Environmental Modelling & Software 100 (2018) 33–47

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

A Web Service ecosystem for high-quality, cost-effective debris-flow hazard assessment

Giorgio Rosatti [a, *], Nadia Zorzi [a], Daniel Zugliani [a], Stefano Piffer [b], Alessandro Rizzi [b, 1]

[a] Department of Civil, Environmental and Mechanical Engineering, University of Trento, Trento, Italy
[b] Trilogis srl, Trento, Italy

ARTICLE INFO

Article history:
Received 14 December 2016
Received in revised form
25 October 2017
Accepted 8 November 2017

Keywords:
Debris-flow hazard assessment
TRENT2D
WEEZARD
WS
Cloud computing

ABSTRACT

In this work, we present WEEZARD, a smart, cost-effective system, aimed at producing, managing and analysing high-quality debris-flow simulations to be used in hazard assessments. Developing the system was a challenge because of the complexity and the number of interconnected operations involved in such assessments. The goal was achieved by using a web client-server system, where a set of web services are exposed in a way similar to that used in OGC ® WPS solutions. The new system is composed of a previously developed advanced two-phase debris-flow model (TRENT2D), re-coded as a web service, and all the functionalities necessary to exploit the model, from the management of input and output (also provided as web services) to GIS features and two- and three-dimensional dynamic visualizations. Future developments include the possibility of using approaches such as the "Model as a Service" to further increase the accessibility and interoperability of the TRENT2D model.

1. Introduction

Among geomorphic processes, debris flows are a common phenomenon in mountain regions. From a physical point of view, they are gravity-driven flows, composed of a mixture of a solid phase (sediments) and a liquid phase (water, commonly the result of heavy rainfall). Because of urbanization, these phenomena often affect buildings and infrastructure, causing severe damage and possibly casualties (see Fig. 1). Moreover, climate change seems to have induced an increase in extreme precipitation and discharge (IPCC, 2014), possibly making debris flows both larger and more frequent (see e.g. Stoffel et al., 2014).

Reducing the risk of adverse consequences, according to the UE Flood Directive (UE, 2007), *is feasible and desirable* by using *appropriate best practice* and *the best available technology.* The cited regulation goes on with an important indication concerning technology: *not entailing excessive costs.* A similar concept is expressed in the USA Disaster Mitigation Act (DMA, 2000), which states that multihazard advisory maps must be developed with *the most cost-effective and efficient technology available.*

Compared to other techniques (checklists, statistics, etc.), computer simulations are undoubtedly the most advanced technology available for what we define here as "debris-flow simulation jobs", consisting in tasks such as understanding the dynamics of past events through a back analysis, anticipating scenarios for hazard assessment, hazard or risk mapping and planning mitigation works. However, high-quality reliable simulations can be obtained only from advanced research models that consider the physical complexity of the phenomenon as much as possible and by using suitable high-resolution computational domains. Unfortunately, these models impose a significant computational burden and, consequently, require an adequate hardware resource, which is not always available, for example, to practitioners working in this field.

Moreover, as presented in detail in Section 2.2, running a simulation is only part of a job. In fact, many other tasks are required, concerning essentially pre- and post-processing of simulation data. Often, each task must be performed using different software, resulting in a high fragmentation of the work and a significant and expensive need for manual work.

Summarizing, the use of advanced software is not an easy matter and seldom a cheap operation. Consequently, rough or very simplified models are commonly used in practical jobs, leading to possible unreliable results in forecasting situations, such as hazard mapping.

* Corresponding author.
E-mail addresses: giorgio.rosatti@unitn.it (G. Rosatti), nadia.zorzi@unitn.it (N. Zorzi), daniel.zugliani@unitn.it (D. Zugliani), stefano.piffer@trilogis.it (S. Piffer), alessandro.rizzi@trilogis.it (A. Rizzi).
[1] Current address: Tecnobit srl, Predaia - Trento, Italy.

https://doi.org/10.1016/j.envsoft.2017.11.017
1364-8152/

Fig. 1. An example of (a) a debris-flow outcome and (b) of the enormous damage that this type of flow can produce. (Photo (a): courtesy of Autonomous Province of Trento, Italy).

The main goal of our work is to incorporate an advanced model into a practical and affordable tool for debris flow hazard assessment. In this paper, we describe the development of a complete platform based on the "Software as a Service" (SaaS) paradigm. The result, although based on consolidated Information Technologies, is an innovative system in the field of debris-flow modelling for hazard assessment, and allows possible interesting developments.

In detail, in Section 2 we present an analysis of who may be interested in debris-flow modelling (i.e. all the possible classes of users) and what must be done (i.e. which work elements are necessary) to perform a numerical study of the phenomenon. Then, we survey the existing integrated solutions that are able to chain the operations needed for a debris-flow job (Section 3). Section 4 shows the fundamental features of the TRENT2D model (Armanini et al., 2009; Rosatti and Begnudelli, 2013b), a high-quality two-phase numerical model for debris flows, developed before this work by some of the present authors, around which we developed our innovative system. In Section 5, we present the architecture of WEEZARD (acronym for WEbgis modElling and haZard Assessment for mountain flows: an integRated system in the clouD, available at http://tool.weezard.eu), an integrated solution tailored to the requirements of practitioners and land protection agency technicians, while in Section 6 the main WEEZARD functionalities are shown. A discussion of the limits of the system at the current state of development and the possible future improvements is presented in Section 7. Our conclusions end the paper.

2. Debris-flow modelling: who and what

In order to achieve the established goal, it is necessary to have a clear outline of the problem we are dealing with. More specifically, it is important both to single out the potential interested parties in modelling the phenomenon and their specific requirements, and to list the components of a modelling job that are necessary in hazard assessment.

2.1. The potential interested parties and their main requirements

Interested parties can be divided into two large groups: people who perform simulations and people who use the results of the simulations.

People belonging to the first group are scientists, engineers, technicians and scholars who perform simulations aimed at studying the basic dynamics of debris-flows, back analyses of real events, outcomes of possible scenarios and the effectiveness of planned mitigation works. This group of people carries out a lot of simulations and therefore they need methods and tools to set up and perform a simulation in an easy and efficient way by means of a suitable concatenation of the required operations. Moreover, they need tools for in-depth analysis of the dynamical behaviour of the flow, such as visualization of scalar and vector fields, sections and time animations. Finally, they are often asked to provide direct and intuitive representations of the simulated phenomena, such as georeferenced static and dynamic maps or three-dimensional realistic representations. It is worth noting that while for most of the possible interested parties it is desirable to use a single environment to perform all the required work, there could be a community of scientists interested in using the bare numerical model independently of the rest of the system, and incorporating this model into their own research framework.

On the other hand, decision makers, land planners, communication experts and educated citizens do not perform simulations but may need to access and possibly manage information produced by debris-flow simulations (for example, static and dynamic maps or videos) quickly and easily.

2.2. The many components of a debris-flow modelling job

As we have already pointed out, modelling requires a large number of operations beyond the numerical simulations. By "component", we mean one of the logical tasks (or logical element) into which a debris-flow modelling job can be decomposed that is completed by using specific software. In this section, we briefly describe the four components that we consider most important and that may produce work fragmentation if performed with software not designed as part of a complex chain of operations.

2.2.1. Mathematical and numerical modelling

The first component is obviously the model. In order to have an acceptable description of the physics of debris flows, it is necessary to use at least a two-dimensional, depth-averaged, shallow flow approach. The resulting mathematical model is composed of a system of Partial Differential Equations (PDEs). However, quite different 2D models based on this premise are present in the literature. They differ in the basic assumptions regarding the nature of the flow (monophasic or biphasic), in the type of bed over which the flow occurs (fixed- or mobile-bed), and in the type of closure relations describing the bed evolution, the concentration of sediments and the bed shear stresses (see Iverson and Ouyang, 2015 for an overview). Different choices give rise to models that show different mathematical complexity and different physical reliability.

Also from a numerical point of view, considerably different approaches can be used. Finite difference schemes are widely employed (see e.g. Wu et al., 2013; Wang et al., 2008), but since debris flows are highly impulsive phenomena that generate shocks, more adequate shock-capturing methods are often used (see e.g. Fraccarollo et al., 2003; Murillo and García-Navarro, 2010; Pelanti et al., 2011). However, these last schemes imply a greater computational burden than the finite differences schemes.

Rough or very simplified models can be obtained from the complete PDE system of a model by substituting some (see e.g. Gregoretti et al., 2016) or all (see e.g. Mergili and Fellin, 2007; Scheidl and Rickenmann, 2011) of the differential equations with algebraic approximated relations, producing so-called cellular automata-like models. Because of their simplicity, they do not require a complex numerical scheme and, consequently, the computational burden is very limited. These models can work well in back analysis, where a careful calibration of the model parameters can be performed. Nevertheless, they can lead to possible unreliable results whenever they are used in forecasting situations, i.e. when an a priori parameter estimate is not easily achievable.

Another important factor is the computational domain resolution. In order to avoid inaccurate simulations, resolutions of less than a meter should be preferred, since only this dimensional scale adequately describes the terrain features that can affect the debris flow dynamics. Obviously, since the higher the resolution, the higher the computational load is, an adequate trade-off between resolution and computational burden must be chosen.

To summarize, different modelling choices significantly affect not only the quality of simulation results but also the related computational costs. This last element, related to the available hardware resources, is actually the major issue in developing an integrated solution, as will be clarified in Section 3.

2.2.2. Data management

Time-dependent numerical models are commonly conceived to read input data and to produce many output data at given time intervals. In desktop applications, normally, input files and output paths are chosen by the user at the beginning of each simulation. Without a rigid strategy for data management, this degree of freedom may produce handling difficulties as the number of performed simulations increases, with possible data mess and wasted time in data finding. A natural solution is to use a registry (database-based or file-based) that must be fed manually at the end of each simulation. However, this solution does not guarantee data matching as files can be moved or deleted without automatic updating of the database.

2.2.3. Geospatial data

In a numerical debris-flow model, almost all input and output data have a geospatial nature. In fact, the computational domain is a discretization of a portion of terrain over which a flow is expected to occur. Since model variables are defined somewhere inside a computational cell, variable values are actually geospatially distributed fields. Sometimes, the domain consists in a regular Cartesian grid oriented in any direction with respect to the common Global Coordinate Systems (GCSs). In these cases, a Local Coordinate System (LCS) is used for localization inside the domain.

