Run Watchers: Automatic Simulation-Based Decision Support in Flood Management

Artem Konev, Jürgen Waser, Bernhard Sadransky, Daniel Cornel, Rui A.P. Perdigão, Zsolt Horváth, and M. Eduard Gröller

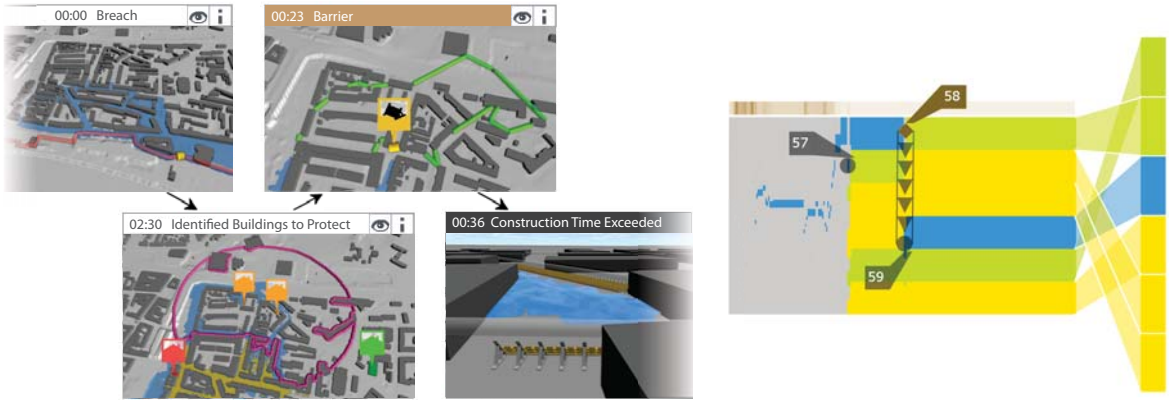

Fig. 1. (Left) Breach scenario in an urban area imposes severe time limitations. Based on flood simulations, Run Watchers automatically compute whether a response plan exists that protects the important infrastructure and, possibly, other buildings. Automatically generated visualization justifies the computed action plans. (Right) During the computation, Run Watchers generate large decision trees. The decision cluster visualization allows the user to navigate the whole set of decisions.

Abstract— In this paper, we introduce a simulation-based approach to design protection plans for flood events. Existing solutions require a lot of computation time for an exhaustive search, or demand for a time-consuming expert supervision and steering. We present a faster alternative based on the automated control of multiple parallel simulation runs. Run Watchers are dedicated system components authorized to monitor simulation runs, terminate them, and start new runs originating from existing ones according to domain-specific rules. This approach allows for a more efficient traversal of the search space and overall performance improvements due to a re-use of simulated states and early termination of failed runs. In the course of search, Run Watchers generate large and complex decision trees. We visualize the entire set of decisions made by Run Watchers using interactive, clustered timelines. In addition, we present visualizations to explain the resulting response plans. Run Watchers automatically generate storyboards to convey plan details and to justify the underlying decisions, including those which leave particular buildings unprotected. We evaluate our solution with domain experts.

1 Introduction

Unpredictable aspects of natural disasters can have a devastating effect on the safety of population and infrastructure. In flood management, an unexpected failure of standard protection measures causes a situation where decisions should be made and executed within a very short time frame. Thus, it is important to have a well-prepared response plan that provides maximal protection in view of severe time limitations.

In Cologne, the primary flood protection line is based on the flood walls along the Rhine river. Our collaboration partners from the Flood Protection Center are concerned with scenarios involving a breach in the flood walls. In such cases, they would like to know whether or not (and how) it is possible to quickly construct from mobile water barriers a secondary *protection line* that covers as large an area as possible and guarantees the safety of what they consider the important infrastructure. If no protection line can be built, the flood managers would like to have an *object protection* plan. The goal of this paper is to provide flood managers with an automated decision support tool that can generate infrastructure protection plans with maximum possible coverage in the light of time and resource limitations. The tool should be able to illustrate the decision making process and justify the decisions made at every step. Justification is important for the verification of decisions and explaining them to the authorities or stakeholders.

