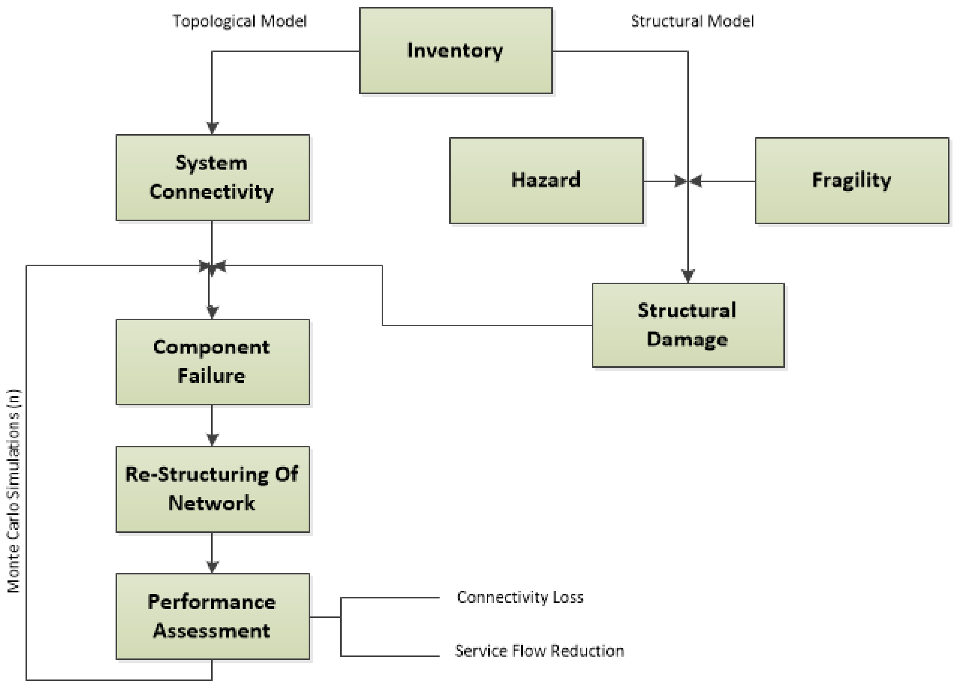


Figure 22: Interdependent Network Performance Analysis flowchart [12].

In the structural model, damage assessment of the inventory items are carried out based on specified hazard and fragility information. Each inventory item is assigned to corresponding fragility functions based on specific attributes. Fragility relations are used to estimate the expected damage based on the corresponding hazard type and value. The estimated damage is then used for failure assessment of network components (for network based analysis) in the succeeding steps of the analysis.

The Topological model is where the networks are modelled based on connectivity and flow relations. Failures of components are determined based on structural damage and interdependency effects exposed through the execution of numerical simulations. Re-structured networks with their surviving components are analysed by applying Monte Carlo Simulation techniques to determine the system performance based on reductions in connectivity and flow. System performance is the quantification of the effect of physical damage on the network flow and system serviceability. Topological analysis estimates the effects of hazards on the end-users by quantifying the amount of service loss for each individual network [12].

Structural Damage Model

Structural damage modelling is the initial step in the performance assessment of interdependent lifeline networks and CIs. All three elements (hazards, inventory, fragility) of structural damage assessment are vitally important for the achievement of accurate assessments. Thus, accurate definitions and utilizations of inventory, hazard, and vulnerability parameters are equally important [17].

Hazard and vulnerability are the agents that define the natural hazard risk. Reduction of risk can be achieved by reduction of vulnerabilities through modifications and improvements in the inventory. Vulnerability of a man-made environment is almost entirely dependent on the human factor, whereas hazard is a natural phenomenon and cannot be reduced nor prevented [12, 18].

Hazard

Hazard is the quantification of an event without any reference to human or structural loss, simply depending on the characteristics of the selected scenario. Each hazard type has a set of characteristics parameters that defines the impact of this hazard on societies of structural inventories. For example, in the case of flood hazards, these parameters can be maximum water depth, the water velocity etc.

Fragility

Structural damage levels are estimated through the application of fragility functions, which give the probability that a limit state is exceeded, or by damage functions that indicate the amount of expected damage, given an input parameter representing a level of hazard. Three approaches can be employed to develop these functions: empirical, theoretical, and judgment-based. Empirical relationships are obtained by estimating observed damage from a site or from laboratory experiments through regression analysis. When empirical data is unavailable or insufficient, modelling the systems with known or estimated capacities provides a theoretical approach. Judgment-based functions, based on expert opinion, are developed in the absence of empirical data and theoretical models [19], [20].

Topological Model

Drawing on in-depth definitions of inventory attributes, finely detailed mappings of state-of-the-art fragility information, and elaborate hazard maps developed using modern simulation tools with detailed site and source definitions result in detailed identification of critical components of lifeline networks. However, knowing the physical state of a network is not sufficient to make predictions regarding the impact of its operational loss after disruption. Interdependent performance analysis tools for topological networks can be employed to simulate the post-event conditions of the analysed networks by applying connectivity and flow algorithms. With the interdependent approach, lifeline networks can be modelled as a mutually dependent system of systems where the state of one network is influenced by its dependencies.