A basic input for the model is the initial condition of the physical system, which consists in a Digital Terrain Model (DTM) defined on the computational domain. This piece of information can be retrieved from existing DTMs obtained with techniques such as photogrammetry, LIDAR, InSAR or land surveying. However, several modifications (e.g. merging of different DTMs, clipping, interpolating or changing resolution) must often be performed on these DTMs in order to produce the required input. Other geospatial data

manipulations are necessary, for example, when original DTMs must be modified in order to account for planned mitigation works, or when outputs deriving from different scenarios must be combined to produce a hazard map.

Other input data concern geospatial information. They can be both distributed data, such as the bed roughness, or localized data, such as the position of the input sections in the computational domain.

Outputs consist essentially in time series of the spatially distributed variables of the model, including flow depth, velocity and bed elevation variations. If the model uses an LCS, a georeferencing procedure must be employed in order to analyse outputs in a GCS.

Geospatial handling is commonly performed by using a desktop Geographic Information System (GIS). However, systematic treatment of large amounts of data may not be fully automated, and a significant amount of work must be done manually.

2.2.4. Dynamic visualization

As already noted, debris flows are highly unsteady processes, in which not only the final situation but also the dynamics of the event are important. Therefore, it is essential to be able to visualize output data in a dynamic way in both 2D and 3D views. There are several desktop programs able to do this task extremely well. However, efficient loading of large amounts of data occurs only with specific software-dependent file formats and this may require lengthy file conversions. Moreover, these software products are not designed to manage geospatial, multi-layer data in an easy way and some important features, such as draping an orthophoto over a DTM in a 3D view, are not available. In contrast, this last feature is commonly available in a GIS, but dynamic visualizations are not. In other words, there is no software able to provide all the features required for a comprehensive analysis, and therefore two different kinds of software must be employed. This often also implies duplication of data in different file formats, with consequent waste of time and storage space.

3. Integrated solutions

In the previous section we have presented a list of tasks/elements that produce fragmentation during a debris-flow job. It is also evident that, to be autonomous in the execution of a job, the users must own and know different tools/applications (i.e. CAD tool, GIS knowledge, etc.) and be aware of the working context in which one must act. Finding a way to chain all the required operations in an efficient way is therefore mandatory in order to reduce the production times and the overall costs.

Integration among components is a problem that can be dealt with in different ways (see e.g. Belete et al., 2017; for a general overview). In the next sections, we introduce the most widely used solutions in the literature on hydraulic-based phenomena, along with their strengths and weaknesses.

3.1. Embedded desktop solutions

In an embedded desktop solution, one component becomes the main desktop environment, while all the others are included in it. In this way, all the components are tightly connected and communication is guaranteed by specific routines. Either the numerical model or a GIS system can be chosen as main environment. In the first case, the numerical model must be equipped with all the required graphical features, while in the second case, the numerical model becomes a particular library function of the GIS. Using a GIS system as the main environment is generally the preferred option as it can be obtained using an existing system and has a development effort far smaller than a numerical-model-based environment, where the possibility of using of existing libraries is instead more limited. In the literature, there are several examples of GIS-based approaches (see e.g. Mergili and Fellin, 2007; Wu et al., 2013; Scheidl and Rickenmann, 2011; Wang et al., 2008; Gregoretti et al., 2016), not only in the field of debris flow but also regarding dam-breaks (see e.g. Cannata and Marzocchi, 2012).

The major limitation of an embedded solution derives precisely from its desktop nature. A GIS-based application is limited by the hardware resources commonly available among practitioners, resources that are adequate to perform classic GIS operations but often inadequate for heavy computing. For this reason, all the models that have been embedded in a GIS so far are cellular automata-like models. Finally, as already noted in the previous section, GIS environments do not guarantee an adequate visualization of the output.

3.2. Tightly coupled desktop solutions

In this approach, all the components involving graphical elements are merged together in a single desktop Graphical User Interface (GUI) framework detached from the numerical model, which remains a stand-alone component running in a separate desktop console environment (see e.g. Christen et al., 2012). Although they are separate (or decoupled), a strict coupling between the components actually exists, since the GUI works only for that specific numerical model. Finally, communications commonly occur through exchange of data files.

Separation of environments is at the same time the strongest and weakest point: the complexity of the model does not affect the GUI but communications are not optimized at all. Moreover, this approach is limited by the locally available resources and by the lack of a database management system.

3.3. Loosely coupled solutions

In a loosely coupled solution, each of the components has, or makes use of, separate components communicating through suitable interfaces. One of the most diffused approaches is the Service Oriented Architecture (SOA) (see e.g. Jones, 2005; Shan and Hua, 2006). It can be used to build a wide range of solutions that, depending on the degree of interoperability, can range from closed (or essentially closed) to completely-open systems. Closed solutions need to provide all the functionality to perform a given job in a single environment (all-in-one solutions) where the web is used to connect distributed components (client-server solutions) or distributed business modules (Web as a Platform solutions) via customized interfaces. Customized interfaces guarantee high-speed communications but limit the interoperability with other systems. In contrast, completely-open solutions maximize the looseness and the exploitation of the potentiality of the web, allowing a component to be accessed through standard interfaces. When the component is a numerical model, a Model Web approach can be followed as proposed in Nativi et al. (2013). This approach, essentially based on the Model as a Service paradigm, is useful when there is a large community that is interested in using only the model, and incorporating it into their own framework (model chaining). Once again, the peculiarity of the approach, in this case offering the model as an independent component, is at the same time an advantage and a limitation, because the use of completely general interfaces make communications slower and the GUI less intuitive than in closed solutions. Finally, one of the greatest limitations of SOA solutions compared to desktop solutions is the amount of development effort that SOA solutions require.

In the literature, there are several examples of SOA solutions for

geospatial data and some have also been equipped with a mathematical model (see e.g. Jia et al., 2009; Kulkarni et al., 2014; Yin et al., 2015; Mure-Ravaud et al., 2016; Tan et al., 2016). However, a solution specifically designed to deal with complex numerical simulations, and particularly in the field of debris flows, is still lacking.

4. The TRENT2D model

As briefly mentioned in the Introduction, high-quality debris-flow simulations can be performed using the TRENT2D model. Since in this work we want to build a smart integrated solution around this building block, we recall here the basic features of the model that are useful to understand the choices we have made in designing the new system (see next section), referring the reader to the original papers for a more detailed description.

4.1. The mathematical formulation

The mathematical model describes the debris flow as a two-phase mixture flowing over a mobile bed. It consists in a set of PDEs expressing the depth-averaged mass and momentum balance laws of each phase under the isokinetic hypothesis, i.e. sediments and water have the same velocity. Because of the high values of sediment concentration reached in debris flows, the bed dynamics is fully coupled with the description of the mixture flow (Garegnani et al., 2013). It is worth noting that this model does not imply a cellular automata-like approach but instead is a complete model.

More specifically, the PDE system comprises the mass balance of the mixture (Equation (1a)), the mass balance of the solid phase (Equation (1b)) and the momentum balance along the x (Equation (1c)) and y (Equation (1d)) directions:

$$\frac{\partial}{\partial t}(z_b + h) + \frac{\partial}{\partial x}(hu_x) + \frac{\partial}{\partial y}(hu_y) = 0 \tag{1a}$$

$$\frac{\partial}{\partial t}(c_b z_b + ch) + \frac{\partial}{\partial x}(chu_x) + \frac{\partial}{\partial y}(chu_y) = 0 \tag{1b}$$

$$\frac{\partial}{\partial t}(\delta hu_x) + \frac{\partial}{\partial x}\left[\delta\left(hu_x^2 + \frac{1}{2}gh^2\right)\right] + \frac{\partial}{\partial y}(\delta hu_x u_y) + \delta gh\frac{\partial z_b}{\partial x} = -\frac{\tau_{bx}}{\rho_w} \tag{1c}$$

$$\frac{\partial}{\partial t}(\delta hu_y) + \frac{\partial}{\partial x}(\delta hu_x u_y) + \frac{\partial}{\partial y}\left[\delta\left(hu_y^2 + \frac{1}{2}gh^2\right)\right] + \delta gh\frac{\partial z_b}{\partial y} = -\frac{\tau_{by}}{\rho_w}. \tag{1d}$$

In this system the unknowns are: the mobile-bed elevation z_b, the flow depth h and the depth-averaged velocity $\vec{u} = (u_x, u_y)$. The other quantities are model parameters or given functions of the unknowns. Namely, c_b is the maximum packing concentration of the bed solid material and c is the depth-averaged concentration, expressed as a function of the local hydrodynamic variables,

$$c = c_b\beta\frac{\|\vec{u}\|^2}{gh}$$

where β is a dimensionless transport parameter, as proposed in Rosatti and Fraccarollo (2006). $\delta = 1 + c\Delta$, where $\Delta = (\rho_s - \rho_w)/\rho_w$, ρ_s is the density of the solid phase and ρ_w is the density of the liquid phase. g is the gravitational acceleration and τ_{bx} and τ_{by} are the x and y components of the bed shear stress ,$\vec{\tau}_b$ described through the relation for the grain-inertial regime proposed by Bagnold (1954) and modified by Takahashi (1978):

$$\vec{\tau}_b = \frac{25}{4}a\rho_s\sin\phi\frac{\lambda^2}{Y^2}\|\vec{u}\|\vec{u}$$

where:

$$\lambda = \left[(c_b/c)^{1/3} - 1\right]^{-1}$$

and $a = 0.32$ according to Takahashi (1978), ϕ is the dynamic friction angle of the solid phase and finally $Y = h/d$ is the submergence of the sediments with diameter d, assumed to be a constant model parameter.

The physical complexity of the debris-flow phenomenon is well represented by the complexity of the PDEs system consisting of a highly nonlinear, partially nonconservative set of hyperbolic equations. Suitable numerical methods are needed to solve the system accurately, since its nature allows the development of shock waves characterized by Generalized Rankine-Hugoniot relations (see Rosatti and Fraccarollo, 2006; Rosatti and Begnudelli, 2010; for details).

4.2. Other significant features

Some other features are useful to understand the framework that must be managed by the new solution.

