

**RiverCure Portal: Exploring Geographic Features on
Context Definition and Integration with the HiSTAV
Hydrometric Tool**

Jorge Miguel da Silva Marques

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Alberto Manuel Rodrigues da Silva

Doutor Jacinto Paulo Simões Estima

Examination Committee

Chairperson: Prof. Luís Manuel Antunes Veiga

Supervisor: Prof. Alberto Manuel Rodrigues da Silva

Members of the Committee: Prof. João Luís Gustavo de Matos

January 2021

Abstract

Flood events are becoming more prominent every day, causing serious problems to the people they affect, such as physical, psychological and material harm. With the recent technological advances, hydrometric modeling tools capable of predicting floods and quantifying their severity, like HiSTAV, are emerging and becoming more prominent. However, the dissemination of the results and the user interaction with these tools is still lacking. By taking advantage of the advancement of Web and GIS technologies, an opportunity to overcome these challenges arises. The objective of this thesis is to develop the RiverCure Portal (RCP), a web application that, by integrating HiSTAV in its workflow, makes it widely available to knowledgeable users, disseminates its results to the interested stakeholders, and streamlines the user interactions necessary to use the tool. The RCP data model was specified using the Application Specification Language (ASL), an ITLingo language focused on simplifying the development process by defining rigorous and platform-independent specifications. This thesis details the motivation behind the RCP, and how these methodologies and technologies were leveraged to create a web application that integrates HiSTAV. A detailed walkthrough on how to create a HiSTAV simulation from RCP is described together with images from the user point of view. Finally, an evaluation and a conclusion are provided, with the former being focused on the comparison of RCP with another similar tool called Mike 21 and, the latter containing the main takeaways from the development of RCP and whether it fulfills its intended goals.

Keywords: RiverCure; Water Management; Geographic Information System (GIS); HiSTAV; Application Specification Language (ASL)

Resumo

Eventos de cheias, como inundações, causam graves problemas às pessoas afectadas por eles, causando danos físicos, psicológicos e/ou materiais. Com o avanço da tecnologia nos dias de hoje, surgem ferramentas de modelização hidrométricas capazes de prever inundações e de quantificar a sua gravidade, como o HiSTAV. No entanto, a divulgação dos resultados e a interação dos utilizadores com estas ferramentas ainda é fraca. Aproveitando a evolução e adoção da tecnologia web, surge uma hipótese de solucionar estes aspetos. O objetivo desta tese é desenvolver o RiverCure Portal (RCP), uma aplicação web que, ao integrar o HiSTAV no seu domínio, o torna amplamente disponível a utilizadores especializados, dissemina os seus resultados aos stakeholders, e melhora as interações dos utilizadores com esta ferramenta. O modelo de dados do Portal RiverCure foi definido em Application Specification Language (ASL), uma linguagem pertencente ao projeto ITLingo destinada a simplificar o processo de desenvolvimento, ao usar especificações rigorosas e independente da plataforma. Esta tese detalha a motivação por detrás do desenvolvimento do RCP, e como estas metodologias e tecnologias foram alavancadas para criar uma aplicação web que integra o HiSTAV. Uma descrição detalhada sobre como criar uma simulação do HiSTAV a partir do RCP acompanhada por algumas imagens do ponto de vista do utilizador é providenciada neste documento. Por último, são fornecidas uma avaliação e uma conclusão, com a primeira a focar-se na comparação do RCP com uma ferramenta semelhante chamada Mike 21 e, a segunda visa responder a se este cumpre os seus objetivos pretendidos.

Palavras-Chave: RiverCure; Gestão no Domínio da Água; Geographic Information System (GIS); HiSTAV; Application Specification Language (ASL)

Acknowledgments

Many challenges were faced, during this last year, to complete this research. As such I would like to leave me my gratitude to the people who helped me walk this path.

First, I would like to thank Professor Alberto Silva and Professor Jacinto Estima, who were always available to help and provide the necessary guidance and feedback for the completion of this dissertation.

Second, to the RiverCure team, who provided the necessary insight, explanations, and time to make sure this project could become a reality.

Third, to Ivo Gamito, who helped me build RiverCure Portal and was always available to discuss ideas and solve problems that emerged during this last year.

Last, but not least, to my family and friends who accompanied me during my journey on Instituto Superior Técnico.

To everyone involved in this project either directly or indirectly, thank you!

Table of Contents

Abstract	iv
Resumo	vi
Acknowledgments	viii
Table of Contents	x
List of Figures	xiii
List of Tables	xvi
List of Specifications	xviii
List of Acronyms	xx
1 Introduction	1
1.1 Context	1
1.2 Work Objectives.....	2
1.3 Research methodology.....	4
1.4 Research Scope	6
1.5 Document structure	7
2 Background	8
2.1 Model Driven Engineering	8
2.2 Web Engineering	9
2.3 IFML.....	10
2.4 ITLingo.....	11
2.5 OGC Standards – ISO 19125.....	14
2.6 WaterML	17
2.7 Coordinate Reference Systems	17
2.8 GIS Data	19
2.9 Technologies.....	21
3 Related Work	28
3.1 SNIRH.....	28

3.2	SVARH.....	29
3.3	HiSTAV	30
3.4	Mike 21	31
4	RiverCure Portal Requirements	33
4.1	RiverCure Portal Interface	33
4.2	RiverCure Portal Context.....	34
4.3	Data Collection	41
4.4	HiSTAV Integration	42
5	RiverCure Portal – Simulation Pipeline	44
5.1	RCP Data Model Definition and Generation from the ASL Specification (Phase 1)	44
5.2	RCP’s Context Creation and Definition (Phase 2).....	45
6	RCP and HiSTAV Integration (Phase 3)	60
6.1	Pre-processor	61
6.2	Paraview Web Visualizer.	62
6.3	Solver2D	62
6.4	RCP and HiSTAV integration API	64
6.5	Simulation Pipeline Overview	66
7	Evaluation.....	69
7.1	Context Definition Evaluation	69
7.2	Simulation Evaluation	69
7.3	Discussion	70
8	Conclusion	72
8.1	Conclusion	72
8.2	Future Work	72
	References	74
	Appendix A: RCP Specifications	74
	Appendix B: RCP Images	78

List of Figures

Figure 1. RCP simulation process in BPMN notation.	3
Figure 2. DSR mapped for this thesis.	5
Figure 3. Main phases of the research project.	6
Figure 4. RiverCure Portal and other associated systems.	6
Figure 5. Software product, platforms, transformations, and models	9
Figure 6. Example of user interface and corresponding IFML model. The user selects an item in the list and displays its details in the same view container.	10
Figure 7. UML diagram suggesting the relationships between RSL's elements.	13
Figure 8. Overview of Geometry object model.	16
Figure 9. Geometry class operations.	16
Figure 10. CRS adaptation of a 3D to a 2D representation example.	18
Figure 11. Geometry representation of a lake.	19
Figure 12. Raster examples.	20
Figure 13. A Mesh example.	21
Figure 14. GeoDjango Admin interface view.	23
Figure 15. SNIRH web portal.	29
Figure 16. From left to right: Data Acquisition, Central Processing and Information Distribution	30
Figure 17. Mike 21 working area example, after import of land and water data.	31
Figure 18. SNIRH system monitorization network and sensor popup	34
Figure 19. Context Visualization on RCP.	38
Figure 20. DTM example.	39
Figure 21. Context data architecture.	46
Figure 22. RCP navbar.	46
Figure 23. RCP admin page view of the context creation section.	47
Figure 24. Context List Page View.	48

Figure 25. Context detail page view.	49
Figure 26. Prompt after clicking the edit button on a Context Detail page.	50
Figure 27. Possible notifications after upload request examples.	50
Figure 28. Dropdown options for selection of geometry to define on RCP.	51
Figure 29. Manage Context page.	52
Figure 30. Popup examples.	53
Figure 31. Geojson representation of a Boundary.	54
Figure 32. Success message on mesh generation request.	55
Figure 33. Available options for the RCP user based on whether the mesh has been generated.	56
Figure 34. Context Event List Page.	56
Figure 35. Event Context Detail page.	57
Figure 36. HiSTAV simulation results displayed on Event results page.	59
Figure 37. HiSTAV integration with RCP.	60
Figure 38. Output of a successful pre-processor request.	61
Figure 39. Coura context generated mesh.	62
Figure 40. Sensor bnd file data example.	63
Figure 41. Resulting VTK from a HiSTAV simulation example.	64
Figure 42. RCP landing page.	78
Figure 43. RCP about page.	78
Figure 44. RCP login screen.	79
Figure 45. RCP register screen.	79
Figure 46. RCP admin Backoffice.	80
Figure 47. RCP Context access request page.	80
Figure 48. RCP hydro feature page.	81
Figure 49. RCP sensors page.	81
Figure 50. RCP Context list page.	82
Figure 51. RCP Context detail page.	82
Figure 52. RCP upload Context files prompt.	83
Figure 53. RCP manage Context page.	83
Figure 54. RCP profile page.	84

List of Tables

- Table 1.** Web Engineering Areas and Topics. 9
- Table 2.** Classification of RSL constructs: abstraction levels versus specific concerns. 12
- Table 3.** GeoDjango Spatial Datatypes. 15
- Table 4.** Spatial Lookup available in GeoDjango according to the DBS. 24
- Table 5.** Datatypes mapping from ASL to Django. 45

List of Specifications

Spec. 1. Data Entity example.....	12
Spec. 2. DataEntityCluster example	12
Spec. 3. Definition of a UIContainer in ASL example.....	13
Spec. 4. Definition of an action in ASL example	14
Spec. 5. Context ASL specification	35
Spec. 6. Sensor ASL specification.	41
Spec. 7. Context Sensor ASL specification.....	42
Spec. 8. Sensor Observation ASL specification.....	42
Spec. 9. RCP data enumerations.	74
Spec. 10. RCP HydroFeature specification.	75
Spec. 11. RCP organization specification.	75
Spec. 12. RCP user specification.	75
Spec. 13. RCP sensor alarm specification.....	76
Spec. 14. RCP boundary line specification.....	76
Spec. 15. RCP Context boundary point specification.	76
Spec. 16. RCP Context event specification.	77
Spec. 17. RCP Context access request specification.....	77

List of Acronyms

APA	<i>Agência Portuguesa do Ambiente</i>
API	Application Programming Interface
ASL	Application Specification Language
BPMN	Business Process Model and Notation
CL	Cell Length
CRS	Coordinate Reference System
DB	Database
DBMS	Database management system
DSL	Domain-specific language
DSML	Domain-specific modeling language
DSR	Design Science Research
DTM	Digital Terrain Model
FCT	<i>Fundação para Ciência e Tecnologia</i>
GIS	Geographic Information Systems
GPS	Global Positioning System
HiSTAV	High performance version of Strong Transients in Alluvial Valleys
INAG	<i>Instituto Nacional da Água</i>
INESC-ID	<i>Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento em Lisboa</i>
IOGP	International Association of Oil and Gas Producers
IST	<i>Instituto Superior Técnico</i>
MDE	Model Driven Engineering
ML	Machine Learning
MVC	Model View Controller

OGC	Open Geospatial Consortium
OOP	Object Oriented Programming
ORM	Object Relational Mapping
RCP	RiverCure Portal
RSL	Requirements Specification Language
SFA	Simple Feature Access
SNIRH	<i>Sistema Nacional de Informação de Recursos Hídricos</i>
SRID	Spatial Reference Identifier
SVARH	<i>Sistema de Vigilância e Alerta de Recursos Hídricos</i>
UML	Unified Modelling Language
WE	Web Engineering

1 Introduction

1.1 Context

Countries and their responsible organizations for the water management require an ever-increasing level of sophistication to protect and provide better conditions to their citizens. Moreover, with the advances in technology and improvement in data collection, the amount of available data is bigger than ever, allowing the creation of richer and more complex hydrometric models to analyze these data. However, to fully leverage the advantages provided by these models, it is necessary to build pipelines responsible for gathering the collected data and feeding it to the models so they can correctly produce reliable and valuable results, which can be then disseminated through the different stakeholders.

In 2018 the **RiverCure** project was promoted to solve this problem. **RiverCure** is a research project supported by *Fundação para a Ciência e a Tecnologia* (FCT), developed by *Instituto Superior Técnico* (IST), *Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento em Lisboa* (INESC-ID) and *Agência Portuguesa do Ambiente* (APA). It addresses the issue of reducing uncertainty and improving forecasting capabilities of hydrodynamic and morphodynamical mathematical models for flood simulation, water resources management and habitat protection [1]. Profiting from recent advances in the collection and processing of crowdsourced information produced by riverine communities and, shared through websites and social media, RiverCure intends to contribute to the improvement of the forecasting capabilities of mathematical hydrometric models of rivers, like HiSTAV [2], by making an efficient and systematic use of this curated crowdsourced and authoritative data through assimilation and calibration [3]. To bring these goals into fruition a web application called RiverCure Portal, or RCP for short, that enables the gathering and funneling of water data collected by riverine communities and crowdsourced data into HiSTAV and, shares the results of the simulation executions of this tool with the stakeholders, was proposed.

RCP is based on different systems (explored in chapter 3) which were partially integrated or emulated during this research. These systems are (i) **SNIRH** (*Sistema Nacional de Informação sobre Recursos Hídricos* or Hydric Resources National Information System in English), a water resources monitorization system, that saves data in APA's databases and releases the information to the public in a web portal (<https://snirh.apambiente.pt>) [4]; (ii) **SVARH** (*Sistema de Vigilância e Alerta de Recursos Hídricos* or Hydric Resources Surveillance and Alert System in English), a subsystem of SNIRH that provides the hydraulic state of rivers and reservoirs (i.e., water levels, flow rate, stored volumes and water quality) and relevant meteorological information in real time, while also allowing the prediction of its possible evolution [5]; and (iii) **HiSTAV**, a modelling effort aimed at delimiting the critical tsunami inundation areas in an urban waterfront and to quantify the associated severity [6]. Moreover, RCP should also integrate the treatment of images obtained from crowdsourcing sources, such as social networks (e.g., Instagram) using machine learning (ML) techniques to automatically classify these images according to

their depicted water level. However, this integration is not part of the scope of this research and shall be treated in future work.

Additionally, this research proposed to tackle the problem related to the fact that the creation and support of web applications are becoming increasingly knowledge demanding, with the involved actors being confronted by increasing demands for improved quality, increased complexity of products and services, higher flexibility, shorter lead-times, etc. [7]. All these problems, result in an increased monetary and temporal costs for the involved companies and individuals. In order to solve this problem, this research takes the development of the web application a step further by first devising a specification of RCP's data model in ASL. Inserted in the ITLingo initiative [8], ASL is a platform-independent application specification language that allows the user to write application specifications, from which a functional program can be generated [9]. Using an ASL specification as a starting point for the RCP development, ensured that the development of the RCP resulted in artifacts aligned with the stakeholder's vision, as these models were defined by sharing a common vision and knowledge among technical and non-technical stakeholders, thus facilitating and promoting the communication among them. Furthermore, they made the project planning more effective and efficient while providing a more appropriate view of the RCP system and, allowing the project control to be achieved according to objective criteria [10]. This development methodology is known as Model Driven Engineering (MDE) [11].

1.2 Work Objectives

This research purpose is divided in 3 objectives. The first objective is the implementation of an initial version of the RiverCure Portal (or RCP for short), with a special focus on the creation of the Simulation Pipeline, which connects RCP to HiSTAV enabling bilateral communication. This implementation starts from an ASL specification of the RCP's data model, from which all the code related with the data models is generated.

The second objective is the development of features for the RCP that can be used to generate valid inputs for a HiSTAV simulation. These features are referred as "Context Definition" and, are a set of functionalities that allow a user to define georeferenced geometries with associated properties and names, that are later used by HiSTAV as an input for its simulation. Moreover, because the data is used across different systems (RCP and HiSTAV), it is necessary to ensure data consistency, compatibility and portability between these two applications through the use of standards like OGC [12] and WaterML [13].

The third objective is the integration of HiSTAV with RiverCure through the implementation of a communication channel. This way, HiSTAV simulations requests and its results visualization are done on the RCP. This communication channel is a pipeline, called "Simulation Pipeline", implemented through the development of a REST API [14], that allows the exchange of data between the two systems. This pipeline is invisible to the final user, who only interacts directly with RCP, by performing certain actions that trigger requests to HiSTAV, accompanied by the necessary data defined during the "Context

1.3 Research methodology

This research followed the Design Science Research methodology (DSR). The DSR is an iterative methodology that combines principles, practices, and procedures. It provides guidance for research in Information Systems (IS) and other disciplines [15], [16]. Design Science lays emphasis on systematic, testable and communicable methods [17].

Hevner et al. [15] proposed a guideline in the application of DSR in information systems area, based on the following aspects:

Design as an Artifact: DSR must produce a viable artifact in the form of a construct, a representation, a technique, or an instantiation.

The relevance of the problem: The basic objective of design-science research is to develop technology-based solutions to important and relevant business problems.

The design evaluation: The utility, the quality, and the efficacy of the design artifact must be demonstrated rigorously through a well-executed evaluation method.

Research contribution: Effective DSR must offer a clear and provable contributions in the areas which design artifact is apply, design foundations, and/or design methodologies.

Research Rigor: The DSR depends upon rigorous methods application in both evaluation and the construction of the design artifact.

Design as a search process: The search for effective artifact depend the utilization of the available ways to reach desired outputs while the laws in the problem environment are still satisfy.

Communication of research: Design-science research presentation must be effective both the technology-oriented as well as the management-oriented consultation.

These guidelines translate into 6 phases of the DSR process [16]:

Problem identification and motivation: Define the specific research problem, justify the value of a solution, and motivate the researcher to investigate the answer.

Define the objectives of a solution: Infer the objectives of a solution for the defined problem and the knowledge of what is achievable.

Design and development: Create the artifactual solution. Such artifacts can be constructs, models, methods or instantiations created to address the designated problem.

Demonstration: Demonstrate the efficacy of the artifacts to solve the problem.

Evaluation: Examine and measure how well the artifacts support the problem's solution, comparing the objectives to the results collected from use of the artifacts in the demonstration.

Communication: Communicate the problem, the artifacts, and the design, considering its relevance, utility, novelty, and effectiveness to researchers and other relevant audiences.

Figure 2 shows which tasks were performed during each phase of DSR during this research. It is important to note that the phases are cyclical resulting in several executions of the same task. The cycle from Define Objectives & Goals to Evaluation was performed weekly, i.e., there was a weekly discussion with the stakeholders about the tasks done during that week. A cyclical development ensured a higher flexibility during the development of RCP, allowing for the adaptation of requirements on necessity (e.g., new problem is discovered). It also gave the stakeholders the opportunity to evaluate the product being developed every week, increasing the trustworthiness of the developed features by reassuring us that they were solving the problem correctly and, according to the stakeholders needs.

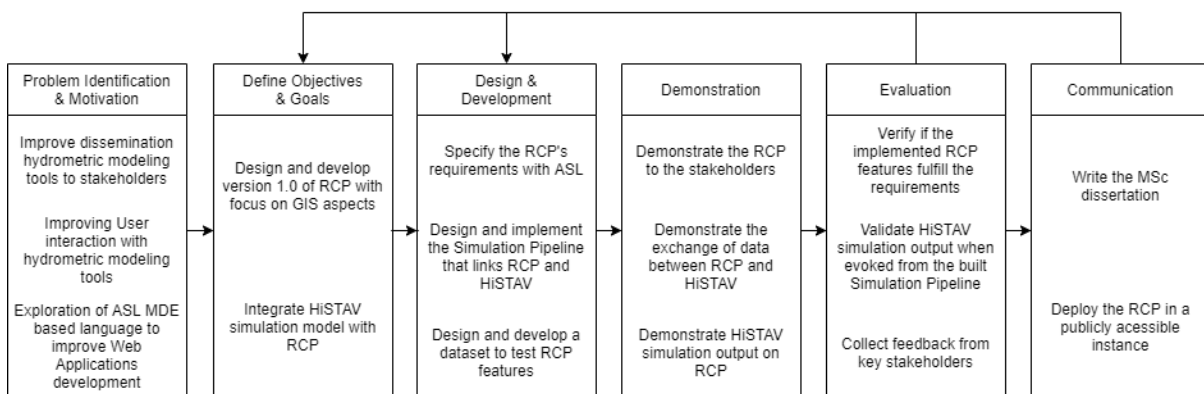


Figure 2. DSR mapped for this thesis (Adapted from [16]).

In practice, the implementation of this project was divided in three distinct phases. By the end of these phases, two main software artifacts were developed which, together, form the Simulation Pipeline, which is the main objective of this project. Which is the process that allows the exchange of data between them and, the request for HiSTAV simulations from the RCP, as explained earlier.

The first two phases consisted in the development of the RCP, using different development approaches in each phase. During phase 1, the RCP was defined in ASL and, from this specification, code generation mechanisms were used to generate a Python/Django [18] data model for the RCP. From the generated data model, conventional development processes (i.e., programming) were used to implement RCP's use cases. This implementation corresponds to phase 2 of this project and its biggest. It was during this phase that all the functionalities and features of the Simulation Pipeline on RCP side were developed. This consisted mainly on the implementation of the Context creation and definition features allowing HiSTAV full integration during phase 3.

The main idea was to create a seamless workflow that any specialized user can use to simulate a given Context on HiSTAV and, later share the results with the intended stakeholders. Ergo, phase 3 consisted in the integration of RCP with HiSTAV by creating mechanisms that allow the communication between them. For this effect, a simple Flask [19] API was developed, to enable communication between RCP

and HiSTAV. Figure 3 shows a holistic view of this research phases order and tasks. The first two tasks are part of phase 1 and, the other two are part of phase 2 and 3, respectively, with each task consisting of at least one entire cycle on the DSR methodology (Define Objectives and Goals to Evaluation).

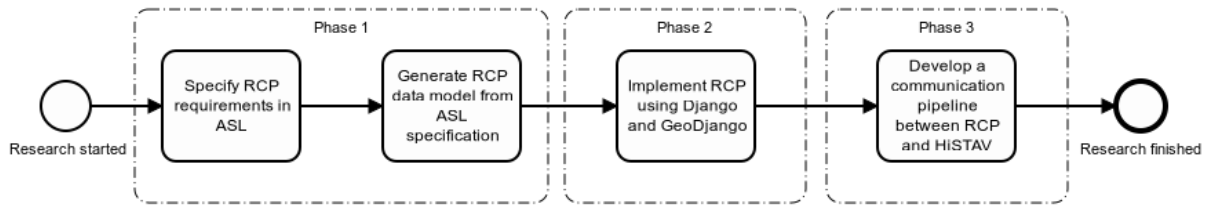


Figure 3. Main phases of the research project.

Following this methodology ensured that the implemented features of the RCP are valuable to the stakeholders and, the correct and trustworthy output of the Simulation Pipeline. Moreover, since some stakeholders have experience with other tools similar to the RCP a comparative evaluation can be performed assuring that the RCP takes the existent technologies one step further, resulting in a more rigorous development and a higher quality and trustworthiness final result.

1.4 Research Scope

Figure 4 shows the architecture of the complete RCP. Since the development of the envisioned RCP would be far too extensive for a MSc thesis it was necessary to focus on specific parts of the whole system. The chosen focus was the creation of the Simulation Pipeline, which is comprised of the integration of HiSTAV with RCP and the creation and definition of a Context which is an input for the HiSTAV simulation.

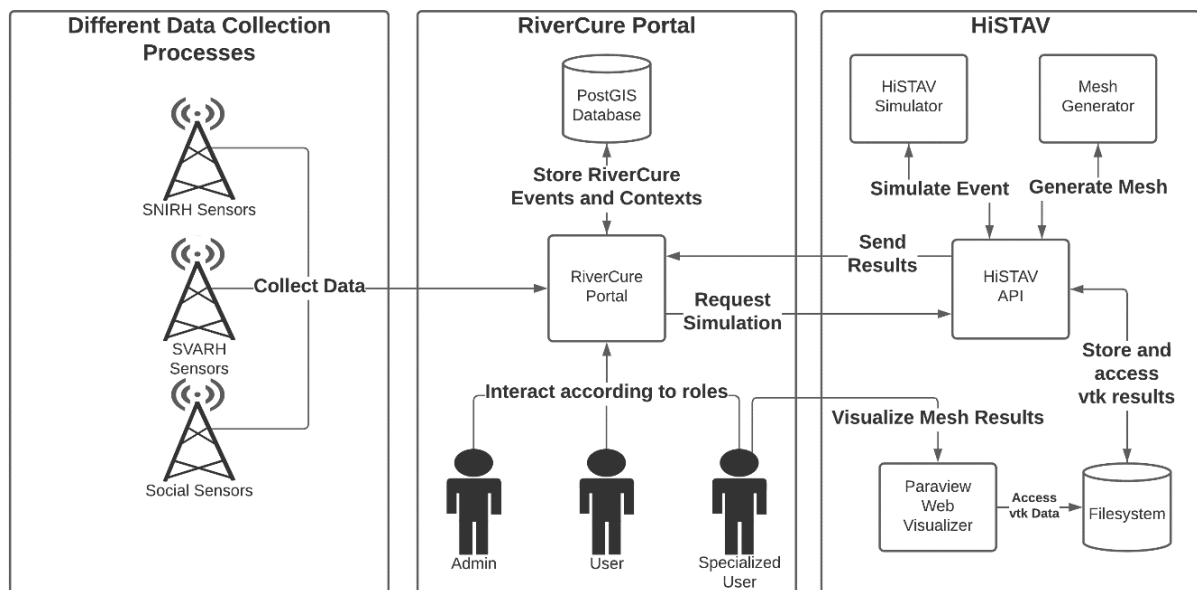


Figure 4. RiverCure Portal and other associated systems.

Regarding the implementation of data collection, only the standardization of sensors and sensor observations, as well as the implementation of features that allow the insertion of new sensors in RCP, both manually and by importing a worksheet, were part of this research scope.

The scope of this research can be summarized as: (i) the standardization of sensors and its observation from the data collection processes; (ii) the implementation of the PostGIS DB and, the implementation of geographic features on RCP as well as the implementation of the necessary interactions users need to leverage these features; (iii) the integration of the HiSTAV system in its whole with RCP;

1.5 Document structure

This document is structured in eight chapters, organized as follows:

Chapter 1 introduces the motivation behind this dissertation and a brief introduction to the RiverCure project.

Chapter 2 gives an overview of all the principles and technologies that were necessary to make this project vision come to fruition.

Chapter 3 goes over the systems that inspired and compose RCP as well as a platform that solve the same problems as RCP, called Mike 21. This is the platform currently in use by *Agência Portuguesa do Ambiente* [20], the main target audience of a tool like RCP.

Chapter 4 details the requirements of RCP, including all its features and functionalities, not limited to this research scope.

Chapter 5 describes the implementation of the RCP described in section 4, limited to this research scope. It provides detailed information on what features were developed and their purpose on the project as a whole. Moreover, it contains a description of the necessary actions to use the developed RCP features, step by step.