Topology, in geographic information systems, defines the interaction between objects independent from their geometry. One of the most significant outcomes of topology in a GIS is to be able to perform network analyses, neighbourhood analyses, and similar related analytical processes without the requirement for specific spatial information.

Data Structure

Topological data structures provided by modern GIS software support three main functions: connectivity, area definition, and contiguity. These functions are in turn supported by three basic topological data structures: arc-node, polygon-arc, and left-right. The arc-node structure will be used in CIRP to handle connectivity models such as road, transportation, utility lifelines, and telecommunication networks where the order of links and connecting nodes on a route is important. The polygon-arc structure is the data structure used to store area definition in the database. This defines the relationships between areas and surrounding arcs such as coastlines or borders. The left-right structure provides contiguity (neighborhood) functionality where information on polygons surrounding neighboring polygons is stored. This enables representation of relationships such as administrative units which are neighbors to other units, or buildings that have sides facing to a road, etc.

The principles of geospatial topology are based on graph theory. The networks must be represented as directed graphs in order that an analysis tool may process them correctly. A graph can be described as a pair $G = (V, E)$ of sets where V is defined as the set of vertices, and E is defined as the set of edges. A graph is defined as connected if any two vertices of a graph are always connected with a path. Moreover, if every edge in a graph is assigned an initial and terminal vertex, defining all flow directions, such a graph is defined as a directed graph. Analysed networks are built as directed graphs. Electric power network, potable water network, and natural gas lifeline systems are examples of CI networks that are able to be represented as directed graphs by the CIRP framework. CI networks generally consist of buried pipelines, power lines, power plants, substations, water wells, pumping stations, water tanks, natural gas pressure regulator stations, gate stations, switching stations, junction units, and other similar elements.

Each node in a topological network, as employed to represent a CI network, is assigned to either one of three types according to their roles in the physical network: generation, intermediary, and distribution.

- Generation Nodes: These are the network facilities where network flow is originated from. Supplier facilities like water wells in potable water networks, power plants in electric power networks, and gate stations in urban natural gas networks are defined in the topological model as origin nodes.
- Distribution Nodes: These are the discharge nodes of the networks where the flow is supplied to the end users. Transformers in electric power distribution networks and water tanks in water transmission networks are defined in the topological model as destination nodes.
- Intermediary Nodes: These are the nodes in the network without any generation or discharge. Incoming flow is transmitted without any change. Pressure regulator pumps and water tanks in water distribution networks, pressure regulator stations in natural gas networks, and all types of junctions are defined in the topological model as intermediary nodes [12].

Interdependent failure mechanism

The interdependency mechanism can be explained with an example of a power and a water system given in the following Figure. The power distribution nodes provide the electric power requirements of water nodes. In the given system, power and water systems operate simultaneously, but not independently. Each connection between power and water nodes represents a dependency and is quantified with a dependency level. Dependency levels determine the functional behavior of water network nodes in case the failure of power distribution nodes supplying electricity to them.

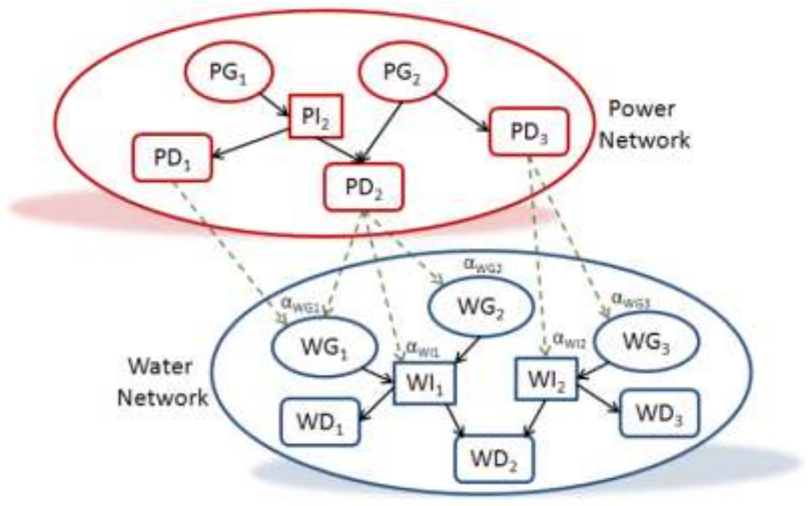


Figure 23: An example for illustrating network interdependency [12]

Failure of a network node can either be structural or operational. Operational failure can further be divided into two as non-functionality of a node due to loss of connectivity within the network and non-functionality due to failure of all feeder electric power nodes. This condition implies that a node can still fail due to interdependency effects although being structurally undamaged. [12]

Network performance measures

Interdependent system behaviour can be assessed by simulating the system response. Based on hazard damage assessment results and interdependency definitions between systems, the post event states of the networks may be obtained and, subsequently, system reliability can be assessed by application of the appropriate simulation models (e.g. Monte Carlo). In order to measure the functional loss of a system when some of the components are likely to be dysfunctional, two performance measures are defined that quantify those losses: Connectivity Loss (CL), and Service Flow Reduction (SFR). These measures assess the network performance with metrics depending on the topological settings of the network, or with more detailed metrics depending on supply, demand, and flow patterns additional to the topological settings.