The numerical method: the system of equations is integrated over a regular Cartesian mesh by means of an explicit, finite-volume method with Godunov-type fluxes. Accuracy is second order in space and time, thanks to a MUSCL-Hancock approach (see e.g. Toro, 2009). Because of the presence of non-conservative fluxes, specifically designed solvers must be employed in order to avoid large errors. In particular, two different approaches are available, the LHLL method (Fraccarollo et al., 2003; Armanini et al., 2009), and the Generalized Roe solvers (Rosatti et al., 2008; Rosatti and Begnudelli, 2013a, b), both considering the nonconservative term in the solution of the Riemann problem.

Input and output information: the first input concerns the computational domain. It consists in a Cartesian grid with given cell size and number of cells along the two sides, sufficient to include the extent of the simulated phenomenon. Over this mesh, initial and boundary conditions must then be provided. As the initial condition, the terrain elevation before the debris-flow event is required, while as boundary conditions, it is necessary to give both geometric information, i.e. the indexes of the cells comprising the inflow section, and dynamic information, i.e. a suitable solid and liquid hydrograph obtained either from measurements or by using a rainfall-runoff model. Finally, model parameters and a reference slope across the input section must be supplied. As output, the model periodically produces a series of data, in which the grid values of the computed variables are stored in matrix form.

Computational burden: the code was implemented in Fortran 90 and fully parallelized with OpenMP. Despite the use advanced optimizations (including processor specific optimizations), the computational burden is commonly not negligible. As an example, a simulation of a 3 h debris-flow occurrence in a conoid in the Alpine region, involving a computational domain of 800×816 square cells with side of 1 m, takes 130 min on a workstation equipped with two Intel four-core XEON X5677 processors @ 3.47 GHz with a RAM usage of 470Mb.

5. The WEEZARD solution

The choice of the best technology for a system depends

essentially on the requirements of the targeted interested parties. In the present work, we have addressed our effort principally to promoting the use of advanced debris-flow modelling among practitioners and land protection agency technicians, who require essentially a system with a single environment, as pointed out in Section 2.1. Therefore, the most convenient solution seemed to be a partially-open Web Service (WS) ecosystem, in which decoupled components use well-defined (and in future, standard) service interfaces according to a service-oriented methodology. This choice allows the development of highly customized and efficient environments.

Following on from this choice, WEEZARD was developed according to a multi-tier approach, i.e. with different software components logically grouped together and physically placed on different computers connected by the web. Some of the components, especially those used in WEEZARD for dealing with geographic information, are based on Free Open Source Software (FOSS), since it guarantees a good compromise among quality, security and costs. All the other components were built specifically for our system. The whole solution was developed according to an iterative and incremental method (Beedle et al., 2001), with recurrent planning, implementation, testing and evaluation, followed by periodic update releases. This cyclic approach provides a made-to-measure solution for modelling purposes.

The following sections present details of both the logical and the physical architecture of the system.

5.1. The logical architecture

As shown in Fig. 2, the system is organized in three layers, namely the Application layer, the Middleware layer and the Data layer, whose functions and software implementations are described below.

Application layer: the application layer is essentially identifiable with the web application of the system and looks like a WebGIS. It is loaded on the client machine when the relevant web page is accessed with an internet browser. The main tasks of this layer are to collect user inputs, transforming them into information usable by the WEEZARD WSs (e.g. the TRENT2D model or the data manager); to manage client-server communication; and finally, to display the data returned in the form of 2D or 3D maps. The operational load of the client is very limited, and so almost any kind of device with a browser and internet access can be used as a client machine (even a simple smartphone).

Middleware layer: the middleware layer is the operational core of the system and hosts all the services supplying system functionalities. The layer consists of several functional elements that were specially developed for this system: the Simple Authentication and Security module for managing user authentication and data security; the TRENT2D core model and relevant auxiliary modules for preparing and harmonizing data; the Data Management module for transferring data back and forth from the middleware layer both to the application and to the data layer; and the geographical information publisher. All these modules are configured as WSs.

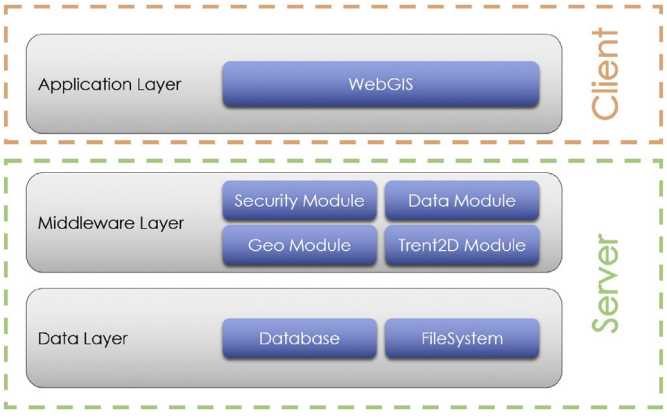

Fig. 2. Logical architecture of the WEEZARD solution.

Data layer: the data layer is the part of the system that handles data storage and retrieval. In our case, the information collected is heterogeneous, varying from user information and model information, to system information and statistical information. The storage strategies are also different, in some cases using the Relational DataBase Management System (RDBMS), with or without spatial schema; in other cases information is stored directly on the FileSystem.

5.2. The physical architecture

The physical architecture of the system is shown in Fig. 3, where the component diagram is sketched. As can be seen, in the current implementation it is composed of a client-server structure in which the server is located on a single machine. More complex configurations will be considered in future releases.

The web application is the entry point of the WEEZARD system and, as described above, looks like a WebGIS webpage. The GUI skeleton was developed using HTML5, CSS3 and JavaScript in the Bootstrap framework (Otto and Thornton, 2016), where several GUI controls (e.g. tabs, radios, checkboxes and trees) derive from JQuery (JQuery, 2016). Two-dimensional visualizations (maps, sections, etc.) are realized using the OpenLayer library (OpenLayers, 2016), while three-dimensional views are obtained using the WebGL library (Khronos, 2016), a low-level 3D graphics API based on OpenGL exposed through HTML5 (Feng et al., 2011; Singh and Garg, 2016).

Following the Infrastructure as a Service (IaaS) principle, middleware services are located on a server rent from an IT cloud provider. The structure of the middleware was designed following the Spring MVC strategy (MVC, 2016), which allows web-based applications to be built that are flexible and decoupled from the underlying view technologies. This strategy divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is accepted from or presented to the user (see e.g. Reenskaug and Coplien, 2009; for details).

All the modules required in the WEEZARD system are provided as RESTful WSs, hosted in the Apache Tomcat (TheApache®Software, 2017) server container. The modules dedicated to geographical data manipulation use Geospatial Data Abstraction Library (GDAL) routines (GDAL, 2017). Geospatial data are shared using the GeoServer (2016) with OGC®-compliant open standards such as Web Map Service (WMS) and Web Feature Service (WFS). The MPlayer library (MPlayer, 2016) is used to encode, in an AVI file, the screenshots produced by the client 3D window (see Section 6.2.3 for details).

Finally, the datastore is located on the same physical server as the middleware services. In future, it will be located in a cluster of servers and managed by using replication and load balancing strategies. Geospatial data are managed through the PostgreSQL (PostgreSQL, 2016) object-relational database with the Postgis (PostGIS, 2016) extension. Simulation output (consisting of many different data points) is stored directly to the filesystem. In some cases, such as in the case of manipulation, they are loaded in runtime into the RDBMS.

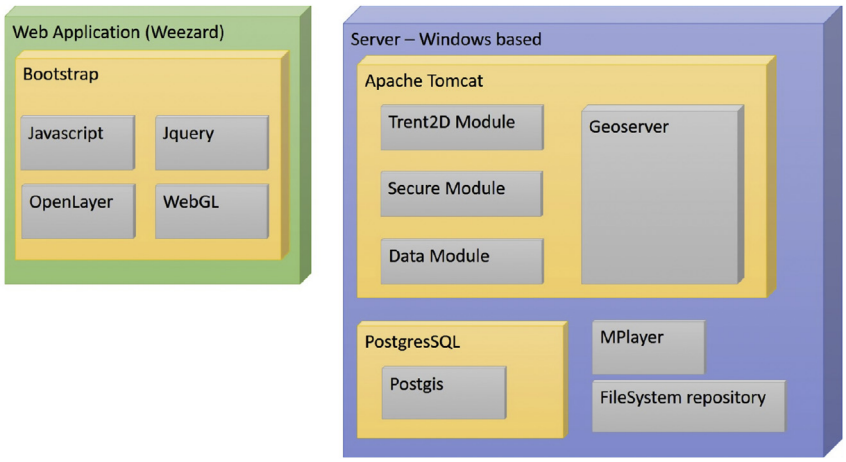

Fig. 3. Component diagram of the WEEZARD solution.

5.3. Client-server communications

Client-server communications is a key element in an efficient and high-performing system. As described in Section 2.2, different modules are used to perform a single simulation job. Therefore, we need to manage operations like passing data to a process instance and processing asynchronous requests. We dealt with these aspects by taking into account how the chain process works in an OGC® WPS and implementing a similar strategy. For passing data from the client to the server, we have three possible approaches:

- the data are already stored on the server (from the point of view of the client);
- the data are directly part of the XML encoded document sent via HTTP POST. The data can be text based (JSON), XML based (GML) or even raster based - in this case, they are usually encoded using Base64;
- the client does not have to pass the data itself, but can just send a reference link to the target data service (or file). In such a case, for example an OGC® WFS GetFeatureType URL can be passed and the server will download the data automatically.

Due to the length of time required to complete a simulation job, we implemented an asynchronous-mode process-request procedure similar to the OGC® WPS standards (see e.g. PyWPS-Development-Team, 2016). Communications between client and server are performed by using the Pull method, where the client is the active element while the server just responds to the client's request without actively pushing any information. The client sends an "ExecuteRequest" message and the server returns back immediately a response with a URL, where the client can check the process execution status. The process status is a generic status of the process instance and provides information on its progress. It can assume four different values:

- ProcessAccepted: the process was accepted by the server and it will start soon;
- ProcessStarted: the process has started and a report on the percentage of calculations already performed is provided;
- ProcessFinished: calculations have ended successfully;
- ProcessFailed: there was something wrong with the process and the server reports an exception, along with a message describing what may have gone wrong;

6. User categories, functionalities and the GUI

For the GUI, as for the system architecture, the choice of system functionalities, their design and their graphical implementation are closely related to the requirements of the type of user that the system is aimed at, namely practitioners and public agency technicians. Singling out the desirable features was accomplished through the systematization both of the observations collected during the development and the practical use of the TRENT2D code, and of informal interviews with the users who employed the desktop code during their work activities.