In order to find a good protection plan, one needs to simulate multiple barrier locations and type variations. This leads to a large search space that can grow exponentially with the number of barrier locations involved. One approach is to manually steer the search [32]. However, this is tedious and requires a lot of user interaction. Alternatively, one can use exhaustive sampling of the search space with multidimensional ensembles of simulation runs [33]. Similar to other brute-force approaches, this can take weeks of computation time for real-world problems. In this paper, we suggest an approach that can be seen as filling the gap between the above two strategies, i.e., automated search space sampling guided by domain-specific rules. We introduce Run Watchers, which are program agents capable of managing multiple parallel simulation runs. Run Watchers can monitor the ongoing simulations of protection plans, terminate those where the protection fails,

- *Artem Konev is with VRVis Vienna. E-mail: konev@vrvis.at.*
- *Jürgen Waser is with VRVis Vienna. E-mail: jwaser@vrvis.at.*
- *Bernhard Sadransky is with VRVis Vienna. E-mail: sadransky@vrvis.at.*
- *Daniel Cornel is with VRVis Vienna. E-mail: cornel@vrvis.at.*
- *Rui A.P. Perdigão is with TU Vienna. E-mail: perdigao@hydro.tuwien.ac.at.*
- *Zsolt Horváth is with TU Vienna. E-mail: zhorvath@vrvis.at.*
- *M. Eduard Gröller is with TU Vienna. E-mail: groeller@cg.tuwien.ac.at.*

Manuscript received 31 March 2014; accepted 1 August 2014; posted online 13 October 2014; mailed on 4 October 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

and create new, more "promising" ones, thus performing an efficient traversal of the search space and saving computation time.

Run Watchers' activities can be represented as decision trees. However, for real-world scenarios, such trees can be large and have a complex structure. Scalable visual representations are needed to facilitate navigation through Run Watchers' decisions and resulting response plans (Figure 1, right). To handle this issue, Run Watchers create storyboard visualizations of the automated decision making process, displaying the reasons for particular decisions and emphasizing the actual response plan. Moreover, it is possible to generate and present the rationale for hard decisions of *not* protecting particular objects. This is necessary if the protection of some less important buildings is compromised in favor of more important ones. The storyboards are interactive and provide multiple levels-of-detail (Figure 1, left).

The case study addressed in this paper deals with a breach in flood protection walls at the time of a high water. If a breach happens, the water starts flooding the area immediately, giving the city services no time to protect the nearby neighborhood. However, protection is still possible for more distant areas. In our case, we attempt to protect the area where the flood wave arrives after some guaranteed response time, i.e., the maximum time needed to detect a breach, alert the emergency services, and start implementing a response plan. The domain experts from Cologne pose the following questions:

Q1: For a given breach incident, does a secondary protection line exist that can be constructed in time? If multiple protection lines are feasible, which one is the best? What is the detailed construction plan?
Q2: If no such line exists, is there an object protection that can be done in time? What is the construction plan for it?
Q3: What is the rationale for a particular plan? Why is it not possible to protect some of the buildings in the area?

The solution presented in this paper provides domain experts with the answers to the above questions. Little user interaction is required, since the decision making and generation of corresponding visualizations and reports are performed automatically by Run Watchers. We summarize our contributions as follows:

- Automated rule-based simulation control for fast and efficient parameter space traversal
- Scalable visualization of multiple simulation runs using condensed decision clusters
- Automatically generated storyboards for displaying derived response plans and explaining the underlying decision making process
- Workflow speed-up for setting up the problem and finding a solution

2 Related Work

Multidimensional parameter spaces require elaborated techniques for their exploration. With exhaustive approaches, distributions of parameter values are sampled. Booshehrian et al. [5] use ensemble simulations to support fisheries management. Waser et al. [33] compute a pool of flood response plans for multiple incident scenarios. Torsney-Weir et al. [31] sparsely sample the parameter space of an image segmentation algorithm to guide the user through a tuning cycle. Bruckner and Möller [8] randomly pick parameter sets for fluid simulation to facilitate the result-driven design of visual effects.

