Nodes on Ropes: A Comprehensive Data and Control Flow for Steering Ensemble Simulations

Jürgen Waser, Hrvoje Ribičić, Raphael Fuchs,
Christian Hirsch, Benjamin Schindler, Günther Blöschl, and M. Eduard Gröller

Abstract—Flood disasters are the most common natural risk and tremendous efforts are spent to improve their simulation and management. However, simulation-based investigation of actions that can be taken in case of flood emergencies is rarely done. This is in part due to the lack of a comprehensive framework which integrates and facilitates these efforts. In this paper, we tackle several problems which are related to steering a flood simulation. One issue is related to uncertainty. We need to account for uncertain knowledge about the environment, such as levee-breach locations. Furthermore, the steering process has to reveal how these uncertainties in the boundary conditions affect the confidence in the simulation outcome. Another important problem is that the simulation setup is often hidden in a black-box. We expose system internals and show that simulation steering can be comprehensible at the same time. This is important because the domain expert needs to be able to modify the simulation setup in order to include local knowledge and experience. In the proposed solution, users steer parameter studies through the World Lines interface to account for input uncertainties. The transport of steering information to the underlying data-flow components is handled by a novel meta-flow. The meta-flow is an extension to a standard data-flow network, comprising additional nodes and ropes to abstract parameter control. The meta-flow has a visual representation to inform the user about which control operations happen. Finally, we present the idea to use the data-flow diagram itself for visualizing steering information and simulation results. We discuss a case-study in collaboration with a domain expert who proposes different actions to protect a virtual city from imminent flooding. The key to choosing the best response strategy is the ability to compare different regions of the parameter space while retaining an understanding of what is happening inside the data-flow system.

Index Terms—Emergency/Disaster Management, Visual Knowledge Discovery, Visualization System and Toolkit Design, Data-Flow, Meta-Flow, Parameter Study, Uncertainty, Visualization of Control.

1 Introduction

The Center for Research on the Epidemiology of Disasters (CRED) has pointed out floods as the most common natural disaster in 2010 [16]. The problem is that most of the populated areas in the world are vulnerable to flood disasters. World-wide, floods are likely to become increasingly severe and more frequent due to climate change [31], population growth, deforestation or change of land-use. We cannot prevent natural hazards but we can prepare to keep damage as low as possible. Simulation technology can help to get a better understanding of these natural phenomena.

Our case study extends the levee-breach scenario that has been explored with the World Lines (WL) approach [3, 44]. The simulation site comprises a river that is located next to a neighborhood of 20 buildings which are protected by levees from flooding. The starting point of the conducted study is a breach in the levees that causes city flooding. The new case study in this work is based on different assumptions, inducing a modified set of requirements for the steering task. Figure 1 shows an overview of the simulation domain and the case-study tasks. We assume that a severe weather condition is steadily rising the water level in the river bed. A domain expert, who has experience and indispensable knowledge about local conditions [13], draws the conclusion that the levees are likely to fail within the next couple of days. The local expert proposes a small number of eligible precautions that can be realized in the given time frame. It is the task of the user to evaluate these measures in the steering environment and pick the best response strategy. The focus is thus not on designing a breach closure on an already flooded city, but to choose a predefined response strategy that can be accomplished prior to the flooding. The decision making is guided by the following rules:

- *Jürgen Waser is with VRVis Vienna, E-mail: jwaser@vrvis.at.*
- *Hrvoje Ribičić is with VRVis Vienna, E-mail: ribicic@vrvis.at.*
- *Raphael Fuchs is with ETH Zürich, E-mail: raphael@inf.ethz.ch.*
- *Christian Hirsch is with VRVis Vienna, E-mail: chirsch@inf.ethz.ch.*
- *Benjamin Schindler is with ETH Zürich, E-mail: bschindl@inf.ethz.ch.*
- *Günther Blöschl is with TU Vienna, E-mail: bloeschl@hydro.tuwien.ac.at.*
- *Eduard Gröller is with TU Vienna, E-mail: groeller@cg.tuwien.ac.at.*

*Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.*

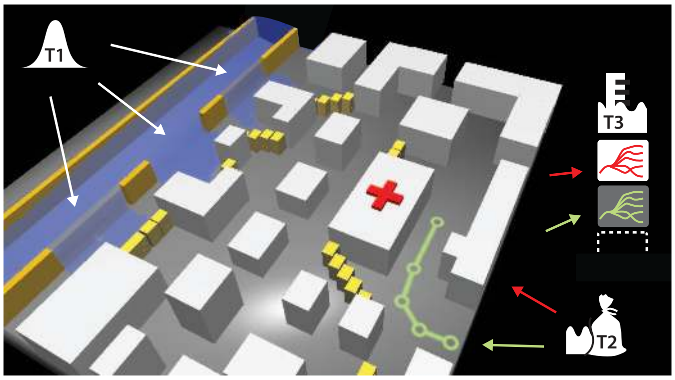


Fig. 1. Scenario and task description: (T1) The flooding expert is uncertain about the location of the imminent levee breach. (T2) Hospital (red cross) and evacuation path (green) have high priority, the damage to buildings has to be kept to a minimum. (T3) The expert needs to analyze many risk curves with respect to important results such as water levels to determine the best response strategy.

Task 1: The domain expert is uncertain about the exact location of the levee breach, but is able to define a confidence interval with respect to the breach location [7, 8]. According to Stewart and Melchers [39], a risk analysis should take into account a thorough investigation of the uncertainties. For this reason, the user has to perform an ensemble simulation to investigate a distribution of possible breach locations (see Figure 1, T1).

Task 2: A successful response strategy needs to fulfill certain requirements (Figure 1, T2). For example, the people in the hospital should be kept safe under any circumstances. An evacuation path from the hospital should be present at any time. Furthermore, the damage to buildings has to be kept as low as possible. The system has to allow

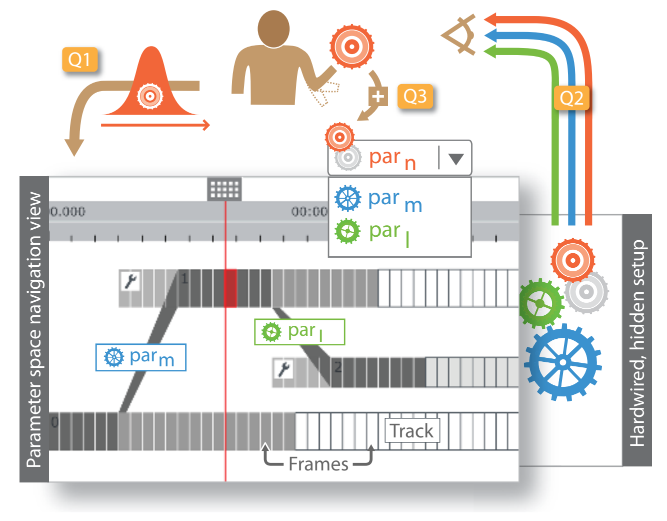


Fig. 2. Problem description: (Q1) How to create parameter studies to test uncertainties with respect to input parameters? (Q2) How to visualize what is going on in the underlying simulation setup? (Q3) How to modify the simulation setup and extend the parameter space to allow for exploration with respect to input parameter par_n (red)?