1. Connectivity Loss (CL) measures the ability of every distribution node to receive flow from generation nodes [21]
2. Service Flow Reduction (SFR) determines the amount of flow that the system can provide compared to the demand before the disturbance [22].

8.5.2 Scenario Manager

The scenario manager will be responsible for the management of the scenarios. A scenario is defined as the process undertaken to calculating the impact of a specified hazard in a set of assets in a predefined area. The Scenario Manager component will be responsible for keeping track of all objects associated with a particular scenario in a user's workspace. A user's workspace can support multiple scenarios simultaneously, and each scenario is independent. A scenario also keeps a record of all analysis workflows associated with that scenario. The scenario contains all of the datasets associated with a user's project as well as any settings applied to the datasets (e.g., styles, visibility, etc.) and any datasets that can be visualised.

The scenario manager will offer the following functionalities:

- Creation of a new scenario
- Editing an existing scenario
- Deleting an existing scenario

- Provision of datasets to scenarios
- Execution of certain analysis
- Display on the map the scenario object that can be rendered

Data Sources

The repository component is used as the data source for the scenario manager. As a result the repository manager can provide the required datasets for each one of the scenarios. The common procedure is that the scenario manager creates a request with all available datasets for specific types of data (based on the metadata) and the repository returns the specified type of data from all available positions.

8.5.3 Analysis Manager

The Analysis Manager is the base component where all analyses are registered. CIRP developers can add a new analysis by registering a new plug-in with this extension point. When a user creates a new scenario through the workbench, they are prompted to define an optional region of interest, make an analysis selection, and identify the inventory data. The Analysis Manager then attempts to populate selected analysis fields with the appropriate data types and default input parameters, and in some cases, the user will need to provide values for the parameter inputs. If a dataset for an analysis field (e.g. Water Treatment Areas) is not available in the user's scenario, then the user will have the option to search all registered repositories to find an appropriate dataset, or to create a new dataset using an analysis. If the user chooses to create a new dataset, the Analysis Manager will first execute the analysis necessary to create the dataset, and then use the result as an input into the original analysis.

Hazard Component

In the context of CIRP a hazard analysis denotes a natural hazard simulation service that will be executed inside the CIRP platform. Hazard analyses are configured in the Workflow Analysis Editor view. The latter will provide users with the ability to load and execute custom scenarios. The user must provide all required information related to the event that will be simulated. When all data is provided, the user will be able to execute the hazard analysis.

The Workflow Analysis Editor view takes advantage of the Graphical Editor Framework (GEF) and the Zest library. These Eclipse libraries provide the ability to easily create intuitive graphical representations of data which is used by the analysis manager; a workflow-type graph of the analysis is generated and updated as the user interacts with the analysis. The coupling of a form-based user interface with a visual representation of the workflow provides the user with multiple options for managing their analyses.

Vulnerability Component

The functionality of the vulnerability component is based on fragility datasets and, therefore, on fragility functions. A fragility function represents the cumulative distribution functions of the capacity of an asset to resist an undesirable limit state. Capacity is measured in terms of the degree of environment excitation at which the asset exceeds the undesirable limit state. For example, a fragility function could express the wind factors that a building structure can tolerate before it collapses. Users are able to import new fragility datasets according to their needs and their access rights.

For each asset (building, network link, and/or node), the user has to map fragilities with the structure type in order to identify the necessary parameters. The discrete probabilities of damage states are obtained by taking the difference between adjacent curves. In the following figure, Flood fragility functions for water transportation pump stations are depicted to illustrate this point.

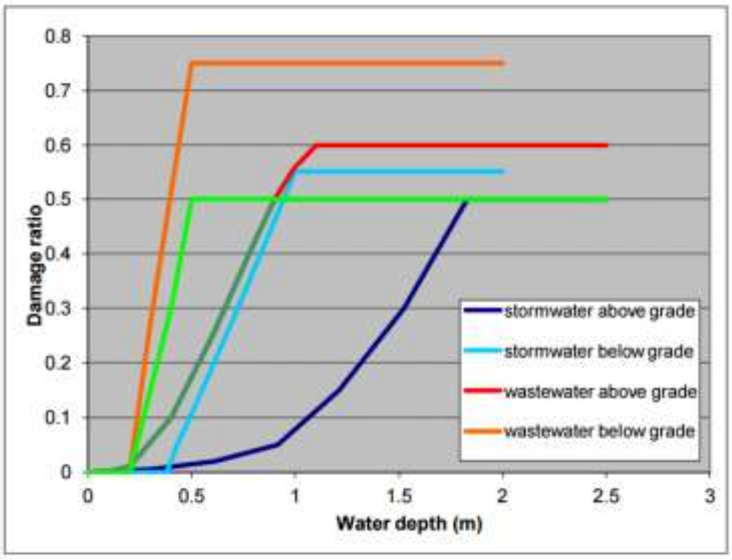


Figure 24: Flood fragility functions for water transportation pump stations [23]