The present release of the system provides for only two user categories, namely the standard user and the administrative user. A standard user is a person (practitioner, technician, researcher, student) mainly interested in doing simulations and producing suitable representations of outputs, while an administrative user is a researcher of the development group who, besides running simulations for practical use and for testing, also manages some basic aspects of the system. Other user categories, which can be singled out among the interested parties listed in Section 2.1, can be conveniently added. Nevertheless, we have left this opportunity to future releases.

6.1. The main environment

Since WEEZARD is mainly intended to host and manage geospatial-related information and functionalities, the main page looks like a WebGIS framework, in which the background map is the worldwide OpenStreetMap® (2016). After the login procedure, other maps and shapefiles can be overlaid, as shown in Fig. 4. Their handling is done through a classical Table Of Contents (TOC) layer manager, in the *Layers* accordion menu located in the *Browsing* side panel. Geodata can be either loaded by the user or retrieved from WMSs. The use of the second menu of the *Browsing* panel will be presented in Section 6.2.3.

Common GIS functionalities (e.g. pan, zoom, identify, add external sources and search) are available through the quick buttons located at the top of the window. The last two buttons allow access to the *scheduling queue functionality*, which will be described in some detail in Section 6.2.4, and to the WEEZARD main menu.

6.2. The functionalities for the standard user

All the functionalities are designed with the purpose of carrying out a modelling job in an easy and time-saving way. Several of them

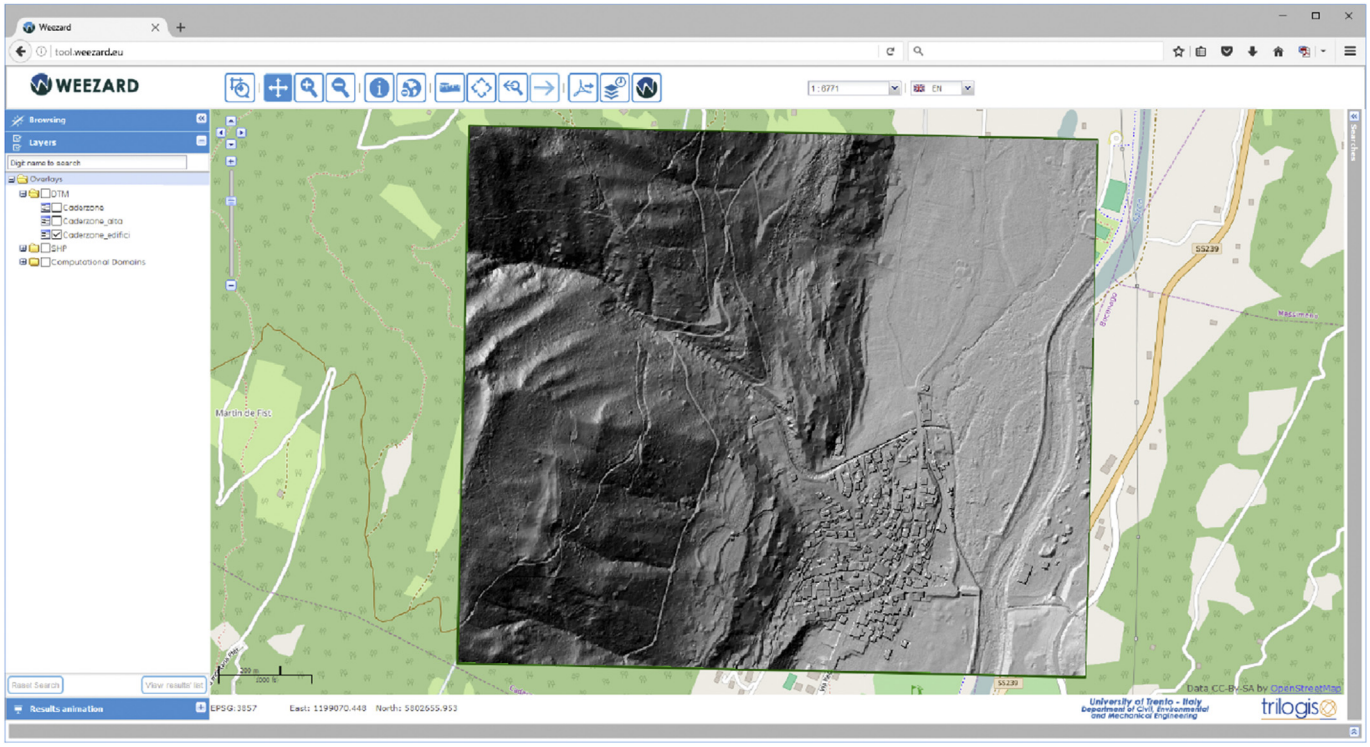

Fig. 4. Main window of the WEEZARD system.

perform operations that either cannot be performed using other GIS software, or would require a succession of awkward procedures. In addition, guided paths allow the user to follow chains of tasks in a clear and efficient way.

The WEEZARD button opens the main menu for the standard user, where functionalities are grouped logically in three categories, namely *GIS functionalities*, *Computational domain* and *Simulations* (see Fig. 5). The main functionalities will be illustrated in the following sections, as along with the *scheduling queue functionality*.

6.2.1. GIS functionalities

GIS functionalities are collected in two groups, called *DTM* and *SHP*. In the *DTM* group (see Fig. 6), beyond some classical GIS operations on DTM data, such as upload, merge and manage, two tasks are explicitly tailored for modification of DTMs to be used in simulations.

Edit with shp allows the user to change the elevation of all the cells inside closed geometries defined in a shapefile. This functionality is particularly useful, for example, when a DTM must be modified in order to account for a given distribution of buildings. In contrast, *Edit cells* enables easy pointwise modifications of a DTM: a three-panel window (see Fig. 7), showing a two-dimensional view, a three-dimensional view of a zoomed part and the matrix of the relevant elevations, allows the user to have an immediate perception of possible cell value modifications.

6.2.2. Computational domain

The *Computational domain* category is devoted to preparing input data for TRENT2D simulations (see Section 4.2). Also in this case, functionalities are collected in two groups (Fig. 5): one is dedicated to creating and managing domains while the other deals with defining boundary conditions.

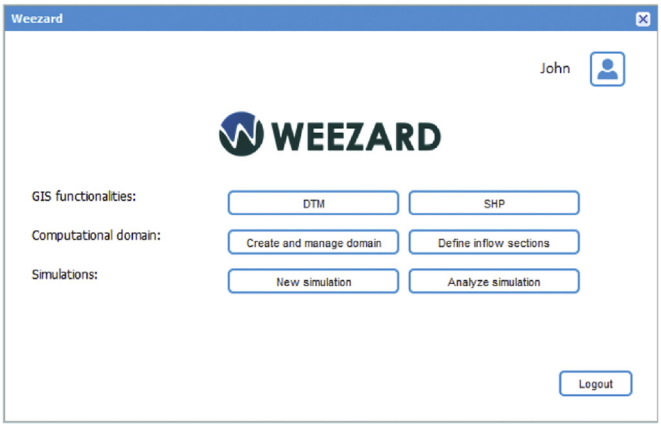

Fig. 5. Main menu for accessing the WEEZARD functionalities.

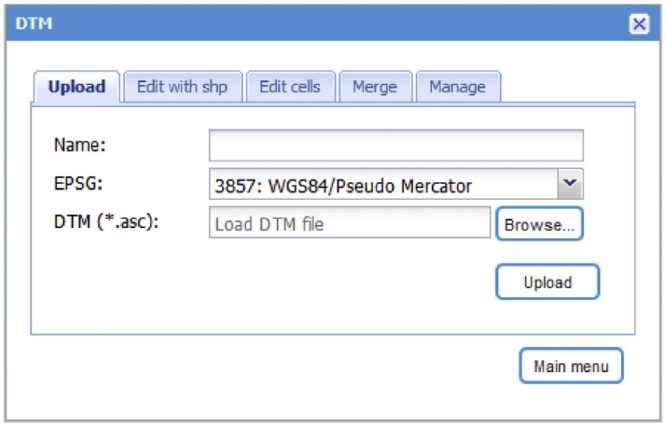

Fig. 6. *DTM* menu tabs.

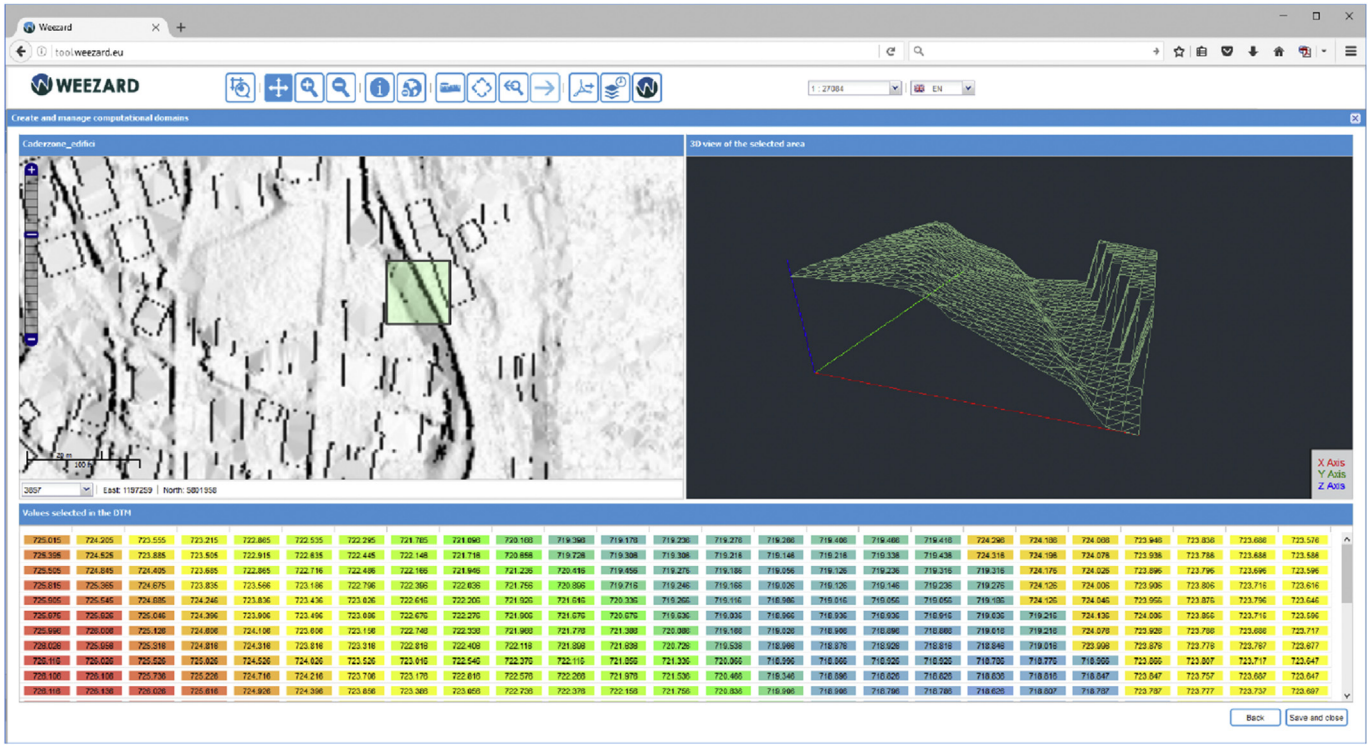

Fig. 7. *Edit cells* three-panel window.