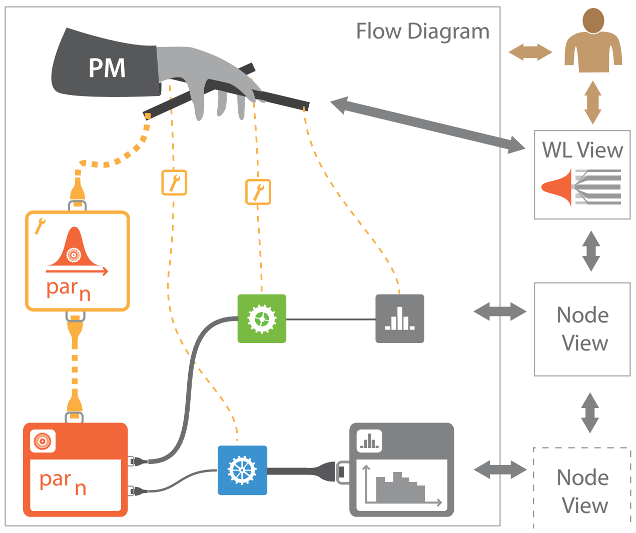


Fig. 3. Proposed solution: The simulation setup is presented in a data-flow diagram that is extended by a meta-flow (yellow, dashed lines) to decouple parameter control from data propagation. World Lines (WL) navigates via the puppet master (PM), a special meta-node which employs meta-communicators (yellow nodes) to control other nodes. The meta-flow highlights the control path to the red node, since in the WL view, the user studies uncertainties with respect to par_n (red). The thickness of data-flow wires (grey lines) reflects scalar values of computed results on a per-wire basis. The robustness-analyzer nodes (grey) show more detailed information.

that all relevant parameters can be entered and that all the necessary information is generated to verify whether these requirements are fulfilled. To accomplish this, the user needs a deeper understanding about what is going on in the underlying simulation setup.
Task 3: The steering process should yield risk visualizations [19, 25] for all generated information (see Figure 1, T3). In a scheduled simulation, these visualizations are updated each time new simulation data is available to allow for reasoning as soon as possible. Each visualization shows the temporal distribution function of a computed result, ideally accompanied by uncertainty bounds. To simplify fast reasoning, the generated visualizations should be automatically organized in a concise overview. To our knowledge, there is no system to accomplish all user tasks in a single steering environment.

2 Problem Description

In this section we generalize the requirements for the presented system. Figure 2 summarizes the problems involved with state-of-the-art applications which we tackle in this paper. The steering process that has to be perfomed is equivalent to interactive navigation in a multi-dimensional parameter space with the goal to generate insight. Modern simulation-steering environments provide interactive navigation views to ease this exploration in parameter space. The World Lines (WL) view [44, 3] is an example for such a view. In this view, simulation runs are represented as tracks that evolve in time (see Figure 2). A track consists of frames, each of which represents a simulation step. With WL, parameter-space navigation is accomplished via the concept of branching: when a parameter changes, a new track is created which covers an additional point in parameter space. The WL view in Figure 2 comprises two branches which have been created through modification of parameters par_m (blue) and par_l (green) respectively. This navigation approach is not sufficient if we have to take uncertainties into account. A logical next step is to introduce the ability to simulate parameter studies according to statistical distributions (Figure 2, Q1). Analogous to interval arithmetic, simulation steering is not done for one or a few values of a specific parameter but for entire intervals or distributions thereof.

Navigation views usually operate on a hardwired simulation setup that is opaque to the user (Figure 2, Q2). Hiding internals in such a black box can be the right choice for the unexperienced user. In many cases however, true understanding of a phenomenon is only gained if the user has access to information contained inside the black box. Better insight is obtained if the user learns about how the system parts change during navigation. State-of-the-art applications lack a concise visualization that shows which system parts are affected by navigation, and that automatically updates the presentation after every relevant change. Moreover, it can be essential to be able to quickly identify important properties of intermediate results such as fluid pressure, barrier movements or water levels, that are generated by the various components of the system. Ideally, a visualization of internal system parts is also capable of presenting an overview of relevant information computed by each part.

The user requires the ability to modify the simulation setup itself (Figure 2, Q3). During interactive exploration, it is possible that the parameter space has to be extended to cover additional input dimensions not considered in the first place. This option also provides the ability to investigate different, related systems. Moreover, users should be able to alter the system dynamically to generate additional information needed without loosing any results that have been computed before the modification.

3 Proposed Solution

Our solution to the problems is based on a standard data-flow system. Each node of a data-flow diagram can contribute dimensions to the parameter space. Even though modular and extensible, a standard data-flow is not sufficient to solve the presented problems. Steering a simulation via dynamic modifiation of a data-flow is not feasible. We suggest a smart data-flow diagram which is extended by a meta-flow of steering information. The data-flow determines the transport of data that is generated during the execution of the simulation algorithm, including simulation results, derived data and the visualization. The purpose of the meta-flow is to provide an advanced control of the parameters required throughout the data-flow. This way, the communication of parameters between nodes is decoupled from the standard data-flow, enabling flexible information exchange and control exchange between nodes.

Inspired by the coordination graph in Improvise [45], we visualize the meta-flow as a an additional network of ropes that is directly embedded into the flow diagram (Figure 3). The meta-flow is built with ropes (dashed lines) to top and bottom connectors of nodes while the data-flow propagates trough wires from left to right.

The interactive visualization of the data-flow and the meta-flow enables straightforward manipulation of the computations performed during simulation. The idea is to add novel types of meta-nodes which do not perform computations but which control other nodes. Such a node operates on a complex flow network in a way comparable to a puppet master pulling the strings. Therefore we call these control nodes puppet masters (PM). A puppet master utilizes special helper nodes (meta-communicators) to gain fine-grained control over nodes, enabling interaction, visualization and steering inside large parameter spaces. WL employ a puppet master to steer a user-defined subset of nodes. In the flow diagram, users can modify the explorable parameter space by configuring the meta-flow.

The meta-flow enables modification of control paths within the flow diagram. Still, the behavior of the underlying flow network and of individual nodes remains difficult to grasp. We believe that the behavior of individual system parts is best understood in the place where users create them, namely in the flow diagram directly. We suggest the extension of the flow diagram with emphasis techniques from visualization such as levels-of-detail (LOD). The goal is to dynamically highlight important per-node information about the meta-flow and computational results. The flow diagram in Figure 3 modifies the thickness of ropes and the LOD of nodes to identify the most relevant parts in the current steering process. With this technique, it can be seen that the user mainly modifies parameter par_n, provided by the red node. In this manner, the meta-flow visualization can be regarded as a concrete example for automatically storing knowledge gathered during interactive exploration [17].

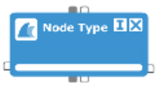
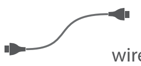
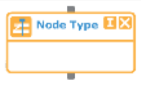
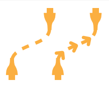
In addition, we create direct visualizations of computed data inside the flow diagram. Figure 3 gives examples for such visualizations. Statistical quantities are displayed on data-flow wires and inside robustness-analyzer nodes. The thickness of the wires encodes scalar values like mean or standard deviation. The diagram itself becomes a dynamic view that is an integral part of a system of multiple linked views associated with certain nodes. This constellation improves the linkage between nodes in the data-flow and the resulting data that is shown in the associated views. The contributions of this work can be summarized as follows:

- Configurable meta-flow as an abstraction for information and control exchange between nodes.
- Puppet masters for centralized navigation in parameter space spanned by nodes.
- Dynamic visualization inside the flow diagram. The ropes and nodes of the meta-flow depict the control-flow between nodes, the work-flow and inherent knowledge. The wires and nodes of the data-flow dynamically indicate the computed data per node and connection.
- Parameter studies in the World Lines navigation view to evaluate uncertainties through calculating with intervals and distributions.

4 Related Work

Modular Visualization Environments (MVEs) Popular MVEs such as AVS [4, 43] or IRIS Explorer [30] are based on the visualization pipeline model [22]. They provide modules to accomplish the constituent tasks in the visualization process [15]. In a graphical interface, the different modules (nodes) are presented as boxes. They have input and output ports and are connected via simple mouse clicks. The connections represent the data-flow between the modules which all run as separate processes. In this work, we refer to data-flow connections as wires. Table 1, row one, illustrates our basic design of the data-flow components which is based on state-of-the-art MVEs [1, 4, 28]. The background color of nodes is determined by the node category (e.g., view, simulation, geometry). A key feature of MVEs is their extensibility [47]. The simulation process can be incorporated into an MVE providing the opportunity to steer while using visualization components [32, 37, 40, 48]. To enable the transport of node parameters

Table 1. Basic design of data-flow and meta-flow components.

flow type	basic node design		established through	
data-flow (horizontal)		background color (acc. to node category)		wires
meta-flow (vertical)		yellow border color		ropes

from node to node in the data-flow of AVS [4], parameters can be converted to node connectors. An alternative solution is to connect node parameters via a special-purpose connection [12, 28]. If connected this way, the values of parameters are synchronized. Demand-driven approaches provide the option to upstream information through the data-flow [18], enabling interaction with visualizations [21]. In Improvise [46], the coordination between views is configured using relational expressions and can then be visualized as a coordination graph.