CIRP users will be able to upload their own fragility curve dataset by following a graphical interface procedure. Fragility curve datasets must be be provided as fully compliant *.xml files similar to:

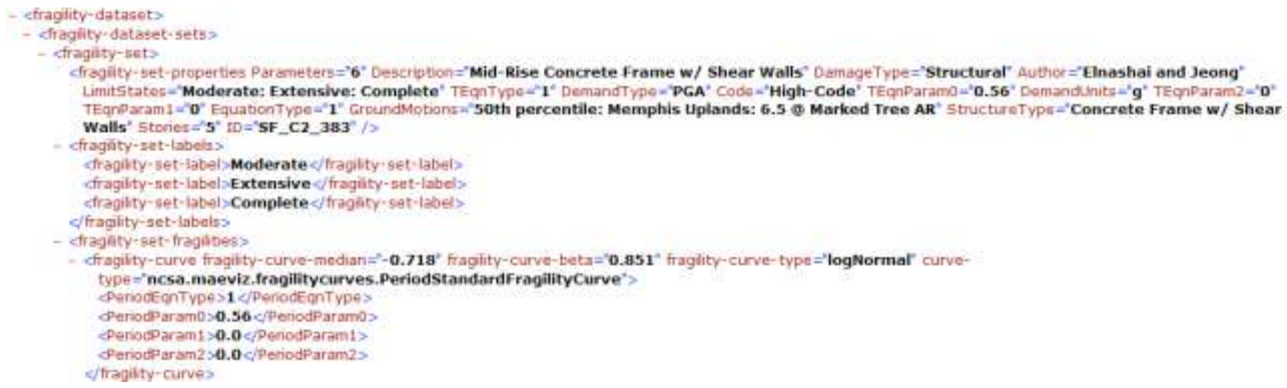

```
- <fragility-dataset>
  - <fragility-dataset-sets>
    - <fragility-set>
        <fragility-set-properties Parameters="6" Description="Mid-Rise Concrete Frame w/ Shear Walls" DamageType="Structural" Author="Elnashai and Jeong"
          LimitStates="Moderate: Extensive: Complete" TEqnType="1" DemandType="PGA" Code="High-Code" TEqnParam0="0.56" DemandUnits="g" TEqnParam2="0"
          TEqnParam1="0" EquationType="1" GroundMotions="50th percentile: Memphis Uplands: 6.5 @ Marked Tree AR" StructureType="Concrete Frame w/ Shear
          Walls" Stories="5" ID="SF_C2_383" />
      - <fragility-set-labels>
          <fragility-set-label>Moderate</fragility-set-label>
          <fragility-set-label>Extensive</fragility-set-label>
          <fragility-set-label>Complete</fragility-set-label>
        </fragility-set-labels>
      - <fragility-set-fragilities>
        - <fragility-curve fragility-curve-median="-0.718" fragility-curve-beta="0.851" fragility-curve-type="logNormal" curve-
            type="ncsa.maeviz.fragilitycurves.PeriodStandardFragilityCurve">
            <PeriodEqnType>1</PeriodEqnType>
            <PeriodParam0>0.56</PeriodParam0>
            <PeriodParam1>0.0</PeriodParam1>
            <PeriodParam2>0.0</PeriodParam2>
          </fragility-curve>
```

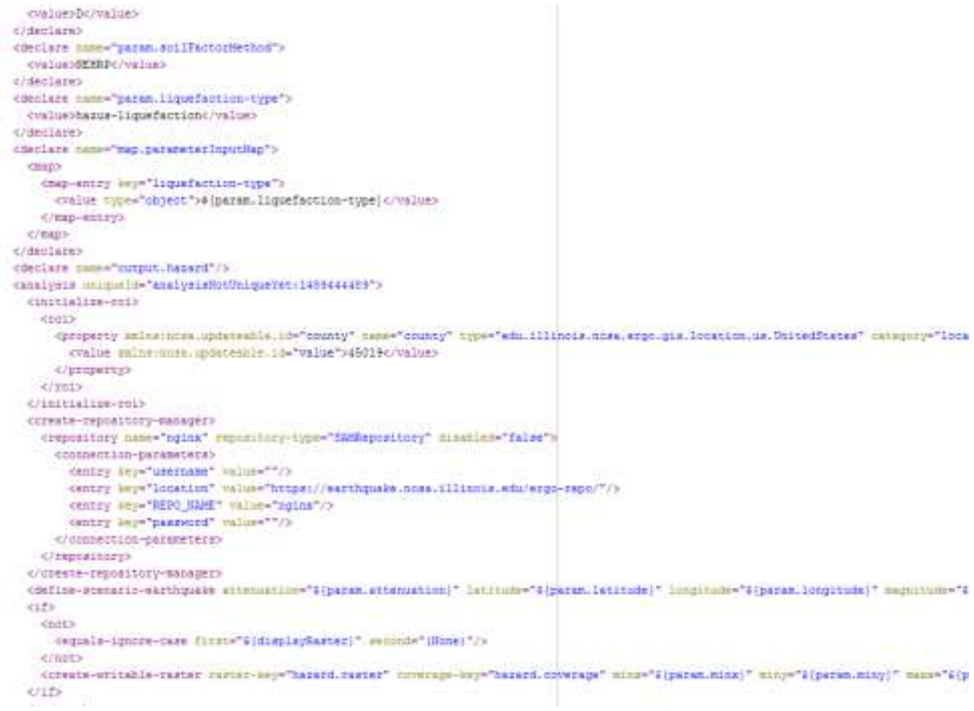
Figure 25: Extract of the fragility dataset provided by ERGO-CORE framework