Chapter 6 describes the implementation of the communication mechanisms between RCP and HiSTAV. A thorough description of the REST API that enable the communication is given, as well as an overview of the entire Simulation Pipeline in the end.

Chapter 7 evaluates the implementation described in chapter 5 and 6 and, how it holds up against the envisioned system of chapter 5 and the system currently being used described in chapter 3. Contains a simple comparison of the implemented RCP and the system currently used by APA.

Chapter 8 describes the conclusion attained by developing RCP and the future work necessary so it can reach its envisioned state and possibly surpass it.

Appendix A contains RCP ASL specification.

Appendix B contains screenshots of the RCP pages.

2 Background

This chapter contextualizes core concepts necessary to understand how RCP was developed and provides the reasoning behind the main development decisions. It starts with an exploration of some important concepts like Model Driven Engineering (MDE) and Web Engineering.

Second, IFML, RSL and ASL are introduced and contextualized. ASL is a language based on a combination of IFML and RSL, it uses concepts from both and, in order to fully understand ASL it is necessary to have a concept on how these language work and the architecture behind them. Then, an overview of ASL is provided along with an explanation on how it adopts concepts from IFML and RSL. RCP is specified in ASL language and, from this specification, RCP data model was generated. Hence the importance of contextualizing ASL.

Third, the OGC standards, a collection of standards, WaterML and Geographic Information System (GIS) concepts are introduced. OGC standards and WaterML are standards that have a high relevancy on the water management field, while GIS concepts are mainly focused on the available data types necessary for the implementation are introduced. Finally, an exploration of all the core technologies necessary to develop the first version of RCP and HiSTAV “Simulation Pipeline” is provided.

2.1 Model Driven Engineering

Model-Driven Engineering (MDE) is a software development paradigm, at its core is the use of abstract models to describe software aspects while systematically transform these models into more concrete ones up to executable code [21]. MDE raises the abstraction level of languages needed to develop software [21]. It shields software developers from the complexities of underlying implementation platforms [22]. MDE approaches offer potential benefits to software engineering including improved productivity, portability, maintainability and interoperability [23].

A model is specified in some specific model language, known as Domain-specific modeling language (DSML). DSML's formalize the applications structure, behavior, and requirements. They are described using meta-models, which define the relationships among concepts in a domain [24]. Figure 5 the relationship between a model and a system. It is easily seen that by using MDE the business applications developed by companies end up far less confusing and much more understandable. In turn, this makes the applications much easier and cheaper to maintain and, requires a lower knowledge level for its development.

MDE is not a new concept. In the last decades numerous techniques and modelling languages have been proposed to support the design and the development of complex software systems [11]. ASL follows MDE because of its advantages, so it is important to understand MDE in order to comprehend the basis behind ASL.

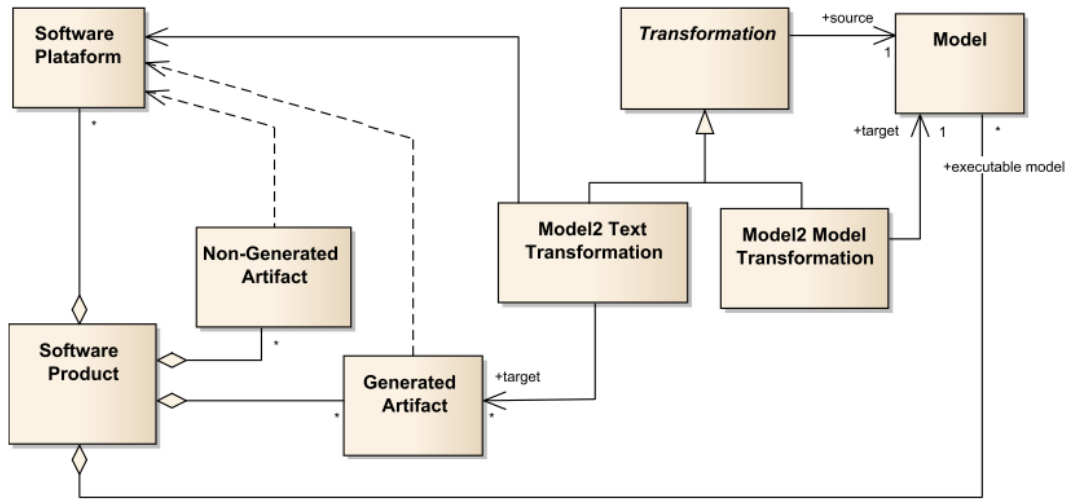


Figure 5. Software product, platforms, transformations, and models (Extracted from [11]).

2.2 Web Engineering

Web Engineering is the application of systematic, disciplined, and quantifiable approaches to development, operation, and maintenance of web-based applications. It is both a pro-active approach and a growing collection of theoretical and empirical research in web application development [25]. Web engineering is also a diverse field that covers extensive topics. These topics are usually related to other disciplines, such as software engineering, hypermedia engineering, human computer interaction, network engineering, project management, web programming, databases and information management [25]. However, there are no commonly and clearly defined core areas in this field, a similar situation with the field of Information Systems [26]. In table 1 it is possible to see the different areas of web engineering.

Table 1. Web Engineering Areas and Topics (Adapted from [26]).

Web Engineering Domain		Example Topics
Core Areas	Issues that directly address the activities in general WIS development and management.	Methodologies, requirements engineering, modeling, web testing, website metrics and quality, maintenance and evolution, project management.
	Development of WIS components	Interface design, usability, information modeling, navigation design, content management, etc.

Closely Related Areas	Development of specific WISs	Web services, semantic web, mobile web, P2P, SoA, intranet, etc.
	Domain-specific development issues	Development of elearning, ecommerce, virtual organization, advertising, collaboration, etc.
	Development technologies and tools	Applications framework, languages, architectures, CASE, etc.
Distant Related Areas	Other web related topics not directly related to WIS development and management activities	Agents, integration, search engine, web mining, development behavior issues, human and cultural aspects, education and training, legal, privacy, security, etc.

2.3 IFML

The standard Interaction Flow Modelling Language (IFML) is designed for expressing the content, user interaction and control behavior of the front-end of software applications [27].

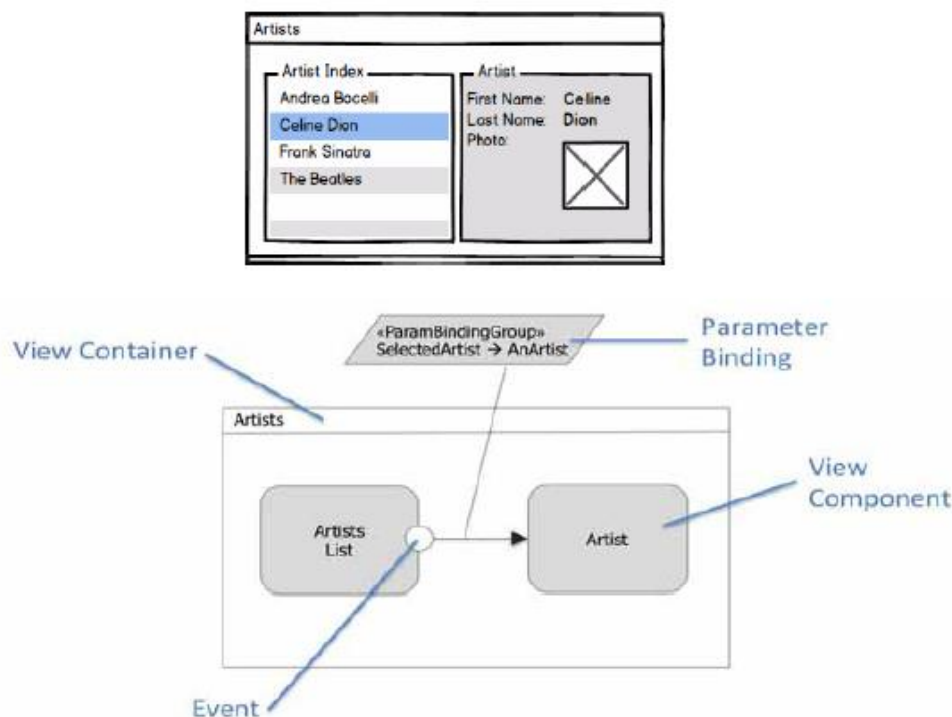


Figure 6. Example of user interface (top) and corresponding IFML model (bottom). The user selects an item in the list and displays its details in the same view container (Extracted from [28])

The objective of IFML is to provide system architects, software engineers, and software developers with tools for the definition of Interaction Flow Models that describe the principal dimensions of an application front-end: the view part of the application, made of view containers and view components; the objects that embody the state of the application, and the references to business logic actions that can be executed; the binding of view components to data objects and events; the control logic that determines the actions to be executed after an event occurrence; and the distribution of control, data, and business logic at the different tiers of the architecture [28]. RCP data model is generated from an ASL specification. ASL uses IFML concepts hence the necessity of a basic understanding of IFML.

2.4 ITLingo

ITLingo is a long term initiative with the objective of researching, developing and applying languages of rigorous specifications [29]. ITLingo approach has the objective of improving technical documentation, and the knowledge extraction and conversion in order to help stakeholders gaining a better understanding of software requirements, which are often misunderstood [30].

ITLingo brings three different languages for three different but close domains: Requirements Engineering, with RSL (Requirements Specification Language); Testing Engineering, with TSL (Testing Specification Language) [31]; and Project Management, with PSL (Projects Specification Language) [32].

Recently, a fourth language, Application Specification Language (ASL) has been developed. This language allows the developer to describe the data model and containers he intends to have in the app. Django [18] code – Python framework for development of web applications – is then generated from this description, originating a functional web application. By developing web apps this way, both the level of expertise required by the developer, as well as the implementation time are decreased.

RSL

RSL is a requirement specification language inserted in the ITLingo Project. It is a process and tool-independent language, meaning it can be used and be adapted by multiple users and organizations with different processes/methodologies and supported by multiple types of software tools. However, in practice, RSL has been implemented with the Xtext framework [33] in an Eclipse-based tool called "ITLingo-Studio" so, its specifications are rigorously defined and can be automatically validated and transformed into multiple representations and formats [32].

RSL provides several constructs logically classified according to two dimensions (see Table 2) abstraction level and specific concerns they address. The abstraction levels are: business, application, software and hardware levels [32]. RSL allows the definition of DataEntities and DataEntitiesCluster, Actors, UseCases and StateMachines. It is important to understand these concepts once that ASL also uses them.

Table 2. Classification of RSL constructs: abstraction levels versus specific concerns (Extracted from [32]).

System (isFinal vs isReusable)	Concerns	Active Structure (Subjects)	Behavior (Actions)	Passive Structure (Objects)	Requirements	Tests	Other	Relations & Sets
	Abstract Levels							
Business		Stakeholder	ActiveElement (Task, Event)	DataEntity	Goal QR	Acceptance CriteriaTest	GlossaryTerm	SystemsRelation
Application		Actor	StateMachine	DataEntityCluster	Constraint FR	UseCaseTest	Risk Vulnerability	Requirements Relation
Software				DataEnumeration	UseCase UserStory	DataEntityTest	Stereotype	TestsRelation SystemElements Relation
Hardware						StateMachine Test	IncludeAll IncludeElement	ActiveFlow
Other								SystemView SystemTheme

DataEntities and **DataEntitiesCluster** represent the structural entities that exist in an information system. More specifically, a DataEntity denotes an individual structural entity that might include the specification of attributes, foreign keys and other check data constraints [32] and a DataEntitiesCluster a cluster of DataEntities and the relations they have between each other..

```

DataEntity e_Customer "Customer": Master [
  isEncrypted
  attribute Id "Customer ID": Integer [isNotNull isUnique]
  attribute Name "Name": String(50) [isNotNull isUnique]
  attribute fiscalID "Fiscal ID": String(12) [isNotNull isUnique]
  attribute phone "Phone #": String(12) [isNotNull isUnique]
  primaryKey(Id)
  check ck_Customer1 "ValidFiscalID(fiscalID)"
  description "Customers"
]

```

Spec. 1. Data Entity example (Extracted from [29])

```

DataEntityCluster ec_Invoice "Invoices (Complex)": Document [
  master e_Invoice
  detail e_InvoiceLine [reference e_Product, e_VAT]
  reference e_Customer
]

```

Spec. 2. DataEntityCluster example (Extracted from [29])

Actors represent the participants of use cases or user stories. Actors represent end-users or other information systems that interact with directly with the system under study [32].

UseCases are a sequence of actions that one or more actors perform in a system to obtain a particular result. However, the RSL's UseCase construct extends this general and vague definition considering some additional aspects [32]. This makes it so that RSL offers a rigorous way to specify use cases, in particular when compared with UML or SysML. Its users can adopt different styles in what concerns use cases specification, depending on their needs and preferences [32].

StateMachines define the behavior of DataEntities in their relationships with use cases. A StateMachine is necessarily assigned to one DataEntity or DataEntityCluster (i.e., a DataEntityGeneric) and classified as Simple or Complex depending on the number of involved states and transitions (e.g., a StateMachine with more than three states might be classified as Complex). A StateMachine includes several states corresponding to the situations that a DataEntity may be find itself during its life cycle (e.g., states like Created, Pending, Approved, Rejected [32]).

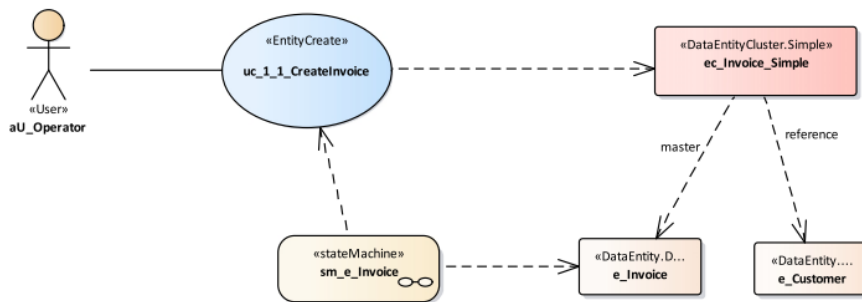


Figure 7. UML diagram suggesting the relationships between RSL's elements (Extracted from [32]).

ASL

ASL [9] was built on top of RSL, (adding a third dimension) that allows the user to describe containers. To understand ASL it is important to first understand RSL. It is the most recent language integrated in ITLingo initiative. Like RSL, ASL has also been implemented with the Xtext framework [33] in an Eclipse-based tool called "ITLingo-Studio". So, its specifications are rigorously defined and can be automatically validated and transformed into multiple representations and formats.

```

UIContainer uiCt_ManageProducts: Window [
    component uiCo_ProductList: List: List_Table [
        isScrollable
        dataBinding e_Product [
            visualizationAttributes e_Product.CategoryID, e_Product.icon,
            e_Product.Name, e_Product.price
            sortAttributes e_Product.CategoryID, e_ProductCategory.Name
            orderBy e_ProductCategory.Name DESC
        ]
        event ev_New "New": Submit: Submit_Create
        event ev_Update "Update": Submit: Submit_Update
        event ev_Delete "Delete": Submit: Submit_Delete
    ]
]
  
```

Spec. 3. Definition of a UIContainer in ASL example

ASL uses concepts from both RSL and IFML which are domain specific languages (DSL). While RSL brings text-based requirements for what business software applications should consist of, IFML brings a visual representation of how the applications should be presented to and manipulated by the final users. ASL keeps RSL concepts like DataEntity, DataEntityCluster, Actors and UseCases, in order to keep the capability of transforming text-based requirements into code. It also introduces some concepts and elements like UIContainers from IFML. These concepts and elements follow the rules to express

them based on the IFML PIM model. With ASL it is then possible to define containers and user interactions with such containers according to the requirements established for the business application in the beginning of the project by the stakeholders.

```
action uiAct_Update_ShoppingCart: NotSpecified [  
    event ev_Ok: Other [  
        navigationFlowTo uiCt_ShoppingCart  
    ]  
]  
action uiAct_Empty_ShoppingCart: NotSpecified [  
    event ev_Ok: Other [  
        navigationFlowTo uiAct_Empty_ShoppingCart  
    ]  
]
```

Spec. 4. Definition of an action in ASL example

ASL can then generate the business application data model in Python code on the Django framework [18]. This gives the possibility to the developer to save enormous amount time when developing a business application while increasing the maintainability of such applications, since the data model is generated according to an ASL specification. Moreover, due to the fact that this specification has a high degree of abstraction, it is possible for software engineering layman stakeholders to contribute and understand this specification. This results in a final product, more faithful to the idealized one.

2.5 OGC Standards – ISO 19125

The Open Geospatial Consortium (OGC) [12] is an international voluntary consensus standards organization, committed to improving access to geospatial, or location information [12]. It provides a consensus process that communities of interest use to solve problems related to the creation, exchange and use of spatial information. OGC standards are technical documents that detail interfaces or encodings [34]. The most relevant of these documents, for this research, is the Simple Feature Access (SFA), also known as ISO 19125. This standard focus on describing the common architecture for simple feature geometry, which is Distributed Computing Platform neutral and uses UML notation [35]. The RCP data model definition follows the SFA standards defined by this organization, in what concerns the geospatial information. Developing the RCP data model based on these standards increases the portability and potential cross-functionality with other systems.

The first step to implement the ISO 19125 was to look at the spatial data types defined by this standard and compare them to the types present in GeoDjango [36], which was the framework used in this project. Since the RCP data model was generated from an ASL specification, it was necessary to define the SFA types in ASL for a correct mapping of geographic data types from ASL to GeoDjango. For that it was necessary to first understand how GeoDjango and SFA data types are related.

Table 3 shows the geometry objects supported by GeoDjango. A juxtaposition for these types with understandable concepts, for a geographic domain layperson, follows. GeometryField is the base class from which all the other geometry fields inherit. In this case only RasterField and GeometryCollectionField are not a geometry field, with the latter being a collection of GeometryFields.

PointField corresponds to a point geometry, LineStringField corresponds to line geometries and PolygonField corresponds to polygon geometries. The types with the prefix Multi denote a collection of the type with the corresponding name [37]. RasterField is equivalent to a GDALRaster, often addressed as raster [38]. By looking at Figure 8, which contains an overview of the geometry object model from ISO 19125, a clear link can be established between SFA standards and GeoDjango spatial field types (e.g., a Point in the geometry object model would be a PointField in GeoDjango).

Table 3. GeoDjango Spatial Datatypes (Based on [37]).

Spatial Field Types	PointField
	LineStringField
	PolygonField
	MultiPointField
	MultiLineStringField
	MultiPolygonField
	GeometryCollectionField
	RasterField

Considering that RCP is developed using GeoDjango with PostGIS, and PostGIS follows OGC standards there is a high degree of certainty that all the spatial types necessary for this project and the RiverCure project are available. Geometry is the root class of the hierarchy, an abstract class [35], which has the operations seen in Figure 9.

Mapping and defining ASL geographic types using SFA, raises the ASL abstraction level, making it possible to develop ASL GIS capabilities independently of the target language and platform that the code will be generated in. If the language or framework we intend to generate the code in (e.g., GeoDjango) also follows these standards, the transformation from ASL to the target language or framework should be linear. Moreover, implementing the RCP data model based on ISO 19125 increases the likelihood of being able to integrate different data streams and introduce new concepts with low effort.

2.6 WaterML

WaterML 2.0 is a technical standard and information model for the representation of water observations data that, by using the existing OGC standards [34], aims at being an interoperable exchange format, allowing the exchange of water observations data sets across information systems [39]. It is divided in 4 parts. Part 1 is about timeseries, Part 2 about ratings gaugings and sections, Part 3 is about surface hydrology features and Part 4 about GroundWaterML. Since RiverCure intends to manage, collect, and distribute data coming from different sources, it is important to standardize data according to WaterML, as interoperability is an important requirement.

Water domain data collection is an important concern in this research. WaterML Part 1 description can be broke down in the representation of hydrological observations data with a specific focus on time series structures [13]. RCP sensors, capture data based on this observation concept (i.e., each data point is an observation). Later, these observations are transformed in a file to be read by the HiSTAV simulation. Observations are defined by Observations and Measurements (O&M) [40] as "...an act associated with a discrete time instant or period through which a number, term or other symbol is assigned to a phenomenon. It involves application of a specified procedure, such as a sensor, instrument, algorithm or process chain. The procedure may be applied in-situ, remotely, or ex-situ with respect to the sampling location. The result of an observation is an estimate of the value of a property of some feature." [13].

2.7 Coordinate Reference Systems

To define a location on the planet, it is necessary to use a coordinate reference system (CRS). With the help of a CRS, every place on the earth can be specified by a set of two or three numbers (2D or 3D), called coordinates. CRS can be divided into 2 distinct systems, **projected coordinate systems**, also called Cartesian or rectangular coordinate systems, and **geographic coordinate systems** [41]. **Geographic coordinate systems** use degrees of latitude and longitude and sometimes also a height value to describe a location on the earth's surface. The locations are determined based on the well-known latitude and longitude lines [41]. **Projected coordinate systems** use two-dimensional CRS which, are commonly defined by two axes. At right angles to each other, these axis, form a XY-plane, with the horizontal axis being normally labelled X, and the vertical axis labelled Y [41]. Since this CRS is 2-dimensional, and the planet Earth is a 3-dimensional "object" that happens to be approximately "round", these CRS need to adapt to the Earth's shape (as seen on Figure 10), in order to correctly locate objects on the map [42].

A CRS has 5 key components [42]: **Coordinate system:** The X, Y grid upon which your data is overlaid and how you define where a point is located in space; **Horizontal and vertical units:** The units used to define the grid along the x, y (and z) axis; **Datum:** A modeled version of the shape of the Earth which defines the origin used to place the coordinate system in space; **Projection Information:** The mathematical equation used to transform locations that are on the surface of the Earth

to a flat surface (e.g., computer screens or a paper map). These approaches are designed to increase the accuracy of the data in terms of length or area.

Individual CRS can be referred to by using Spatial Reference Identifiers (SRID), including EPSG codes defined by the International Association of Oil and Gas Producers (IOGP). The IOGP's EPSG Geodetic Parameter Dataset is a collection of definitions of coordinate reference systems maintained by the Geodesy Subcommittee of the IOGP Geomatics Committee [43]. Each entity is assigned an **EPSG code** between 1024 and 32767, and can be used to access the mathematical approach of each CRS. A well-known CRS is that of EPSG 4326, also known as WSG 84 which uses latitude and longitude and, is the reference system used by the Global Positioning System (GPS).

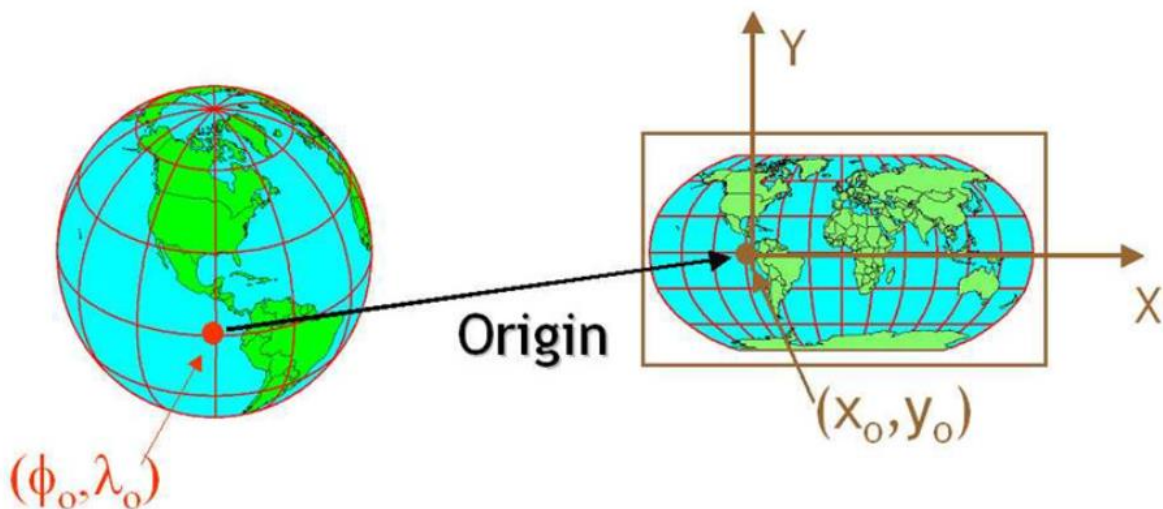


Figure 10. CRS adaptation of a 3D to a 2D representation example (Extracted from [42]).

Since RCP deals with geographic data, choosing the appropriate CRS according to the current situation and, the capability to correctly transform between CRS systems is crucial. Having data from the same location that are stored in different CRS, will result in them not lining up in any GIS or other program unless they are transformed to the same CRS [42].

For this effect, RCP uses 3 different EPSGs. For the base tiles of every map represented in RCP, EPSG 3857 is used, as this EPSG has by far the most available tiles on the web. It is also the most used EPSG for tiles by different web GIS software. However, for the coordinates of all the vertex geometries “drawn” on these maps EPSG 4326 is used. It can be considered an industry standard, when it comes to web applications, to use EPSG 3857 for the map tiles, as it is the easiest to render on the web context and, EPSG 4326 for the coordinates values, as most people think in latitude and longitude when faced with the word coordinates, due to the widespread use of GPS. The used technology for the map representation, leaflet [44] (explored in the next section) is capable of transforming these coordinates to EPSG 3857 on the fly for representation. Finally, for operations related with the HiSTAV simulations EPSG 3762 (EPSG used in Portugal) is the appropriate system, due to the fact that HiSTAV simulations

are done using this EPSG as it was developed primarily for simulations occurring in Portugal. Given that each geographic projection of a country and its spheric reality onto a plane surface generate specific and well-known distance, shape and, area deformations, country needs are different from each other in order to reduce these distortions [45]. Portugal uses EPSG 3762 as it is more adequate for its terrain.

2.8 GIS Data

A web application like RCP is highly dependent on GIS functionalities. RCP needs to handle a high number of GIS data formats. These data types are known as spatially referenced data and can be separated into two categories, vector and raster formats (including imagery) [46]. Additionally, there is a third data format used by RCP, called mesh, which is an input needed to perform HiSTAV simulation.

Vector data is a way to represent real world features within a GIS environment. Things like roads, trees, rivers can be represented by a vector feature. Vector features are shapes represented by geometries such as point, line and polygon [47], with every point coordinate value being dependent on the CRS in use. Looking at SFA standards already gives a good idea of what each feature is, however, they are detailed individually next. A point feature is self-explanatory, it is a point with x and y coordinates, optionally z. In the scope of this project a point can be used to represent, for example, a sensor. A line feature or, more commonly referred to as polyline, is a group of ordered points, connected by lines. Once again, each point has a x, y and optionally z coordinate and, is connected to another point forming what is more commonly known as a line. In RiverCure, a polyline can be used to represent, for example a river.



Figure 11. Geometry representation (green line) of a lake (Extracted from [47]).