Further advancements in data-flow systems are related to the process of building visualizations. Macros are reusable groups of nodes that can be treated effectively as one module [1, 4, 34]. SmartLink [41] suggests assistants which take the history of user decisions into account to propose connections between nodes. Scheidegger et al. [35] use provenance information to guide the creation of data-flow nodes and provide automatic suggestions. VisTrails [38] adapts a history tree for capturing and reusing provenance while modifying the data-flow setup. MeVisLab [1] uses halos to improve the visibility of connections between nodes in a complex network. To reduce clutter, the user can toggle the visibility of connectors.

Level-of-detail Context sensitive level-of-detail (LOD) is a common technique to improve the layout of interfaces [27]. Continuous Zoom [9] is a fisheye-based method for navigating large networks. The technique divides the display into rectangular regions and enlarges individual sections based on scale factors and degree-of-interest.

Visual Programming Visual-programming approaches focus on creating executable data-flows at a very low level, similar to a text-based programming environment [23]. Therefore the resulting networks tend to be more difficult to understand than those which are created in MVEs. LabVIEW [29] is the most prominent example. Because of the complexity of the created data-flows, there are several proposals to visually augment the flow diagram. For example, wires are colored according to the data type they transport [20]. The data-flow allows for loop and switch statements that are shown as boxes which can contain sub-data-flows [26]. Impure is a visual programming language written in ActionScript to visualize data from the internet [11].

Navigation in parameter spaces Parameters are core components of any computation. By modifying parameters, the user navigates from point to point in a multi-dimensional parameter space, often to find an optimal set of parameters. There are a number of different interaction techniques to ease this navigation which are usually tailored to the task at hand. The most common way of navigation in parameter space is through selection from parameter studies. Often this is done by entering numbers into a regular grid [6, 24]. The AVS [4] animator module is a data-flow node for generating keyframe animations. The node has access to all available node parameters in order to create snapshots of points in parameter space. Bruckner and Möller [14] sample the entire input parameter space to perform many simulation runs offline. The results are sorted according to similarity and presented for exploration in a cluster-timeline view. Amirkhanov et al. [5] present a parameter-exploration system to test the stability of specimen placements in computed tomography. Berger et al. [10] provide a real-time navigation through a continuous parameter space based on simulation-proxy models. The WL approach [3, 44] follows the concept of steering where simulation data is generated on-the-fly. Here, parameter space traversal is accomplished via the creation of new simulation runs (branching).

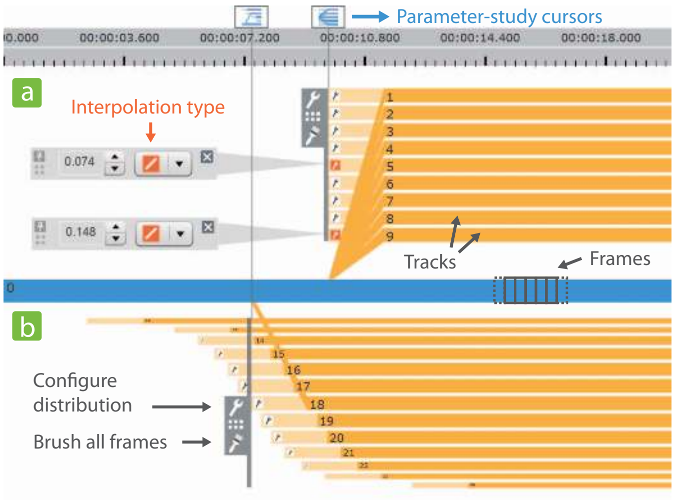


Fig. 4. Parameter studies in World Lines. (a) Parallel parameter study using a custom distribution function. (b) Temporal parameter study using a Gaussian distribution of track start times.

5 Parameter Studies with World Lines

In this section we propose an approach to account for uncertainties with respect to simulation input parameters using World Lines (WL). Figure 4 indicates the most important components of WL to recap their visual representation. The user can explore the parameter space by interactive branching: any modification of a parameter creates a new track that visually originates from the parent track. However, this way only one parameter can be modified at a time and it can be time-consuming to test many alternative choices. To speed up the exploration process, we suggest the introduction of parameter studies. Instead of single branches, the investigator opts for one of the following parameter-study types:

Parallel parameter study Creates an ensemble of tracks that originate from a common frame and that share a common start time (Figure 4a). This can be used for example to test different levee-breach positions given by a statistical distribution.

Temporal parameter study This type investigates different branching times for the same parameter change, resulting in an ensemble of tracks that have different start times. (Figure 4b). This can be used for example to test alternative times of levee-breach occurrence again given by a statistical distribution.

In the WL view, a *parameter-study cursor* serves as a component to visually group all tracks that are contained in the associated parameter study (Figure 4). Users can choose between sampling according to a predefined statistical distribution or according to a custom function. Figure 4b shows a temporal parameter study that is based on a Gaussian distribution of the track start times. Here, the cursor's temporal position is given by the mean value of the distribution. Dragging the cursor in horizontal direction adjusts the mean value and shifts all tracks in time. An embedded configuration panel can be used to further configure the distribution properties. In addition, the parameter-study cursor provides convenient interaction techniques to apply operations to all associated tracks at once. This includes dragging, to customize the layout, brushing, to select all frames contained in the parameter study, and collapsing, to reduce visual clutter by showing only the track that is associated with the mean value of the distribution. Custom parameter-study functions are evaluated by interpolation. The user selects two or more tracks of the parameter study that should contain parameter values to be interpolated. Value and interpolation type are specified through inline widgets (Figure 4a).

6 Meta-Flow

The previous section discusses a novel way to setup parameter studies using the World Lines (WL) view. This section describes how this information is transmitted to the underlying data-flow network to perform the required computations. A standard data-flow is a static graph that defines a specific application. In such a hardwired setup, it is not easy to define and understand how nodes communicate. We propose the management of node parameters and node relations in a separated meta-flow network which is directly embedded into the data-flow diagram. Via ropes, nodes are able to control other nodes. To provide the user with more precise control over this steering mechanism, we employ special meta-flow nodes called *meta-communicators* that are treated as part of a rope. These meta-communicators are automatically created as soon as a rope is established. The meta-flow allows the user to configure and perform complex steering tasks without the need for extensions to the linear data-flow model, such as looping or upstreaming [33]. The main advantage is that the control operations are now decoupled from the data-flow execution. In the flow diagram, this separation is visualized as a vertical meta-flow through ropes and a horizontal data-flow through wires. Users can now cleary see and modify how the nodes control each other. Table 1 compares the basic design of meta-flow components and data-flow components. Table 2 provides an overview of all meta-flow components and their functionality in the presented system. In the following sub-sections we give a more detailed explanation of these components.

6.1 Puppet Masters

Puppet masters are a generic concept to steer many nodes. A puppet master is realized as a meta-node that is able to gain full control over all meta-connected nodes. Puppet masters are able to automatically navigate a data-flow system in parameter space. The connection between a puppet master and a controlled node needs to be established via the meta-flow connector named *meta* located both at the top and at the bottom of each node. An unlimited number of ropes is allowed to be attached to this connector. Users can influence the steering behavior with respect to individual nodes via the special meta-communicator named *configurator*. A configurator is automatically placed onto a rope as soon as the rope is created. In addition, a puppet master can be associated with an optional view to simplify parameter-space navigation.