In a simulation-based search, high computational costs per sample constrain the exhaustive exploration. Manual exploration relies on the user's expertise in picking better samples. Effective monitoring and control of running simulations have been relevant issues in computational steering for over a decade [23]. World Lines [32] give control over multiple parallel simulation runs and allow for various types of aggregative and comparative analysis [25]. Matković et al. [19] use computational steering to speed up the prototyping of automotive system components. Santos et al. [27] assist the user in identifying and

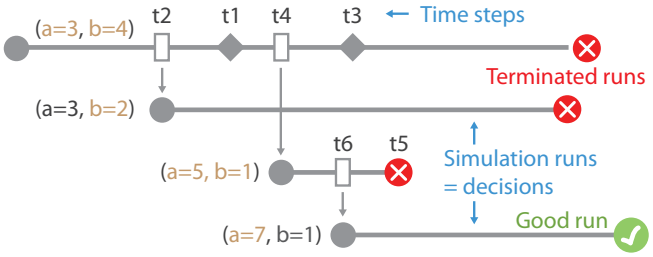

Fig. 2. Multiple simulations running in parallel are managed. Simulation runs originate from other simulation runs, sharing simulation states. Some runs are terminated in the course of search, others proceed until the end. The surviving ones are considered successful.

analyzing characteristic isosurfaces in turbulence combustion simulations. Borrmann et al. [6] create a collaborative steering environment for designing ventilation and illumination. Eickermann et al. describe a visualization and steering extension to supercomputer interfaces [10].

Computational steering requires a lot of user attention. In this context, systems providing automated simulation monitoring and control are of interest. Kapadia et al. [14] automatically compute time-varying metrics to identify unwanted patterns in multi-agent simulations. Swan II et al. [29] assist the user with parameter tuning to produce desired behaviors in simulations of microwaves penetrating missile bodies. A variety of works describe systems that automatically vary intrinsic simulation parameters such as mesh resolution and time step, or even the application itself [3, 9, 15, 21, 26]. Most of them manipulate a single simulation run. The co-simulation approach operates the simulations of multiple subsystems of the target system [11, 24, 36]. Yau et al. [35] automatically manage parallel simulations for a helium model validation study using the SimX system. They utilize application-specific knowledge to re-use simulation results and to improve the performance. To our knowledge, no decision support system exists for flood management that automatically controls parallel simulations, trying multiple protection measures in an efficient way.

Visual storytelling is an acknowledged way to present stories through interactive visualizations. Lundblad and Jern [17] utilize visual storytelling techniques to explain variations of economic performance and social indicators within and between countries. Jern et al. [13] embed semi-guided interactive visualizations into standard HTML documents, thus letting the users analyze data and save the insights for sharing. Lu and Shen [16] generate interactive storyboards from sample volume renderings and descriptive geometric primitives gained through a data analysis process. Wohlfart and Hauser [34] suggest that, along with comprehensibility of a visualization message, credibility should be addressed as well. They present an approach to volume visualization that delivers not only the results, but also the creation process.

Providing interactive visualizations gives users freedom to explore, yet lessens the control that visualization designers have over the actual course of the presented story. Ma et al. [18] suggest that a compromise might be to start with displaying the most salient features, and then to allow the user to explore the rest of the data. Using levels-of-detail in interactive visualizations can handle this. Matković et al. [20] introduce levels-of-detail in virtual instruments for process monitoring. Tominski et al. [30] visualize multiple time-dependent attributes on maps and hide information to balance the amount of data displayed. Balzer and Deussen [4] employ levels-of-detail techniques for displaying complex clustered graph layouts in 2D and 3D.

3 Automated Decision Making with Run Watchers

In this section, we present Run Watchers, the fundamental components of our solution realizing an automated decision making. First, we introduce the basic concepts and functions of Run Watchers. Further on,

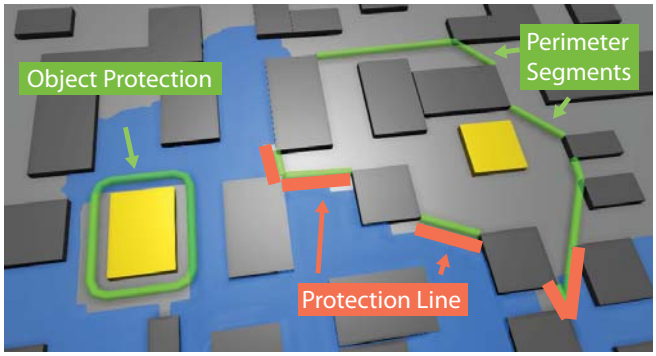

Fig. 3. Problem setup. Example protection line and object protection are shown. A perimeter is a closed line around one or more important buildings. Perimeter segments are possible locations for water barriers. A set of barriers forms a protection line. A smallest possible protection line corresponds to an object protection.