Lastly, a polygon feature is, like a polyline, a series of points, with x, y and optionally z coordinates, that are connected with a continuous line but, where the coordinates of the last point are the same of the first point, forming a closed shape. A polygon can be used to define, an area to test for flood possibility, such as the Domain area of a given Context in RCP. On Figure 11, delimited by a green line, a geometry example, more specifically a polyline, is seen.

Moreover, each of these features has associated attributes to describe them. These attributes can be a timeseries in the case of sensors (point), a type in the case of a boundary (polyline) or a cell length (CL) in the case of the Domain (polygon). CL is used to divide the Domain area in smaller cells by HiSTAV for the computation of simulations. These features, besides being stored in a GIS DB, can also be stored in file formats such as geojson [48] and shapefiles [49], which are both used by RCP.

A Raster is a matrix of cells (i.e., pixels) organized into rows and columns (i.e., a grid) where each cell contains a value representing a variable, such as temperature or height [50] (example on Figure 12). Rasters are well suited for representing data that changes continuously across a landscape (surface). A digital terrain model (DTM) is a popular example of a raster representation. DTMs are an important input for both the HiSTAV Simulation pipeline and the user of RCP.

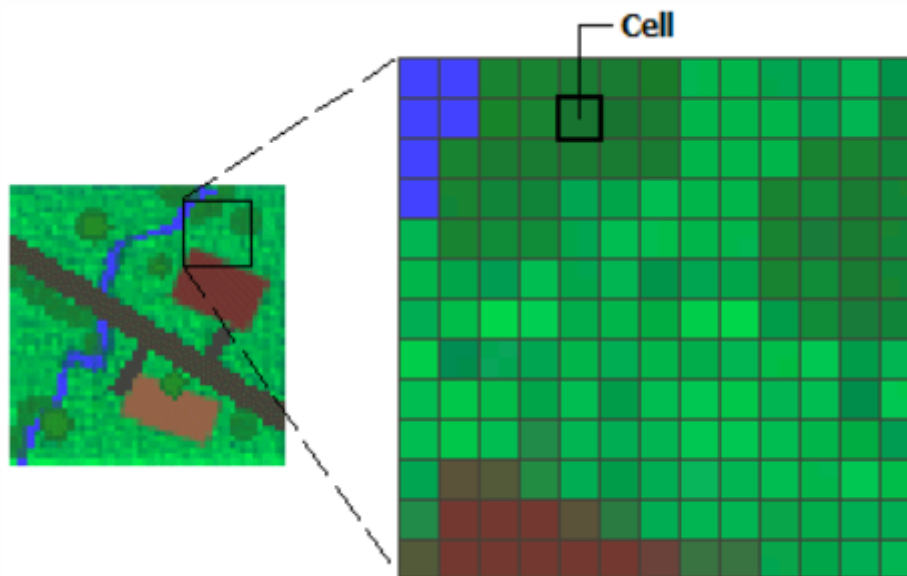


Figure 12. Raster examples (Extracted from [50]).

A Mesh is an irregular triangle network, usually with temporal and spatial components. It provides information about the spatial structure that contains a collection of vertices, edges and faces in 2D or 3D space [51] and, can have datasets that assign a value to every vertex. A vertex is a XY(Z) point, an edge is a line connecting two vertices and, a face is a set of edges forming a closed shape. Each vertex

can store different datasets with or without a temporal dimension [51], containing information like, for example, wind speed through time. HiSTAV needs a mesh as input for the execution of a simulation, and in RCP Context, a program is used to generate a mesh from a combination of vector and raster data, as is explained in section 5.3. A Mesh example can be seen on Figure 13.

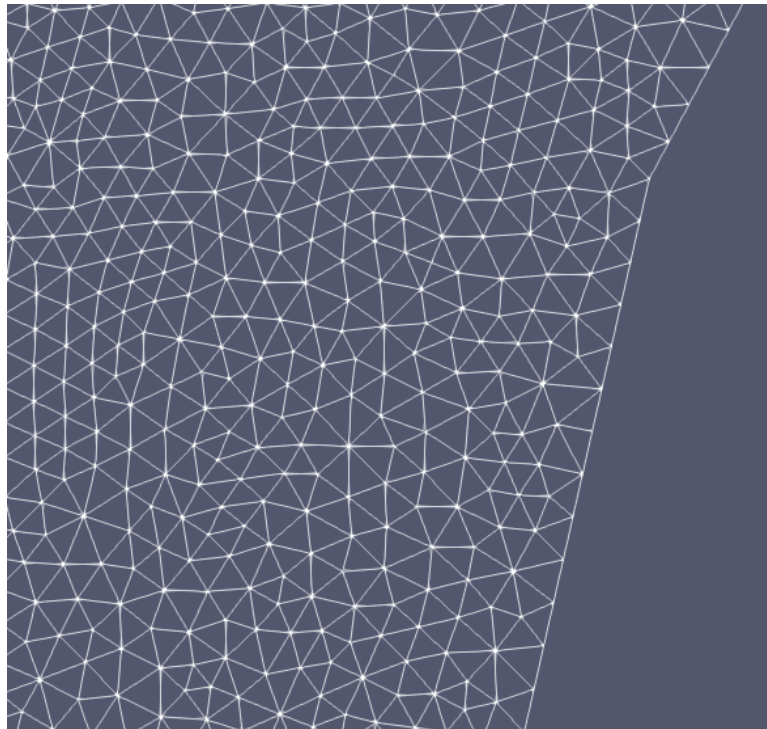


Figure 13. A Mesh example.

2.9 Technologies

To develop a web application like RCP, it is necessary to leverage a substantial number of technologies. These are technologies for geographic operations in the web context. In this section the technologies used in the development of RCP are explained, as well as the reasoning behind their needs.

Python

Python is a general-purpose interpreted programming language used for web development, machine learning, and complex data analysis. Python is a perfect language for beginners as it is easy to learn and understand [52]. Moreover, Python has a vast set of open source standard. RCP requires functionalities like creating geojson files programmatically, as geojson is the standard format chosen for geographic data exchange between RCP and HiSTAV, which are covered by these libraries. Additionally, Python has compatibility with major platforms and paradigms. Since RCP wants to leverage external systems this interoperability attribute might be important in the future. Due to the nature of RCP, these Python qualities are essential to a correct implementation in reasonable time.

However, the main reason Python was chosen was because it is the language used by Django framework [18], a Python web framework capable of developing complex web application like RCP. Moreover, Django has an extension, called GeoDjango, which introduces spatial data types and spatial operations, significantly facilitating the development of RCP. Both Django and GeoDjango are introduced in the next two sections.

Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design [18]. It is focused in developing web application following a Model-View-Controller (MVC) [53] architecture and has 3 main principles [54]: **Don't repeat yourself (DRY)**, **explicit is better than implicit** and, **loosely coupled architecture**.

These principles are advantageous both in the development of RCP and, in ASL code generation. DRY simplifies the development process as the developer does not need to rewrite the same code, and an explicit approach makes the code more readable and easier to transfer to another person. Finally, a loosely coupled architecture contributes to an easier change and expansion process, making it easier for the developers to focus on specific parts of RCP that need to be improved or changed. All these characteristics lead to a more resilient application, with higher maintainability, scalability and code readability. As for ASL, no hidden features, a high degree of segregation of features and simplicity and readability of Django code, makes it easier to design the code generation mechanisms for Django and outline its development process, making ASL the abstract and platform-independent language it ought to be. Additionally, a vast set of data formats, especially GIS formats are contained in a Django extension called GeoDjango [55] (this extension is detailed in the next section) making it easier to develop the equivalent of these types in ASL, turning it in a more complete and holistic language, while keeping its main principles intact, its abstraction and platform independence.

Django inherited Python's "batteries-included" approach and includes out-of-the box support for common tasks in web development [56]: user authentication, templates, routes, views, an admin interface, robust security and, support for multiple database backends. The admin interface is particularly important as it is an automatically generated, user-friendly interface that allows a user to make CRUD operations without writing code. With respect to support for multiple database backends, Django has a database abstraction API [57], known as object relational mapping (ORM) [58], which simplifies DB interaction by allowing the developer to use object oriented programming (OOP) principles to access the DB, instead of using traditional SQL queries (i.e., each table is an object type and each entry an instance of the said object). The existence of ORM is also advantageous for ASL, as it allows the scoping of code generation, in the context of this project, solely for Python language and Django framework. Both the admin interface and ORM are inherited and expanded by the GeoDjango [36], which is explored in more detail in the next section.

As of 2018 Django has been under active development for over 13 years, making it a grizzled veteran in software years, and has a big user base, which further attests to the maturity of the software. Web development is hard. It does not make sense to repeat the same code and mistakes, when a large

community of developers has already solved these problems. At the same time, Django has extensive documentation and support forums while remains under active development and has a yearly release schedule. The Django community is constantly adding new features and security improvements [56].

GeoDjango

GeoDjango [36] is an included contributed module for Django [18] that turns it into a world-class geographic web framework. Since RCP needs to include geographic functionalities and deals with different data formats, GeoDjango functionalities integration is a right decision. Moreover, GeoDjango strives to make it as simple as possible to create geographic web applications like location-based services [55]. Its features include: (i) Django model fields for OGC geometries and raster data; (ii) extensions to Django's ORM for querying and manipulating spatial data; (iii) a loosely coupled, high-level Python interfaces for GIS geometry and raster operations and data manipulation in different formats.

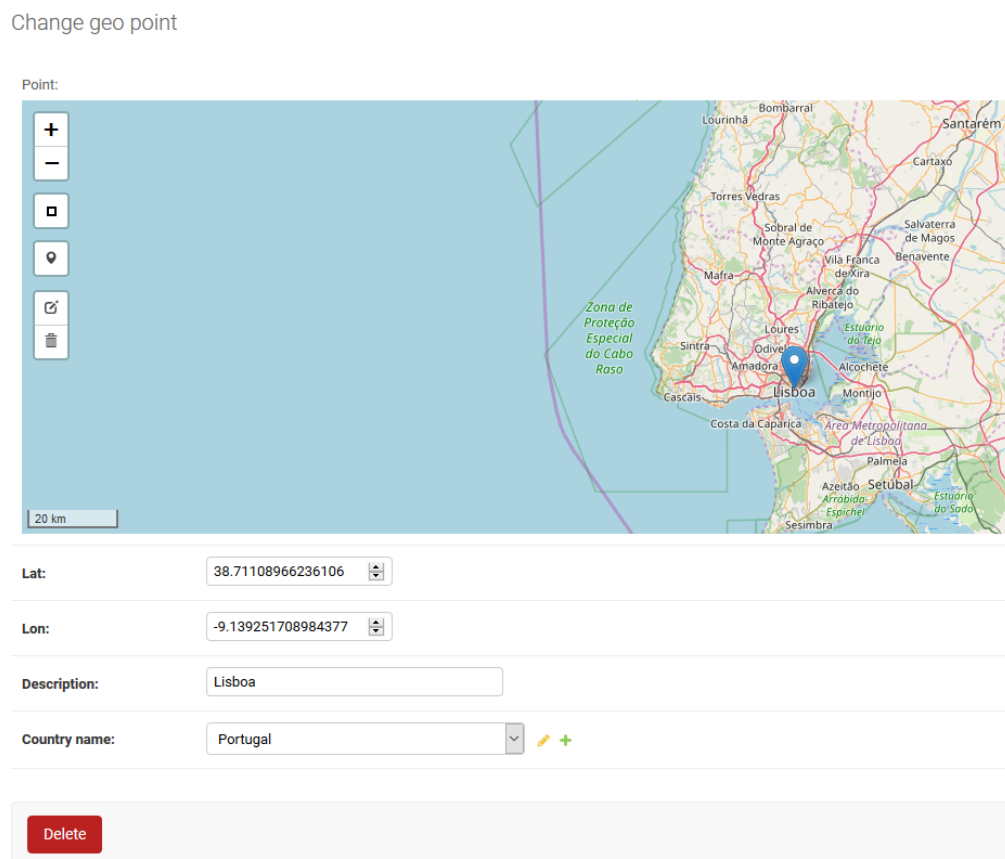


Figure 14. GeoDjango Admin interface view.

The ORM extension adds functionalities from the GIS DBMS chosen as the backend. In this project the chosen backend was PostgreSQL with PostGIS, as it is the DBMS with the highest compatibility with GeoDjango [59]. There are different levels of compatibility, but most GIS DBMS allow geographic operations like intersection of polygons, checking if a geometry is contained inside another, retrieving

the overlapping area of different polygons or transforming between different CRSs. All these operations are performed on the DBMS layer. However, it is necessary to be able to perform these operations on the application layer as well. Either because the data must be treated before being stored on the DB or because the information being calculated is of temporary nature and is not stored on the DB. For this purpose GeoDjango integrates geospatial libraries capable of manipulating spatial data like GDAL [60] and PROJ [61]. The GIS DBMS and geospatial libraries used are explained in the next sections.

GeoDjango also features a powerful built-in admin interface inherited from Django [18], adding geographical types and operations to it. This interface is generated automatically when a developer starts a Django project and allows a user, normally an admin, to perform CRUD operation on the defined models, just like the admin interface inherited from Django. Figure 14 shows an example of the GeoDjango Admin interface where all fields are editable, including a geometry field (i.e., a point in this case).

PostgreSQL and PostGIS

As seen in the last section, the geographic operations available on GeoDjango are highly dependent on the chosen DBMS, as GeoDjango ORM merely bridges the GIS DBMS functions to a Python equivalent, treating each table as an object type. It is necessary to choose a complete DBMS in terms of geographic functionalities to fully leverage this advantage.

PostgreSQL DBMS is a powerful, open source object-relational database system with over 30 years of active development that has earned a strong reputation for reliability, feature robustness, and performance [62]. Moreover, it has an extension, known as PostGIS [63], that has a vast set of geographic operations based on the OGC standards (e.g., CRS transformations, geometries intersection) while having the highest compatibility level, of all DBMS, with GeoDjango as seen on Table 4 [59].

Table 4. Spatial Lookup available in GeoDjango according to the DBS (Extracted from [59]).

Lookup Type	PostGIS	Oracle	MariaDB	MySQL [5]	Spatialite	PGRaster
<u>bbcontains</u>	X		X	X	X	N
<u>bboverlaps</u>	X		X	X	X	N
<u>contained</u>	X		X	X	X	N
<u>contains</u>	X	X	X	X	X	B
<u>contains_properly</u>	X					B
<u>coveredby</u>	X	X			X	B
<u>covers</u>	X	X			X	B
<u>crosses</u>	X		X	X	X	C
<u>disjoint</u>	X	X	X	X	X	B
<u>distance_gt</u>	X	X	X	X	X	N
<u>distance_gte</u>	X	X	X	X	X	N
<u>distance_lt</u>	X	X	X	X	X	N
<u>distance_lte</u>	X	X	X	X	X	N
<u>dwithin</u>	X	X			X	B
<u>equals</u>	X	X	X	X	X	C
<u>exact</u>	X	X	X	X	X	B

intersects	X	X	X	X	X	B
isvalid	X	X		X ($\geq 5.7.5$)	X (LWGEOM)	
overlaps	X	X	X	X	X	B
relate	X	X			X	C
same_as	X	X	X	X	X	B
touches	X	X	X	X	X	B
within	X	X	X	X	X	B
left	X					C
right	X					C
overlaps_left	X					B
overlaps_right	X					B
overlaps_above	X					C
overlaps_below	X					C
strictly_above	X					C
strictly_below	X					C

It is important to use a DBMS that follows some widely used and accepted standards, as this allows the application of these same standards to the environment being developed, increasing its maintainability, flexibility and, interoperability. In this research case, developing ASL with these standards in mind, makes it compatible with any DBMS that also follows these standards like PostgreSQL and PostGIS, creating the desirable platform-independent characteristic.

In sum, PostgreSQL + PostGIS is the DBMS that offers a vast set of spatial lookups operations, database functions and aggregate functions [63] and, a high degree of compatibility with GeoDjango framework [59], when compared to other DBMS used in the software GIS software scope.

Geospatial Libraries

Like explained on the previous section about GeoDjango, in order to be able to perform geographic operations on the application layer, it is necessary to integrate a set of libraries for the effect, namely GDAL/OGR [38], GEOS [64] and PROJ [61]. The importance of each individual library is explained below.

GDAL/OGR stands for Geospatial Data Abstraction Library and is a veritable “Swiss army knife” of GIS data functionality. OGR Simple Features Library is a subset of GDAL, which specializes in reading and writing vector geographic data in a variety of standard formats [38]. It presents a single raster abstract data model and single vector abstract data model to the importing application for all its supported formats [60]. Most relevant data and datasets available use **GDAL/OGR**. So, by using **GDAL/OGR** we ease the handling of such formats (e.g., importing and analyzing files). Importing a water basin border to Rivercure, is an example of a situation where using **GDAL/OGR** allows for a deeper and more curated analysis of a vector file contents, in this case a file containing the water basin borders, it programmatically extracts the shape of the basin as well as any properties associated in the file, allowing the data to be treated, changed and stored according to the requirements. This library, when integrated with GeoDjango, also facilitates the creation of scripts for batch handling this kind of data, allowing, for example, the batch load of several files in one operation.

GEOS (Geometry Engine - Open Source) is a C++ port of the JTS Topology Suite (JTS), an API for modelling and manipulating 2-dimensional linear geometries [65]. It aims to contain the complete functionality of JTS in C++. This includes all the OpenGIS Simple Features for SQL spatial predicate functions and spatial operators, as well as specific JTS enhanced functions [64]. It is owned by OSGeo [66]. Moreover, GeoDjango implements a high-level Python wrapper for the GEOS library, including the following features [67]: (i) a BSD-licensed interface to the GEOS geometry routines, implemented purely in Python using ctypes; (ii) it is loosely-coupled to GeoDjango (e.g., GEOSGeometry objects may be used outside of a Django project/application); (iii) it has mutability: GEOSGeometry objects may be modified; (iv) it is cross-platform and tested (i.e., compatible with Windows, Linux, Solaris, and macOS platforms).

Simply put, this library allows the seamlessly creation of geometries in GeoDjango with simple predicate functions for simple geographical operations (e.g., checking if 2 geometries intersect one another). Thus, RCP geometries have great flexibility and portability, while guaranteeing the strong confidence a library owned by OSGeo [66] provides.

PROJ is a generic coordinate transformation software that transforms geospatial coordinates from one coordinate reference system (CRS) to another [61], based on SRID or EPSG code recognized by OGC. PROJ is a crucial library to enable RCP to transform geometries and rasters between the necessary CRSs. Integrating PROJ in the RCP increases its flexibility when dealing with different spatial references systems, making it a holistic geographic platform.

Leaflet and Django-leaflet

Since RCP is heavy on geographic data functionalities, there is a necessity for a way to visually represent this data in an understandable way for the user. A common solution for this problem is to use a map renderer. In this research the chosen renderer was leaflet [44]. Leaflet is the leading open-source JavaScript library for mobile-friendly interactive maps and has all the mapping features most developers need [44]. Moreover, leaflet also provides high portability between platforms, which is highly desirable, and an extensive and thorough API. This API allows the creation of elegant maps and geometries representations, which can be used to easily analyze or define georeferenced geometries. This is done by using mainly 2 concepts, the base layers and the overlay layers. The base layers are mutually exclusive [68] and, in the context of this project, are usually represented by a world map seen from the top. These layers are usually seen from anywhere inside the renderer as they serve to offer geographic context to the user on the definition of the overlay layers, which are all the other geometries (e.g polygons, points, polylines) defined inside the renderer. The base layer is always underneath the overlay layers which are not mutually exclusive. Overlay layers have a Z parameter which define the order of the layers from a vertical perspective.

Leaflet base layer uses EPSG 3857 and, as such, all the representations rendered by leaflet are done so in the same EPSG for simplicity sake. However, on RCP all the geometries are stored in EPSG 4326 as it is a more commonly used coordinate system, when working with maps, because it is the system

used by the GPS. This means all the geometries visualized on leaflet must be transformed on the fly, in order for the location of the different geometries to be correctly represented on the world map.

Additionally, it was necessary to integrate leaflet with Django [18]. We used a library called Django-leaflet [69] which simplifies this process by making all leaflet functionalities available in Django projects but and adding some functionalities for the powerful Django admin interface, for instance to allow the rendering of geometry fields and spatial data.

Paraview Web Visualizer

ParaView Visualizer is a standalone application that leverages ParaView by Kitware, Inc. [70] capabilities on the backend to produce interactive visualizations over the web [71]. It is able to show interactive visualizations of VTK files. The VTK format is a format associated with Paraview [72], offering a consistent data representation scheme for a variety of dataset types, and to provide a simple method to communicate data between software, especially when dealing with large datasets [73].

The results of a mesh generation and of a HiSTAV simulation are, a single VTK file in the case of the former, and a collection of VTK files in the case of the latter. This makes this file format an important part of the simulation process as is explained later in the RCP (Simulation Pipeline section). The user might need to validate these VTK files and, as such, it was necessary to implement a way to visualize these VTK files in the web, without the necessity for the user to download the file and visualize them in a desktop application. Paraview Web Visualizer serves this purpose by providing a VTK visualizer that can be setup in the web.

3 Related Work

RCP is, in essence, an integration and streamlining of certain features and functionalities of systems in use today. There are 3 main systems that need to be understood, **SNIRH** [4], **SVARH** [5], and **HiSTAV** [2]. The first two serve both as a feature requirement collection and a data collection location. When it comes to features, these systems have functionalities that were emulated in RCP like the visualization of sensors on the map, where the user can collect all the information about the sensors by clicking on them. In terms of data, these systems have a vast network of sensors spread around Portugal that collect hydrometric data. These sensors should be integrated with RCP in the future, allowing the flow of data from these sensors to RCP. As for **HiSTAV**, a system that simulates a Context design on RCP, it is fully integrated with RCP, allowing the full leverage of all its functionalities, in a way that is imperceptible for the user (i.e., the user does not interact with **HiSTAV** directly).

Moreover, to validate RCP and its utility, it is necessary to compare its features with existing tools available in the market. One of such tools is Mike Powered by DHI [74]. Mike offers several suits to solve different problems, but in the scope of this research, the main focus is on Mike 21 [75], as this suit is the most similar to HiSTAV. In this section a detailed description of the aforementioned tools is provided.

3.1 SNIRH

Sistema Nacional de Informação sobre Recursos Hídricos (SNIRH) or Hydric Resources National Information System in English, was announced on October first, 1995 by *Instituto da Água* (INAG) and is a water resources monitorization system, that saves data in APA's SQL-Server DB and releases the information to the public in a web portal (<https://snirh.apambiente.pt>). The portal gets 600 visits per day, between teachers, students, researchers, journalists and public administration personal and is divided in three sub-systems, SNIR-Lit, SNIRH-Júnior and, SVARH [4].

The web portal shares relevant information, like reports and maps of flood areas. However, it is quite old, and the user interaction is poor. RCP emulates the most relevant functionalities of SNIRH, mainly the display and availability of the vast amount of information collected in APA's SQL DB, while significantly improve the usability for the user. RCP then further expands on SNIRH's concept by integrating a hydrometric model known as HiSTAV (explored in section 3.3) creating the ability to perform simulation through interaction with the portal, strengthening the amount of data available on the RCP.

In summary, SNIRH's concept is the base from which RCP was built. Both systems share a great deal of requirements and functionalities. The main differences being the improvement of user interaction, visual appeal and the integration of HiSTAV in RCP. A screenshot of SNIRH web portal is seen on Figure 15.



Figure 15. SNIRH web portal (Extracted from [4]).

3.2 SVARH

Sistema de Vigilância e Alerta de Recursos Hídricos (SVARH) or Hydric Resources Surveillance and Alert System in English is a subsystem of SNIRH that provides the hydraulic state of rivers and reservoirs (i.e., water levels, flow rate, stored volumes and water quality) and relevant meteorological information in real time, while also allowing the prediction of its possible evolution. It is composed by a network of stations with autonomous transmissions, that measure several hydrological variables and water quality, and by a technology infrastructure for storage and dissemination of the data collected by the stations.

This network is composed by three types of automated stations 311 hydrometric and 620 meteorological, totaling approximately 931 stations [76]. The data collected by each station is related with the station type (e.g., meteorological measures water level, wind speed and direction, air temperature, relative humidity, and water temperature). A station is composed by sensors, a datalogger, a power supply (battery and solar panel), and a communication system (usually GSM modem) [77]. The flow of the collected data between these components is seen on Figure 16. These stations are located in critical points for surveillance of floods, droughts and pollution accidents [5]. The whole system is divided in three parts.

- **Data Acquisition** – Automatic stations with transmission.
- **Central Processing** – Data gathering informatic system from the automatic stations, and its storage with hydric and hydraulic models.
- **Information Distribution** – Real-time information distribution software from the automatic stations.

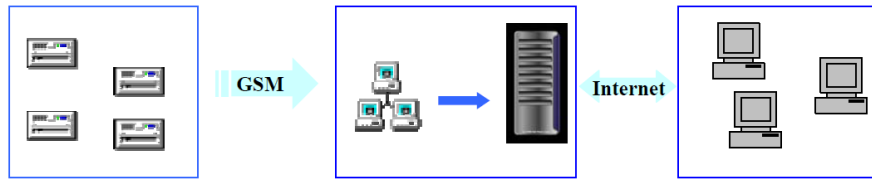


Figure 16. From left to right: Data Acquisition, Central Processing and Information Distribution (Extracted from [5]).

The main relevance of this system in RCP context is the possibility to leverage the data distribution of SVARH and funnel the data to RCP. The higher the amount of quality data available on RCP, the higher the amount of data that can be fed to HiSTAV for simulations, resulting in a higher quality and trustworthy simulation. More data should result in more geographic areas with available data that can be used by HiSTAV simulations. This makes it possible to define distinct Contexts for these areas, increasing the number of areas being simulated by HiSTAV. Hence, RCP's stakeholders would have more precise and trustworthiness results, as well as more distinct Contexts available on RCP (i.e., more area is covered/protected by RCP).

3.3 HiSTAV

HiSTAV is a modelling effort aimed at delimiting the critical tsunami inundation areas in an urban waterfront and to quantify the associated severity [6]. Concretely, it is a reliable and performant tool for faster than real time (FTRT) high-resolution simulations, leveraging parallel, distributed and graphics-based computing technologies for this effect. Moreover, HiSTAV was released as a standalone product, available to both engineering and research communities, by supporting a close integration with open-source Geographic Information Systems and scientific visualization toolkits for complex data handling and analysis. Its applicability remains valid for multiple fields in water resources [2].

For its simulation, HiSTAV needs a collection of geometries and rasters known as Context as input. This Context is composed by 5 geometries known as Domain, Refinement, Alignment, Boundaries and, Boundary Points, and 2 raster known as Digital Terrain Model (DTM) and Friction Coefficient. These are all explained in section 5, but for now it is important to understand that all these components are georeferenced and have associated properties. They are necessary for a correct execution of a HiSTAV simulation.

RCP integrates this tool by implementing features that allow a user to define or upload the Context described above, as well as its corresponding properties, and establishing a pipeline between itself and HiSTAV enabling bidirectional data exchange. This pipeline allows RCP users to execute simulations based on defined "Contexts" and visualize the results.