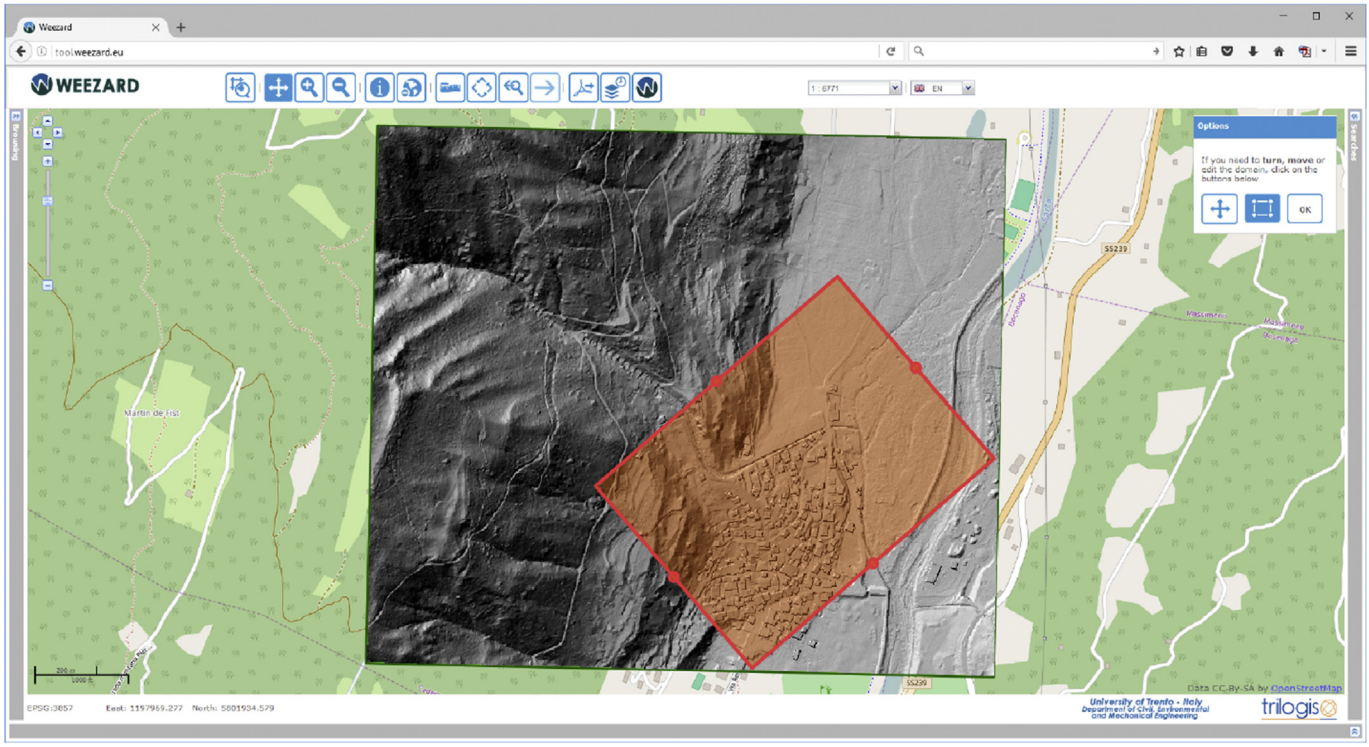

Fig. 8. Definition of the computational domain by drawing a rectangle.

The location in the world and the extent of a computational domain are determined graphically with the mouse by drawing a rectangle over a suitable 2D georeferenced layout, such as a grey-scale shaded DTM or any other useful layer (see Fig. 8). The user effort ends here, while the backstage work is just starting. The client sends some data and a request to the server, which performs all the operations necessary to prepare the computational domain as an input file for the model, namely defining an LCS, determining

the number of cells in the computational domain, interpolating cell elevation from the underlying DTM in order to obtain the initial condition for the model, writing files in specific locations and finally registering information into the Datastore.

A graphical approach is also used to select the geometric boundary conditions. As shown in Fig. 9(a), the boundary cells are highlighted over a suitable reference layer (commonly, the same layer used to locate the computational domain) and, with the mouse, a preliminary selection can be made. A small window displays the section and a slider allows the final extension to be easily delimited. Moreover, the averaged slope of the main channel entering the domain can be easily estimated as shown in Fig. 9(b).

6.2.3. Simulations

This category too is divided into two groups of tasks: *New simulation* and *Analyze simulation*. The first button starts a guided procedure that leads the user through running a TRENT2D simulation, spanning from the definition of solid and liquid inflow

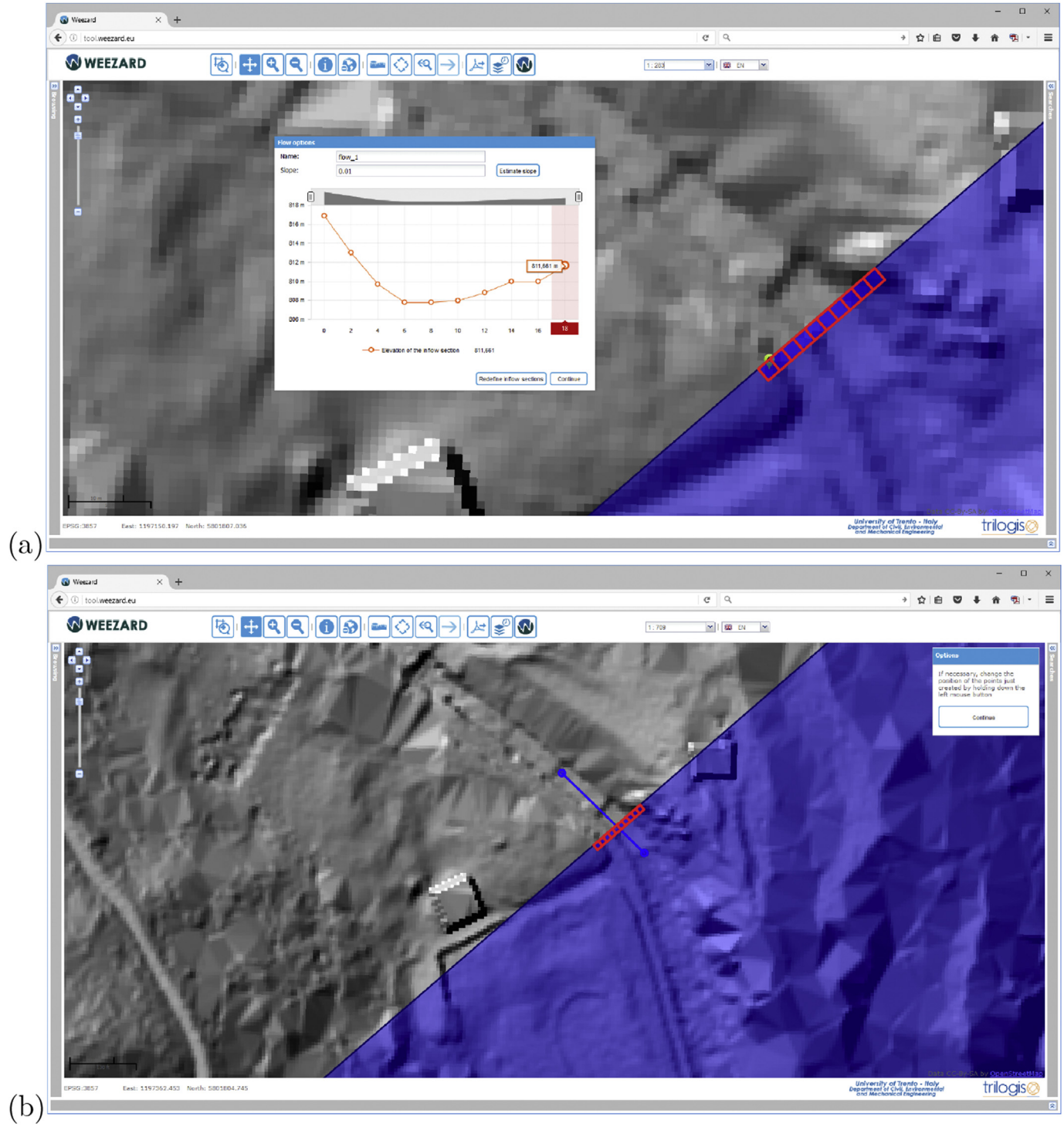

Fig. 9. (a) Selection of the input flow section; (b) the estimation of the bed slope across the inflow section.

hydrographs to the choice of the model parameter values. Since these calculations are already explained in Armanini et al. (2009) and no significant graphical support is needed, we do not present the details of this chain of operations. At the end of the procedure, the user can choose to start the simulation. In this case, the client sends a request to the server, which runs the simulation in background, remaining available for any other client request. The user can check the working progress and analyse partial results at any time.