Analysis Execution Engine

In order to execute an analysis, the analysis component compiles the analysis chain into a self-contained script. The implementation of the script will be built using Ogrescript. Ogrescript is an XML-based scripting language developed at NCSA that runs inside of a container called ELF. Ogrescript is also an Eclipse RCP application; so it is easily customised and adapted to run as a plug-in inside the ERGO-CORE framework. The extensibility of Ogrescript was one reason it was chosen for executing framework analyses. The primary reason, however, was to meet an important design goal that the execution of a single task (e.g. computing the damage of an asset) should be decoupled from the execution of an entire analysis (e.g. computing the hazard risk an area).

This execution plug-in will allow users to focus on analysis requirements individually and provides the proper separation from the execution engine. Thus, the developer of a new task does not need to understand the inner workings of the mechanism behind the iteration across an entire dataset. It also allows new and more advanced iteration and scaling functionality to be deployed without affecting the domain-focused analyses. The scripts generated by the analysis component can be exported and stored in a human- and machine- readable XML format. This ability provides the user a history of the run, as well as the ability to execute the script elsewhere.

The main reason Ogrescript was developed was to manage scientific software on supercomputing clusters and to enable an analysis to run as a standalone command line client that can be executed on many different operating systems and architectures. Ogrescript provides the ability for users to develop analysis workflows in an intuitive graphical user interface on their desktop and execute the workflow in an environment that makes the most sense for the scale of a use case; running on a local machine or on a remote cluster or supercomputer.



```
    <value>D</value>
  </declare>
  <declare name="param.soilFactorMethod">
    <value>SEEBP</value>
  </declare>
  <declare name="param.liquefaction-type">
    <value>hazus-liquefaction</value>
  </declare>
  <declare name="map.parameterInputMap">
    <map>
      <map-entry key="liquefaction-type">
        <value type="object">${param.liquefaction-type}</value>
      </map-entry>
    </map>
  </declare>
  <declare name="output.hazard"/>
  <analysis uniqueId="analysisNotUniqueYet:1489444489">
    <initialize-roi>
      <roi>
        <property xmlns:ncsa.updateable.id="county" name="county" type="edu.illinois.ncsa.ergo.gis.location.us.UnitedStates" category="loca
          <value xmlns:ncsa.updateable.id="value">45019</value>
        </property>
      </roi>
    </initialize-roi>
    <create-repository-manager>
      <repository name="nginx" repository-type="SAMRepository" disabled="false">
        <connection-parameters>
          <entry key="username" value=""/>
          <entry key="location" value="https://earthquake.ncsa.illinois.edu/ergo-repo/"/>
          <entry key="REPO_NAME" value="nginx"/>
          <entry key="password" value=""/>
        </connection-parameters>
      </repository>
    </create-repository-manager>
    <define-scenario-earthquake attenuation="${param.attenuation}" latitude="${param.latitude}" longitude="${param.longitude}" magnitude="$
    <if>
      <not>
        <equals-ignore-case first="${displayRaster}" second="(None)"/>
      </not>
      <create-writable-raster raster-key="hazard.raster" coverage-key="hazard.coverage" minx="${param.minx}" miny="${param.miny}" maxx="${p
    </if>
```

Figure 26: Ogrescript XML example

The following figure displays a brief overview of the Analysis Execution Engine procedure. A number of analysis progress listeners are registered with the Analysis Manager. When a new analysis is sent for execution through the Analysis Manager, an analysis ogrescript containing all the input/output datasets and additional parameters specified by the user during analysis creation is formed. This ogrescript is then fed into the Analysis Execution Engine, which combines the analysis ogrescript with host information (to load and store the specified datasets) as well as scheduling services (for load balancing) to generate an ogrescript executable. This base class of this executable implements the ELF-internal interface IScript. In detail, this executable consists of a payload, or script, along with initial properties, input and required output.

ELF is a transient, compute-resource-resident container for running such payload scripts. The ELF wrapper serves as a container whereby the execution of the aforementioned ogrescript executable is transformed to an ELF executable and monitored, with failures or success being reported back to the Analysis Manager through the registered listeners. While the ELF executable is being executed, analysis execution listeners notify its status to the Analysis Manager, triggering the update of the progress bar on the display screen accordingly. Internally, the analysis included in the correspondent scenario (.xml) gets updated too, and saved in the repository after passing through the Analysis Manager.