3.4 Mike 21

Mike 21 is a versatile desktop tool for 2D coastal and sea modelling. It is capable of simulating physical, chemical, and biological processes in these contexts. The hydrodynamic model in the MIKE 21 Flow Model is a general numerical modelling system for the simulation of water levels and flows in estuaries, bays and coastal areas. It simulates unsteady two-dimensional flows in one layer (vertically homogeneous) fluids and has been applied in a large number of studies [78]. It is capable of considering different flood factors like **Boundary Conditions**, **Flooding and Drying**, **Infiltration and Leakage**, and **Multi-Cell Overland Solver**. Additionally, it contains a **Flood Screening Tool** [78]. It has a proven track record as it has been used for many coastal and marine engineering projects around the world [75]. Some of its benefits and typical applications, according to its developer, are [75]: Benefits: (i) proven technology and more than a 25-year track record of successful applications; (ii) offers maximum flexibility, higher productivity, and full confidence in the results; (iii) is modular; (iv) it comes with a wealth of first-class tools that enhance and ease modelling possibilities; In terms of typical applications: (i) design of data assessment for coastal and offshore structures; (ii) optimisation of port layouts and coastal protection measures; (iii) cooling water, desalination, and recirculation analysis; (iv) optimisation of coastal outfalls; (v) environmental impact assessment of marine infrastructures; (vi) ecological modelling including optimisation of aquaculture systems; (vii) optimisation of renewable energy systems; (viii) water forecast for safe marine operations and navigation; (ix) coastal flooding and storm surge warnings; (x) inland flooding and overland flow modelling;

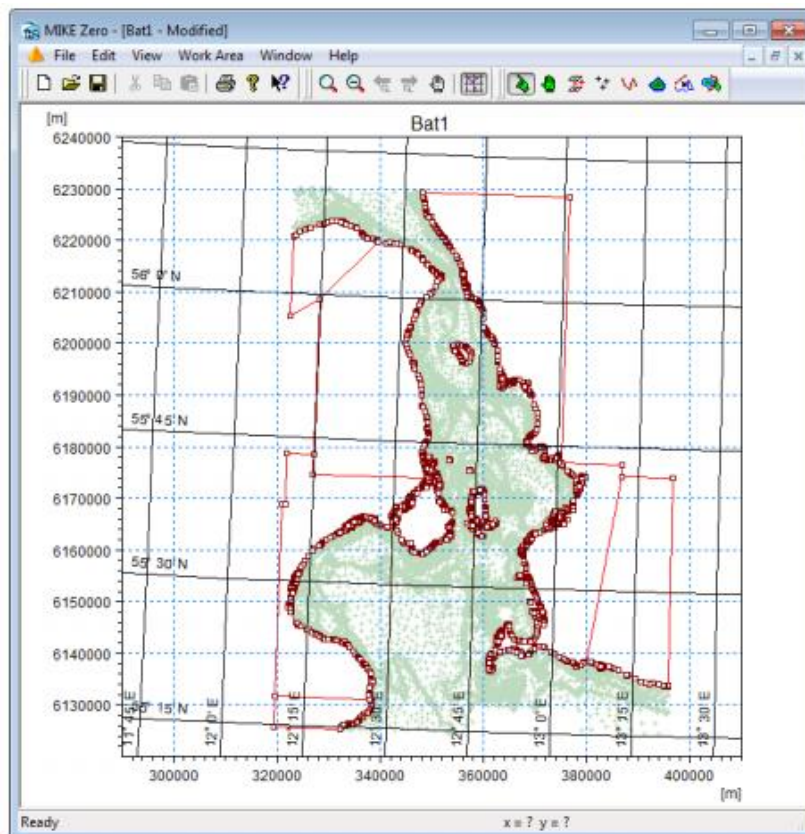


Figure 17. Mike 21 working area example, after import of land and water data (Extracted from [79]).

As for the user interaction, to correctly use Mike 21, the user needs a background in coastal hydraulics and oceanography, which is sufficient to check whether the results are reasonable or not [80]. The general steps taken to simulate an area in Mike 21 are [79]: (i) the bathymetry needs to be setup by importing all the necessary geographical data with soundings based on a survey or digitized from nautical chart; (ii) the user needs to create the boundary conditions by setting the water levels at the boundaries; (iii) define a data set with all the values to simulate (e.g., water level through time); Figure 17 shows an example of a Mike 21 working area.

There are several similarities between Mike 21 and RCP, both in user qualifications, general guidelines to simulate an area and, business goals. Chapter 6 provides a detailed comparison between the two tools.

4 RiverCure Portal Requirements

This research only covers a part of the entire RCP system. To have a better understanding of what was developed in this research and its purpose, it is necessary to have a better idea of the entire RCP system. For this effect, the RCP requirements are detailed in this chapter followed by a description of what part of RCP was implemented in the scope of this research.

The RCP intends to integrate SNIRH and SVARH functionalities with HiSTAV itself in one web application [81]. As such, there are four main points that need explaining. First, the RCP interface, i.e., which interactions should be available to the user and the general structure of the data visualization. Second, the definition of the Context concept, which is the main focus of this research and is the input for HiSTAV simulations. Third, how data is collected and funneled to RCP, as well as what RCP does with it. Fourth, how RCP should integrate HiSTAV into its workflow, and what are the data flows between these two systems. Additionally, a last section is provided detailing which components were developed within this research project.

4.1 RiverCure Portal Interface

RCP interface is inspired on SNIRH [4] interface. However, it intends to significantly improve the user interaction as well as the overall appearance of the site. For this effect, RCP shall contain less navigation buttons and the information available shall be more selective and curated. RCP also relies on the segregation of responsibilities between users according to their specific roles (e.g., a visitor will not have access to the HiSTAV simulation request but, a Context Admin will). In terms of user interaction, this segregation is leveraged by providing a custom interface for each user role (i.e., each navigation button is only available to a user which role can use the associated functionality).

The information and functionalities that should be available in RCP are the following: (i) information about all the created “Contexts”, and a set of actions to create such “Contexts”; (ii) information about all the sensors associated with RCP system, and functionalities to associate new sensors. The sensor information should contain the details of the sensor itself (e.g., code, location, type) as well as the data being collected, in the form of observations; (iii) information about past, present and future flood events, as well as a way to create such events; (iv) full integration with HiSTAV, allowing the user to use its functionalities without interacting with it directly.

RCP information and functionalities are geographically heavy. As such, there is a particular interface element that requires special attention. This element is the map, which allows the user to geographically contextualize spatial information being defined or available on RCP. Figure 18 left image shows an example of SNIRH interface. RCP should contain a similar view. Each blue circle corresponds to a hydrometric station. The letter inside each circle represents the type of the station (e.g., hydrometric, weather station). Clicking on a circle results in a popup containing information about the data being collected by the corresponding sensor, in a defined time interval, as seen on the right of Figure 18. The RCP sensors representation shall follow a similar principle.

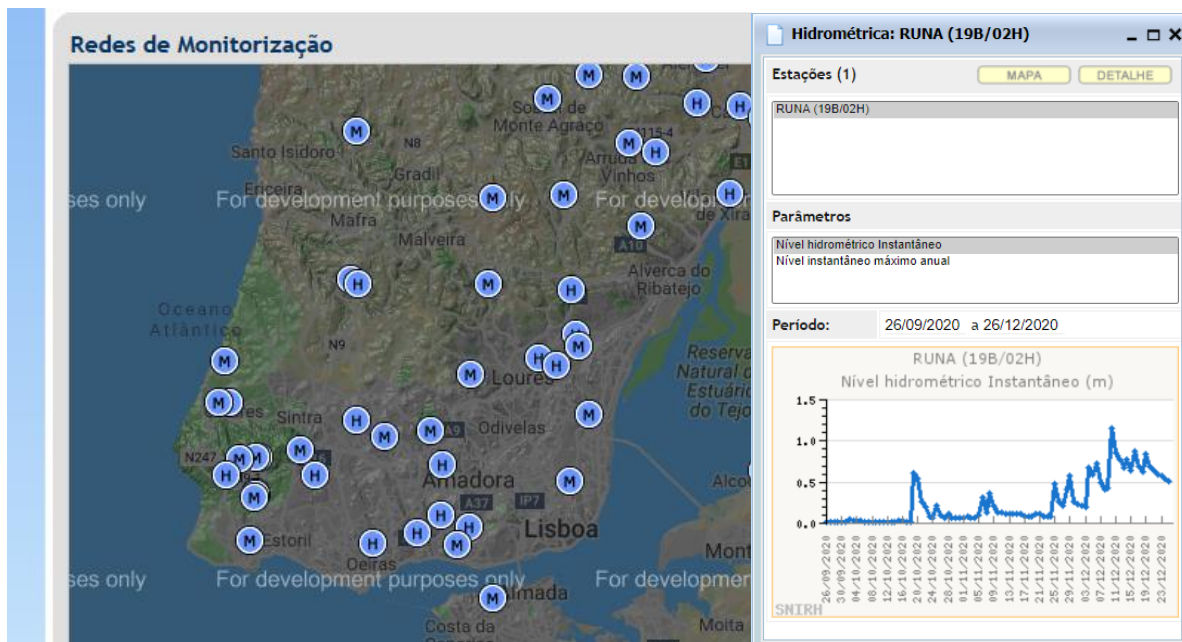


Figure 18. SNIRH system monitorization network (left) and sensor popup (right) (Extracted from [4]).

4.2 RiverCure Portal Context

An authorized user should be able to define a Context. This Context is the input for a HiSTAV simulation. Before understanding how to define a Context it must be first understood what a Context is. The Context concept is composed of several features, which are divided into 2 different groups: **Context Geometries** and **Context Attributes**. **Context Geometries** are composed of distinct geometries named: Domain, Refinement, Alignment, Boundary and Boundary Point. On the other hand, **Context Attributes** are: DTM Raster, Context Sensor, Hydro Feature, and Context Event. Spec. 5 shows the rigorous Context specification in the ASL language.

All these concepts are further explained in this section. The division of these features into 2 groups is due to purpose and structural differences between them, which, come up when preparing the data for a HiSTAV simulation. The first 5 concepts, under the geometries section, are geometries that, have a visual representation, and need to be defined or imported by the user on RCP. These concepts are known by HiSTAV as Context files and, are the core input for its simulation. The other 4 are either defined through associations of concepts (e.g., associating a sensor to a Context, in the case of the Context Sensor, or associating an Event to a Context, in the case of Context Event), are an imported file (DTM Raster) or were defined previously and are transversal to the entire system (Hydro Features). Except for the DTM Raster, whose file representation is needed by HiSTAV, all the other three concepts are not used directly by HiSTAV and are instead used by RCP to configure a simulation request.

```

DataEntity e_Context "Context": Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute code "Code": String [constraints (NotNull Unique)]
  attribute Name "Name": String [constraints (NotNull Unique)]
  attribute hydroFeature "HydroFeature": Integer [constraints (NotNull
ForeignKey(e_HydroFeature))]

  attribute geomExternalBoundary "Domain" GeoPolygon
  attribute CLExternalBoundary "Domain CL": Double
  attribute geomRefinement "Refinement": GeoPolygon [constraints (multiplicity"1..*")]
  attribute CLRefinement "Refinement CL": Double [constraints (multiplicity"1..*")]
  attribute geomAlignment "Alignment": GeoPolyline [constraints (multiplicity"0..*")]
  attribute CLAlignment "Alignment CL": Double [constraints (multiplicity"1..*")]
  attribute userResponsible "User Responsible": Integer [constraints (NotNull
ForeignKey(e_User))]

  attribute isPublic "Context Access Restrictions": Boolean [defaultValue "False"
constraints (NotNull)]

  tag (name "tenant-main" value "true")
]

```

Spec. 5. Context ASL specification

Context Geometries

The definition of these geometries is the bulk of the work related to the definition of a Context. These geometries represent the visual part of a Context. When the user is visualizing a Context what the user is seeing is these 5 geometries definition. The DTM also has a visual representation within a Context, however, since it is not a geometry, but a raster, it cannot be defined by the user, only uploaded in a raster file format form, and is treated differently by HiSTAV, it is under a different section as it is not considered primary in a Context definition.

In a structural sense, these geometries are similar because they follow the same design principles, as they are all geometries and can be defined by the user by drawing on a map. These geometries are stored in a PostGIS DB [63] and have a corresponding geojson file [48] representation, which is not stored but generated on demand.

Lastly, the definition of these geometries is subject to 2 geographic constraints: (i) the Domain geometry must contain, inside its boundaries all the other geometries; (ii) the boundaries of these geometries cannot intersect each other under any circumstance.

1. Domain

The layman definition of Domain is that of a general area which is under analysis for the potential risk of flood. In most cases, this is the first geometry defined by the specialized user. The refinement and alignment geometries must be contained inside this geometry while the other 2 geometries, boundary and boundary points, must be **on** the boundary line. An example can be seen on Figure 19, where the Domain is defined by a grey polygon.

A Domain geometry is stored in the DB as a **Polygon**. Each Context can only have 1 Domain that is defined by a single Polygon. For the geojson file representation, the entire Domain geometry is

converted into a geojson feature that contains a polygon geometry definition. This geometry only has 1 property, cell length (CL), which is a Float, and is defined by the user, after defining the Domain Polygon.

In summary: **Geometry Type:** Polygon; **Properties:** CL (Float); **Multiplicity:** 1;

2. Refinement

A Refinement geometry is a Polygon. It must be contained inside the Domain geometry and, it is usually defined around the area of interest of a river, resulting in said river being contained inside the polygon. However, rivers can have tributaries, or other features, requiring more than 1 polygon to be represented as a correct pre-processor geometry and input. Thus, unlike the Domain, for a given Context there can be several refinement definitions, optionally nested inside each other, each defined by a polygon with individual property values. Therefore, a Context, can have several refinements, stored as **Polygons** in the DB.

The only property of this geometry is also named CL and, in case there are several refinements defined for the same Context, each refinement has its own CL value. For the geojson representation, each refinement is converted in a feature containing a polygon geometry and corresponding CL value (Float) as a property.

In summary: **Geometry Type:** Polygon; **Properties:** CL (Float); **Multiplicity:** 1..* (As long as they are inside the Domain definition and their boundaries don't intersect any other geometry boundaries);

3. Alignment

An Alignment is defined by a LineString (GeoPolyline in ASL). It must be contained inside the Domain and, it usually outlines a river. In a way, it can be seen as drawing the river on the map as a LineString. Like explained in the refinement section, it might be necessary to define several alignments for a Context due to the existence of tributaries and other features of a river. It is possible to define several alignments for a given Context, as long as these alignments do not intersect each other. However, contrary to the refinements, whose definition of at least 1 is mandatory, the definition of alignments is not, as long as the refinements have a sufficient level of detail to form the Mesh grid. Like the geometries described before, the alignments are also stored in a PostGIS database, but in this case, they are stored as LineStrings.

The only property of the alignment geometries is CL, defined by the user after defining the geometry points. Alignments follow the same principles of the Domain and Refinements when being converted to geojson. This means each geometry is converted to a feature, and each feature has a LineString geometry and, 1 property named CL (Float), which is defined by the user for each individual LineString.

In summary: **Geometry Type:** LineString; **Properties:** CL (Float); **Multiplicity:** 0..* (As long as they are inside the Domain definition and their boundaries don't intersect any other geometry boundaries);

4. Boundary

A Boundary is defined by a LineString. Its definition must be overlaid on the Domain boundaries and all its points must be points from the Domain definition. Consequently, it is defined by joining at least 2 sequential points from the Domain. If the points are not sequential, the simulation will not be performed

correctly. In almost all cases a Context will have more than 1 Boundary, the user can therefore define several Boundaries for a Context as long as they don't overlap or share any points. Since a Boundary is defined by connecting different sequential points of the Domain, not sharing any Domain points is a good enough safeguard against intersection when defining more than 1 Boundary for one Context.

Each Boundary is stored in a PostGIS DB as a LineString along with 2 properties, the Type and the Data Type. Both properties are multiple choice fields and are defined by the user after defining the boundaries. The Type can be either Input, Output or InputOutput. Regarding the Data Type the choices are H for depth, Q for discharge, Z for elevation and V for velocity. When converting this geometry to geojson, once again, each geometry is transformed to a feature, and each feature has its LineString geometry and properties with corresponding values.

In summary: **Geometry Type:** LineString; **Properties:** Type (Input, Output, InputOutput), Data Type (H – Depth, Q – Discharge, Z – Elevation, V – Velocity); **Multiplicity:** $1 \cdot \frac{nr\ domain\ points}{2}$ (rounded down);

5. Boundary Point

A Boundary Point is a point from a Boundary. These points are not defined by the user directly but instead automatically created when the user defines a Boundary, 1 Boundary Point for each point of the Boundary. Since this is an automatically generated geometry, all the possible constraints are automatically addressed.

A Boundary Point only has 1 property named series, which can be defined multiple times. This property represents an association of a boundary point to a sensor, which becomes a Context Sensor after the fact (Context Sensor concept is explained next, in the Context Attributes section). The association is done by selecting a sensor, which is within a defined distance to the Boundary, from a boundary point popup interface. The distance for possible sensors association is also defined in this popup interface.

The reason for the choice of the property name series is that what is important to HiSTAV is the data series of the sensor and not the sensor itself. In the end, what is really associated with the boundary point is a file with the data of said sensor in a given time interval. This file is a text file with the name of the corresponding sensor as its name and a bnd extension. For now, it is enough for the reader to know that every Boundary Point must have at least 1 sensor associated, and that it is possible for a Boundary Point to have more than 1 sensor associated, resulting in several series properties, the rest is explained in the HiSTAV Integration section. The sensor will also be automatically associated to the Context when associated with a Boundary Point (this will become relevant later).

The defined geometries are stored in a PostGIS DB, and their associated sensors are also stored in said database but on their own table. There is a table called e_ContextSensor dedicated to storing the association of a Boundary Point to a sensor.

This geometry follows the principles of the previous geometries when converting to geojson. This means each geometry is transformed into a geojson feature with its own properties, which, in this case, is called series and corresponds to the name of the bnd file with the data from the associated sensors.

In summary: **Geometry Type:** Point; **Properties:** Series - Sensor bnd file; **Multiplicity:** 2..sum of all points of the defined boundaries;

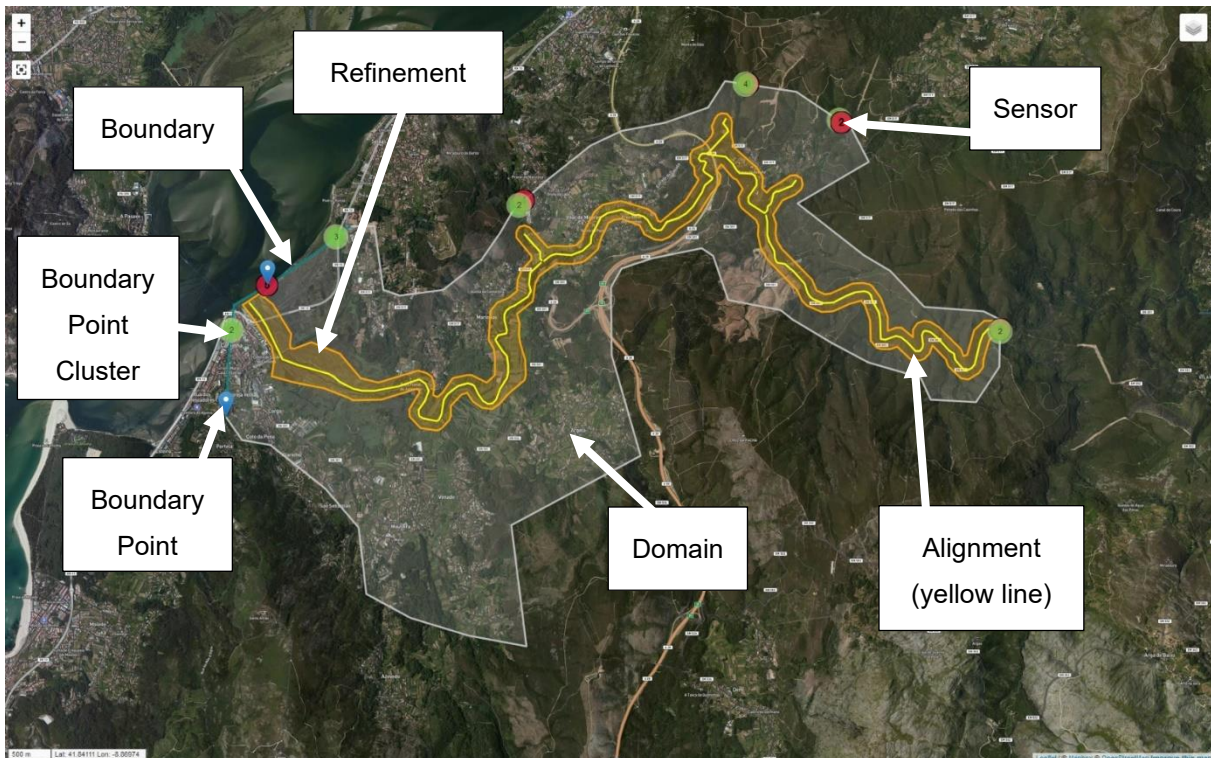


Figure 19. Context Visualization on RCP.

Context Attributes

In this section the Context attributes are explained. The name attributes was chosen to distinguish these Context properties from the Context geometries. While the Context geometries are structurally similar, and, are defined by the user drawing geometries on a map, these attributes are more distinct between themselves. Not all of them are essential (as the reader will see later) and they are defined in a more linear way by the user (e.g., filling a form).

Moreover, while the Context geometries can be considered the main components of a Context, the attributes are more akin to support data, even being directly related with the Boundary Points in the case of the Context Sensor, as it is explained later. For simplicity sake, let us consider the geometries as the Core properties of a Context, with the attributes being the secondary properties.

1. DTM Raster

The digital terrain model (DTM) is a topographic model of the bare Earth that can be manipulated by computer programs. The dataset contain the elevation data of the terrain in a digital format which relates to a rectangular grid [82].

A DTM is very useful visual aid for a specialized user when defining a Context. As such, it is important for such user to be able to add a DTM and have a representation of it on the same map where he is supposed to draw the Context.

Moreover, the DTM is used by the Pre-processor to generate a mesh, that is used by the HiSTAV simulator, making the capability to associate a DTM to a Context essential. The mesh generation and its use by the Pre-processor is further explained in the HiSTAV integration section. Adding a DTM is, by consequence, mandatory before proceeding with the HiSTAV Simulation. On Figure 20 the reader can see a DTM where a whiter color corresponds to higher altitudes and darker to lower altitudes.



Figure 20. DTM example (as seen on RCP).

In summary: **Geometry Type:** Raster; **Properties:** None; **Multiplicity:** 1; **Purpose:** Visual aid for the user during the definition of a Context, used by the Pre-processor for mesh generation, which is an input for the HiSTAV simulator.

2. Context Sensors

In theory, a Context Sensor is a Sensor that is associated with a Context. This association is not done directly, instead it is done by associating a Sensor to a Boundary Point. The Sensor is then automatically considered as a Context Sensor. The Sensor itself, can be described as a point on the map, that collect data through observations in the form of a timeseries. These procedure of collecting data is the adaptation from the WaterML standards [13].

A Context Sensor, in practice, is just a concept representing a link between a Boundary Point and a Sensor, keeping track of which sensors are associated to which boundary points. Keep in mind that a Boundary Point can have more than 1 Sensor associated and 1 Sensor can be associated to more than 1 Boundary Point. A Boundary Point cannot be however associated with the same Sensor more than once. This link is relevant so that it is possible to associate the correct series to a Boundary Point. The series property of Boundary Point is explained in detail in the previous section Context Geometries under Boundary Point section.

In summary: **Geometry Type:** None; **Properties:** Boundary Point, Sensor; **Multiplicity:** 1..*; **Purpose:** Linking a Boundary Point to a Sensor;

3. HydroFeature

A HydroFeature is a type of terrain related with water (e.g., a river). At the time of writing the defined HydroFeatures are: River, Estuary, Lake, River Basin, Drainage Basin and Dam. There is the possibility of extending these types as necessary in the future.

Like the Context Geometries, a HydroFeature also has a georeferenced geometric definition. However, in this case, this definition is not done by the user, it is instead loaded from a shapefile [49] through a script developed specifically for this purpose. Moreover, it is not visible to the specialized user, who will define the Context and it is not used by the HiSTAV Simulation. The purpose of a HydroFeature is solely for the admin who creates the Context to have a way to specify for the specialized user defining the Context, where it should be defined and what HydroFeature is supposed to be analyzed later in the HiSTAV Simulation.

In summary: **Geometry Type:** Polygon; **Properties:** Type (River, Estuary, Lake, River Basin, Drainage Basin, Dam) with possibility for the addition of more types; **Multiplicity:** 0..1;

4. Context Event

A Context Event is a time bounded Event, created by an authorized user and associated with a specific Context. A given Context can have as many Events as necessary. It is through a Context Event creation that a simulation is requested for the said Context. A Context Event has several properties that can be divided in 2 groups, with all properties being defined at the time of the Context Event creation.

The first group are properties necessary for the HitStav simulation request. These properties will influence the outcome of a simulation and the user that creates the Context Event must be very considerate when defining its values. These properties include: (i) the Context, the only property defined automatically, since the user is creating a Context Event and not an Event; (ii) start and end date, and time interval for which the HiSTAV Simulation should be performed; (iii) writing periodicity, which represents the frequency the user wants HiSTAV to write the results of the simulation; (iv) and finally the update maximum values periodicity, which represents how frequently the user wants HiSTAV to update its maximum values. These values have a high degree of complexity and as such must be defined by a specialized user with knowledge of HiSTAV Simulator which has the corresponding authorization given by an RCP admin.

The second group of properties aim to provide a precise classification of the Context Event. Properties like name, description, type, subtype, and state are all present in a Context Event even though they are not used by the HiSTAV simulation. However, these properties are of the utmost importance when analyzing a complete event and its results in retrospective. A pundit user can quickly understand the reasoning behind the creation of a certain Context Event by looking at these properties, as well as interpret the results from the associated HiSTAV simulation.

In summary: **Geometry Type:** None; **Properties:** For HiSTAV Simulation: Context, Start Date and Time (2 fields, 1 for date and 1 for time); End Data and Time (2 fields, 1 for date and 1 for time); Writing Periodicity (uses hz for computation but the user can define it in hours, minutes, or seconds); Update Maximum Values Periodicity (uses hz for computation but the user can define it in hours, minutes, or seconds); For descriptive purposes: Name (String); Description (String); Type (Flood, Heavy Precipitation, Hydrological Drought, Meteorological Drought, Hurricane, Tsunami, Storm, Landslide); SubType (Forecast, Hindcast, Planning); State (Announced, Occurring, Concluded); **Multiplicity:** 0..*; **Purpose:** Starting a simulation bounded by the time frame provided by the user who created the event;

The Context definition is the first step in the long simulation pipeline which will finish with the HiSTAV simulation and its output, which allows a specialized user to infer if the area from the Context under simulation is in risk of being flooded.