Much more complex is the *Analyze simulation* task group. As shown in Fig. 10, it consists in a panel where the whole set of simulations, whether already performed (labelled with a green traffic light), still running (yellow light) or not yet run (red light), is present along with the name of the simulations, the computational domain names and the saved model variables. Once a simulation is selected, different tasks are available depending on the status of the simulation. We describe here only the case of a completed simulation.

The first line of buttons concerns information and management operations while the second line regards visualization. Here, three different display frameworks are available, each fulfilling different target requirements:

2D local view. This framework is tailored for in-depth quantitative analyses. Model outputs are displayed in a 2D, LCS view. With the time slider the user can select a particular simulation instant and with the side panel a particular variable can be visualized. Values can be displayed point by point or through multipoint sections (see Fig. 11(a)). The geometrical position of the sections can be stored on the server and used for analyzing different simulations on the same computational domain. This feature is particularly useful during hazard assessment, when different mitigation works must be compared in order to choose the most efficient solution for a given site, or when the same computational domain is used with different boundary conditions (i.e. different hydrographs associated with different return periods).

2D georeferenced view. As shown in Fig. 11(b), this framework contextualizes results in the 2D, GCS main window. Here, the accordion menu *Results animation,* located in the *Browsing* left panel, reveals the relevant management tools. Thanks to some simple buttons, scalar field maps (or velocity field maps) can be animated in time, showing the dynamic development of the simulated phenomenon over a background layer (or layers).

3D view. This last framework provides realistic 3D representation of a debris-flow simulation. For this reason, the framework is particularly useful for communication with non-experts. As pointed out by several authors (Hagemeier-Klose and Wagner, 2009; Albano et al., 2015; Sanders et al., 2016), this element is not a secondary goal in the field of risk management, because citizen awareness can be increased significantly through the use of advanced web tools, enhancing the resilience of communities. Image orientation and zooming are controlled with a mouse, while other controls are available in the side panel, as shown in Fig. 11(c). In particular, the *3D Cover* control group is used to manipulate the surface appearance (wireframe, shadowed greyscale, orthophoto draping, etc.), while the *Animation* collection permits dynamic visualizations and video production.

6.2.4. Scheduling queue functionality

The multi-user architecture of the system implies that many requests to run simulations can reach the server simultaneously or that a new request may arrive when other simulations are still running. Since only a given number of simulations can run concurrently (depending on the available computational power of the system), a scheduling queue is necessary to manage the situation. The scheduling queue functionality provides all information regarding the progress of a user's run. A window, shown in Fig. 12, gives a graphical and quantitative overview of the estimated

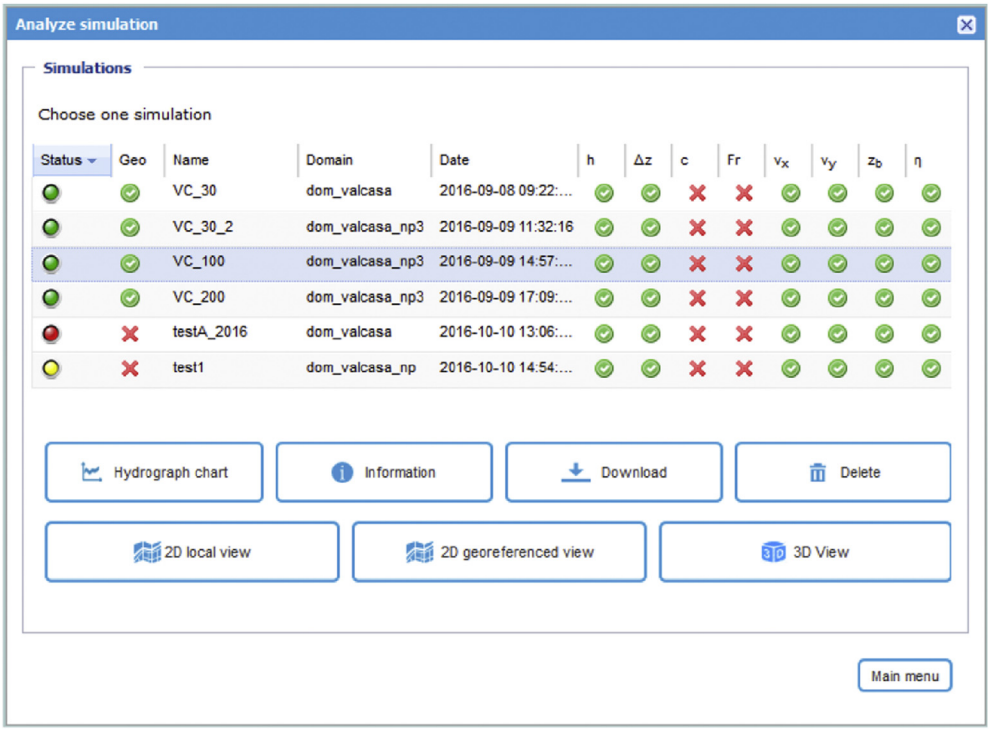

Fig. 10. *Analyze simulation* panel.

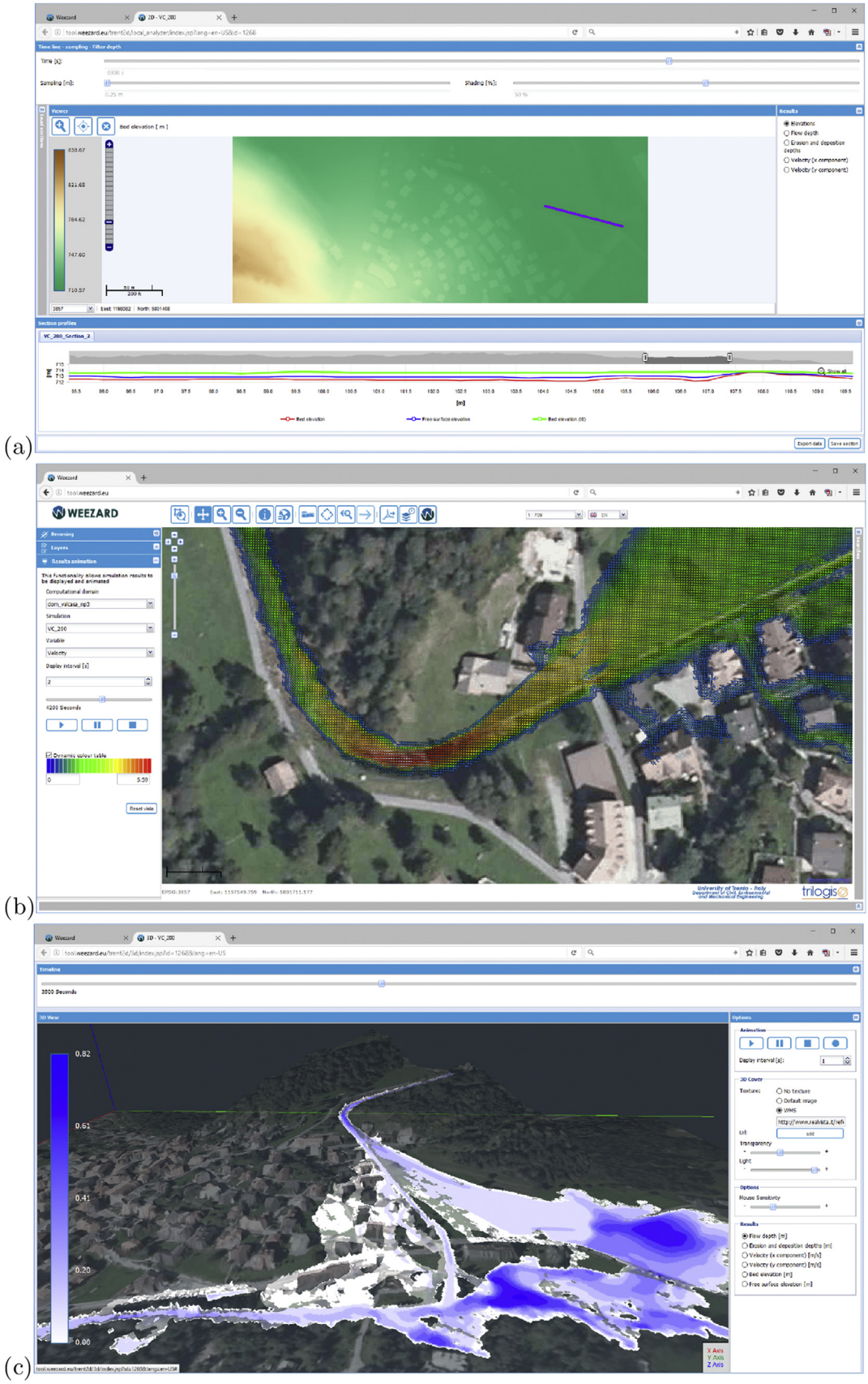

Fig. 11. Visualization frameworks: (a) 2D local view with sections, (b) 2D georeferenced view, (c) 3D view.

computational time of each simulation in the queue, the simulations currently running along with their estimated ending time and, finally, the position of the user's simulation in the queue.

6.3. The administrative user functionalities

Administrative users can access all the standard user functionalities and, in addition, a control panel for the management of some aspects of the system. The panel has four tabs (see Fig. 13): one is dedicated to user management, another gives the simulation list, the third gives a graphical and quantitative overview of the disk usage, while the last tab allows the administrator to set some system options (e.g. timeout values or maximum number of concurrent simulations).

6.4. GUI design and validation

GUI design and validation followed an incremental approach:

- each functionality was subdivided into logical steps consisting of a number of operations/input data that can fit into a single window layout; each layout was then sketched and subsequently implemented;
- a first validation derived from a systematic use of the functionality by the working group: bugs, non-intuitive sequences or operations, and difficulties in using windows were reported and corrected;
- a second validation derived from the use of the functionality by a beta tester group, made up of people outside the working group; suggestions for possible improvements were collected and, where appropriate, implemented.

Particular attention has been paid to giving the GUI an intuitive nature. In order to verify this feature, no detailed user manual was given to the beta tester group, who had to learn how to use the functionalities with only the support provided by the GUI itself. Indications of possible improvements to the GUI were collected and the necessary modifications were made.

7. Discussion and future developments

Besides the advantages of the chosen technologies with respect to other alternatives, it is necessary also to highlight possible limitations of the system and barriers to its use at the current state of development.