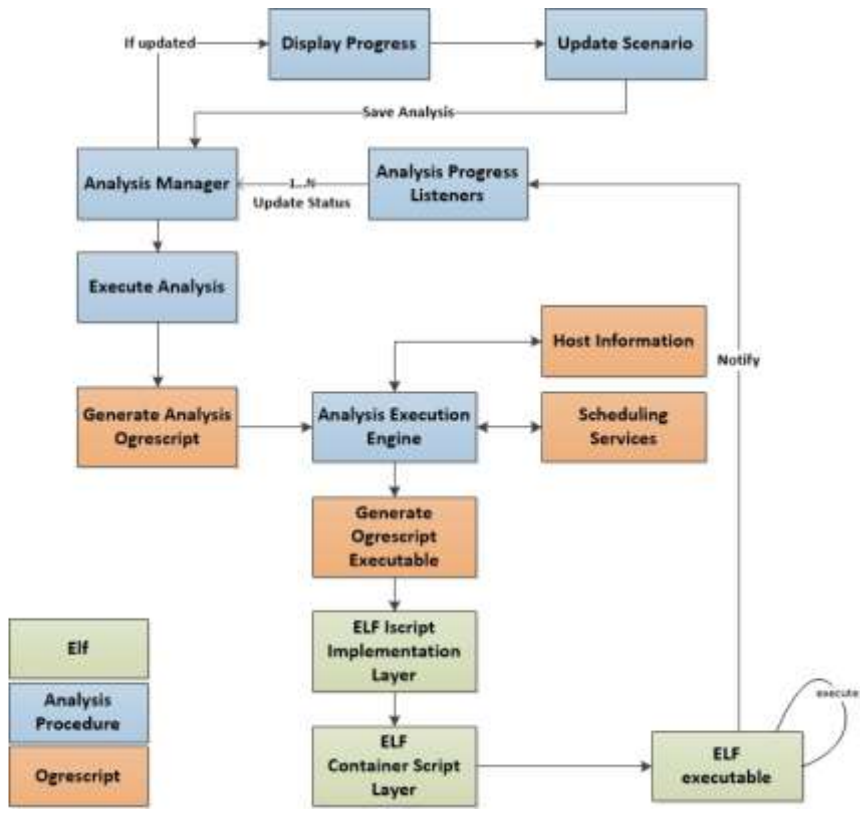


Figure 27: Analysis Execution Engine example

8.6 Output Manager

The Output Manager will provide a number of tools for displaying the output of an analysis. In detail, the output can be rendered using different formats, such as maps, charts, reports or tables. The technologies used for each output format are:

- 2D map
- 3D map
- JFreeChart (charts)
- JasperReports (reports)
- Ktable / NatTables (tables)

Below follows a brief description of each technology per output format:

8.6.1 2D Map Plugin

The rendering for 2D Maps will be implemented using the GeoTools library. GeoTools functionality is described previously in Section 8.4. GeoTools is used for dataset visualisation in 2D, supporting different file formats like shapefiles (.shp), rasters (.asc), .csv, .xml , .txt, and compressed files (.gz, .zip). The 2D Map component of GeoTools will render each loaded spatial dataset and will support different styling based on SLDs. Other supported actions include user-definable zoom through on-map tools or by mouse-drawing an bounding box to zoom to (i.e. drawing a rectangle).

8.6.2 3D Map Plugins

The 3D Visualization component will be based on the various 3D Map components of the CEF that offer, to users, the capability to navigate through three-dimensional landscapes created by the combination of aerial and satellite imagery, elevation data, and other levels of two-dimensional or three-dimensional information. Users can import vector layers, images and elevation data from multiple sources, to add information in landmarks such as image tags or text, buildings, point-cloud models, two-dimensional and three-dimensional entities, and default routes from GIS files and databases. Empowering a user to import unique or proprietary information into a three-dimensional map creates an exciting and interactive application that supports highlighting of specific elements, their function, their relationships, and proximity aspects with a separate area display. Extension points for 3D map services will be adapted for the Output Manager

8.6.3 Chart Plugin

Charts are generated using the JFreeChart library. JFreeChart is an open-source Java chart library that supports many output types, including Swing and JavaFX components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG). In the CIRP framework, JFreeChart will be used to display charts of an analysis input or output datasets and various statistics. Charts can be zoomed for further details, and are customizable in terms of changes in properties and display. Charts can be rendered as bar charts, stacked bars charts, line charts or pie charts.

8.6.4 Report Plugins

The Report plugins will provide the ability to visualize scenario results in a report form. This tool consists of a visual report designer and a runtime component for Java. The visual report designer is responsible for the design of the report template, whereas the runtime component is responsible for the visualization of scenario results according to the predefined templates. The template of these reports consists of lists, charts, crosstabs, forms and documents, and compound reports with multiple features and can be embedded on rich client applications. Information gleaned from such embedded Reporting tools may be used in both real-time decision-making and to track and analyse historical data. These reports are able to be exported to different file formats, such as pdf, .excel, html, xml, csv, and text.