The Simulation Pipeline steps are as follows: (i) Context creation by an authorized user and attribution of a Context owner. (ii) Context definition by the Context owner; (iii) pre-processing request by the Context owner; (iv) an authorized user creates a Context simulation event.

4.3 Data Collection

The RCP result trustworthiness is dependent on the quality and quantity of data available on it. RCP intends to leverage SNIRH and SVARH sensors and use them as its own sensors. The RCP users should be able to associate these sensors to their created Contexts.

```

DataEntity e_Sensor "Sensor": Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute code "code": String [constraints (NotNull Unique)]
  attribute Name "name": String [constraints (NotNull)]
  attribute type "Type": DataEnumeration SensorKind [constraints (NotNull)]
  attribute modalityType "Modality Type": DataEnumeration SensorModalityKind
[constraints (NotNull)]

  attribute Description "Description": Text
  attribute Version "Version": String
  attribute responsible "User Responsible": Integer [constraints (NotNull
ForeignKey(e_User))]

  attribute geom "Geometry": GeoPoint
]

```

Spec. 6. Sensor ASL specification.

In the future, it should be possible to add new sensors to RCP from any source as long as they follow the OGC standards [35]. Spec. 6 and 7 show the Sensor and the Context Sensor ASL specifications respectively. These two specifications together form the RCP Sensor concept.

```
DataEntity e_ContextSensor "ContextSensor": Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute context "Context": Integer [constraints (NotNull ForeignKey(e_Context))]
  attribute Sensor "Sensor": Integer [constraints (NotNull ForeignKey(e_Sensor))]
  attribute Description "Description": Text
  attribute associateDatetime "Associate Datetime": Datetime [constraints (NotNull)]
  attribute associateUser "Associate User": String [constraints (NotNull ForeignKey
(e_User))]
]
```

Spec. 7. Context Sensor ASL specification.

All the data is collected as an observation according to the WaterML specification [13] as to standardize the data collected by different sensors. For this effect, a Sensor Observation ASL specification was defined as shown on Spec. 8.

```
DataEntity e_SensorObservation "Sensor observation": Document [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute sensorId "Sensor": Integer [constraints (NotNull ForeignKey (e_Sensor))]
  attribute sensorType "Sensor Type": DataEnumeration SensorKind [constraints
(Derived ("sensorId.type"))]

  attribute startDatetime "Start Datetime": Datetime [constraints (NotNull)]
  attribute endDatetime "End Datetime": Datetime [constraints (NotNull)]
  attribute date "Date": Date [constraints (NotNull)]
  attribute time "Time": Time [constraints (NotNull)]
  attribute data "Data": String
]
```

Spec. 8. Sensor Observation ASL specification.

Additionally, the possibility for Social Sensors should be implemented. A Social Sensor is, simply put, a geo-referenced photo uploaded by a user. Ideally, the photos would be scrapped from social media platforms or directly uploaded by a user to the RCP. This photo would then be submitted to a machine learning algorithm [83] that estimates the water height depicted on the photo. The resulting classification results would be stored on RCP. This would result in a significant increase of data available on RCP.

4.4 HiSTAV Integration

RCP needs to communicate with HiSTAV to request simulations and receive their results to be displayed to the user. The exchange of data between these two systems is done through the implementation of a REST API in each of them. Most requests sent to HiSTAV use geojson files [48] in their bodies, containing the definition of the Context to simulate. The results of HiSTAV simulations are raster images sent in TIFF format and, are processed in RCP side, which involves their transformation in tiles in PNG format so they can be displayed in the implemented leaflet map as an overlay layer.

RCP should leverage all the data observed by the sensors and funnel it to HiSTAV Model. The result should then be sent back to RCP where the users can see them. Ideally this data feed to RCP is

constant, enabling the HiSTAV Model to be constantly running in a powerful enough machine. Having this constant data feed enables the automatic creation of noteworthy events. These events should be time bounded (i.e., have a start time and end time) and be highly detailed. They should also have a Context and simulation results associated with them. This way users can analyze past and current events and understand what impact they had on the associated Context area. The last feature would be the possibility of predicting future events. Allowing the organizations using RCP to better protect the population of areas in risk of flooding and their belongings.

5 RiverCure Portal – Simulation Pipeline

As explained in section 1.3, this research development was divided in three phases. The first two consisting on the specification and implementation of RCP, and the last phase on the integration of RCP with HiSTAV. In this chapter, the 2 phases related with RiverCure development and implementation decisions, as well as the reasoning behind these decisions, are looked into in depth starting from phase 1. Last phase is covered on chapter 6 – HiSTAV Integration.

5.1 RCP Data Model Definition and Generation from the ASL Specification (Phase 1)

The goal of phase 1 was to specify the RCP using the ASL language (contextualized on section 2.4) and, from this specification, generate a data model in Django/GeoDjango/Python code [18] (GeoDjango and Django terms are used interchangeably from this point onwards) from which RCP use cases could be implemented.

However, at the time of this generation ASL lacked spatial data types and a corresponding mapping to equivalent types in GeoDjango [36]. Thus, the first step was to decide which spatial types were necessary in ASL, so it could fully specify the intended RCP and, after completing this step, how to map these types into the spatial types of GeoDjango.

Since the scope of this project does not include the specification process of RCP in the ASL language, this phase is short, including only the tasks described in the previous paragraph. The ASL geometry data types definition and the exploration on how to transform these ASL geometries data types to GeoDjango data types.

ASL Data Types

After analyzing the objectives and goals of RCP four spatial data types, based on OGC standards [35], were chosen. These data types are generalist to provide the necessary flexibility for the development of a web application of this kind, as well as allowing the exploration of ASL geographic code generation mechanisms for other programming languages in the future, since ASL is a platform-independent language. These types were named: **GeoPoint**; **GeoPolyline**; **GeoPolygn**; **GeoRaster**.

These four spatial types have enough flexibility to cover a plethora of use cases necessary for geo applications. Specifically, the Context definition, which is the main feature of RCP on the scope of this project. This definition is explained in detail in section 5.2.

Data Types Mapping

After settling on the necessary data types for the holistic RCP definition, all that was left to do was to map these types to the equivalent in Django/Python datatypes, known as field types [37], based on SFA [35]. Table 2 shows the chosen mapping, along with a short description of the datatype in Django.

It is important to note that in the case of the GeoPolyline and GeoPolygn a field type capable of storing multiple polygons was chosen. The reason behind this choice is that it is extremely common, in the cases where a data entry is a geometry field like a polygon or a polyline, for this geometry field to be represented by multiple spatially independent polygons or polylines. This is not true for the cases where the geometry field is a point, and so it was decided to keep the GeoPoint as a PointField, and the GeoPolyline and GeoPolygn as MultiGeomFields. When it comes to raster, we picked the only available field type in Django for raster data type, i.e., RasterField.

Table 5. Datatypes mapping from ASL to Django (Python).

ASL Data Type	GeoDjango Data Type	GeoDjango Data Type Description
GeoPoint	PointField	Stores a Point
GeoPolyline	MultiLineStringField	Stores a MultiLineString
GeoPolygn	MultiPolygonField	Stores a MultiPolygon
GeoRaster	RasterField	Stores a GDALRaster

5.2 RCP's Context Creation and Definition (Phase 2)

In this phase the first iteration of RCP was developed, from its ASL generated data model. Focused mainly on the creation of the capability of designing and simulating a Context all in the same application from the user point of view.

To achieve this goal, it was necessary to integrate the HiSTAV simulation, the simulation software that predicts and calculates water levels based on a given Context and data from related sensors, with RCP. For this effect, it was necessary to develop mechanisms that allow a user to create the necessary input required by HiSTAV and integrate it with RCP through the creation of communication mechanisms, in a way that it is seamless and effortless for the user.

The HiSTAV simulation uses as input the Context, described on section 4.2, which is a collection of geometries and rasters with associated properties. The first step on the development of features associated with Contexts was its specification, seen of section 4.2. From this specification, it was necessary to transform it into a Django model that is automatically transformed in SQL tables. On Figure

21, the chosen structure is represented in UML. After rigorously defining the Context concept in a data model, the implementation of features that allow a user to define and create Contexts was performed. Both the creation and definition of a Context is covered in depth both in the perspective of the developer and of the user.

Lastly, it was necessary to have in mind that RCP would need to communicate with HiSTAV. So, there was a necessity to establish a format for communication between the two system later on. The chosen format was geojson file [48]. The representations for each geometry of Context in geojson is covered in this section.

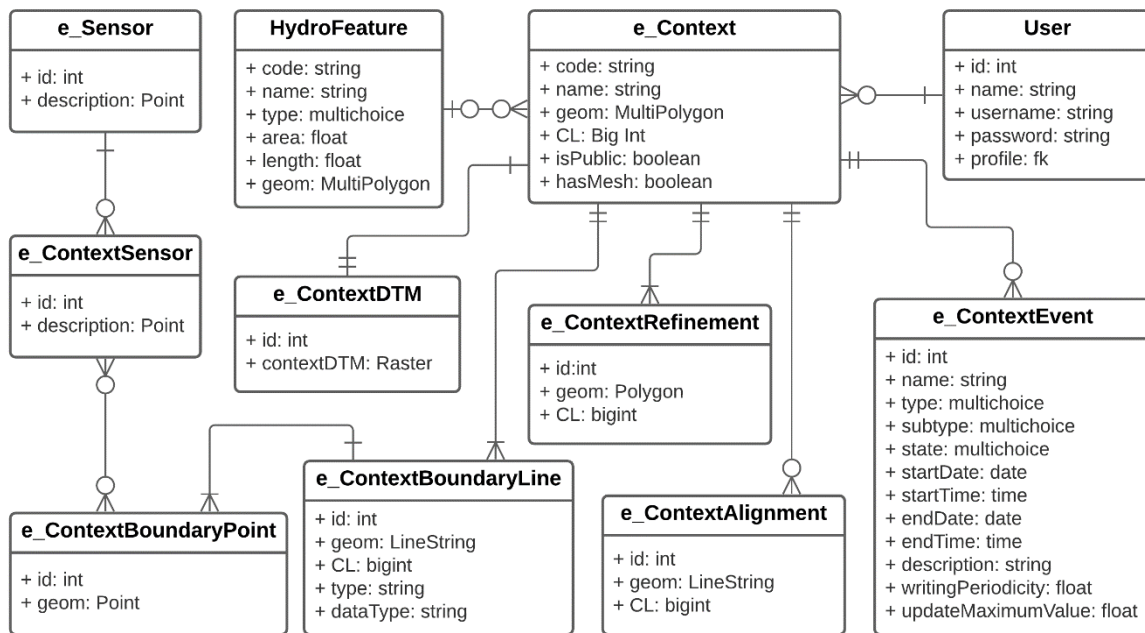


Figure 21. Context data architecture (ER notation).

Context Creation

In this section the creation of a Context workflow is explained in detail.

When accessing RCP, an authorized user should have a navbar similar to the one seen on Figure 22.

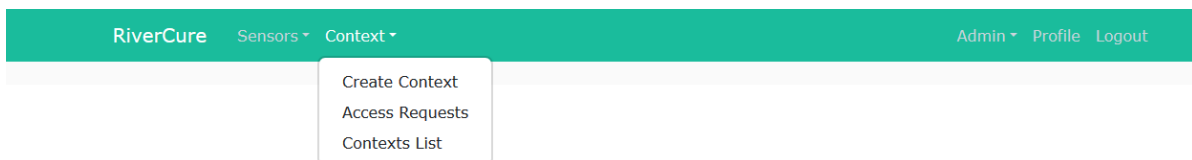


Figure 22. RCP navbar.

From here the user can select Create Context from the Context dropdown list. This option will only show up in case the user has a role and permission that allow for Context creation. After clicking the Create

Context option, the user is redirected to the admin panel of RCP, to a section where a Context creation is possible. An example of such a section can be seen on Figure 23.

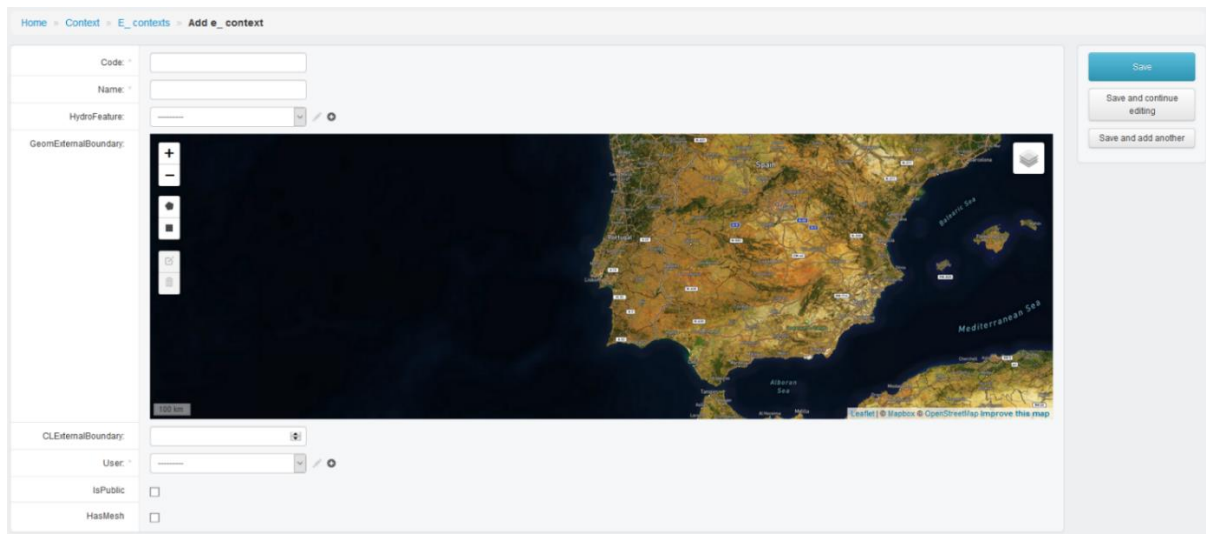


Figure 23. RCP admin page view of the Context creation section.

In this view the user can create a Context by defining a code, a name and defining an owner for the Context. It is possible to define other properties during the Context creation but these 3 are the only mandatory ones for a successful creation. The Context owner will be the user responsible for the definition of the Context. After all mandatory properties are defined the user can click save (the blue button on the top right corner) and conclude the Context creation. For the purpose of this demonstration we will assume that a Context named Coura has been created.

Context Definition

In this section the definition of a Context workflow on RCP is explained in detail, as well as all the necessary actions in each step to achieve the intended outcome.

By the end of this section the reader should have a good grasp of how a Context is created and defined in addition to what a Pre-processing ready Context looks like. For the purpose of this demonstration, RCP interface is shown as is when the user is logged in as a user with all the permissions.

Context List

After the Context is created, the Context owner, defined at Context creation, can now access it with the intent of defining it. On the Context navbar dropdown (Figure 22) there is an option called Contexts Lists. By clicking on this option, the user is redirected to a page looking like the one shown on Figure 24.

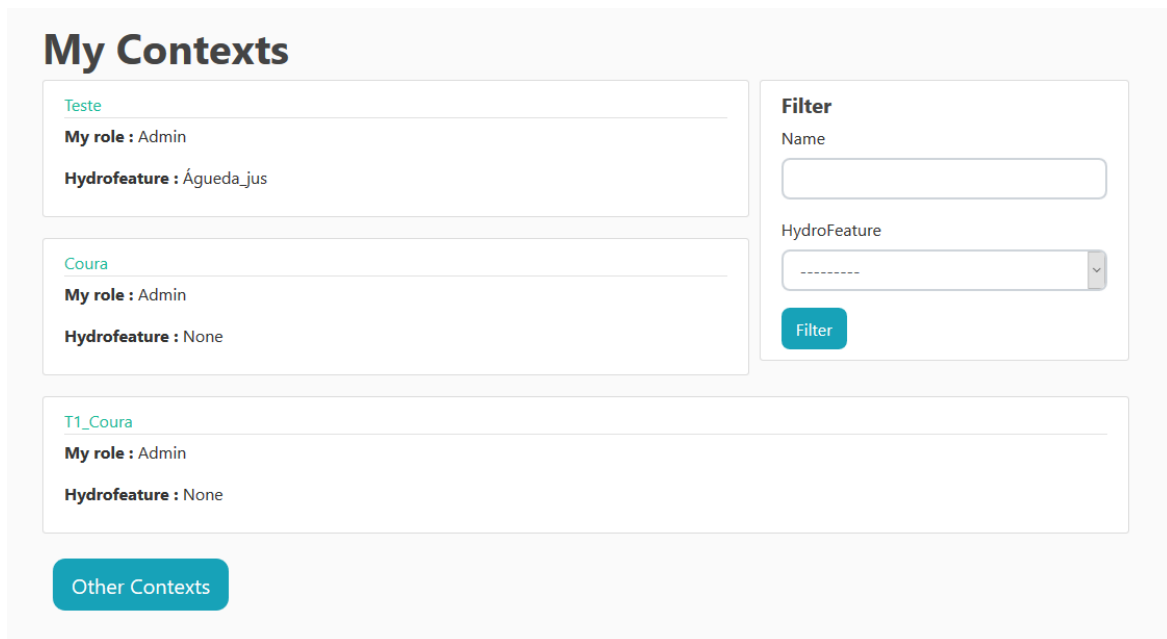


Figure 24. Context List Page View.

Here the user can see a list of the “Contexts” he owns. There is also a filter feature in case the user wants to search for an owned Context either by Name or HydroFeature. The Other Context button redirects the user to a similar list but where the “Contexts” present are not owned by him or her as long as the Contexts are public. The Context has a Boolean property named isPublic, seen on Figure 23, which defines if the Context should be visible for non-owner users.

The user can select one Context from his “Contexts” list, by clicking on the Context name. This will redirect the user to the Context Manage page, where he can see its definition in detail and, if necessary, edit the current definition. Assuming the user selected the Coura Context that was created during the creation phase, he would be redirected to a page similar to the one seen on Figure 25. The next section details the use cases of this page.

Context Manage Page

Figure 25 shows our hypothetical Context Definition, which is empty, as the Context has just been created. If some geometry was already defined, the map would show the geometries defining the Context. The only visual representation available on the map is that of Sensors. It is important to note that these Sensors are not yet Context Sensors at this point. The number on the circles relates to the code of the Sensor, which in this hypothetical case correspond to the codes 1, 2, 3, 4 and 5.

The Context Manage Page is where the heavy lifting of the Context Definition is done. This workflow is long. As such for readability sake, it is defined in the following distinct numbered sections.

Coura context

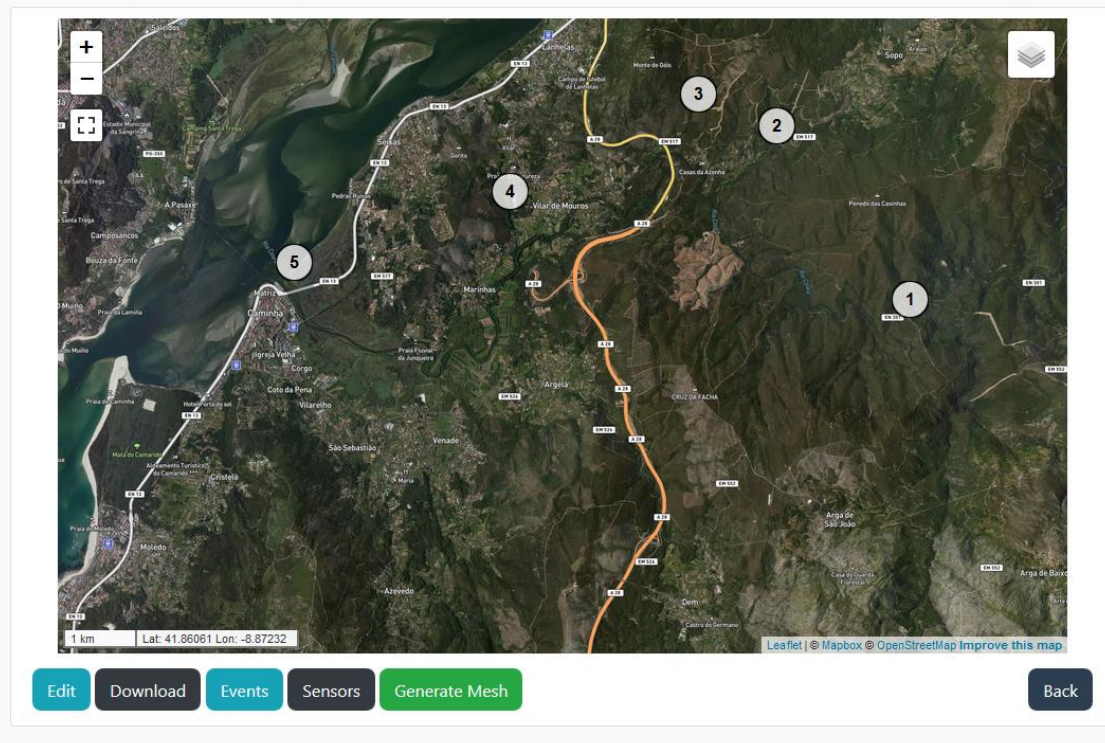


Figure 25. Context detail page view.

1. Starting a Definition & Upload Option

On the bottom of Figure 25 a wide selection of buttons is visible. Having an empty definition, the user would want to define the Context. For that he must click on the edit button. After clicking the edit button, a form prompt is displayed under the Context Detail map as seen on Figure 26. Since the scope of this section is to explain how a Context is defined, only the Edit button is relevant for now. The other options relevant for the scope of this dissertation are explained later in this document.

The purpose of this prompt is to allow the user to upload any definition of a Context Geometry he might have in the geojson file format before starting a manual edit. In case the user has a complete definition, a manual edit might not even be necessary. Besides the Context Geometries, the user can also upload a DTM file and a Contour lines file for visual aid when defining the Context, and in the case of the DTM, as an input for the HiSTAV Simulation. The upload of these files must be done in this phase as there is no other place where the user can upload them.

As for the buttons in the form prompt, their names are self-explanatory. The Upload button uploads the files and redirects the user to the Context Detail page. The Upload & Define Manually button uploads the files and redirects the user to the Context Manage page (this page detailed next). Finally, the Define Manually skips the upload and redirects the user directly to the Context Manage page. In case of the execution of an upload, a message alert is displayed on top of the page to which the user is redirected

to, notifying the user of the result of the upload. The two messages possibilities can be seen on Figure 27.

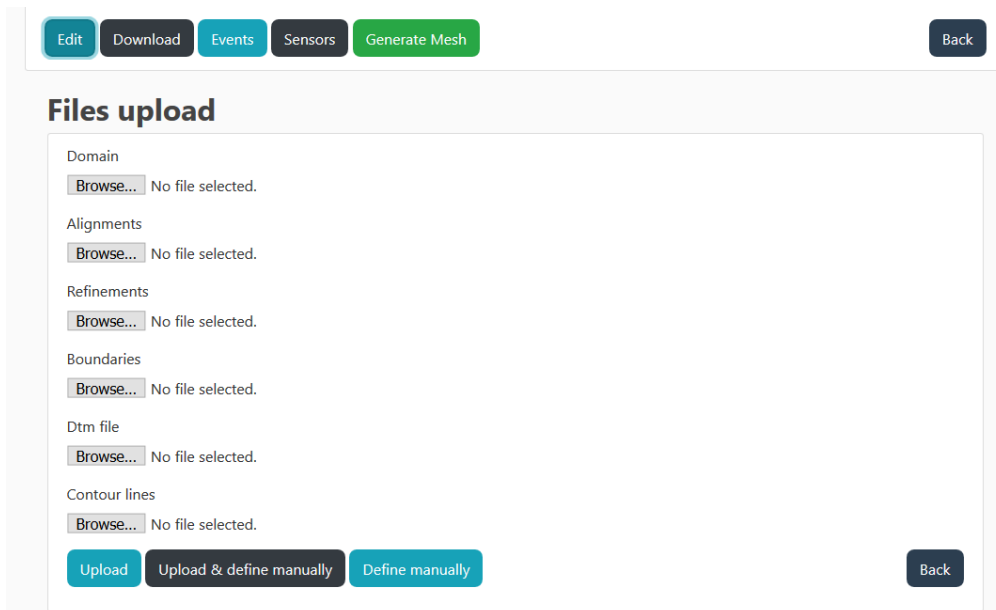


Figure 26. Prompt after clicking the edit button on a Context Detail page.

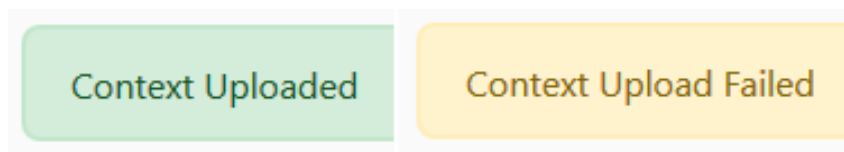


Figure 27. Possible notifications after upload request examples.

In case the user clicks a button that leads to a manual edit, the user will be redirected to the Context Manage page, where he can make all the necessary adjustments to the definition of the Context. Let us assume that we have uploaded all the geometry files, for our previously created Context Coura, except the Refinement. The resulting Context Manage page would look similar to the one seen on Figure 29.

2. Map Tiles & Layering

All distinct geometries have their own layer on the map. This makes it possible to hide a certain Context geometry type. This is done by toggling the types on the dropdown list present on the top right corner of the map.

By looking at this dropdown it is possible to see that there are 3 options which are a radio option while the rest are a toggle. This is due to the fact that these 3 options are base layers, while the others are overlay layers (the distinction between layers is provided in section 2.9 under leaflet). The presence of a contour lines base layer is intended to provide the user with a lightweight version of contour lines for

the world. If the users prefer to use their own contours lines, he can do so by uploading them as described before.

3. Drawing the Geometries

If the user does not have a geojson file definition for the different Context geometries, he needs to define them manually. This is done through the drawing of identified geometries directly on the map.

Before beginning to describe how to manually define the different Context geometries, the reader should notice the dropdown under the map, on the Manage Context view (seen on Figure 29) with the placeholder “Select the polygon to define”. The options available in the dropdown are shown on Figure 28. From the four options only the Boundary option needs special attention explaining. So, it will be left for last.