A first issue is portability. The Achilles heel is the numerical model, since in order to obtain high computational efficiency, the Fortran 90 code is compiled with processor-specific optimization in a Windows OS and so it does not have the complete portability of the rest of the system. Since the time necessary to do a simulation job can be quite long, in the initial phase of system development, we have preferred to prioritize efficiency over portability. However, our intent is to develop in a short time a UNIUX version and then to develop a fully portable release, able to exploit the potentialities of elastic cloud technologies.

A possible barrier to the use of the technology employed in the present work is that the IT skills necessary to develop the numerical code and the system environment are quite different, and therefore a proper collaboration and coordination among different development groups is necessary. In our case, the collaboration is between a university research department and a private company, namely Trilogis srl. We believe that this blended public/private collaboration is an effective way to carry out goals such as that of this work, consisting in a technology transfer process aimed at making an advanced numerical model accessible to practitioners and technicians.

Regarding data interoperability, the system is able to use shared data with OGC® WMS standards, while, in output, simulation data are not exposed as they are user sensitive and so an open access is not allowed. Other forms of data sharing (e.g. FTP; cloud repository; Web Distributed Authoring and Versioning, WebDAV), for example between practitioners and public agencies, can be easily activated since the system structure is ready. On the other hand, interoperability with other processing systems has not been pursued in this work, because of the type of target users we have chosen for the system. However, our idea is to move towards a fully compliant OGC® WPS V. 2.0 in the near future, also considering the possibility to join the Model Web (Nativi et al., 2013) or other similar

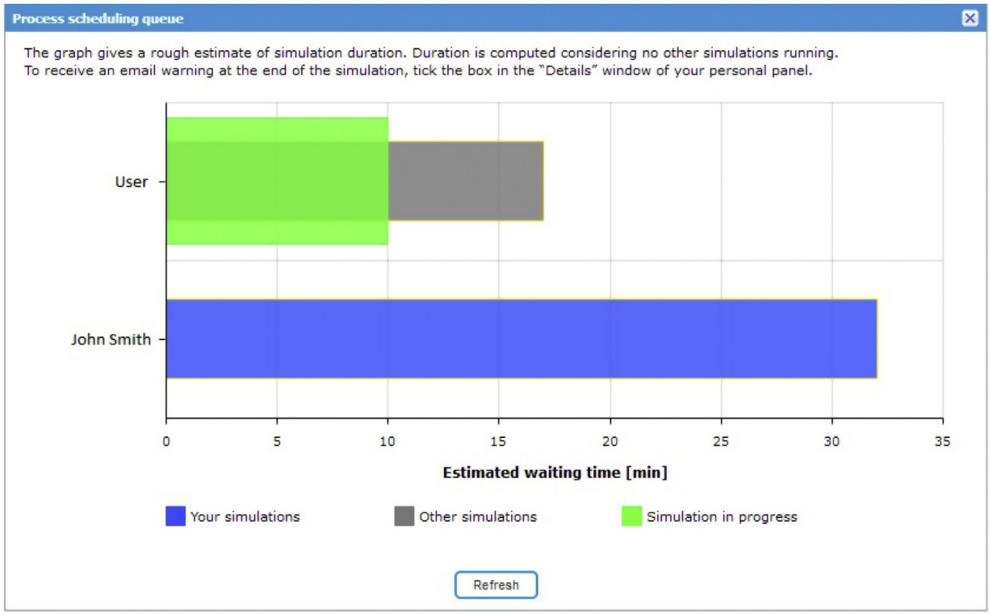

Fig. 12. Scheduling queue window.

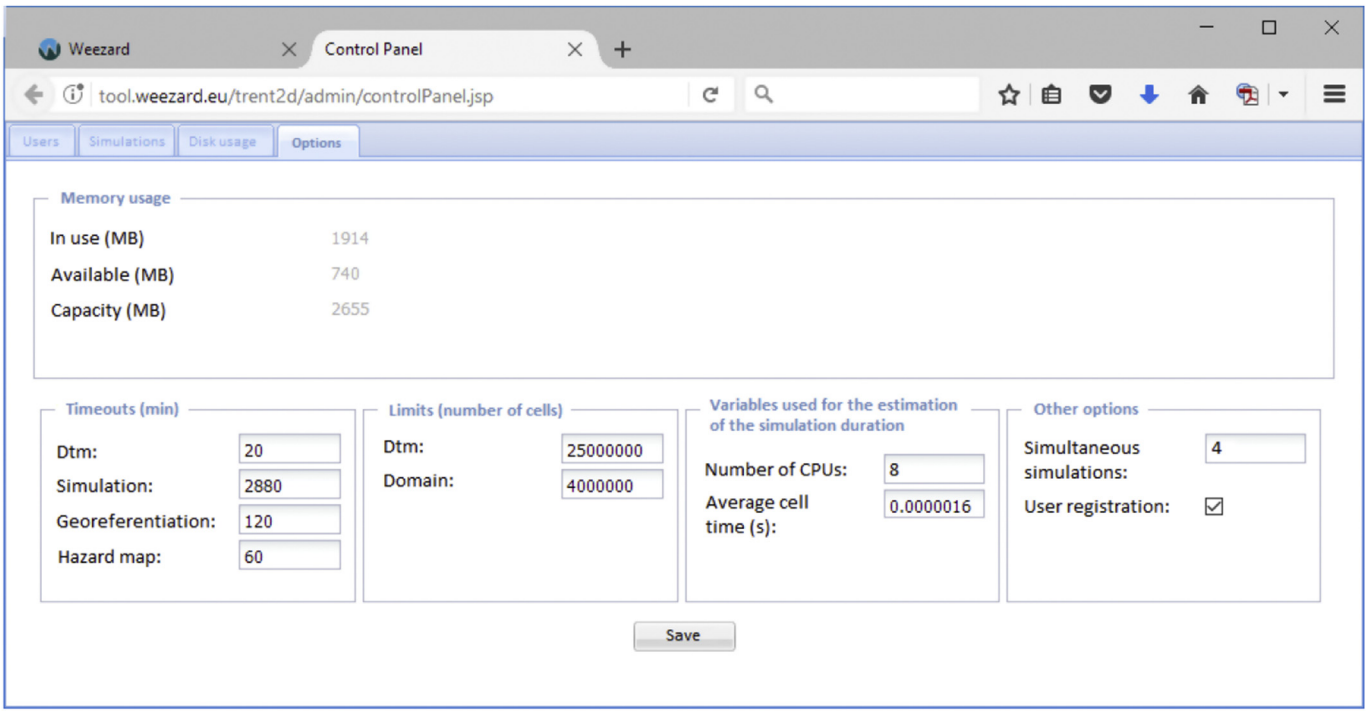

Fig. 13. Administrative panel.

initiatives. Wrapping the TRENT2D module as a WPS will also be considered (at the moment we have simply made TRENT2D as a service), to be addressed using containers with better performance, like 52North (2017) or GeoServer itself.

Other future developments regard the possibility of equipping WEEZARD with new functionalities and models, in response to user requests and observations. In particular, we have planned the following points:

- adding automatic production of hazard maps according to, for example, the so-called BUWAL approach (Heinimann et al., 1998),
- extending the range of applicability of the model, for example following the result presented in Rosatti and Zugliani (2015) for a debris-flow description over mixed fixed- and mobile-bed conditions,
- introducing models describing other geo-based phenomena such as avalanches,
- adding a hydrological model to produce inflow conditions for the debris-flow model.

8. Conclusions

Assessing debris-flow hazard using the best available technology while limiting costs, in accordance with the requirements of the EU and USA regulations, is no longer a “mission impossible” for practitioners and land protection public agencies. In this paper, we have presented a WS system in which a state-of-the-art numerical model, namely the TRENT2D model, is inserted into an efficient all-in-one environment strongly customized for the chosen end users. The system, called WEEZARD, is available at http://tool.weezard.eu.

User feedback confirms the effectiveness of the approach, which allows a significant reduction in the time and cost of production for a debris-flow simulaton job. We hope that WEEZARD's innovative features and its easy accessibility through the web will contribute to raising the standard of tools used to assess debris-flow hazard.

A future challenge for our system is undoubtedly its integration into a Decision Support System (DSS) for risk management in mountain regions. DSSs are still at an early stage of definition and development (see e.g. Muste and Firoozfar, 2016), but are one of the most promising tools for the protection of our communities. We are confident that WEEZARD, thanks to its WS structure, can be easily adapted to match any DSS standard that is established.

Acknowledgements

This work has been supported and funded by CARITRO Foundation - Cassa di Risparmio di Trento e Rovereto (Italy), within the project *Nuove frontiere per l'analisi provisionale di fenomeni alluvionali con elevata concentrazione di sedimenti: uno strumento numerico di seconda generazione per la salvaguardia territoriale montata,* and by the University of Trento, within the project *CLIMAWARE - CLIMatic change impacts on future Availability of WAter REsources and hydro-geological risks*.

The authors would like to thank Laura Rech and Cristian Sannicolò for their precious support in the WEEZARD testing phases and for the preparation of the figures.

Appendix A. WEEZARD system availability

The WEEZARD system is a WS ecosystem available on the web at http://tool.weezard.eu since September 2016. The cost of WEEZARD for professional and academic purposes depends on the actual use of the service. A free demo version is available upon registration, while some pre-packaged offers for the full version can be found at http://www.weezard.eu. Free access can be arranged for possible joint research projects with the authors.

Since WEEZARD is a web application, the only software product that a user requires is a web browser. Systematic debugging has been performed with Chrome and Firefox, but the system also works well with other browsers. The hardware demand is also

extremely limited. Only the 3D visualization has a minimum hardware requirement, consisting in a video card supporting WebGL and 2GB of system memory. Nowadays, these features are available on the majority of personal computers. On the server side, the hardware configuration is scalable according to the number of concurrent users. The most demanding element of the system is obviously the numerical kernel. The hardware resources necessary on the server depend on the size of the simulation domain. The current minimum configuration is composed of an eight Xeon CPU machine, with 8 GB of RAM and 500 GB of disk.