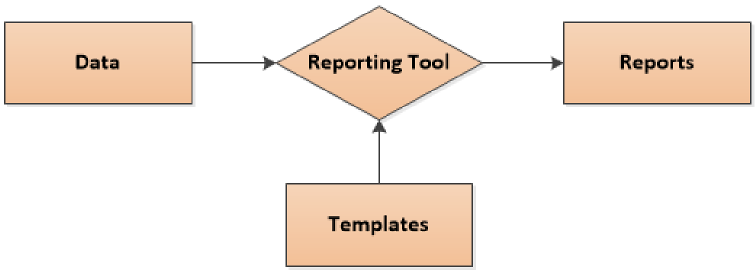


Figure 28: Reporting Tool Architecture

8.6.5 Tables

Tables are also used for result visualization purposes. Such tables are based on the NatTable framework. NatTable is a framework to create table, grid, and tree controls. It is designed to handle very large data sets and is as flexible as possible in terms of functionality and styling. NatTable provides a user-orientated table experience including sorting, filtering, grouping, and fixed/frozen columns and rows.

8.7 User Collaboration Components

The user collaboration components in CIRP are a set of OSGi bundles that will support the initiation, control, and participation in collaborative user sessions. The goal here is to allow users to navigate simultaneously and synchronously into the CIRP map based environment and to then exchange information related to analysis results, messages, and annotated map areas where - for instance - risk resilience adaptation measures should be considered. The following functionality will be provided:

- A user will be able to create a collaboration session and invite other CIRP users
- CIRP users invited are notified via the message broker component

- Once a user participates in the collaborative session they are able to select scenario results to be exchanged with the session user group. In addition a set of tools for map annotation, guided map navigations, and chat are activated.

8.8 Graphical User Interface

This section presents in brief the CIRP Graphical User Interface (the detailed UI as designed will be described in the second iteration of this deliverable). The CIRP GUI will be based on the CEF workbench which consists of a set of perspectives, views, toolbars and popup wizards. A perspective is a set of Views organised in the application window in an appropriate manner. Each perspective determines the visible actions and views within a window. A View when visible takes a part of the perspective, it can be minimised (in any of the right, left and bottom toolbars) or it can be extracted from the main window (detached View). In addition each View can contain one or more controls or buttons in the view bar area. The following perspectives are envisaged:

- Administrator
- Scenario Manager
- Collaboration
- Help

The main toolbar will contain a number of buttons that activate each of the perspectives. The administrator perspective will only be visible to the Administrator user role.

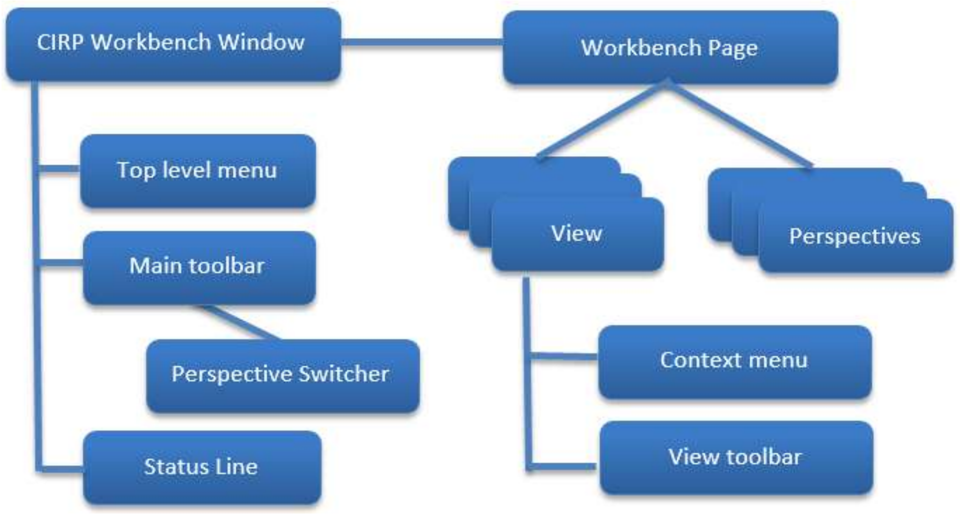


Figure 29: The CIRP UI components

'Drag-n-drop' functionality will be implemented throughout the user interface as it provides a quick and easy mechanism for users to re-order and transfer data within the CIRP application views (repository, map, scenario management, etc.).

In addition, context sensitive help will provide the ability to perform a single search in order to find information from any number of sources (federated information search).

Finally cheat sheets will guide users through tasks. A task is broken down into steps and presented to the user one step at a time, and the user checks off the steps as he/she completes them. Cheat sheets will come in two forms: simple (one task, several steps), and composite (many sub-tasks, each having many steps).

9 Conclusions

The CIRP detailed design as presented in this Deliverable has the aim of defining the CIRP architectural elements and identifying both the scope of the individual components and services and the context in which they will operate.

The CIRP has been designed as a collaborative modelling environment where new risk assessment and geospatial analyses can extend the analysis workflow and where multiple scientific disciplines can work together to understand interdependencies, validate results, and present findings in a unified manner. This provides an efficient, pragmatic, and effective solution that integrates existing modelling tools and data into a holistic resilience model in a standardised fashion.