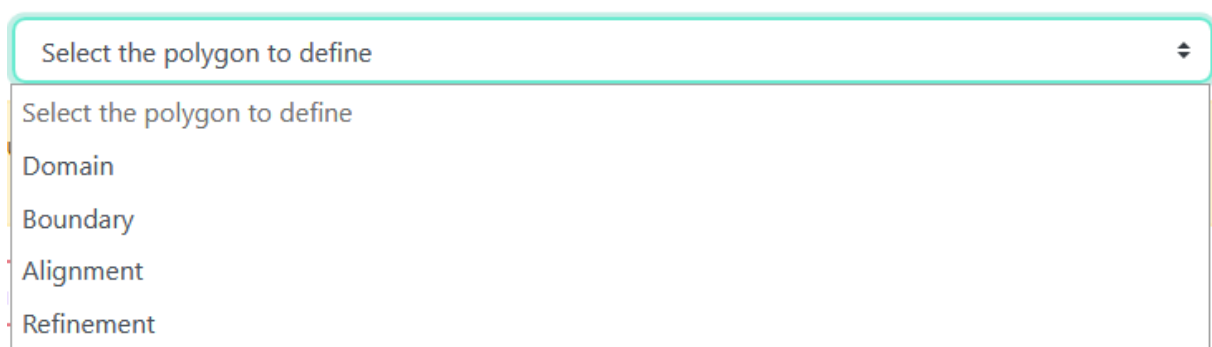


Figure 28. Dropdown options for selection of geometry to define on RCP.

The user starts by choosing which geometry to define. It is mandatory to define the Domain before all others and, some controls were implemented to guarantee that the user cannot define any other geometry before defining the Domain.

Counting from the top left corner of the map, the third and fourth square, allow the user to select a geometric shape to draw. The third square is for a LineString and the fourth for a Polygon. Once again, the user choices are limited, as he can only choose the corresponding geometric shape for the selected Context Geometry, Polygon for Domain and Refinement, LineString for Alignment. This constraint is also assured by controls limiting and warning the user in case of invalid choices. After selecting a shape, the user can draw it by just clicking on the map, as long as the shapes don't intersect each other or themselves.

The Boundaries are defined in a different way. When defining the Domain, all its points stay drawn on the map. The user can then define the Boundaries by selecting Boundary from the dropdown and linking sequential points of the Domain, clicking on an empty spot on the map when the Boundary is complete.

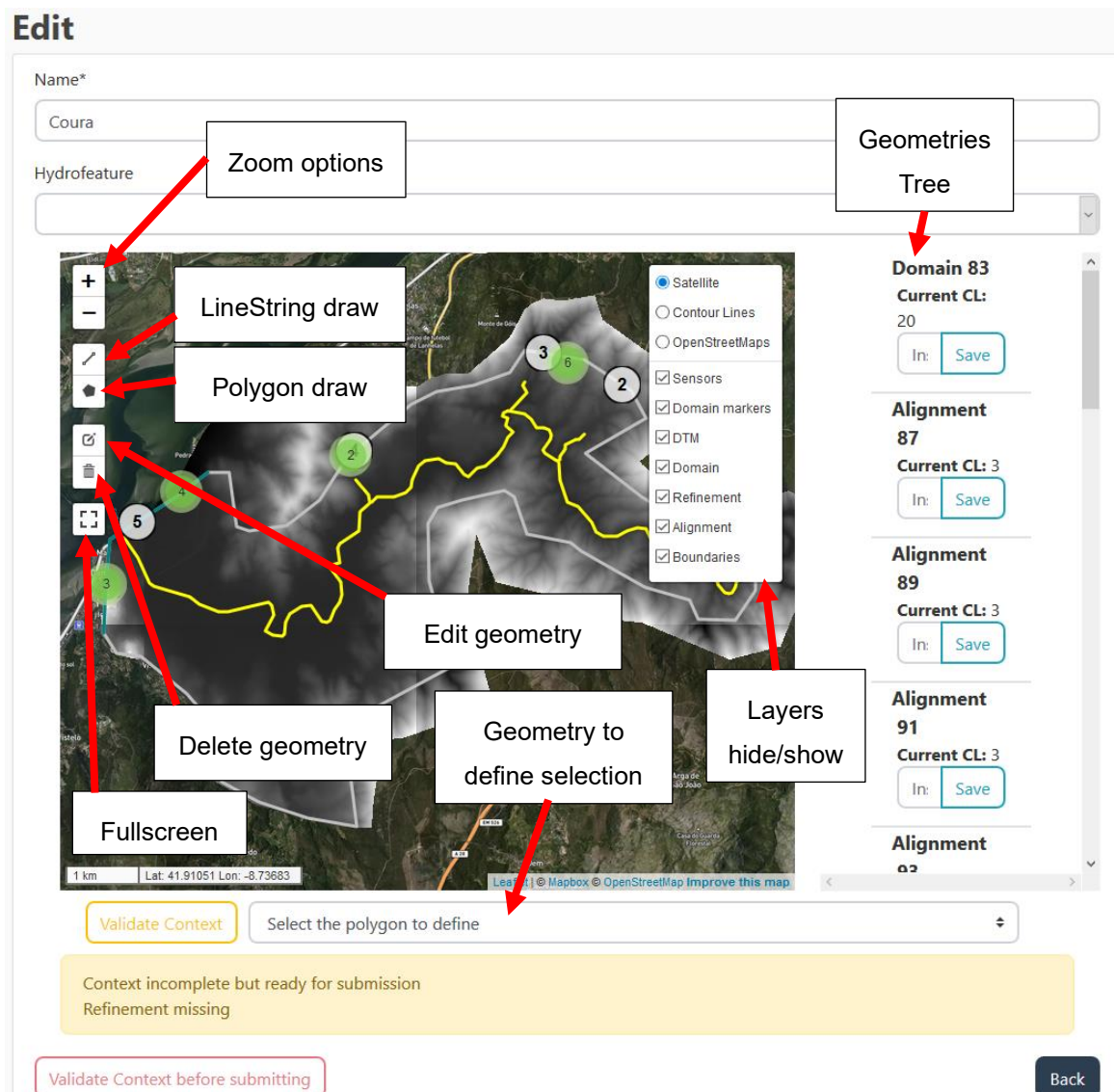


Figure 29. Manage Context page.

There is also an option to edit or remove any geometry from the map. This is done by selecting the fifth and sixth squares. The fifth square is for editing and the sixth for deleting. The meaning of editing here is related to the change of position or the addition of points to a geometry. These operations are independent from the currently selected geometry on the dropdown. Every time an edit or deletion is performed, it is mandatory for the user to redefine the Boundaries of the Context and, as such, the edit functionalities are not available for the Boundaries. Deletion is however possible by double clicking a Boundary.

As the reader can see, the workflow for drawing of the Context geometries is relaxed and flexible while having controls to keep the user on the right path. There are also some visual aids as to whether the Context is complete or not, as explained later in the Saving a Context subsection.

4. Boundary Points and Sensors Clustering

Both the Sensors and Boundary Points are clustered when the user is zoomed out (e.g., the green circles seen on Figure 29 are clusters of Boundary Points). These properties are not clustered together but in their own distinct group.

5. Geometry properties definition

On the right side of the map, a list of all the geometries defined on the map is available. Each entry on the list has input(s) field(s) according to its type of Context geometry. This input fields can be used to change the value of each geometry property (e.g., CL in the case of Domain, Refinements or Alignments). In alternative, the user can also click on the geometry directly on the map, which will trigger a popup to appear. This popup is designed according to the properties of its geometry. Since Domain, Refinements and Alignments have the same property, their popups are the same, changing only the name on top that identifies the geometry to which the popup belongs. An example of the different possible popups can be seen on Figure 30.

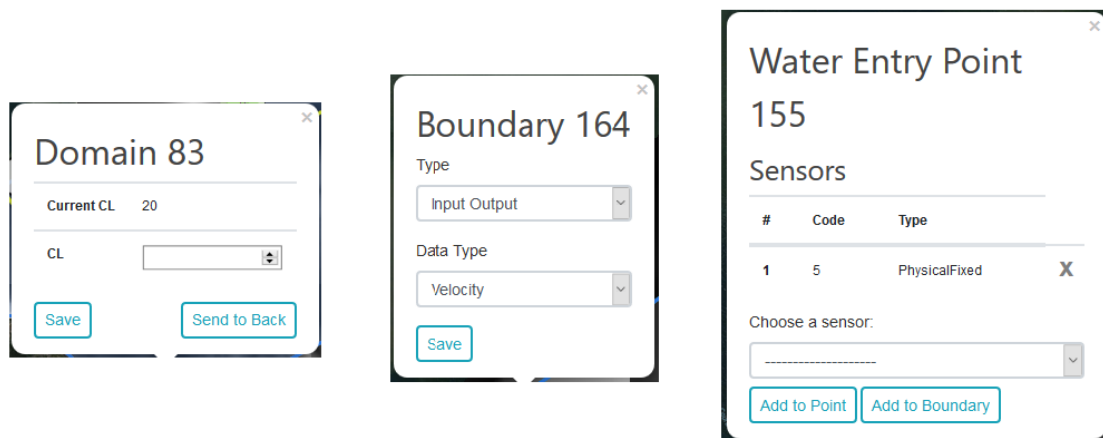


Figure 30. Popup examples (from left to right; Domain/Refinement/Alignment; Boundary; Boundary Point).

In the Domain/Refinement/Alignment popups the reader can notice that there is a button named Send to Back. Since a Context can have several different geometries that can be nested inside each other, there is a possibility of a geometry being defined on top of a smaller geometry. This will result in the smaller geometry being under the larger geometry making it non selectable for editing. The button Send to Back addresses this problem by allowing the user to reorder the different geometries in the Z axis. Other noteworthy difference is on the rightmost popup, where there are 2 buttons. The difference is simple: the Add to Point button adds the selected Sensor to the Boundary Point to which the open popup belongs, while the Add to Boundary adds the Sensor to every Boundary Point that belong to the same Boundary as the Boundary Point to which the open popup belongs.

Moreover, by hovering any entry on this list or on the geometry itself, the corresponding geometry is highlighted making it easier for the user to identify and distinguish each geometry. Moreover, each geometry has an id on the right of its name to further help distinguish it from the other geometries.

6. Saving a Context Definition

The last feature of note on this page is the alert at the bottom of the page, informing the user that the Refinement is not defined. This alert is dynamic and will notify the user of which are the missing geometries, or in case nothing is missing, that the Context is complete. The reader should also notice the 2 buttons near the alert bar named Validate Context and the Validate Context Before Submitting. Both buttons change color according to the state of the Context definition (e.g., green for complete, yellow for incomplete) with the latter changing the text to Save Context after the user successfully validates the Context under definition by clicking the former button. It is possible to validate and save the current progress at any time.

The reader will also notice that it is possible for the Context owner to change the HydroFeature and the Name of the Context. This is done by just writing a new name for the case of the Name and, for the HydroFeature the user can just select from from a dropdown list. At the moment of writing the HydroFeatures present in the dropdown are all the HydroFeatures in the system database.

Context Download

It is possible for the user to download a Context definition in the form of geojson files by clicking the Download button seen on Figure 25. By clicking this button, the user will get a prompt to download a compressed folder with 5 geojson files inside (one for each Context Geometry).

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            -49892.437846,
            245495.638554
          ],
          [
            -49896.968514,
            245453.238859
          ]
        ]
      },
      "properties": {
        "Geometry type": "Boundary Line",
        "Type": "InputOutput",
        "Data type": "Q"
      }
    },
  ]
}
```

Figure 31. Geojson representation of a Boundary.

An example of a Boundary geojson file can be seen on Figure 31. All the other files follow the same principles. A feature for each independent geometry, with the geometry type, coordinates, and its

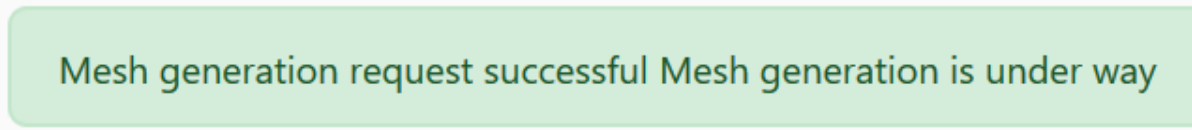
properties. In the case of the Boundary, the properties are the type of geometry in the Context, the Type, and the Data Type. All the coordinates values are in the EPSG 3763 as this is standard is mandatory for the HiSTAV Simulation.

Pre-processor Request

In this section the request is explained from the optic of a RCP user. The implementation on HiSTAV side is explained in the Phase 3 section.

Looking back at the buttons on the bottom of Figure 25, the reader can see a green button named Generate Mesh. By clicking this button, the user can request a mesh generation. This generation is done by an application independent from RCP (this application is covered on Phase 3 section). This means that on the scope of RCP the pre-processing is just a POST request.

After clicking the Generate Mesh button RCP will transform the Context Geometries in several different serialized geojsons, one file for each geometry type that are sent in the request body. The user receives a response from the other application indicating whether the pre-processing request was successful or not. A visual example of this response can be seen on Figure 32.



Mesh generation request successful Mesh generation is under way

Figure 32. Success message on mesh generation request.

Since the pre-processing is an asynchronous operation, the user does not get immediate feedback of whether the pre-processing was successful or not. Instead, the other application will notify RCP when the mesh generation is finished. This notification is done through a RCP endpoint detailed on section 4.2.6.

The user can find out if a mesh for a given Context is generated after loading the respective Context Detail page. Here, the Generate Mesh button can have the color yellow and the name Regenerate Mesh, instead of the green color and Generate Mesh name, indicating that this detailed Context already has a generated mesh. In this case, there is an additional button on the page called View Mesh, which will redirect the user to a Paraview Web Visualizer [71] instance, where the user can manually validate whether the mesh was correctly generated. The two possible variations of buttons combinations can be seen on Figure 33.

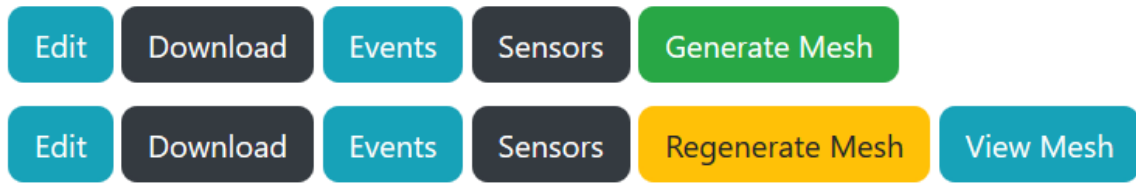


Figure 33. Available options for the RCP user based on whether the mesh has been generated.

Context Event Simulation

Requesting a HiSTAV Simulation is not as linear as requesting a Pre-Processing. For the simulation it is necessary for the user to create a Simulation Event. This Event will be associated to a Context as we will see later. Looking at Figure, 33 the reader can see a button called Events. Clicking on this button will direct the user to a list with all the Events associated with the Context detailed on the page Context Detail that contains this button. An example of such list is seen on Figure 34.

Coura

Events

Name	Context	Type	SubType	Start	End	State
ss	Coura	flood	Forecast	Jan. 1, 2020	Jan. 1, 2020	occurring

Add Event
All Events

Figure 34. Context Event List Page.

On the Context Event List page there is a button called Add Event that allows the user to create a new Context Event. Alternatively the user can click on a Context Event from the list to inspect the details of the clicked Context Event. Both options are covered on this section starting with the Event Creation.

By clicking on the Add Event the user is redirect to a page with a form with the fields of a Context Event properties. These properties were detailed on section 4.2.1 on the Context Event section. It is important to note that the user decision related with the start and end date and time, as well as the writing periodicity and update maximums will influence the results of the simulation. After filling all the mandatory fields, the user can click a button named create to create the Context Event.

This action will trigger a POST request to the HiSTAV Simulation API similar to the one described in the previous section. But this time instead of submitting the geojson representation of the Context geometries, RCP creates a text file, with the bnd extension, for each Context Sensor, with the Sensor Observations values found inside the time interval defined by the user. An example of such a file is shown Figure 40, on the Solver2D section. Additionally, another text file with the bnd extension, called output, is created with both update periodicities with one value per line, meaning the file has only two lines. Both the output file and the Sensors data files are submitted to the HiSTAV Simulator. The simulation can take hours to complete, so, just like the Pre-processing its execution is asynchronous. At the moment of request, the user is only notified on the successful start of the simulation. Only after HiSTAV finishes the simulation, does it notify RCP.

In case the user wants to inspect an already created Context Event, he can click on the Context Event name as described before. The user is then presented with a page with all the Context Event properties (example on Figure 35).

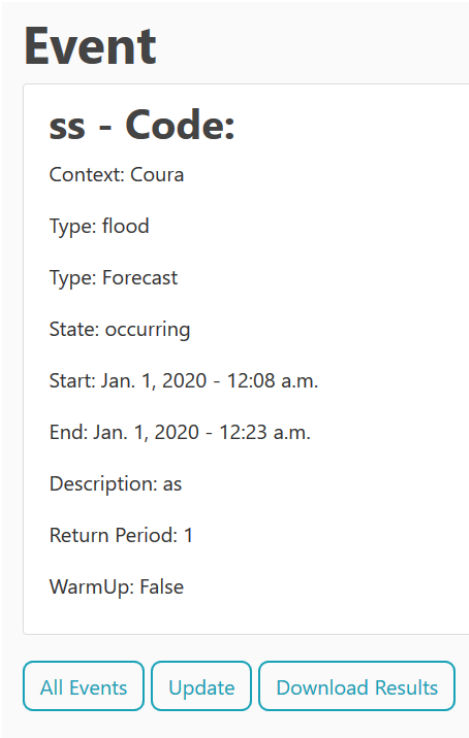


Figure 35. Event Context Detail page.

From this page the user can choose to update the Context Event or download the results, if they exist, by clicking on the respective description buttons located at the bottom of the page. Updating the properties does not change the simulation performed or being performed. However, this it is intended to implement this feature in future work.

Endpoints for HiSTAV Simulation

The execution of an external simulation known as HiSTAV. External meaning it is executed by a different and decoupled software on another machine, requiring the implementation of some endpoints for the communication between the two machines, these endpoints are addressed in this section. Some executions resulting from requests to this API can take hours to complete (e.g., HiSTAV Simulation) so, in order to avoid locking the user from performing other actions while the execution is underway, or, possible https timeouts by the requests, these executions were made asynchronous, as.

In order for the user to have some feedback and status information on these requests, it was necessary to implement 2 endpoints, one for each request available on the API. This API uses these endpoints to change the status of the executions resulting from the requests. Each of these endpoints is further explained in the subsequent sub sections.

Pre-processor endpoint

This endpoint is used by the Simulation API to update the status of the mesh generation, known also as pre-processing, of a Context.

The Context model has a Boolean field named hasMesh which indicates if the Context as an updated and valid mesh generated on the Simulation API side. This mesh is a pre-requirement for a simulation request on this Context.

Summary:

Url: {RivercurePortal_url}/contexts/mesh-status/<context_name>?status=<value>

(e.g., <https://RivercurePortalURL/contexts/mesh-status/Coura?status=True>)

Type: GET

Arguments:

- Context name – name of the Context to pre-process
- Value – either true or false, true if pre-processing completed successfully, false otherwise.

Solver2D endpoint

This endpoint is used by the Simulation API after a HiSTAV Simulation finishes. This endpoint signals RCP that the results for a specific event are available. The event identification is done through its id, and the simulation API knows the RiverCure internal event id because it is provided with the simulation request as seen in section 4.3.3.

Receiving a request on this endpoint will trigger the RCP to make a request to the Simulation API for the maximums results of said Simulation Event. The endpoint available for the results download is explored on section 4.3.4 under Simulation Results. After concluding the download of the maximums results RCP transforms the results into tiles that can be used to overlay the leaflet [44] map used on

RCP using Django Raster [84]. On Figure 36 an overlay example of the maximums results can be seen. The reader can see there are 4 maximums. Each of these maximums can be hidden enabling the user to examine each maximum individually.

Summary:

Url: {RivercurePortal_url}/contexts/results/handle/<event_id>

(e.g., <https://RivercurePortalURL/results/handle/7>)

Type: GET

Arguments:

- Event Id – internal RCP identifier of the event to which the simulation results belong to

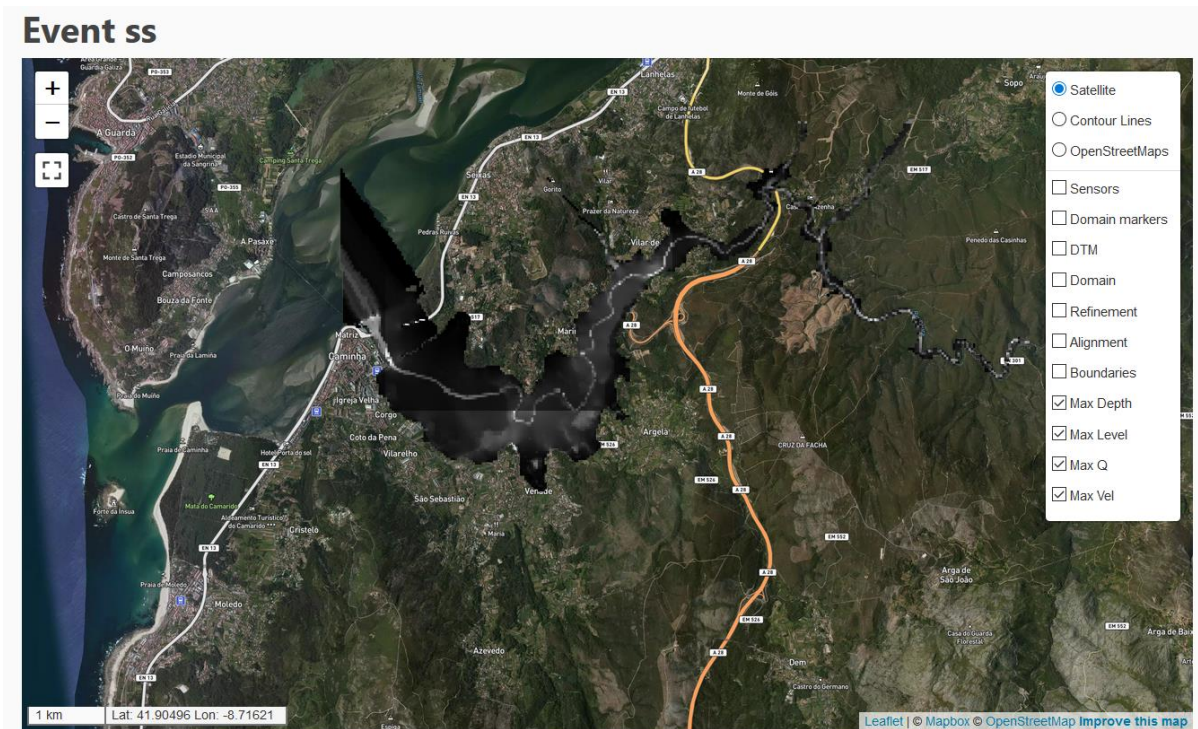


Figure 36. HiSTAV simulation results displayed on Event results page (maximums only).

6 RCP and HiSTAV Integration (Phase 3)

The Simulation pipeline consists of 2 parts. Firstly, a pre-processing of the Context is necessary, followed by the simulation using a software called solver2D. In the pre-processing part, a user needs to make a request and provide the 5 geometries files: (i) Domain; (ii) refinement; (iii) alignment; (iv) boundary; (v) and boundary points. This is done by simply clicking a button as seen on the previous Pre-processing request section. These files are used by a software called mesh to generate a raster (example shown on Figure 39), saved in VTK format, which is an input for the solver2D software. From this point onward this mesh software will be addressed as pre-processor. After this VTK file is created, the Context is ready for the simulation by solver2D (explained in section 6.2). It is possible for the user to verify and validate the output of mesh by using Paraview Web Visualizer. Although it is not part of the simulation pipeline directly, it is an important validation tool, and, as such, it is addressed in this section.

The executions of the pre-processor and solver2D are independent of each other. This means that, in theory, the execution of the solver2D does not have to be preceded mandatorily by the execution of the pre-processor. However, since the output of the pre-processor is an input for the solver2D, in practice, the pre-processor needs to be executed at least once before executing the solver2D. After generating the mesh, solver2D can be execute as many times as needed with different constraints (e.g., time intervals or frequency outputs), without the need to execute the pre-processor again, as long as the Context definition does not change. A more detailed BPMN of this pipeline, following the structure of Figure 1, can be seen on Figure 37.

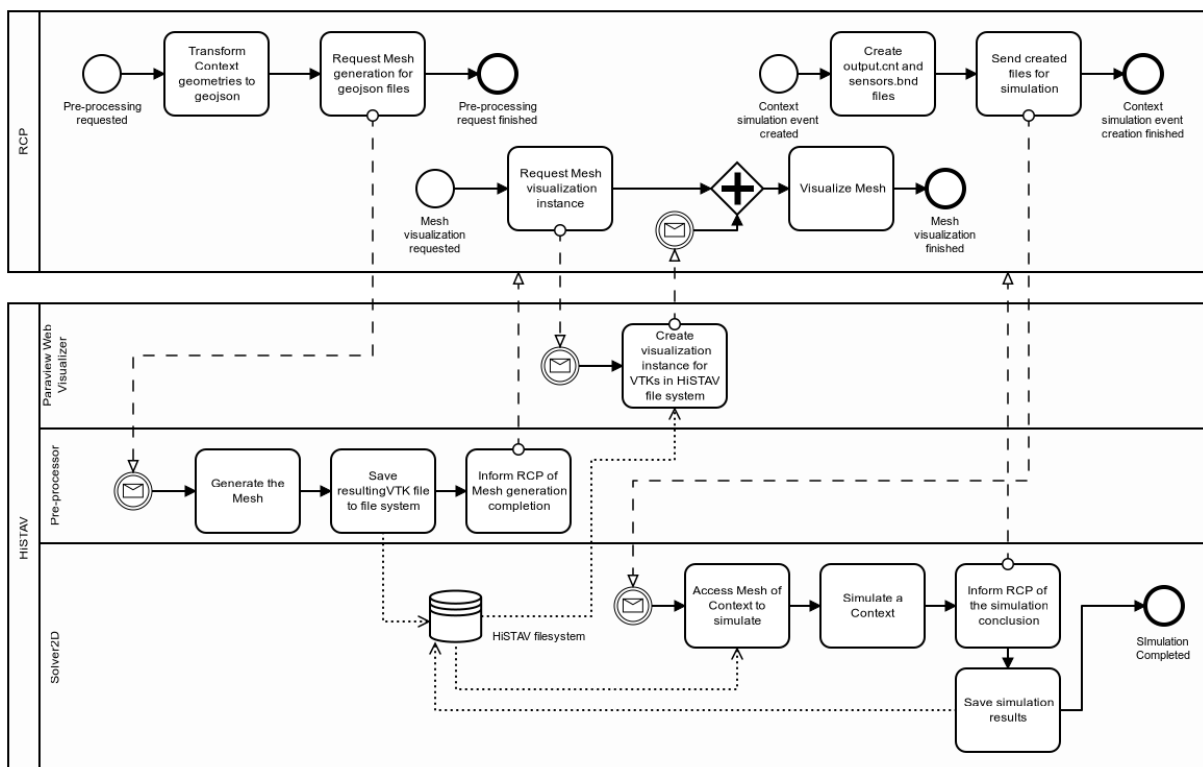


Figure 37. HiSTAV integration with RCP.

Both the pre-processor and solver2D, are part of HiSTAV and are done using external programs, mesh software for the pre-processing and solver2D. Architecturally, it makes sense to decouple RCP from the software that executes these 2 steps. This way overloading RCP is avoided, as some simulations can take hours to complete, and the flexibility is increased in the case of designing or redesigning the Portal, as we can worry more about the usability of the Portal for the end-user and less about the heavy back-office simulation operation. By consequence, it was necessary to create a simple REST API [14], that RCP can use to request the pre-processing and simulation operations.

In the next 3 section these 3 main parts of the whole simulation are addressed, starting with the pre-processor and finishing with the API. After, a concise overview of Paraview Web Visualizer role on the simulation pipeline is provided.

6.1 Pre-processor

The objective of the pre-processor is to generate a mesh called **meshQuality.vtk**. This VTK file is an input for a Solver2D execution is requested. Without this file, Solver2D cannot execute successfully and, as such, our value stream is compromised. As a consequence, the end-user should always verify if a given Context has an updated mesh before requesting a simulation for such Context. The pre-processing inputs and output are detailed in the next paragraphs.

The Geometry files are geojson files that contain the geo-information and its properties of the several geometries designed on RCP and analyzed in the previous section. Each file contains the information corresponding to its name. Since we already went over these geometries in detail and their geojson representation on the previous section, 5.2 Context definition, we refrain from doing it again here. An enumeration of the different files follows: (i) domain.geojson; (ii) alignments.geojson; (iii) refinements.geojson; (iv) boundaries.geojson; (v) boundaries_points.geojson;

The DTM is a file, called dtm.tif, that contains a raster in geoTIFF format known as the digital terrain model or for short, DTM. The end-user is responsible for uploading this file to RCP, and it should correspond to an area overlapping and slightly larger than the Domain geometry area.

The Friction Coefficient is a file, called frictionCoef.tif, that contains a raster in geoTIFF where each pixel has the friction coefficient value of the terrain located in the same coordinates as the pixel.

On Figure 38, the reader can see the console output of a successful pre-processor execution.