Figure 5 shows a screenshot of a data-flow network for simulation steering that utilizes WL as a puppet master to modify parameters across several nodes. The WL view is coupled with this puppet master which has the required knowledge and power to transmit the user-defined steering information down to the meta-connected nodes. The user can modify the available parameter space by adding and removing ropes to specific nodes. For example, to enable modelling of a Levee breach through branching in WL, we have to establish a rope from the puppet master to the Levees node (ochre node in Figure 5). The puppet master makes use of the configurator nodes to give the user detailed control over which parameters should be steered. To reveal its interface, the configurator has to be switched to a higher level-of-detail (Figure 6a, LOD 2). In this interface, the user selects steerable parameters for modification in WL, e.g., to make parameters of the Levees node available for branching, such as breach position or breach width. This configuration of the steering behavior is then stored inside the meta-flow. The exploration process with WL does not require any further, manual modifications in the flow diagram. Furthermore, the configurator node can be used to directly initiate branch events and parameter studies in the WL view. The computed parameter distributions of parameter studies can be visualized as histograms within the configurator node (Figure 6a). The key advantage of employing WL as a puppet master is to achieve a generic, centralized control strategy, separated from the data-flow and presented in a concise visualization. Since parameters are no longer passed through the data-flow, we do not have to overload the data-flow with all the connectors and wires that are required to transport time values, track information, or the various parameter values. The WL node has input connectors for wires as well. These connectors are present, because the WL view

Table 2. Overview of components in the meta-flow.

meta-node category	creation	requires rope (from - to)	purpose
puppet master (PM)	manual	meta - meta	controlling many nodes e.g., World Lines
meta-communicator	automatic (part of rope)	meta (of PM) - meta	Configurator: fine-tuning of control behavior per rope
		settings/results - settings	Linker: parameter linking, interaction
parameter provider	manual	settings - settings	provide item to list-based parameter e.g., Particle Emitter

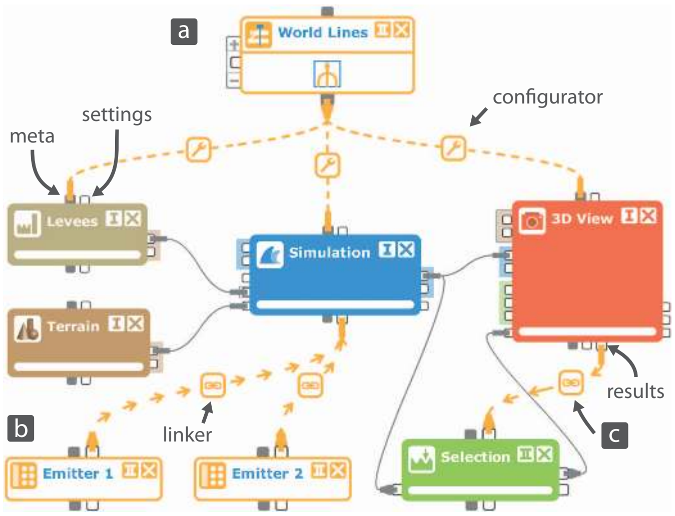


Fig. 5. The role of the meta-flow for simulation steering with World Lines (WL). (a) WL puppet master controls nodes via configurators that are part of ropes. (b) Parameter providers supply emitter settings to the fluid simulation node (blue) through linker nodes. (c) Interaction (spatial selection in fluid) accomplished through linking from the results meta-connector of a 3D view (red) to the settings connector of an upstream node (green).

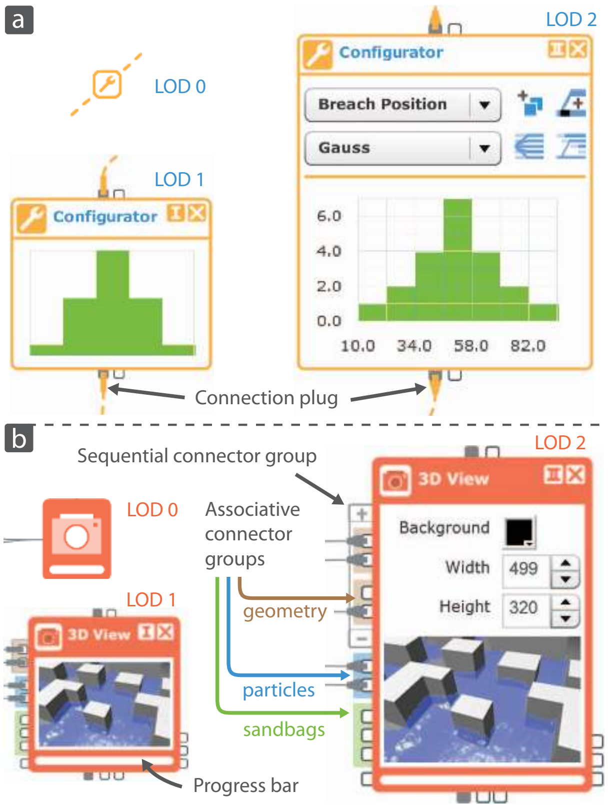


Fig. 6. Nodes at different levels of detail (LOD). (a) At higher LODs, the meta-communicator node (configurator) shows the input distribution of a parameter study (green bars) associated with the data-flow node the configurator is responsible of. (b) Data-flow node (3D View) with connector groups and inline components.

has the ability to directly visualize data-flow results through coloring of tracks and frames [3, 44]. In the presented framework, it is simple to implement additional puppet masters. For example, we employ an animation puppet master for changing parameters when creating animations. We are currently working on a unit-tester puppet master which supplies meta-connected nodes with fixed input parameters and checks their output for correctness. A perturbation puppet master adds small random changes to the parameters of all meta-connected nodes.

6.2 Parameter Linking

Voreen [28] uses connections that are separated from the data-flow, to enable parameter linking between nodes, such as coupling the camera perspective between views. We utilize the meta-flow concept to accomplish this task. Nodes are equipped with two optional meta-flow connectors, namely the *settings* connector placed on top and bottom of a node and the *results* connector, located at the bottom only since node results can only be read. The results connector allows for linkage to information that has been generated by a node during data-flow execution. The properties of the linking mechanism are customized in a special meta-communicator, the *linker*. The rope visually reflects the direction property of the linkage (directed or bi-directional) through arrow glyphs.

The introduction of *parameter providers* offers a flexible way for the specification and sharing of list-based parameters. A typical list-based parameter is the set of particle emitters of a simulation node or the set of light sources of a 3D view. A parameter provider is a meta-node that supplies its value (e.g., a single light source) as an item to the list-based parameter of meta-connected nodes. Figure 5 shows two emitter-parameter providers that are meta-connected in order to set the particle emitters of a fluid-simulation node. The advantage of this approach over existing systems is that we can supply an arbitrary number of items to a list-based parameter while sharing individual items among nodes. A standard data-flow system would require a variable number of connectors to accomplish this task.

Parameter linking of results to settings can be used to implement user interactions. In this work, we use this mechanism to enable spatial selection in views (e.g., to define an evacuation path or to select a building). When clicking into a 3D view, the underlying view node is executed and transforms the mouse coordinates into a point in 3D space. This point is available as a node result to other nodes and can be used to determine selected items in space. This approach is comparable to the upstreaming-technique in standard data-flow systems. With our approach, we avoid such changes to the data-flow model and we do not have to introduce new data-flow connectors for each parameter that the user wants to link. Moreover, for our spatial selection nodes, we provide the necessary linking specifications by default and the user only has to establish a rope from a 3D view to the node. The 3D view then automatically supplies the necessary tools (such as a picker tool) to let users accomplish the spatial selections.

7 Dynamic Visualization with the Flow Diagram

Until now, the presented meta-flow improves the user situation in terms of building a network for parameter-space navigation. Puppet masters utilize the meta-flow to transport steering information to the meta-connected nodes in a fully automated manner. To this point, the details of this communication remain transparent to the user. However, in many cases it is desirable to understand what is happening in the network at runtime while navigating in parameter space. In this section, we explain how to use the meta-flow components to visualize the control-flow between nodes, the work-flow of the user and his/her inherent knowledge. In addition, we show how to use the data-flow to present a concise overview of relevant computed data inside the flow diagram. Before we go into details about the dynamic visualization, we explain basic design decisions.

7.1 Design

Figure 6b, LOD 2, shows the design of a node in the flow diagram. The most important widgets for parameter modification are displayed inline. A node may contain an optional image to show a generated rendering or visualization inline. Meta-flow connectors are simply arranged on top and at the bottom of a node. The connector structure with respect to the horizontal data-flow tends to be more complicated and demanding. For this reason, we organize data-flow connectors internally and visually into *sequential and associative connector groups*.

Associative connector groups contain a set of connectors that are related to each other. For example, the associative connector group *geometry* comprises a mesh connector and a color connector. Another example is the connector group *particles* which contains a velocity connector and particle positions. This way it is possible to access the whole particle field and to provide quick access to the important velocity attribute. The 3D-view node in Figure 6 comprises the three dominant associative groups that are required for the flooding scenario: The *geometry* group (brown), to handle terrain, buildings and levees, the *particles* group (blue), representing water, and the *sandbags* group (green), representing the sandbags.

A sequential connector group contains a dynamically alterable number of the same connector or associative connector group. Using such dynamic input often leads to a reduced number of required nodes to accomplish a task. As an example, consider a view node like in Figure 6 that is capable to display geometry. With the sequential connector group, it is possible to connect several geometry objects to this view node without the need for a series of nodes that merge the geometry into one data structure beforehand. To our knowledge, there is currently no data-flow system that allows the user to dynamically change the number of input connectors of a node.