The detailed CIRP design offers an environment for what-if scenario analyses with the selection of model chains, climate data, and CI inventories in order to calculate damages and assess the resulting risk. The CIRP platform as designed provides a user friendly environment to enable the intuitive design and analysis of modelling scenarios created for any combination of climate hazard and CI assets. In this way, users are able to understand the impact of various adaptation strategies or to quantify the potential impact of a catastrophic event on society.

The CIRP extensible modular architecture can be shared across multiple communities to enable CI policy maker, owners, and scientists to leverage existing software analysis types and algorithms, inventory types, and fragilities while not binding the underlying platform to a particular scientific domain. This pluggable, open architecture is what will allow CIRP to support a wide variety of domain specific functionality isolated in plugins; to repackage different functionalities as a starting point for new applications, and to be extended to add new analytical capabilities in the future.

10 Bibliography

[1] Understanding risk. Review of open source and open access software packages available to quantify risk from natural hazards

[2] ERGO Framework (Multi-Hazard Assessment, Response and Planning) http://ergo.ncsa.illinois.edu/

[3] FEMA & NIBS (1999) HAZUS99 user and technical manuals, Federal Emergency Management Agency Report: HAZUS 1999,Washington D.C.,USA.

[4] FEMA (2003) HAZUS-MH Technical Manual . Federal Emergency Management Agency, Washington, DC, U.S.A. Giovinazzi S.

[5] Crowley H., Colombi M., Crempien J., Erduran E., Lopez M., Liu H., Mayfield M., Milanesi M. (2010) GEM1 Seismic Risk Report: Part 1, GEM Technical Report 2010-5, GEM Foundation, Pavia, Italy.

[6] Crowley H., Cerisara A., Jaiswal K., Keller N., Luco N., Pagani M., Porter K., Silva V., Wald D., Wyss B. (2010) GEM1 Seismic Risk Report: Part 2, GEM Technical Report 2010-5, GEM Foundation, Pavia, Italy

[7] Keith Porter, "A Beginner's Guide to Fragility, Vulnerability and Risk", University of Colorado Boulder and SPA Risk LLC, Denver CO USA, 11 February 2016

[8] Open Geospatial Consortium, Inc., Retrieved September 21; website: http:// www.opengeospatial.org/

[9] ''The open source Java GIS toolkit,'' Retrieved September 21; website: http:// geotools.codehaus.org/

[10] T. McLaren, C. Navarro, J. S. Lee, S. Hampton, N. Tolbert, J. Myers, B.F. Spencer, A. Elnashai, "Coupling Network Analysis using MAEviz Building Blocks", Engineering Research and Innovation Conference, Honolulu, Hawaii, 2009

[11] "ESRI Shapefile Technical Description.",White Paper. Environmental Systems Research Institute, California.

[12] H. C. Ünen, "Seismic Performance Analysis of Interdependent Utility Network Systems", PhD Thesis, Department of Geodesy & Photogrammetry Engineering, Geomatics Engineering Programme, November 2011

[13] R. Shafranovich,"Common Format and MIME Type for CSV Files.",SolidMatrix Technologies Inc., 2005

[14] Open Geospatial Consortium, "Styled layer descriptor (SLD) implementation specification", document 02-070, http://portal.opengeospatial.org/files/?artifact id=1188 , 2002

[15] Open Geospatial Consortium, "Filter encoding implementation specification", document 04-095 http://www.opengeospatial.org/standards/filter , 2004

[16] I. Turton, "Geo tools" In Open source approaches in spatial data handling (pp. 153-169), Springer Berlin Heidelberg, 2008

[17] J. Betbeder-Matibet, "Seismic engineering", London, UK: ISTE/Wiley, 2008

[18] A. S. Elnashai and L. Di Sarno, "Fundamentals of earthquake engineering.", Chicester, UK: Wiley, 2008

[19] K.A. Porter and A.S. Kiremidjian, "Assembly-based vulnerability of buildings and its uses in seismic performance evaluation and risk-management decision-making.", Stanford, CA: John A. Blume Earthquake Engineering Research Center, 2001

[20] Michelle T. Bensi, Fernando Ferrante, Jacob Philip, "Challenges Associated with the Fragility and Reliability Assessment of Flood Protection Measures and Recent Initiatives", http://pbadupws.nrc.gov/docs/ML1435/ML14350B131.pdf

[21] Y.S. Kim, B. Spencer, and A.S. Elnashai, "Seismic Loss Assessment and Mitigation for Critical Urban Infrastructure Systems.", Urbana: NSEL Report Series, NSEL-007. University of Illinois at Urbana-Champaign, 2008

[22] Kim, Y. S., Spencer, B., Song, J., Elnashai, A. S., and Stokes, T. (2007). Seismic Performance Assessment of Interdependent Lifeline Systems. Retrieved from https://www.ideals.uiuc.edu/handle/2142/8927.

[23] Stefan Reese, Doug Ramsay, "RiskScape: Flood fragility methodology", http://www.victoria.ac.nz/sgees/research-centres/documents/riskscape-flood-fragility-methodology.pdf