```
Pre-processing completed in 31 seconds

Writing Nodes      [=====] 100 %
Writing FrictionMap [=====] 100 %
Writing Facets     [=====] 100 %
Writing Boundaries [=====] 100 %
Writing VTK File   [=====] 100 %

All files written in 21 seconds
```

Figure 38. Output of a successful pre-processor request.

The output of the pre-processor is a mesh file, containing a triangular grid, corresponding to the Context given as input. Figure 39 shows an example of a successful mesh generation.

6.2 Paraview Web Visualizer.

As we have seen, before performing the HiSTAV simulation, the user needs to perform a pre-processing task to generate a mesh for the Context under simulation. This mesh is generated and saved in a VTK format and is an important input for the HiSTAV simulation. As such, the user might want or need to check if it was correctly generated before starting a simulation. Hence a tool that allows the visualization and interaction with such a file was necessary. For this project Paraview Web Visualizer [71] was chosen as it provides all the necessary features for the validation of the mesh. A user can easily access this visualizer by clicking View Mesh in the Context Detail page in RCP as seen in the previous section. Figure 39 shows an example of a visualization of a mesh on Paraview Web Visualizer.

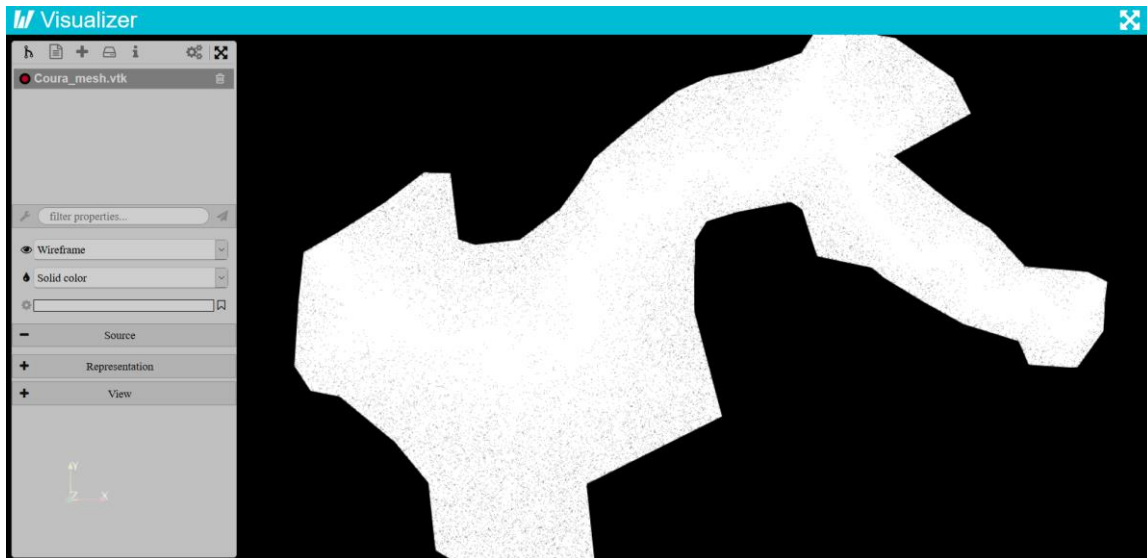


Figure 39. Coura Context generated mesh (seen on Paraview Web Visualizer).

6.3 Solver2D

Solver2D is the software responsible for simulating a given Context. The execution is the last step on the value stream this project set out to achieve. Solver 2D has two inputs, which are named sensor.bnd(s) and output.cnt, and a series of outputs explained in the next paragraphs.

Sensor bnds are several text files, each corresponding to a certain sensor associated with a Context point of the Context under simulation. In order to distinguish to which sensor belongs each file, the files are named after the sensors (e.g., sensor_mare.bnd, where mare is the name of the sensor). The file data is comprised of 2 columns, separated by a tab. The column on the left contains the instant corresponding to a value on the right of a reading by the sensor, of for example, a discharge. Every file left column starts on instant 0 adding 60 seconds per each row until reaching the duration of the

simulation. The instant 0 is the instant of the beginning of the simulation and contains the value of the sensor observation with the corresponding timestamp, e.g., if the simulation was configured to start at 01/12/2021 at midday, then the value on the right of 0 would be the sensor observation with the date 01/12/2021 and time 12:00. The same principle applies for all the other values. The start and end datetime of the simulation are defined by the user on RCP when creating a simulation event through a form (seen in section 5.2 Context Event Simulation). Figure 40 shows an example of a complete Sensor bnd file contents.

0	0.001
60	0.002
120	0.002
180	0.003
240	0.003
300	0.004
360	0.005
420	0.007
480	0.008
540	0.009
600	0.011
660	0.013
720	0.015
780	0.017
840	0.02
900	0.022

Figure 40. Sensor bnd file data example.

Output.cnt file is comprised of two lines. Each line contains one value in hz. The first line corresponds to the frequency at which the user wants to write to the files being generated. The second line corresponds to the frequency at which the user wants to update the maximums during the simulation. Both output.cnt values and sensors.bnd(s) are provided during an event generation in RCP just like the start and end datetime.

Solver 2D Output is a VTK file. An example can be seen on Figure 41. A specialized user can analyse this file to assess the risk of flood for the Context Event from which the Simulation resulted. Additionally, there is the possibility of transforming this VTK into the following four different TIFF: (i) max depth; (ii) max level; (iii) max q; (iv) max vel. This transformation is made using a python script named stavResults.py that was not developed in the scope of this project. These TIFFs allow the specialized user to reach the same conclusion with the advantage of being significantly smaller in size, resulting in the decision to use these TIFFs instead of the VTK to increase performance.

Additionally, during the simulation, other VTK files related with hydrodynamics are generated on HiSTAV. The number and size of these files is large and, dependent on the Writing Periodicity defined by the user during Context Event creation (i.e., the higher the frequency, the higher the number of files). This makes

these files not suitable for display on RCP. As such, these files are only available to users with access to the HiSTAV machine. Possibly, in the future, these files could be made available on the Paraview Web Visualizer since they are VTKs.

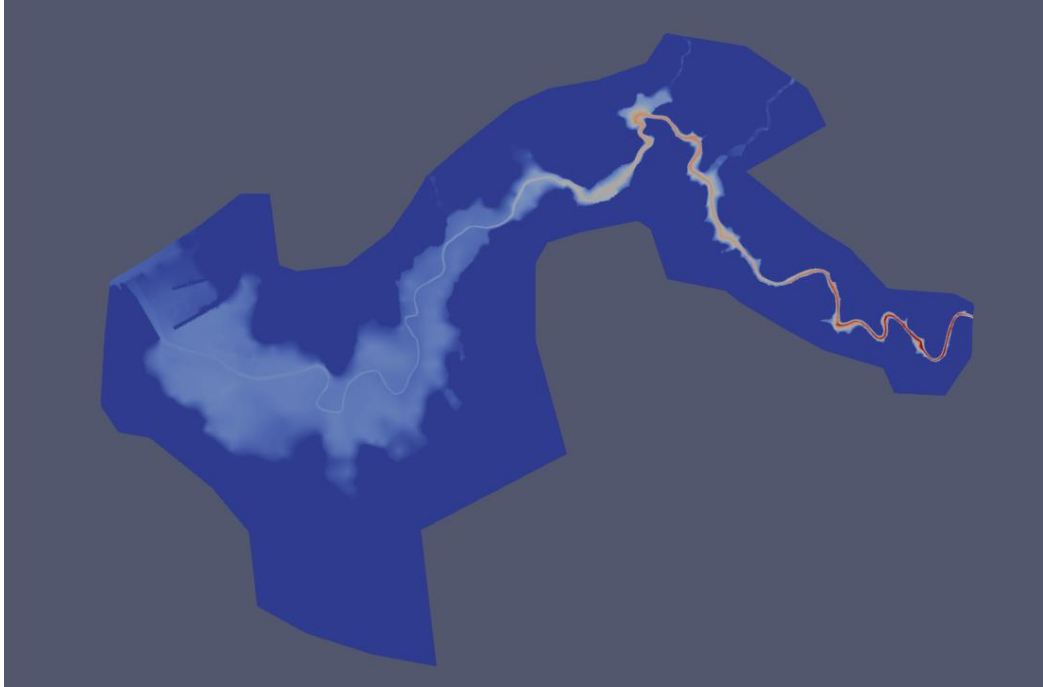


Figure 41. Resulting VTK from a HiSTAV simulation example.

6.4 RCP and HiSTAV integration API

This API serves as the communication medium between RCP and HiSTAV, and is built with Flask [19]. It has three available requests, which are pre-processing, simulate and, simulation results request. It is important to note that the call and execution of both the pre-processing and the simulation programs are asynchronous, while the simulation result is not. This implies that the API, in the asynchronous cases, will respond to the RCP request before the end of the execution and, by consequence it was necessary to implement the capability for bilateral requests to inform RCP of the asynchronous processes' completion. The RCP has two endpoints that HiSTAV can use for this effect, described on RiverCure section.

Pre-processing

This endpoint enables RCP to request a mesh generation for a given Context. For this effect RCP must provide the geojson files representation of the five "Context Geometries", one file for each geometry, on the request body. The request will return a string with either "success" or "fail" value, with the former representing a successful mesh generation process start and the latter the contrary. Since the mesh

generation is done asynchronously, this request also makes a request of its own to the RCP. Its purpose is to signal RCP that the Context pre-processing has been finalized and has a valid generated mesh. This request is made using cURL [85] after the mesh program exits with a successful code. In case the processing is unsuccessful, no request is made.

Summary:

Url: {API_Location_url}/process/

(e.g., https://HiSTAVURL/process/?context_name=Coura&event_id=7)

Type: POST

Data: Geojson files

Arguments:

- Context name – name of the Context to pre-process
- Event Id – internal RCP identifier of the event requesting the simulation

Response:

- **Type:** string
- **Successful:** “success”
- **Failed:** “fail”

Simulate

This endpoint enables RiverCure to request a simulation for a given Context Event. For this effect RCP must provide a text file named output.cnt and the collection of bnds files with the Context Sensors Observations data. The request will return a string with either “success” or “fail” value. The former representing a successful mesh generation process start and the latter the contrary. Since the mesh generation is done asynchronously, HiSTAV also makes a request of its own to the RCP to signal the end of the simulation using cURL. This request will trigger RCP to request HiSTAV the Simulation Results.

Summary:

Url: {API_Location_url}/simulate/

(e.g., https://HiSTAVURL/simulate/?context_name=Coura&event_id=7)

Type: POST

Data: Frequency file & sensor data files

Arguments:

- Context name – name of the Context to pre-process

- Event Id – internal RCP identifier of the event requesting the simulation

Response:

- **Type:** string
- **Successful:** “success”
- **Failed:** “fail”

Simulation Results

This endpoint provides the requester with a zip folder containing the 4 tifs generated from the maximum.VTK with the stavResults.py. This transformation was explained in section 4.3.2. The generation from VTK to TIFF is done after the request to this endpoint is made. The arguments serve as identifiers of which Context and Context Event belong the results being requested.

Summary:

Url: {API_Location_url}/simulation/results

(e.g., https://HiSTAVURL/simulation/results/?context_name=Coura)

Type: GET

Arguments:

- Context name – name of the Context to pre-process
- Event Id – internal RCP identifier of the event requesting the simulation

Response:

- Type: zip
- Contents: The 4 tifs generated from the maximum VTK

6.5 Simulation Pipeline Overview

The simulation pipeline is one of the main value streams created by RCP, and the main focus of this project. It leverages the capabilities of the HiSTAV simulation to allow a user to design a Context, simulate it, and analyze the simulation results without interacting with HiSTAV directly. From the result analyzes a specialized user can make decisions and protect the populace of these areas with the necessary confidence. All while providing a seamless user experience in the definition of the Contexts necessary for these simulations, along with the accessibility the web has provided all these years.

Ergo, it is important that the reader has an understanding on how all the three phases come together to form the simulation pipeline. For this purpose, an overview of the complete pipeline is provided, with general steps and rules for each step. This pipeline consists in the following steps:

1. Context creation by an authorized user and attribution to a Context owner
 - a. When creating a Context, the authorized user will only define a code and a name followed by attributing the Context to an user who will be responsible by point 2 of the simulation pipeline. This user is known as the Context owner.
2. Context definition by the Context owner
 - a. It is required to have a complete Context definition to move on to the next point of the simulation pipeline, but a Context can be saved and stored in an incomplete state at any time.
 - b. A Context is considered complete when all the different geometries and each respective properties have been defined, with the exception of the alignment, which is optional.
 - c. All the Context information defined by the user is stored in a geographic database, in this case a PostGIS DB, when submitted by the user in a valid state to the web server.
 - d. The developed system, RCP, can then transform this information into five different geojson files, each containing one type of geometry and the corresponding properties. Inside the properties tag, in a key/value pair format, where key is the name and identifier of the property and value the corresponding value.
3. Pre-processing request by the Context owner
 - a. The Context definition must be complete before requesting the pre-processing.
 - b. The purpose of the pre-processing is to generate a mesh for a given Context, which will be used as an input for the HiSTAV simulation.
 - c. The pre-processing is executed by an independent application that is part of HiSTAV.
 - d. This independent application has an API RiverCure uses to make the pre-processing requests.
 - e. The request is a POST request, in which the five geometries (Domain, refinements, alignments, boundaries, boundaries points) are sent in its body as a geojson files.
 - f. These geojson files are generated at the moment of the request from the data in the PostGIS database.
 - g. The pre-processing is asynchronous and can take some time, so the user will not get immediate feedback as on if the pre-processing was successful. It will however receive feedback on if the request was successful and the pre-processing was started.
 - h. After concluding the pre-processing, the processor will use a RCP endpoint update the status of the pre-processing for the Context.
 - i. In the Context detail page, the user has a visual confirmation as of whether the mesh for the Context is generated or not.
 - j. The user has a button in the Context detail page for redirection to an instance of Paraview Web Visualizer [71], where visualization and subsequent evaluation of the generated mesh is possible.
4. A user creates a Context simulation event
 - a. The event must have a start and end date and time.

- b. The simulation is executed by an independent application that is part of HiSTAV.
- c. This independent application has an API RiverCure uses to make the simulation requests.
- d. The request is a POST requests, in which an output file and the sensors data files are sent in its body.
- e. All the files are text files generated at the moment of the event creation. This includes the output writing periodicity file, and the sensor data files.
- f. The file with the data of the frequency of output writing is based on the writing periodicity and the update maximums frequency defined by the user at the event creation
- g. The file with the data of the Context sensors is defined based on the sensor observations in the interval between the start and end date and time interval defined by the user at the time of the event creation.
- h. The simulation is asynchronous and takes a long time, so, just like on the pre-processing request, the user does not get feedback as to whether the simulation was successful immediately, only if the simulation started correctly. The RCP is informed by HiSTAV when the simulation finishes. After being notified the RCP request the results to HiSTAV.

For a full execution of the Simulation Pipeline, a given Context must have a created Event and this Event must have results, that are generated by the execution of HiSTAV with values defined during the creation of the Event, and the mesh generated from the Context geometries definition by the pre-processor.

7 Evaluation

This chapter describes in detail how the evaluation of this thesis was tackled. This evaluation is focused mainly on the comparison between RCP and Mike 21 [75], as Mike 21 is considered an important competitor of RCP. As explained in section 1.3, RCP was evaluated almost every week by a majority of stakeholders. In the context of this thesis, this evaluation is focused on the entire pipeline necessary to simulate a flood impact in a given Context, which integrates HiSTAV. This pipeline is divided in 2 main tasks. First it is necessary to define the Context, and only after that, can the simulation be performed. These tasks are evaluated independently in this chapter.

7.1 Context Definition Evaluation

The first part of the evaluation was focused on the use case of the Context Definition. This definition, as described in section 5.2, is essentially the definition of the different required geometries. On Mike 21, this definition is done on the desktop application itself. For RCP, a different approach was selected. The main idea was to join every operation in the same program from the user perspective, so that he only has to interact with one application. This application is RCP. For this effect, every week, RCP features were implemented that fulfill the smaller use cases of the Context Definition, (e.g., Domain definition, Boundary definition...) until fulfilling 100% of the Context definition requirements. Tools like ArcGIS [86] or QGIS [87] can also be used to treat or define data to be uploaded to RCP, bypassing some of its limitations, but sacrificing some interaction streamlining (i.e., an additional step is added), or Mike 21. The implementation of these features was considered successful when the output from the Context Definition on ArcGIS and RCP produced the same output. This means both definition outputs could be fed to either HiSTAV or Mike 21 for a simulation with confidence that the result would be valid.

In the definition of auxiliary files (e.g., the DTM) RCP is severely limited, requiring the user to upload an already existing file, previously defined elsewhere. With Mike 21 the user uses ArcGIS, which does not have this limitation, as it is possible to define such files on ArcGIS. In RCP favor is the fact that it is a web application, contrary to ArcGIS which is a desktop application. This results in a high availability of RCP as it is accessible to anyone, with an authorized account, from anywhere where internet is available. It is also possible to easily share defined Contexts between users. Moreover, the introduction of new sensors in the system is more linear on RCP. Sensors can be added to the RCP with the corresponding data and associate such sensor to a Boundary Point in a streamlined way. In Mike 21 it is required for the user to find and associate the sensor data manually.

7.2 Simulation Evaluation

In both RCP and Mike 21 pipeline cases, after the Context is defined, the user should proceed with a simulation execution. This step is performed differently depending on the system in use.

On Mike 21, the user gets the output from the ArcGIS definition, consisting mainly of shapefiles and TIFFs. Additionally, the user must also get a timeseries file for each sensor associated with the Boundary Points and define some parameters like the writing of new maximums frequency. Only after these steps are taken, can the user initialize Mike 21, providing these files as input for a simulation. As the reader can understand, these steps are error prone as there are too many manual definitions which can result in an invalid simulation.

To solve these problems, the simulation execution in RCP has been streamlined. Each sensor timeseries is automatically created, based on the time interval defined by the user when creating a simulation Event. Since each sensor was associated to a Boundary Point during the Context Definition phase, the association of each timeseries files to the corresponding geometry is also done automatically. As for values like the frequency of writing of new maximums, they are also provided by the user on the simulation Event creation. From the user point of view this involves clicking 2 buttons and filling out a small form. On RCP simulation pipeline the simulation execution process is much more streamlined than in Mike 21. This results in a more trustworthy result for each simulation as well as a much better user experience, since the user does not have to perform as many steps on RCP.

7.3 Discussion

Overall, the differences between RCP and Mike 21 are considerable. Whereas Mike 21 provides a more complete set of simulations, RCP focus more on the user experience and dissemination of results in real time. Most importantly, however, is the fact that RCP can produce a valid and complete simulation output. For these reasons, the evaluation performed is positive, as the primary goal was to prove that it was possible to build a valid web-based simulation tool. A more detailed comparison between the two systems will be needed in the future, when RCP is further polished, with more users and rigorous Context definition use cases. Summarizing, the pros and cons of RCP are as follows:

Pros:

- Accessible from anywhere since it is a web application.
- Easy sharing of Contexts and other data between different users.
- Introducing new sensors and associating them with a Boundary Point is streamlined.
- No installation required, due to RCP being a web application that supports multiple organizations.
- Seamless integration with the simulator.
- Possibility of integrations of data streams from different sources.
- Simulations results available to every authorized user.
- Possibility of continuous execution of simulations in the future.
- Better user interaction and experience.
- All data is stored in a GIS Database (in this case PostGIS).
- Potential to implement new features like continuous simulation.

Cons:

- RCP's Context definition only allows the manual definition of geometries and sensors, with every other data file having to be uploaded by a user (e.g., DTM).
- HiSTAV simulator is not as complete and extensive as Mike 21 (RCP is focused on a single type of simulation, at the moment of writing).
- RCP does not have a proven track record or the level of professional support available on Mike 21
- Overall RCP is a less fledged system in terms of simulation options

8 Conclusion

This chapter covers the conclusion achieved after completing this first iteration of RCP and, the possible future work that can be performed to further improve RCP.

8.1 Conclusion

The necessity of tools such as RCP is more relevant than ever. However, there has been a lack of leverage of the web from existing tools. RCP successfully solves this problem. The fact that it is a web application with a PostGIS Database makes it easier to share and develop results and opens the door to the integrations of diverse data streams due to the use of OGC standards.

Moreover, RCP streamlines the interaction of the user with Water Management tools by automating some crucial steps that had to be done manually in other tools. This results in an improved user experience, which results in a higher likelihood of the specialized user performing a good job.

It is important to not forget that, at the moment of writing, RCP is still in its first iteration, still missing some of its envisioned features like the constant stream of data to the Portal. A fully fledged RCP would constantly simulate, requiring user interference only when defining new rules or Contexts.

A tool like RCP can change how governments and civil protection organizations protect their citizens, allowing these organizations to predict moderate to serious flood events. With these predictions these organizations can act in order to minimize damage and save lives.

Additionally, it was also proved with this work that an ASL specification can generate a valid data model, from which an entire application can be developed. By having a transversal specification that can be understood by most stakeholders, like ASL, we are minimizing the risk that the software under development does not correspond to the expectations of all stakeholders.

8.2 Future Work

The potential for a collaborative web application like RiverCure is almost limitless. However, during the development and discussion of RCP, some of the potential features were left out due to time constraints. Some of these features have high potential of increasing the value of the RCP in the future. These features are:

Standardize Sensor Data. A stronger and more precise standardization of the data present in the sensor could bring a big advantage in terms of maintainability, portability, and compatibility to RiverCure in the future. By implementing a standard (e.g., the OGC standard [12]) to the sensors present in RCP, new sensors can be easily added almost indefinitely. Easily because the probability of the candidate new sensor also following a standard is high. Indefinitely because, since if all the sensor data follows the same metadata, a single task can be implemented to handle the data coming from the sensors.

Creation of a continuous data stream from the sensors. If RCP could access or receive a data stream from each sensor, its potential would increase by a wide margin. In that way, RiverCure could treat and send this data continuously to HiSTAV, where it would be used to perform continuous simulations, significantly increasing flood predictability accuracy. This feature requires the participation of several stakeholders and as such requires a high amount of effort to implement.

Implementation of “Social Sensors”. Previously, a machine learning algorithm that classifies water heights was developed [83] to be integrated in RCP. By allowing users to upload georeferenced and timestamped images and photos of floods, we could leverage this algorithm to extract standardized data from these pictures. Such pictures would then serve as sensors, aka “Social Sensors”.

Automation of Event Creation. Right now, all events are user created. Moreover, in practice there is only one kind of event, which is the HiSTAV Simulation. If some of the points talked about on this section were implemented, the automatic creation of certain events under certain conditions (e.g., sudden rise of water level on an area with a defined Context resulting in the creation of an ongoing flood event and alert to relevant stakeholders of RCP), could also elevate the value of RCP. This way an authorized user would have a lot more information about the past, present and maybe even the future (i.e., prediction) of an event.

Insertion of new data in an occurring simulation. To leverage 6.2.2. it is necessary to implement a way to introduce new data into an ongoing simulation. This is however, beyond the scope of RCP as it most certainly needs some structural changes to HiSTAV.

Specify RCP use cases in ASL and explore code generation mechanisms for the specification. As explained earlier, RCP was specified in ASL language. However, this specification was only used to generate the data model used to develop RCP. To fully flesh ASL it would be important to reverse engineer RCP and define its use cases in ASL language. This specification should be followed by the exploration of code generation mechanisms to enable the generation of code, from the use cases specification, functionally equivalent to the one developed during this dissertation.

Appendix A: RCP Specifications

```
DataEnumeration SensorKind "Sensor Kind" values ("HydrometricSensor",  
"WeatherSensor", "SocialNetworkScanner", "HumanSensor", "TBD Sensor")  
  
DataEnumeration SensorModalityKind "Sensor Modality Kind" values ("PhysicalFixed",  
"PhysicalMobile", "DigitalSocialNetworkScanner", "DigitalHumanUpload")  
  
DataEnumeration TimeserieType "Timeserie type" values ("Continuous",  
"Discontinuous", "Statistical")  
  
DataEnumeration HydroFeatureKind "HydrometricFeature Kind" values ("River",  
"Estuary", "Lake", "RiverBasin", "DrainageBasin", "Dam")  
  
DataEnumeration SimulationKind "Simulation Kind" values ("Simulation",  
"Scenario")  
  
DataEnumeration InformationKind "Information Kind" values ("Simulation", "Event",  
"Sensor", "Alarm")  
  
DataEnumeration OrganizationKind values ("WaterAuthority", "Municipality",  
"ResearchLab", "Partner", "Other")  
  
DataEnumeration UserState "User State" values ("Suspended", "Active", "Inactive",  
"Deleted")  
  
DataEnumeration AlarmPermission "Alarm Permission" values ("Yes", "Yes (only  
authorities alarms)", "No", "Depend on secondary role")  
  
DataEnumeration MetricKind "Metric" values ("Sec", "Min", "Hour", "Day", "Week",  
"Month", "Year")  
  
DataEnumeration ColourKind "Colour" values ("Red", "Yellow", "Green")  
  
DataEnumeration ContextBoundaryLineKind values ("Input", "Output", "InputOutput")  
  
DataEnumeration ContextBoundaryLineDataKind values ("Depth (H)", "Discharge (Q)",  
"Velocity (V)", "Elevation (Z)")  
  
DataEnumeration EventKind values ("Flood", "HeavyPrecipitation",  
"HydrologicalDrought", "MeteorologicalDrought", "Hurricane", "Tsunami", "Storm",  
"Landslide")  
  
DataEnumeration EventState values ("Announced", "Occurring", "Concluded")
```

Spec. 9. RCP data enumerations.