Connectors and connector groups are styled and shaped according to the nature of the transported information (e.g., geometry). Connection styles such as color and line style depend on connection type (meta-flow rope, data-flow wire, ropes that represent parameter links). Connections are further equipped with plugs on either side. These plugs can be dragged interactively to change connections.

7.2 Flow Diagram Simplification

To create concise, dynamic visualizations in the flow diagram, it is vital to reduce the visual clutter introduced by a potentially large number of meta-flow and data-flow components. In this section, we adapt a number of simplification strategies to reduce the information content displayed. Other systems let users combine nodes into a sub-network and place them into an aggregated node [29, 34]. In this section, we investigate other simplification strategies to reduce the information amount displayed. Since the goal is a dynamic visualization, it should be possible to perform these simplifications automatically while preserving the mental model for the user. The discussed simplifications are level-of-detail and connection reduction.

Level-of-detail (LOD) When networks become larger, it is increasingly more difficult to get an overview of the complete setup. In case of an interactive flow diagram, it is not feasible to apply a standard zooming technique where elements are simply scaled. Relevant

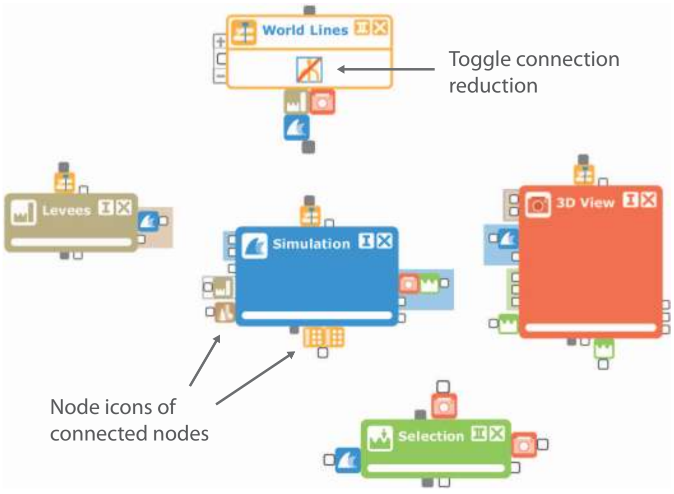


Fig. 7. Screenshot of the flow diagram in Figure 5 after applying connection reduction to all ropes and wires. If a node has only one connection, it is not visible.

information has to stay readily perceivable. The most important interaction points have to remain large enough for user interaction. This is why we propose a level-of-detail zooming approach. In this work, we are able to reduce the visual clutter of our flow diagrams significantly by introducing three different levels of detail. The appearance of a data-flow node and a meta-communicator for each LOD is depicted in Figure 6. At LOD 2, a node displays all inline elements and shows all connectors and connector groups at their full extent. At LOD 1, sequence connector groups are reduced to display only one item. All connectors are scaled down, connection plugs receive a smaller shape and some inline components are hidden. If inline space remains, the node tries to fit in an optional image or other inline components. At LOD 0, the node is completely reduced to a small icon and all connections to a node lead to the same point. Meta-communicators have a smaller size than other nodes at LOD 0 since they are part of ropes and occur quite frequently. To give an adequate overview of larger networks, we apply scaling to node positions at lower levels of detail. A simple coordination of node space takes care of avoiding node overlap when increasing the level-of-detail. Animated transitions are used to give the impression of smooth scaling.

Connection reduction The flow diagram can become confusing if all ropes and wires are visible at once. A *connection reduction* can be performed to decrease the number of connections visible at a time. In the reduced state, a connection is invisible. If one of the nodes has no other connection than the reduced one, the node is invisible as well. To retain a visual correspondence between two connected nodes, each of them displays a small icon representation of the other node (Figure 7). If the mouse is moved over the icon representation, the connection is temporarily displayed. By clicking it, the user can make the node and its connections fully visible again. Puppet masters provide an inline button to quickly toggle reduction of all their ropes at once.

7.3 Visualization of Control

Any user interaction in the system, such as a change of the perspective in a 3D view or the creation of new tracks in World Lines (WL), causes a modification of parameters in a subset of the nodes. For some nodes parameter changes occur more frequently, while others remain unchanged most of the time. This fact suggests to introduce an importance value for nodes. These values are updated after every control operation. If the node k is directly controlled, the importance value I_k is increased according to the expression

$$I_k = min(1, I_k + c_{inc} \cdot w) \qquad I_k, w, c_{inc} \in [0,1] \tag{1}$$

where c_{inc} is an empiric constant (set to 0.05) and w defines the weight of the control operation. We set this value to 1 for all operations except for changes to the camera perspective in the 3D view, where we assign a value of 0.01. This operation occurs very often and should have a smaller impact to the layout of the flow diagram. All other nodes, i.e., the nodes that are not directly affected by the control operation, receive a small decrease of their importance value I_k according to

$$I_k = max(0, I_k \cdot (c_{dec} + (1-w)^{1-c_{dec}})) \qquad I_k, w, c_{dec} \in [0,1] \qquad (2)$$

where c_{dec} is a constant we set to 0.95. While interacting, the flow diagram automatically changes shape to direct the user attention to the most important parts. To accomplish this, the aforementioned simplifications are automatically applied according to the importance values using the following lookup:

- $I_k < 0.1$: node k receives LOD 0; apply connection reduction to all ropes connected to node k
- $I_k \in [0.1, 0.5)$: node k receives LOD 0
- $I_k \in [0.5, 0.8]$: node k receives LOD 1
- $I_k > 0.8$: node k receives LOD 2

Meta-communicators are excluded from these automatic LOD modifications. If any LOD value changes a re-layout of the node network is necessary. We try to move a subset of the nodes, if possible, to fill the gained space and if required, to provide the missing space. These automatic movements respect the relative node positioning of the layout to retain the mental model. We apply animated transitions to let users more easily perceive the modifications. If the flow diagram does not fit into the screen space of the containing window, we change the window-scroll positions to put the focus on the center of importance $\vec{S}_I$, which is determined as

$$\vec{S}_I = \frac{1}{\sum_n I_k} \cdot \sum_n I_k \cdot \vec{p}_k \qquad (3)$$

where n is the number of nodes and $\vec{p}_k$ is the screen position of the k-th node. In addition, we alter the thickness of ropes to further emphasize the control-flow. If a meta-connected node is not controlled at all, we apply connection reduction. With the proposed course of action, the meta-flow supports the user in understanding the internal control operations. Moreover, the importance based display of the meta-flow offers a way to record and visualize work-flow and knowledge of the current user. The window layout in the presented tool is configurable, allowing the user to show or hide the flow diagram.

7.4 Visualization per Wire

In the WL view, we can visualize information on a per-time and per-track level. However, this information is restricted to the output of a single connector in the data-flow, or at most, to a derived quantity that describes many outputs. For example the frames can be colored by the number of flooded buildings at the given simulation step. WL cannot provide a comprehensive overview of results from several node outputs. We believe that such an overview is best shown in the flow diagram itself. However, the components of the flow diagram can only give restricted information on a per-time and per-track level. Figure 8 illustrates how WL and the flow diagram can compensate each others shortcomings. In this example, we show progress information in both views: the WL view displays which tracks are already simulated, the progress bars of the nodes display which portions of every attribute in the simulation are already computed.

Progress monitoring The presented system utilizes the data-flow processing-algorithms as described by Schindler et al. [36]. By this means, we have important frame information about the data elements each node can compute. More specifically, we know for what frames (i.e., time steps and tracks), a node can generate data. If we compare this information to the data that a node has already computed, we can define the progress per node. Figure 8 shows how the WL view indicates the processed frames by coloring them in yellow. The progress is associated with one node in the data-flow. To give an overview on the

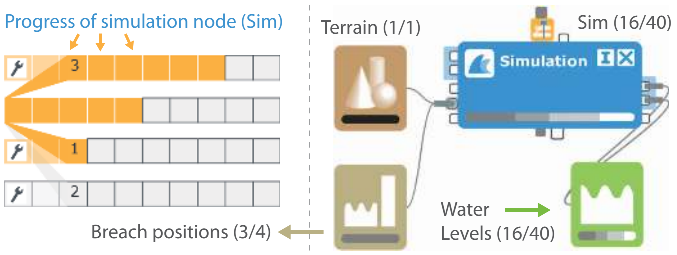


Fig. 8. Progress visualization with World Lines (WL) vs. inside the flow diagram. WL comprise 4 tracks for 4 different breach positions. WL show progress on a per-frame level, the flow diagram on a per-node level. The figure labels depict (calculated frames/frames to calculate).

system progress, each data-flow node in the diagram is equipped with a progress bar. It is straightforward to display the progress with respect to time. To also give a hint on progress related to tracks, we employ transparency inside the progress bar. This means each track is represented by a semi-transparent rectangular area which are all blended together to give the resulting progress bar.