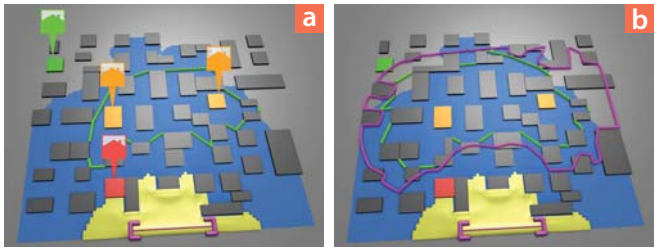

Fig. 4. Initial perimeter derivation. (a) Important buildings are labeled. The yellow area marks the flood extent after the guaranteed response time, the blue area shows the region flooded at the end of the event duration. From the four important buildings, the red one is classified as unprotectable, the green one is safe, the orange ones can and need to be protected. The green line shows the initial perimeter derived with an importance radius of 50 m around the orange buildings. (b) An importance radius of 80 m leads to a larger initial perimeter. The violet line shows the concave hull used for deriving the initial perimeter.

we explain the details of the approach and describe the rules defining the Run Watchers' behavior.

3.1 Concepts and Functionality

Every simulation run is essentially a sequence of time steps. Run Watchers control multiple simulations that run in parallel, having access to all simulation data at every time step. They have authorities to terminate simulation runs or create new ones originating from any time step of any existing run by modifying simulation parameters. The mechanism of originating simulations from other simulations allows for re-use of simulation states and thus saves computation time. At every simulated time step, Run Watchers inspect the data and make decisions whether to leave the simulation running or to terminate it, and whether a new simulation run should be initiated. If a new simulation run is needed, Run Watchers decide how the parameters should be changed, find the proper simulation run and time step to originate from, and launch the new simulation.

When creating a new simulation originating from an existing one, Run Watchers can terminate the original simulation run or let it continue. Generally, this depends on the problem specifics. In our case, Run Watchers operate in two modes. In the normal mode, the original simulation is kept running. In the greedy mode, the original simulation is terminated as the new run is started. This results in having only one simulation run at a time and leads to a much faster search. However, this approach also cuts off possible solutions. In the end, the greedy mode delivers at most one successful solution.

Simulation runs can be visually represented by horizontally oriented timelines, as in Figure 2. In the demonstrated example, we assume that simulations are parametrized by two quantities, a and b. The first simulation run starts with $a = 3$ and $b = 4$. The Run Watcher monitors every time step of the simulation and at the time step t_1 detects a reason to start a new simulation with $b = 2$. It tracks the simulation back and finds the proper time step t_2 to originate the new simulation run. The two simulations for $(3,4)$ and $(3,2)$ are now running in parallel. The simulation data until the time step t_2 is re-used for both runs. Subsequently, at the time step t_3 of the run $(3,4)$, the Run Watcher detects a reason to create a simulation for $a = 5$ and $b = 1$. The proper time step for this is t_4, and the run $(5,1)$ is added to the other two. At the time step t_5 of the run $(5,1)$, the Run Watcher detects a reason for termination, as well as a reason to launch a new run $(7,1)$ originating from t_6. Eventually, the Run Watcher terminates $(3,2)$ and $(3,4)$. Only the run (7,1) survives until the end, which we consider a success. The successful scenario can be extracted from the corresponding decision history. This is: use $a = 3$ and $b = 4$ until the time t_4, then change a to 5 and b to 1, and finally, at time t_6, set a equal to 7.

Although Run Watchers require no user supervision of the decision making process, the decision makers may want to try some specific decisions, too. Our solution provides this functionality, allowing the user to manually initiate new decisions at any time step of any simulation run and thus to answer emerging "what-if" questions. Summing up these two aspects, we provide a highly automated decision support tool, where domain experts are involved in the problem setup and optional computational steering. The actual plan execution and coordination is, however, solely the decision makers' task.