```

DataEntity e_HydroFeature "Hydrometric feature": Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute code "HydroFeature Code": Regex [constraints (NotNull Unique)]
  attribute Name "Name": String [constraints (NotNull )]
  attribute type "Type": DataEnumeration HydroFeatureKind [constraints (NotNull)]
  attribute area "Area": Decimal
  attribute length "Length": Decimal
  attribute PartOf "PartOf": Integer [constraints(ForeignKey (e_HydroFeature))]
  attribute flowsInto "Flows Into": Integer [constraints (
  ForeignKey(e_HydroFeature))]

  attribute geom "Geometry": GeoPolygon
  constraints (showAs (Name))
]

```

Spec. 10. RCP HydroFeature specification.

```

DataEntity e_Organization "Entity": Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute Name "Name": String [constraints (NotNull Unique)]
  attribute type "Type": DataEnumeration OrganizationKind [constraints (NotNull)]
  attribute sector "Sector": String [constraints (NotNull)]

  attribute address "Address": String [constraints(multiplicity "0..2"
DataEnumeration)]
  attribute city "City": Integer [constraints (ForeignKey(e_City))]
  attribute country "Country": Integer [constraints (NotNull Encrypted
ForeignKey(e_Country))]
  attribute email "Email": Email [constraints (multiplicity "0..2" Encrypted)]
  attribute phone "Telephone": String [constraints (multiplicity "0..2" Encrypted)]

  attribute geom "Geometry": GeoPoint [constraints (NotNull)]
  constraints (showAs (Name))
]

```

Spec. 11. RCP organization specification.

```

DataEntity e_User "User": Master [
  attribute id "ID": Integer [constraints (PrimaryKey)]
  attribute login "Login": String [constraints (NotNull Unique)]
  attribute password "Password": Regex [constraints (NotNull Encrypted Check
(RegexValidationExpression "r'[A-Za-z0-9@#$_]{6,12}'"))]

  attribute Name "Name": Text [constraints (NotNull Encrypted)]
  attribute email "Main Email": Email [constraints (Unique Encrypted)]
  attribute state "State": DataEnumeration UserState [constraints (NotNull)]

  attribute emails "Email address(es)": Email [constraints (multiplicity "0..3"
NotNull Unique Encrypted )]

  attribute phoneNumbers "Telephone number(s)": Regex [constraints (
multiplicity "0..3" NotNull Unique Encrypted Check (RegexValidationExpression
"r'^([0-9]{9})") )]

  constraints (showAs (login))
  tag (name "tenant" value "user")
]

```

Spec. 12. RCP user specification.


```

DataEntity e_SensorAlarm "Sensor Alarm": Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute sensorId "Sensor": Integer [constraints (NotNull ForeignKey
(e_Sensor))]
  attribute Name "Name": String [constraints (NotNull)]
  attribute Action "Action": String
  attribute Description "Description": Text
  attribute colour "Colour": DataEnumeration ColourKind [constraints (NotNull
Derived ("TBD"))]

  attribute redMinthreshold "Low threshold": Decimal [constraints (NotNull)]
  attribute redMaxthreshold "Up threshold": Decimal [constraints (NotNull)]
  attribute yellowMinthreshold "Low threshold": Decimal [constraints (NotNull)]

  attribute yellowMaxthreshold "Up threshold": Decimal [constraints (NotNull)]
  attribute greenMinthreshold "Low threshold": Decimal [constraints (NotNull)]
  attribute greenMaxthreshold "Up threshold": Decimal [constraints (NotNull)]
]

```

Spec. 13. RCP sensor alarm specification.

```

DataEntity e_ContextBoundaryLine: Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute context "Context": Integer [constraints (NotNull
ForeignKey(e_Context))]

  attribute geom: GeoPolyline [constraints (NotNull Check (superimposed "must be
a line superimposed on context.geomExternalBoundary"))]

  attribute type: DataEnumeration ContextBoundaryLineKind [constraints
(NotNull)]

  attribute dataType: DataEnumeration ContextBoundaryLineDataKind

```

Spec. 14. RCP boundary line specification.

```

DataEntity e_ContextBoundaryPointSensor: Master [
  attribute id "Id": Integer [constraints (PrimaryKey)]
  attribute point "Context BoundaryPoint": Integer [constraints (NotNull
ForeignKey(e_ContextBoundaryPoint))]
  attribute Sensor "Sensor": Integer [constraints (NotNull
ForeignKey(e_ContextSensor))]
]

```

Spec. 15. RCP Context boundary point specification.

```

DataEntity e_ContextEvent "Event": Master [
    attribute id "Id": Integer [constraints (PrimaryKey)]
    attribute context "Context": Integer [constraints (NotNull
ForeignKey(e_Context))]

    attribute Name "Name": String [constraints (PrimaryKey)]
    attribute type "Event Type": DataEnumeration EventKind
[constraints (NotNull)]

    attribute subType "Event SubType": DataEnumeration EventSubKind
[constraints (NotNull)]

    attribute state "State": DataEnumeration EventState [constraints (NotNull)]

    attribute startDatetime "Start Event": Datetime [constraints (NotNull)]
    attribute endDatetime "End Event": Datetime [constraints (Check
(ck_datesValidation "endDatetime >= startDatetime"))]

    attribute Description "Description": Text

    // Attributes for "Flood Simulation" event, with HiStav
    attribute returnPeriod "Return Period": Integer
    attribute warmUp "Warm up": Boolean
    attribute simulationType "Simulation Type": DataEnumeration EventKind
[constraints (NotNull)]
]

```

Spec. 16. RCP Context event specification.

```

DataEntity e_ContextAccessRequest "Context Request Access": Parameter [
    attribute context "Context": Integer [constraints (NotNull
ForeignKey(e_Context))]

    attribute access_granted "Access granted": Boolean [defaultValue "False"]
    attribute state "request state": DataEnumeration AccessRequestState
[constraints (NotNull)]
    attribute requestUser "request user": String [constraints (NotNull
ForeignKey(e_User))]
    attribute type "access type": DataEnumeration ContextUserKind [constraints
(NotNull)]

```

Spec. 17. RCP Context access request specification.

Appendix B: RCP Images

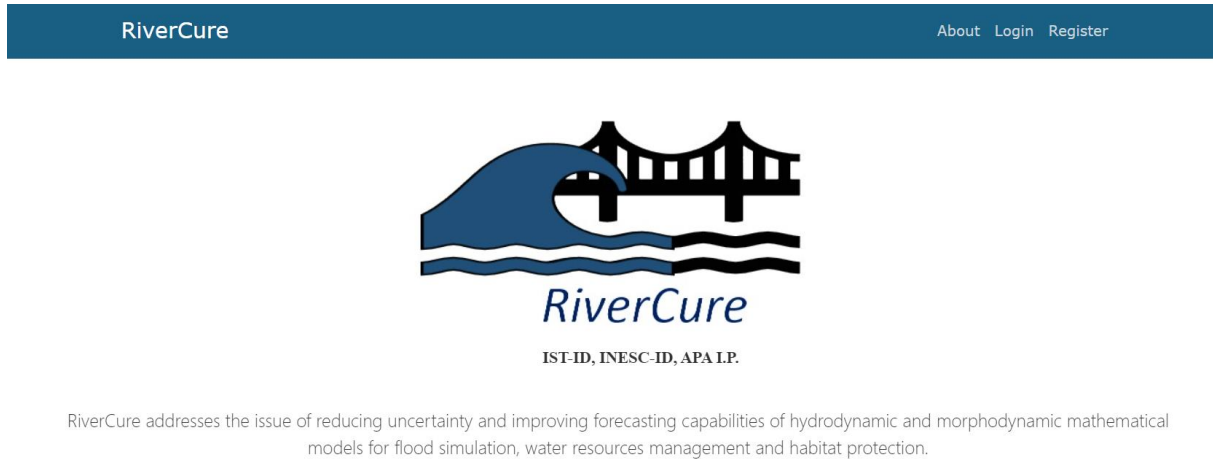


Figure 42. RCP landing page.



Figure 43. RCP about page.

Log In

Username*

Password*

[Log In](#) [Forgot Password?](#)

[Need An Account?](#) [Sign Up Now](#)

Figure 44. RCP login screen.

Join Rivercure

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

[Sign Up](#)

[Already Have An Account?](#) [Sign In](#)

Figure 45. RCP register screen.

RiverCurePortal Monday, 28th December 2020
16:51

Home

Authentication and Authorization

Celery Results

Context

Raster

Rivercureportal

Sensors

Users

Authentication and Authorization

Groups	Change	+ Add
Users	Change	+ Add

Celery Results

Task results	Change	+ Add
--------------	------------------------	-----------------------

Context

E_ context access requests	Change	+ Add
E_ context boundary lines	Change	+ Add
E_ context boundary points	Change	+ Add
E_ context contour lines	Change	+ Add
E_ context dtms	Change	+ Add
E_ context event results	Change	+ Add
E_ context events	Change	+ Add
E_ context sensors	Change	+ Add
E_ context users	Change	+ Add
E_ contexts	Change	+ Add

Raster

Legend entrys	Change	+ Add
Legend semanticss	Change	+ Add

My Actions

- Changed user [RCP_Tester](#)
- Changed e_ context [HiSTAV-test](#)
- Changed e_ context [HiSTAV-test](#)
- Changed e_ context [HiSTAV-test](#)
- Changed user [AS](#)
- Changed e_ context [HiSTAV-test](#)
- Changed user [RuIF](#)
- Changed user [jms](#)
- Changed user [AS](#)
- Changed user [AMR](#)

Figure 46. RCP admin Backoffice.

RiverCure Contexts ▾ Hydrofeatures Sensors About Admin ▾ Profile Logout

Contexts Permissions

User	Context	Role	State	Administer
<i>No pending requests!</i>				

Figure 47. RCP Context access request page.

Hydrofeatures

Name	Type
Águeda_jus	RiverBasin
Águeda_Mont	RiverBasin
Amoreira da Gândara	RiverBasin
Arões	RiverBasin
Avanca	RiverBasin
Avelãs de Caminho	RiverBasin
Bodiosa	RiverBasin
Bunheiro	RiverBasin
Cacia	RiverBasin
Campia (RIBEIRO)	RiverBasin
Esgueira	RiverBasin

Filter

Type

▾

[Filter](#)

[Add Hydrofeature](#)

Figure 48. RCP hydro feature page.

Sensors

Code	Name	Type	Modality	Observations
3	sensor_BH3	HydrometricSensor	PhysicalFixed	
4	sensor_BH4	HydrometricSensor	PhysicalFixed	
5	sensor_mare	HydrometricSensor	PhysicalFixed	
1	sensor_BH1	HydrometricSensor	PhysicalFixed	
2	sensor_BH2	HydrometricSensor	PhysicalFixed	
45	sensor_teste-X	WeatherSensor	PhysicalMobile	
122	new_Test	WeatherSensor	PhysicalMobile	
S_122	Sensor_teste	HydrometricSensor	DigitalHumanUpload	
T1	sensor_BH1	HydrometricSensor	PhysicalFixed	
T2	sensor_BH2	HydrometricSensor	PhysicalFixed	

Filter

Type

▾

Code

ModalityType

▾

[Filter](#)

[Add Sensor](#)

[Upload Sensor](#)

Figure 49. RCP sensors page.

My Contexts

Context	Hydrofeature	Privacy	My Role
my_context_ig	Hydro	Private	Admin
rivercure	None	Public	Admin
NAME	Hydro	Public	Admin
HiSTAV-test	None	Private	Admin
Context-AS-Test3	Amoreira da Gândara	Private	admin viewer manager manager

Filter

Name

HydroFeature

Filter

Add Context **Other Contexts**

Figure 50. RCP Context list page.

Map showing the Iberian Peninsula with two context markers: a yellow circle labeled '10' in northern Spain and a green circle labeled '8' in western Portugal. The map includes standard navigation controls (zoom in/out, full screen) and a scale bar indicating 100 km. Below the map, there are several action buttons: Edit, Download, Events, Sensors, Delete, Generate Mesh, and a Back button.

Figure 51. RCP Context detail page.

Files upload

Domain
 Nenhum ficheiro selecionado

Alignments
 Nenhum ficheiro selecionado

Refinements
 Nenhum ficheiro selecionado

Boundaries
 Nenhum ficheiro selecionado

Dtm file
 Nenhum ficheiro selecionado

Contour lines
 Nenhum ficheiro selecionado

Figure 52. RCP upload Context files prompt.

RiverCure Contexts ▾ Hydrofeatures Sensors About Admin ▾ Profile Logout

Domain 112
Current CL: 20

Refinement 114
Current CL: 15


Alignment 116
Current CL: 3

Alignment 118
Current CL: 3

Select the polygon to define

Context complete stage it!

Figure 53. RCP manage Context page.



admin1

Profile Info

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Institution

Image*

Currently: default.jpg
Change:

Nenhum ficheiro selecionado

First name

Last name

Figure 54. RCP profile page.

References

- [1] "About – RiverCure Project." <https://www.inesc-id.pt/projects/II11041/> (accessed Dec. 10, 2020).
- [2] D. Conde, "High-Performance Modelling of Tsunami Impacts on Built Environments," PhD dissertation, Instituto Superior Tecnico, Universidade de Lisboa, 2018.
- [3] "INESC-ID." <https://www.inesc-id.pt/> (accessed Dec. 27, 2020).
- [4] "APA SNIRH :: Sistema Nacional de Informação de Recursos Hídricos." <https://snirh.apambiente.pt/index.php?idMain=5&idItem=5> (accessed Dec. 07, 2020).
- [5] R. Gomes, M. Saramago, and R. Rodrigues, "Svarh – Sistema de Vigilância e Alerta de Recursos Hídricos," Technical Report, Instituto da Água, Lisboa, 2003.
- [6] D. Conde, M. Telhado, M. Viana Baptista, and R. Ferreira, "Severity and exposure associated with tsunami actions in urban waterfronts: the case of Lisbon, Portugal," *Natural Hazards*, vol. 79, no. 3, pp. 2125–2144, 2015, doi: 10.1007/s11069-015-1951-z.
- [7] P. H. Carstensen and L. Vogelsang, "Design of Web-Based Information Systems - New Challenges for Systems Development?," in *Proceedings of Ecis*, 2001, pp. 536–547.
- [8] A. R. da Silva, "ITLingo Research Initiative," Technical Report, Instituto Superior Técnico, Universidade de Lisboa, 2017.
- [9] I. Gamito and A. R. da Silva, "From rigorous requirements and user interfaces specifications into software business applications," *Communications in Computer and Information Science*, vol. 1266 CCIS, pp. 459–473, 2020, doi: 10.1007/978-3-030-58793-2_37.
- [10] D. Ince, *Object oriented design with applications*, 2nd ed., vol. 34, no. 9. Addison Wesley Longman, Inc., 1992.
- [11] A. R. da Silva, "Model-driven engineering: A survey supported by the unified conceptual model," *Computer Languages, Systems and Structures, Elsevier*, vol. 43, pp. 139–155, 2015, doi: 10.1016/j.cl.2015.06.001.
- [12] "OGC." <https://www.ogc.org/> (accessed Dec. 10, 2020).
- [13] C. OGC, "OGC WaterML 2.0: Part 1- Timeseries," 2011. [Online]. Available: <http://www.opengeospatial.org/>.
- [14] "REST API Tutorial." <https://restfulapi.net/> (accessed Dec. 10, 2020).
- [15] A. Hevner, S. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004, doi: 10.1007/BF01205282.
- [16] Ken Peppers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management*

Information Systems, vol. 24, no. 3, pp. 45–77, 2007.

- [17] F. Gacenga, A. Cater-Steel, M. Toleman, and W. Tan, “A proposal and evaluation of a design method in design science research,” *Electronic Journal of Business Research Methods*, no. 10, pp. 89–100, 2012.
- [18] “The Web framework for perfectionists with deadlines | Django.” <https://www.djangoproject.com/> (accessed Dec. 10, 2020).
- [19] “Welcome to Flask — Flask Documentation (1.1.x).” <https://flask.palletsprojects.com/en/1.1.x/foreword/> (accessed Dec. 10, 2020).
- [20] “Agência Portuguesa do Ambiente.” <https://www.apambiente.pt/> (accessed Oct. 19, 2020).
- [21] G. Early, “Celebrating knowledge,” *Proceedings of the American Philosophical Society*, vol. 151, p. 74, 2007.
- [22] R. France and B. Rumpe, “Model-driven development of complex software: A research roadmap,” *International Conference on Model-Driven Engineering and Software Development*, pp. 37–54, 2007, doi: 10.1109/FOSE.2007.14.
- [23] A. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture : Practice and Promise*. 2003.
- [24] M. Voelter, *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. CreateSpace Independent Publishing Platform (January 23, 2013), 2013.
- [25] S. P. Christodoulou and T. S. Papatheodorou, “Web Engineering Resources Portal (WEP): A Reference Model and Guide,” *Web Engineering Principles and Techniques*, pp. 31–74, 2005.
- [26] J. G. Zheng, “A Historical Perspective of Web Engineering,” *Encyclopedia of Networked and Virtual Organizations*. IGI Global, pp. 660–667, 2008, [Online]. Available: <http://www.igi-global.com/chapter/historical-perspective-web-engineering/17673>.
- [27] “IFML: The Interaction Flow Modeling Language | The OMG standard for front-end design.” <https://www.ifml.org/> (accessed Dec. 12, 2019).
- [28] M. Brambilla and P. Fraternali, “Implementation of applications specified with IFML,” *Interaction Flow Modeling Languages*, pp. 279–334, 2015, doi: 10.1016/b978-0-12-800108-0.00010-2.
- [29] R. Pereira, “Técnicas Avançadas de Modelação e Produção Semi-Automática de Aplicações Web Responsivas,” MSc dissertation, Instituto Superior Técnico, Universidade de Lisboa, 2019.
- [30] I. Gamito, “From Rigorous Requirements and User Interfaces Specifications into Software Business Applications: The ASL Approach,” MSc dissertation, Instituto Superior Técnico, Universidade de Lisboa, 2020.
- [31] A. R. da Silva, A. Paiva, and V. Silva, “Towards a test specification language for information

- systems: Focus on data entity and state machine tests,” *Model. 2018 - Proc. 6th International Conference on Model-Driven Engineering and Software Development*, pp. 213–224, 2018, doi: 10.5220/0006608002130224.
- [32] A. R. da Silva, “Rigorous Requirements Specification with the RSL Language: Focus on Uses Cases,” in *ISD’2019, AIS*, 2018, pp. 1–32.
- [33] “Xtext - Language Engineering Made Easy!” <https://www.eclipse.org/Xtext/> (accessed Dec. 10, 2020).
- [34] “OGC Standards.” <https://www.ogc.org/standards> (accessed Dec. 10, 2020).
- [35] OGC, *OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture*. 2011, p. 93.
- [36] “GeoDjango | Django documentation | Django.” <https://docs.djangoproject.com/en/3.0/ref/contrib/gis/> (accessed Dec. 04, 2020).
- [37] “GeoDjango Model API | Django documentation | Django.” <https://docs.djangoproject.com/en/3.0/ref/contrib/gis/model-api/> (accessed Dec. 04, 2020).
- [38] “GDAL API | Django documentation | Django.” <https://docs.djangoproject.com/en/3.1/ref/contrib/gis/gdal/> (accessed Dec. 10, 2020).
- [39] “OGC® WaterML | OGC.” <https://www.ogc.org/standards/waterml#overview> (accessed Oct. 27, 2020).
- [40] “Observations and Measurements | OGC.” <https://www.ogc.org/standards/om> (accessed Oct. 27, 2020).
- [41] “Coordinate Reference Systems — QGIS Documentation documentation.” https://docs.qgis.org/3.16/en/docs/gentle_gis_introduction/coordinate_reference_systems.html (accessed Dec. 07, 2020).
- [42] “Coordinate Reference System and Spatial Projection | Earth Data Science - Earth Lab.” <https://www.earthdatascience.org/courses/earth-analytics/spatial-data-r/intro-to-coordinate-reference-systems/> (accessed Oct. 28, 2020).
- [43] “EPSG Geodetic Parameter Dataset.” <https://epsg.org/home.html> (accessed Oct. 29, 2020).
- [44] “Leaflet - a JavaScript library for interactive maps.” <https://leafletjs.com/> (accessed Dec. 10, 2020).
- [45] “Spatial Reference List -- Spatial Reference.” <https://spatialreference.org/ref/epsg/> (accessed Dec. 29, 2020).
- [46] “Types of GIS Data Explored: Vector and Raster - GIS Lounge.” <https://www.gislounge.com/geodatabases-explored-vector-and-raster-data/> (accessed Dec. 10,

- 2020).
- [47] “Vector Data.” https://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/vector_data.html (accessed Dec. 10, 2020).
- [48] “GeoJSON.” <https://geojson.org/> (accessed Oct. 16, 2020).
- [49] “What is a shapefile?—Help | ArcGIS for Desktop.” <https://desktop.arcgis.com/en/arcmap/10.3/manage-data/shapefiles/what-is-a-shapefile.htm> (accessed Oct. 20, 2020).
- [50] “What is raster data?—Help | ArcGIS for Desktop.” <https://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/what-is-raster-data.htm> (accessed Oct. 28, 2020).
- [51] “Working with Mesh Data — QGIS Documentation documentation.” https://docs.qgis.org/3.10/en/docs/user_manual/working_with_mesh/mesh_properties.html#what-s-a-mesh (accessed Oct. 28, 2020).
- [52] “10 Best Python Books for Beginners & Advanced Programmers.” <https://hackr.io/blog/best-python-books-for-beginners-and-advanced-programmers> (accessed Dec. 10, 2020).
- [53] “MVC Framework - Introduction - Tutorialspoint.” https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (accessed Dec. 10, 2020).
- [54] D. Rubio, *Beginning Django*. Apress, 2017.
- [55] “GeoDjango Tutorial | Django documentation | Django.” <https://docs.djangoproject.com/en/3.0/ref/contrib/gis/tutorial/#introduction> (accessed Dec. 10, 2020).
- [56] W. S. Vincent, *Django for Beginners*. 2018.
- [57] “Making queries | Django documentation | Django.” <https://docs.djangoproject.com/en/3.1/topics/db/queries/> (accessed Dec. 04, 2020).
- [58] “Django ORM Cookbook — Django ORM Cookbook 2.0 documentation.” <https://books.agiliq.com/projects/django-orm-cookbook/en/latest/> (accessed Dec. 04, 2020).
- [59] “GeoDjango Database API | Django documentation | Django.” <https://docs.djangoproject.com/en/3.0/ref/contrib/gis/db-api/> (accessed Dec. 04, 2020).
- [60] “GDAL-2.” <https://gdal.org/> (accessed Dec. 10, 2020).
- [61] “PROJ - PROJ 7.2.0 documentation.” <https://proj.org/> (accessed Dec. 10, 2020).
- [62] “PostgreSQL: The world’s most advanced open source database.” <https://www.postgresql.org/> (accessed Dec. 04, 2020).

- [63] “PostGIS — Spatial and Geographic Objects for PostgreSQL.” <https://postgis.net/> (accessed Oct. 19, 2020).
- [64] “GEOS.” <https://trac.osgeo.org/geos/> (accessed Dec. 10, 2020).
- [65] “JTS Topology Suite.” <https://sourceforge.net/projects/jts-topo-suite/> (accessed Dec. 10, 2020).
- [66] “About OSGeo - OSGeo.” <https://www.osgeo.org/about/> (accessed Dec. 10, 2020).
- [67] “GEOS API | Django documentation | Django.” <https://docs.djangoproject.com/en/3.1/ref/contrib/gis/geos/> (accessed Dec. 10, 2020).
- [68] “Layer Groups and Layers Control - Leaflet - a JavaScript library for interactive maps.” <https://leafletjs.com/examples/layers-control/> (accessed Dec. 04, 2020).
- [69] “Django-Leaflet.” <https://github.com/makinacorp/django-leaflet> (accessed Dec. 10, 2020).
- [70] “ParaView.” <https://www.paraview.org/> (accessed Dec. 04, 2020).
- [71] “Documentation | Visualizer.” <https://kitware.github.io/visualizer/docs/> (accessed Oct. 12, 2020).
- [72] “VTK File Extension - What is it? How to open a VTK file?” <https://filext.com/file-extension/VTK> (accessed Dec. 04, 2020).
- [73] “File Formats for VTK Version 4.2 VTK File Formats.” Accessed: Dec. 04, 2020. [Online]. Available: www.kitware.com.
- [74] “MIKE 2020.” <https://www.mikepoweredbydhi.com/mike-2020> (accessed Nov. 01, 2020).
- [75] “MIKE 21.” <https://www.mikepoweredbydhi.com/products/mike-21> (accessed Nov. 01, 2020).
- [76] M. Saramago, “Redes de Monitorização Hidrometeorológicas,” *Revista Recursos Hídricos, APRH*, vol. 38, no. 1, pp. 33–39, 2017.
- [77] M. Saramago, “Hydrologic Surveillance System Using Wireless Technologies,” 2006, [Online]. Available: http://projects.knmi.nl/geoss/ICEAWS/ICEAWS-4/CD/docs/ORAL/17_oral.pdf.
- [78] “Mike 21 Flow Model & Mike 21 Flood Screening Tool.” DHI, Horsholm, 2017, Accessed: Dec. 08, 2020. [Online]. Available: www.mikepoweredbydhi.com.
- [79] DHI, “Mike 21 Hd.” DHI, 2011.
- [80] DHI, “MIKE 21 Flow Model User Manual.” DHI, p. 120, 2017.
- [81] M. Gonzalez, “RiverCure Portal : Collaborative GeoPortal for Curatorship of Digital Resources in the Water Management Domain,” Instituto Superior Tecnico, Universidade de Lisboa, 2020.
- [82] “Digital Terrain Model — European Environment Agency.” <https://www.eea.europa.eu/help/glossary/eea-glossary/digital-terrain-model>.
- [83] J. Pereira, “Classifying Geo-Referenced Photos and Segmenting Satellite Imagery for the

Assessment of Flood Severity,” Instituto Superior Técnico, Universidade de Lisboa, 2019.

- [84] “Django Raster — Django Raster 0.3 documentation.” <https://django-raster.readthedocs.io/en/latest/> (accessed Oct. 26, 2020).
- [85] “curl - Documentation Overview.” <https://curl.haxx.se/> (accessed Dec. 10, 2020).
- [86] “ArcGIS Desktop | Documentation.” <https://desktop.arcgis.com/en/> (accessed Dec. 27, 2020).
- [87] “Discover QGIS.” <https://www.qgis.org/en/site/about/index.html> (accessed Dec. 10, 2020).