Robustness analysis In a simulation-steering environment, we require the ability to verify whether a solution, i.e., a choice of parameters, is stable with respect to uncertain input. A robust solution is given, if the variation in the output is smaller than the variation in the input. In Section 5 we have discussed how to enter a distribution of input values. Via the configurator node, we display the input distribution inside the flow diagram. Now, to evaluate the robustness of a solution, we compute the resulting distributions for all simulation attributes the user selects to be included in the robustness analysis. These output distributions are then visualized inside the flow diagram.

The outputs that have to be included in the robustness analysis, are defined interactively by selecting output-connectors. This interaction automatically creates and connects a special data-flow node that we term *robustness analyzer* (grey nodes in Figure 9). Initially, this node is only visible through the small icon at the selected output connector (i.e., its connections are reduced to LOD 0). The robustness analyzers are in charge of computing the output distributions. The results of these calculations are visualized in several ways. To get an overview, the thickness of each data-flow wire is changed to reflect the relative standard deviation of the data which flows through the wire. If this variation is very small, the wire remains reduced and the corresponding robustness analyzer hidden. The higher the standard deviation, the thicker the wire and the higher the LOD of the associated analyzer node. At LOD 1, the analyzer displays a visualization of the distribution of data which flows through the respective wire. This can be done using an information-visualization technique such as a histogram or an ensemble line graph. At LOD 2, this information visualization provides even more details. We point out that this flow-diagram visualization itself is updated during the ensemble simulation.

8 Evaluation

In this section we demonstrate the approach on the flooding scenario introduced in Figure 1. For details on the simulation setup, we refer to the WL case study [44]. For this paper, we have rearranged the buildings and increased the size of the terrain to $140 \times 140\,m^2$.

8.1 Case Study Tasks

As a preliminary task, the domain expert defines three alternative precautions to protect the neighborhood from flooding. We employ WL to create a track for each of these multi-barrier arrangments. Figure 9a contains images to show the difference between the arrangments.

Before we can create a parameter study with respect to uncertain levee-breach locations as demanded by task T1 (Figure 1, T1), we have to extend the parameter space of WL. This is accomplished by establishing a rope from the related puppet master to the levees node. In the auto-created configurator of the rope, we select the breach-position parameter and add it to the parameter space. Now it is possible to

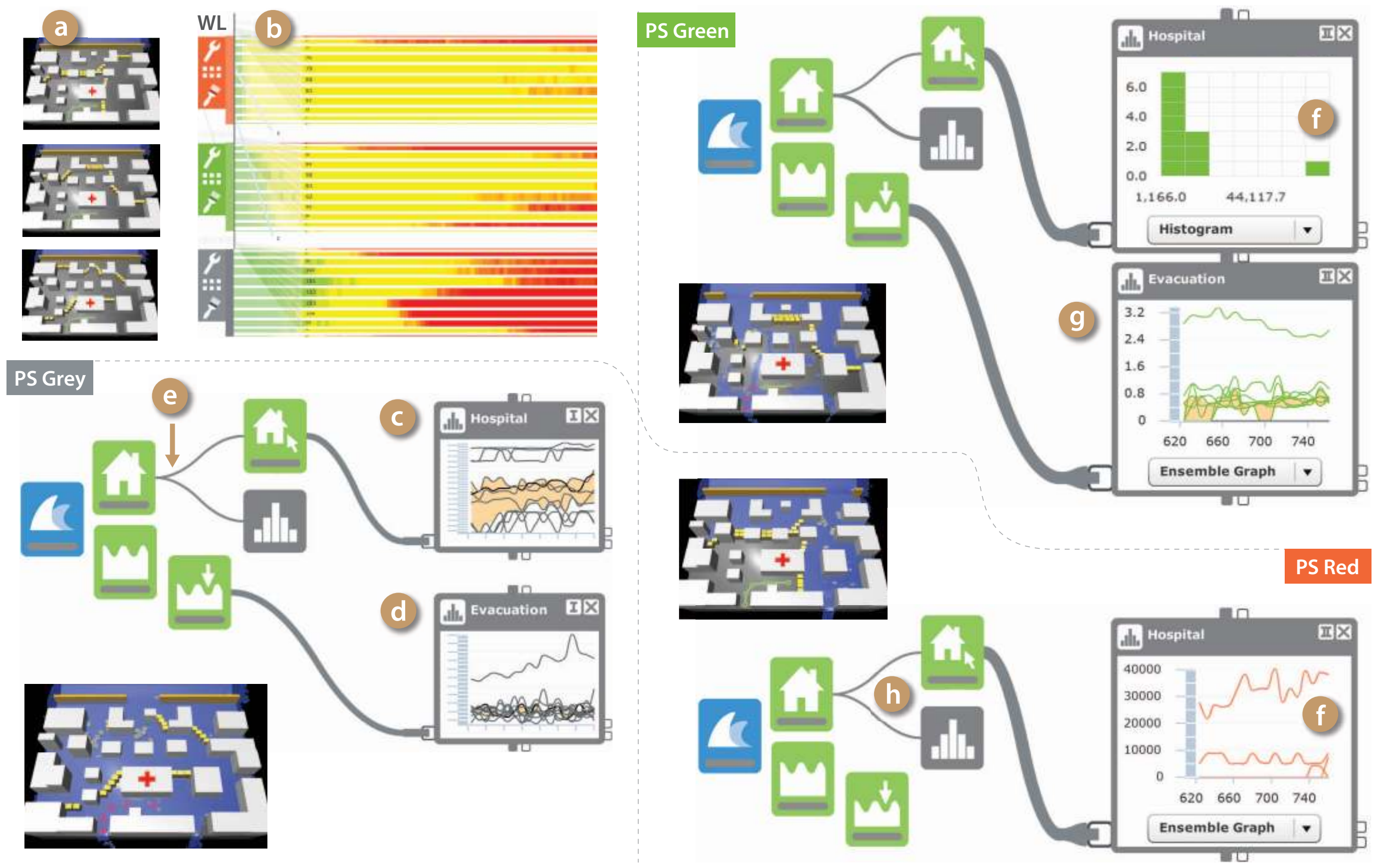


Fig. 9. Robustness analysis of three response strategies to protect a neighborhood from immenent flooding. The images in the upper left depict the three barrier arrangments. World Lines (WL) show the global damage. The flow diagram shows detailed risk curves for the damage of the hospital and the water levels along the evacuation path. A comparison reveals that PS Red performs best with respect to damage and safety.

account for uncertain levee-breach locations by creating a parallel parameter study for each of the three tracks respectively. Each track in the parameter study has a duration of 12 *min*, 40 *sec*. The breach-position values are sampled according to a Gaussian distribution with a mean value of 52 *m* and a standard deviation of 12 *m*. The sample density is 11, resulting in a total number of 33 tracks to be simulated. The resulting Gaussian distribution can be inspected in the related configurator node. For the robustness-analysis procedures, we need to be able to inspect the results for one parameter study at a time. In the flow diagram, one parameter study is considered active at a time, identified by the frames that are currently brushed in the WL view. The three parameter-study cursors are visually distinct with respect to coloring (Figure 9). We use this color in the visualizations of the configurator and the robustness analyzers to clearly identify the active parameter study. In the remainder of this section, we refer to a parameter study by the color of its associated cursor and term them PS Grey, PS Red and PS Green respectively.