Regarding the developers, the numerical kernel of the system, i.e. the TRENT2D model and other ancillary codes written in Fortran90, in the present version was developed by some of the authors of this paper, namely Prof. G. Rosatti, Dr. D. Zugliani, PhD and N. Zorzi, PhD student, of the Department of Civil, Environmental and Mechanical Engineering of the University of Trento. All the other parts of the system were written by Dr. S. Piffer and A. Rizzi of the company Trilogis srl, coauthors of this paper.

Appendix B. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.envsoft.2017.11.017.

References

52North, 2017. URL http://52north.org.

Albano, R., Sole, A., Adamowski, J., 2015. READY: a web-based geographical information system for enhanced flood resilience through raising awareness in citizens. Nat. Hazards Earth Syst. Sci. 15, 1645–1658.

Armanini, A., Fraccarollo, L., Rosatti, G., 2009. Two-dimensional simulation of debris flows in erodible channels. Comput. Geosciences 35 (5), 993–1006.

Bagnold, R.A., 1954. Experiments on a gravity-free dispersion of large solid spheres in a newtonian fluid under shear. Proc. R. Soc. A Math. Phys. Eng. Sci. 225 (1160), 49–63.

Beedle, M., van Bennekum, A., Cunningham, W., Cockburn, A., Fowler, M., Highsmith, j., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Schwaber, K., Sutherland, J., Thomas, D., 2001. Manifesto for Agile Software Development. http://agilemanifesto.org/.

Belete, G., Voinov, A., Laniak, G., 2017. An overview of the model integration process: from pre-integration assessment to testing. Environ. Model. Softw. 87, 49–63.

Cannata, M., Marzocchi, R., 2012. Two-dimensional dam break flooding simulation: a GIS-embedded approach. Nat. Hazards 61, 1143–1159.

Christen, M., Bühler, Y., Bartelt, P., Leine, R., Glover, J., Schweizer, A., Graf, C., McArdell, B.W., Gerber, W., Deubelbeiss, Y., Feistl, T., Volkwein, A., 2012. Integral hazard management using a unified software environment. Numerical simulation tool "RAMMS"for gravitational natural hazard. In: Conference Proceedings of the 12th Congress INTERPRAEVENT 2012 Grenoble/France, pp. 77–86.

DMA, 2000. Disaster mitigation act of 2000. U.S.A. Public Law 106–390.

Feng, L., Wang, C., Li, C., Li, Z., 2011. A research for 3D WebGIS based on WebGL. In: computer science and network technology (ICCSNT). In: 2011 International Conference on, vol. 1, pp. 348–351.

Fraccarollo, L., Capart, H., Zech, Y., 2003. A Godunov method for the computation of erosional shallow water transients. Int. J. Numer. Methods Fluids 41 (9), 951–976.

Garegnani, G., Rosatti, G., Bonaventura, L., 2013. On the range of validity of the Exner-based models for mobile-bed river flow simulations. J. Hydraulic Res. 51 (4), 380–391.

GDAL, 2017. URL http://www.gdal.org/.

GeoServer, 2016. URL http://geoserver.org/.

Gregoretti, C., Degetto, M., Boreggio, M., 2016. GIS-based cell model for simulating debris flow runout on a fan. J. Hydrology 534, 326–340.

Hagemeier-Klose, M., Wagner, K., 2009. Evaluation of flood hazard maps in print and web mapping services as information tools in flood risk communication. Nat. Hazards Earth Syst. Sci. 9 (2), 563–574.

Heinimann, H.R., Hollenstein, K., Kienholz, H., Krummenacher, B., Mani, P., 1998. Methoden zur Analyse und Bewertung von Naturgefahren. Tech. Rep. 85 (Bundesamt für Umwelt, Bern).

IPCC, 2014. Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. IPCC, Geneva, Switzerland.

Iverson, R.M., Ouyang, C., 2015. Entrainment of bed material by Earth-surface mass flows: Review and reformulation of depth-integrated theory. Rev. Geophys. 53 (1), 27–58.

Jia, Y., Zhao, H., Niu, C., Jiang, Y., Gan, H., Xing, Z., Zhao, X., Zhao, Z., 2009. A WebGIS-based system for rainfall-runoff prediction and real-time water resources assessment for Beijing. Comput. Geosciences 35, 1517–1528.

Jones, S., 2005. Toward an acceptable definition of service. IEEE Software.

JQuery, 2016. URL http://www.jquery.com.

Khronos, 2016. URL https://www.khronos.org/webgl/.

Kulkarni, a. T., Mohanty, J., Eldho, T.I., Rao, E.P., Mohan, B.K., 2014. A web GIS based integrated flood assessment modeling tool for coastal urban watersheds. Comput. Geosciences 64, 7–14.

Mergili, M., Fellin, W., 2007. GRASS GIS and modelling of natural hazards. An integrated approach for debris flow simulation. OSGeo J. 3, 53–59.

MPlayer, 2016. URL http://www.mplayerhu.hu.

Mure-Ravaud, M., Binet, G., Bracq, M., Perarnaud, J.-J., Fradin, A., Litrico, X., 2016. A web based tool for operational real-time flood forecasting using data assimilation to update hydraulic states. Environ. Model. Softw. 84, 35–49.

Murillo, J., García-Navarro, P., 2010. An Exner-based coupled model for twodimensional transient flow over erodible bed. J. Comput. Phys. 229, 8704–8732.

Muste, M.V., Firoozfar, A.R., 2016. Toward generalized decision support systems for flood risk management. In: E3S Web of Conferences, vol. 7. www.scopus.com.

MVC, S., 2016. URL http://spring.io.

Nativi, S., Mazzetti, P., Geller, G., 2013. Environmental model access and interoperability: the geo model web initiative. Environmental Modelling & Software 39.

OpenLayers, 2016. URL http://openlayers.org/.

OpenStreetMap®, 2016. URL http://www.openstreetmap.org.

Otto, M., Thornton, J., 2016. URL http://getbootstrap.com/.

Pelanti, M., Bouchut, F., Mangeney, A., 2011. Riemann solver for single-phase and two-phase shallow flow models based on relaxation. Relations with Roe and VFRoe solvers. J. Comput. Phys. 230, 515–550.

PostGIS, 2016. URL http://postgis.net/.

PostgreSQL, 2016. URL https://www.postgresql.org/.

PyWPS-Development-Team, 2016. URL http://pywps.readthedocs.io/en/latest/wps.html.

Reenskaug, T., Coplien, J.O., March 2009. The DCI architecture: a new vision of object-oriented programming. Online publication. http://www.artima.com/articles/dci_vision.html.

Rosatti, G., Begnudelli, L., 2010. The Riemann problem for the one-dimensional, free-surface shallow water equations with a bed step: theoretical analysis and numerical simulations. J. Comput. Phys. 229 (3), 760–787.

Rosatti, G., Begnudelli, L., 2013a. A closure-independent Generalized Roe solver for free-surface, two-phase flows over mobile bed. J. Comput. Phys. 255, 362–383.

Rosatti, G., Begnudelli, L., 2013b. Two-dimensional simulation of debris flows over mobile bed: enhancing the TRENT2D model by using a well-balanced Generalized Roe-type solver. Comput. Fluids 71, 179–195.

Rosatti, G., Fraccarollo, L., 2006. A well-balanced approach for flows over mobile-bed with high sediment-transport. J. Comput. Phys. 220 (1), 312–338.

Rosatti, G., Zugliani, D., 2015. Modelling the transition between fixed and mobile bed conditions in two-phase free-surface flows: the Composite Riemann Problem and its numerical solution. J. Comput. Phys. 285, 226–250.

Rosatti, G., Murillo, J., Fraccarollo, L., 2008. Generalized Roe schemes for 1D two-phase, free-surface flows over a mobile bed. J. Comput. Phys. 227 (24), 10058–10077.

Sanders, B., Luke, A., Schubert, J., Moftakhari, H., AghaKouchak, A., Matthew, R., Goodrich, K., Cheung, W., Feldman, D., Basolo, V., Houston, D., Serrano, K., Boudreau, D., Eguiarte, A., 2016. Co-development of costal flood models: Making the leap from expert analysis to decision support. In: Sustainable Hydraulics in the Era of Global Change, p. 3. https://doi.org/10.1201/b21902-3.

Scheidl, C., Rickenmann, D., 2011. TopFlowDF - a simple gis based model to simulate debris-flow runout on the fan. In: 5th International Conference on Debris-flow Hazards "Mitigation, Mechanics, Prediction and Assessment", pp. 253–262.

Shan, T.C., Hua, W.W., 2006. Solution Architecture for N-tier Applications. In: Proceedings of the IEEE International Conference on Services Computing. SCC '06. IEEE Computer Society, Washington, DC, USA, pp. 349–356. https://doi.org/10.1109/SCC.2006.99.

Singh, H., Garg, R., 2016. Web 3D GIS application for flood simulation and querying through open source technology. J. Indian Soc. Remote Sens. 1–10.

Stoffel, M., Mendlik, T., Schneuwly-Bollschweiler, M., Gobiet, A., 2014. Possible impacts of climate change on debris-flow activity in the Swiss Alps. Clim. Change 122 (1–2), 141–155.

Takahashi, T., 1978. Mechanical characteristics of debris flow. J. Hydraulics Div. 104 (HY8), 1153–1169.

Tan, X., Di, L., Deng, M., Huang, F., Ye, X., Sha, Z., Sun, Z., Gong, W., Shao, Y., Huang, C., 2016. Agent-as-a-service-based geospatial service aggregation in the cloud: a case study of flood response. Environ. Model. Softw. 84, 210–225.

TheApache®Software, 2017. URL https://tomcat.apache.org/.

Toro, E.F., 2009. Riemann solvers and numerical methods for fluid dynamics: a practical introduction. Springer Science & Business Media.

UE, 2007. Flood Directive. 2007/60/EC.

Wang, C., Li, S., Esaki, T., 2008. GIS-based two-dimensional numerical simulation of rainfall-induced debris flow. Nat. Hazards Earth Syst. Sci. 8, 47–58.

Wu, J., Chen, G., Zheng, L., Zhang, Y., 2013. GIS-based numerical simulation of Amamioshima debris flow in Japan. Front. Struct. Civ. Eng. 7 (2), 206–214.

Yin, L., Zhu, J., Zhang, X., Li, Y., Wang, J., Zhang, H., Yang2, X., 2015. Visual analysis and simulation of dam-break flood spatiotemporal process in a network environment. Environ. Earth Sci. 74, 7133–7146.