As a next step, the flow diagram is extended to fulfill the requirements of task T2 (Figure 1, T2). We add the required nodes for building-damage estimation and for the calculation of water levels. For both output quantities, we require spatial selection nodes that we meta-connect to the results of the 3D view. Hereby, we enable inspection of water levels and of damage at user-defined points in space. We select the hospital to enable monitoring of its individual damage calculations. The regression model used for building-damage estimation in Euros is based on investigations by Thieken et al. [42]. The given formula requires a value of household content which can be entered by the user. We set this value particularly high for the hospital. After this, the domain expert defines the evacuation path through mouse clicks in the 3D view. The related spatial selection node is then capable to output the water levels in meters in the vicinity of the evacuation path. At

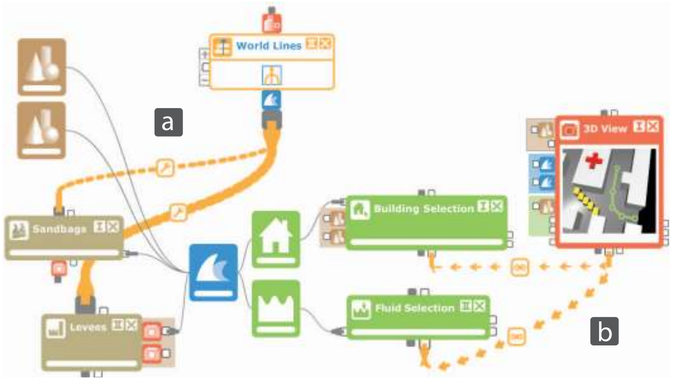


Fig. 10. Visualization of control after the user has setup the system according to the requirements of the case study. The sandbags node and the levees node (ochre) provide the most important parameters to the simulation node (blue). The user did spatial selections through the 3D View (red) to prioritize the hospital and define the evacuation path.

this point, the flow diagram has changed shape several times to reflect the user work-flow and control operations (Figure 10). The emphasis is put on the ropes from the puppet master to the sandbags node and the levees node (Figure 10a) which receive a high level-of-detail. The two thick ropes originating from the results connector of the 3D view clearly highlight the importance of spatial selection (Figure 10b). To be able to produce the risk visualizations as required by task T3 (Figure 1, T3), we have to create a robustness-analyzer node for each data-flow output of interest. We choose the global damage, the damage to the hospital as well as the water levels along the evacuation path. After this, we are ready for simulation. WL provide a scheduling scheme that allows for real-time monitoring of results. During simulation, the risk curves in the flow diagram are regularly updated. Thus, we can reason as soon as simulation data is available. The total simulation time is approximately 50% of real-time.

To produce the visualizations in the flow diagram, we brush all frames of a parameter study that lie within the time interval of 620 *sec* to 760 *sec*. For each track in the distribution, the robustness analyzer averages results across this time interval to calculate the standard deviation and the histogram. The ensemble graphs plot the temporal evolutions of quantities for each track and display a confidence interval (light orange area, see Figure 9c). The final results are shown in Figure 9. In the following, we use the letters a-h to direct the readers attention to the relevant parts in the Figure. PS Grey clearly fails with respect to all requirements. The damage to the hospital is high in most of the tracks (c). The evacuation path cannot be considered safe, no matter where the levee-breach takes place (d). The standard deviation with respect to the global damage estimation is low (e) since in all cases, the neighborhood suffers from severe flooding. For this reason, the related robustness analyzer shows no detailed information. A quick peek into the visualization in the WL view (b) verifies the found results. Here, frames are colored according to the global damage (green=low, red=high). Through the WL visualization alone, we cannot say whether the solution tested with PS Red or the one tested with PS Green is more robust. In both cases, the damage to the complete neighborhood is high at breach locations that are farther off the most likely one. The hospital-damage visualizations in both flow diagrams comprise an outlier that puts the hospital in danger (f). However, the precautionary measures tested with PS Green are incapable of keeping the evacuation path water free (g). On the other hand, the flow diagram of PS Red (h) does not show an output distribution for the water levels at the evacuation path at all. This is due to the fact that the related barrier arrangement protects the evacuation path with respect to each value in the parameter study. Even though we need to choose a response strategy that puts the hospital in danger in case of an unlikely outcome, we are able to keep the people in the hospital safe.

8.2 Domain Expert Feedback

The case study was assessed by an expert with experience in flood forecasting and management systems [13]. It was immediately clear to him that the main strength of Nodes on Ropes (NR) is the ability to account for the uncertainty in the forecasts and management implications. Uncertainty can come from weather predictions and uncertain locations of levee breaches, among other factors. Having an understanding of the uncertainty is a key criterion in flood management decisions as one usually chooses robust flood management designs. One attempts to minimize the risk of making a poor decision that may possibly aggravate the adverse effects of a flood on infrastructure and people. NR allows the user to monitor the evolution of the uncertainty in time with respect to different outputs (such as the accessibility of the evacuation path) which makes it ideally suited for the uncertainty assessment by flood managers. NR is more flexible than WL as one can assign boundary conditions (such as the potential levee-breach position) as well as a range of other parameters. This is important for real-world applications as they always differ on a case by case basis. Another advantage over WL is that the data and control-flow has now become transparent. The modular design makes it easy for the flood manager to change parameters related to the inundation in real time. Also, the flow of uncertainty has become transparent which helps to detect the sources of uncertainties in the decision making. The system could be used for real-time application during a flood event. The real-time updating of the ensemble forecasts in a separate window is an interesting feature that helps keep track of the evolution of the forecasts and their uncertainty.

9 Implementation

Nodes on Ropes is part of the Visdom [2] system, comprising WL and the presented approach. The separation of meta-flow and data-flow is reflected by the client-server architecture of the system. The meta-flow is completely evaluated on the client, the data-flow on the server. Parameters are represented in XML, enabling a flexible and generic implementation of all meta-flow responsibilities. When a control operation on the client side is initiated (e.g., due to a user interaction), the meta-flow is executed first. During this process, a global synchronizer instance bundles the changed data-flow parameters into one XML request structure. When the meta-flow is complete, this request is sent to the server. The server is in charge of executing the data-flow which involves the compute-intensive parts of the process such as simulation or rendering. We leverage GPU power to perform these processes in real-time. Only images and results are sent back to the client. The results are then available for further processing through the meta-flow connector results. The client is a light-weight application implemented in Actionscript that runs in a browser. Hereby, we are working towards our vision for the future to provide mobile decision support on-site.

10 Conclusion

So far simulations have been predominantly concerned with the data-flow. The control-flow was often simple and/or hardwired. Increasingly complex simulations are characterized by an intricate interplay of various heterogeneous components. The elaborate and dynamic nature of the evolving control-flow requires novel visualization and interaction functionality. The goal of this paper is to tackle three important questions related to interactive simulation steering: How can we investigate uncertainties with respect to input parameters? How can we visualize what is going on in a simulation? How can the user modify the simulation setup and extend the explored parameter space? To answer these questions we suggest the extension of the classical data-flow concept by four novel schemes: First, we propose to add a configurable meta-flow as an abstraction for control- and information-flow between nodes. Second, based on the meta-flow, we suggest interactive parameter studies through World Lines. One benefit of this approach is the possibility to create customized steering mechanisms which combine visualization, simulation design, and parameter-space navigation. Hereby, the generic data-flow approach can become as user friendly as special purpose turn-key systems. In the general setup we retain the concept of the data-flow in a consistent manner, even though complex node interactions are possible using the meta-flow. Third, we suggest puppet-master nodes which can control other nodes via ropes. The concept of puppet masters is our answer to the rising complexity when interactive steering comes into play. System complexity involves a trade-off between simplicity and power of the system. At one end of the spectrum, using a programming language like C++, one can set up a complete dynamical system and control any detail of its behavior. At the other end of the spectrum, there are ready-made tools for simulation and visualization which are easier to handle than a programming language. In this paper, we suggest a system which minimizes the capabilities required from the user while retaining complex system behavior at the same time. This is possible by allowing the user to perform modifications via puppet masters. Fourth, we suggest to use the flow diagram itself for visualization. The diagram can highlight important system components and display relevant information about statistical distributions in a parameter study which are difficult to show using World Lines alone.

For future work, we consider the automatic analysis of simulation steps as an important tool to control which simulations can be kept running and which should be stopped. For large parameter studies, early simulation termination can save a lot of computational resources.

The contributions of this paper are guided by the basic insight that a more complex framework needs smarter tools to control its features. By augmenting the data-flow by an elaborate meta-flow, it becomes possible to not only use the data-flow for building the application, but to learn from it as well.

Acknowledgments

This work was supported by grants from the Austrian Science Fund (FWF): P 22542-N23 (Semantic Steering) and P 21695-N23 (ViMal), and a grant from the Future and Emerging Technologies (FET) programme of the European Commision: 226042 (SemSeg).

REFERENCES

[1] Mevislab: A development environment for medical image processing and visualization. http://www.mevislab.de/ (last visited on March, 17th 2011).

[2] Visdom - An integrated visualization system. http://visdom.at (last visited on May, 11th 2011).

[3] World Lines Video. http://visdom.at/media/slides/world_lines.mp4 (last visited on March, 25th 2011).

[4] Advanced Visual Systems Inc. AVS - Advanced Visual System. http://www.avs.com/ (last visited on March, 21st 2011).

[5] A. Amirkhanov, C. Heinzl, M. Reiter, and M. E. Gröller. Visual optimality and stability analysis of 3dct scan positions. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1477 –1487, 2010.

[6] ANSYS, Inc. *Explicit Dynamics, Chapter 10: Optimization Studies*. http://www.cadfamily.com/download/CAE/ANSYS-Explicit/Explicit_Dynamics_Optimization_Studies.pdf (last visited on March, 2nd 2011), 2009.

[7] H. Apel, A. Thieken, B. Merz, and G. Blöschl. Flood risk assessment and associated uncertainty. *Natural Hazards and Earth System Science*, 4:295–308, 2004.

[8] H. Apel, A. Thieken, B. Merz, and G. Blöschl. A probabilistic modelling system for assessing flood risks. *Natural Hazards*, 38:79–100, 2006.

[9] L. Bartram, A. Ho, J. Dill, and F. Henigman. The continuous zoom: A constrained fisheye technique for viewing and navigating large information spaces. pages 207–215. ACM Press, 1995.

[10] W. Berger, H. Piringer, P. Filzmoser, and E. Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum*, 30:911–920, 2011.

[11] Bestiario company. Impure - A web-based visual programming tool written in ActionScript to visualize data from the internet. http://www.impure.com/ (last visited on February, 8th 2011).

[12] I. Bitter, R. Van Uitert, I. Wolf, L. Ibanez, and J.-M. Kuhnigk. Comparison of four freely available frameworks for image processing and visualization that use itk. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):483–493, 2007.

[13] G. Blöschl. Flood warning - on the value of local information. *Intl. J. River Basin Management*, 6(1):41–50, 2008.

[14] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1467–1475, 2010.

[15] G. Cameron. Modular visualization environments: past, present, and future. *ACM SIGGRAPH Computer Graphics 1995*, 29(2):3–4, 1995.

[16] Centre for Research on the Epidemiology of Disasters (CRED). 2010 disasters in numbers. http://cred.be/sites/default/files/PressConference2010.pdf (last visited on March, 3rd 2011).

[17] M. Chen, D. Ebert, H. Hagen, R. S. Laramee, R. van Liere, K.-L. Ma, W. Ribarsky, G. Scheuermann, and D. Silver. Data, information, and knowledge in visualization. *IEEE Computer Graphics and Applications*, 29:12–19, January 2009.

[18] H. Childs, E. Brugger, K. Bonnell, J. Meredith, M. Miller, B. Whitlock, and N. Max. A contract based system for large data visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 190–198, 2005.

[19] H. Cloke and F. Pappenberger. Ensemble flood forecasting: a review. *Journal of Hydrology*, 375(3-4):613–626, 2009.

[20] C. Elliott, V. Vijayakumar, W. Zink, and R. Hansen. National instruments labview: A programming environment for laboratory automation and measurement. *Journal of the Association for Laboratory Automation*, 12(1):17–24, 2007.

[21] W. Felger and F. Schröder. The visualization input pipeline - enabling semantic interaction in scientific visualization. *Computer Graphics Forum*, 11:139–151, 1992.

[22] R. B. Haber and D. A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. *IEEE Visualization in Scientific Computing*, pages 74–93, 1990.

[23] W. M. Johnston, J. R. P. Hanna, and R. J. Millar. Advances in dataflow programming languages. *ACM Computing Surveys*, 36(1):1–34, 2004.

[24] Kitware. Paraview - an open source, multi-platform data analysis and visualization application. http://www.paraview.org/ (last visited on March, 17th 2011).

[25] J. Komma, C. Reszler, G. Blöschl, and T. Haiden. Ensemble prediction of floods - catchment non-linearity and forecast probabilities. *Natural Hazards and Earth System Sciences*, 7:431–444, 2007.

[26] M. Marttila-Kontio and R. Honkanen. Not-so-free data flow in a visual data flow programming language. *Computer Science and Information Technology, International Conference on*, pages 613–619, 2009.

[27] K. Matković, H. Hauser, R. Sainitzer, and M. E. Gröller. Process visualization with levels of detail. In *Proceedings IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pages 67–70, 2002.

[28] J. Meyer-Spradow, T. Ropinski, J. Mensmann, and K. H. Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications*, 29(6):6–13, 2009.

[29] National Instruments (NI). LabVIEW: A graphical programming environment for engineers to develop measurement, test, and control systems. http://www.ni.com/labview (last visited on March, 21st 2011).

[30] Numerical Algorithms Group (NAG). IRIS Explorer: A visual programming environment to develop, prototype and build visualization applications. http://www.nag.co.uk/Welcome_IEC.asp (last visited on March, 21st 2011).

[31] I. P. on Climate Change (IPCC). Ipcc fourth assessment report: Climate change 2007 (ar4). http://www.ipcc.ch/publications_and_data/publications_and_data_reports.shtml (last visited on March, 17th 2011), 2007.

[32] S. G. Parker and C. R. Johnson. SCIRun: A scientific programming environment for computational steering. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, page 52, 1995.

[33] A. M. Pérez, J. M. Gómez, and R. C. Pérez. Semantic interaction in enterprise data-flow visualization environments: An exploratory study. In *ICT Innovations 2009*, pages 217–226. Springer Berlin Heidelberg, 2010.

[34] E. Santos, L. Lins, J. Ahrens, J. Freire, and C. Silva. VisMashup: Streamlining the creation of custom visualization applications. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1539–1546, 2009.

[35] C. Scheidegger, H. Vo, D. Koop, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560 –1567, 2007.

[36] B. Schindler, J. Waser, R. Fuchs, and R. Peikert. Multiverse data-flow control. Technical Report 720, ETH Zürich Comp. Science, 2010.

[37] Scientific Computing and Imaging Institute (SCI). SCIRun: A scientific computing problem solving environment. http://www.scirun.org (last visited on March, st 2011).

[38] C. T. Silva, J. Freire, and S. P. Callahan. Provenance for visualizations: Reproducibility and beyond. *Computing in Science and Engineering*, 9(5):82–89, 2007.

[39] M. G. Stewart and R. E. Melchers. *Probabilistic risk assessment of engineering systems*. Chapman and Hall, London, 1997.

[40] A. Telea and J. J. van Wijk. Vission: An object oriented dataflow system for simulation and visualization. In *Proceedings IEEE Symposium on Visualization (VisSym 1999)*, pages 95–104, 1999.

[41] A. Telea and J. J. van Wijk. Smartlink: An agent for supporting dataflow application construction. In *Proceedings IEEE Symposium on Visualization (VisSym 2000)*, pages 189–198, 2000.

[42] A. H. Thieken, M. Müller, H. Kreibich, and B. Merz. Flood damage and influencing factors: New insights from the August 2002 flood in Germany. *Water Resources Research*, 41:16pp, 2005.

[43] C. Upson, T. Faulhaber Jr., D. Kamins, D. H. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9:30–42, July 1989.

[44] J. Waser, R. Fuchs, H. Ribičić, B. Schindler, G. Blöschl, and M. E. Gröller. World Lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1458–1467, 2010.

[45] C. Weaver. Building highly-coordinated visualizations in Improvise. In *Proceedings IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 159–166, 2004.

[46] C. Weaver. Visualizing coordination in situ. In *Proceedings IEEE Symposium on Information Visualization 2005 (InfoVis 2005)*, pages 165–172, 2005.

[47] H. Wright, K. Brodlie, and M. Brown. The dataflow visualization pipeline as a problem solving environment. In *Proceedings of the Eurographics Workshop on Virtual Environments and Scientific Visualization 1996*, pages 267–276, 1996.

[48] H. Wright and J. Walton. Hyperscribe: A data management facility for the dataflow visualization pipeline. In *IRIS Explorer Technical Report IETR/4, NAG Ltd*, 1996.