3.2 Perimeters

The search of a good protection line is based on the notion of perimeters. We define a perimeter as a polyline whose vertices correspond to buildings and whose segments are potential candidates for barrier placement. Since the flood water can enter the region of interest from any side, perimeters should be closed. Additionally, we assume that perimeters contain no self-intersections. In the course of search, Run Watchers place barriers at some perimeter segments according to simulation results, thus obtaining a candidate protection line.

Normally, a perimeter is meant to enclose a set of buildings. Object protection corresponds to a special case of a perimeter that encloses just one building. This can be seen as the "last chance" to protect an important infrastructure object in cases when no larger perimeters succeeded (Figure 3).

3.3 Initial Perimeter

The search starts with an initial perimeter. The initial perimeter is derived automatically from the user-defined parameters. These parameters are: guaranteed response time, event duration, set of important buildings, and a number specifying the radius of a circular area that the flood managers wish to protect around each important building (importance radius). To compute the initial perimeter, Run Watchers initiate a first, single simulation of the breach event with no barriers placed. The flood is simulated for the specified event duration. The flood wavefronts at the end of the guaranteed response time (initial wavefront) and, subsequently, at the end of the emergency event duration (final wavefront) are used to classify the important buildings picked by the domain experts. We assume that the area within the initial wavefront cannot be protected, and therefore leave all important infrastructure in this area out of consideration. The important buildings outside the final wavefront are considered safe, hence no protection for them is needed. Only the important buildings in the area between the two wavefronts have to be considered. From now on, when referring to important buildings, we mean these buildings only.

For circular areas of the given radius around all important buildings, the common convex hull is computed. It is then intersected with the region between the initial and final wavefronts (Figure 4). Finally, each circular area that is partly outside the final wavefront is added. For all the buildings in the resulting region, a perimeter should be computed. This computation involves a concave hull subroutine based on

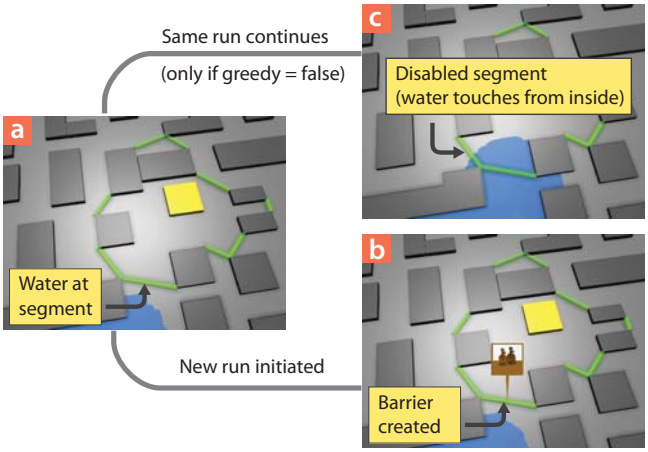

Fig. 5. (a) When water is detected at a perimeter segment, a new run is initiated by placing a barrier at that segment (b). In the greedy mode, the original run is terminated. In the normal mode, the original run continues without a barrier. Subsequently, the flowing water can touch other segments from inside the perimeter, and they should be disabled to prevent placing barriers there (c).

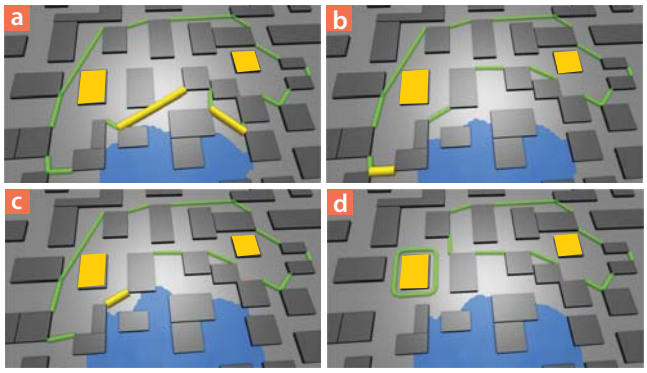

Fig. 6. Perimeter shrinking and splitting. If barriers are overtopped or cannot be built in time, Run Watchers remove the respective perimeter segments and consequently shrink the area to protect. (a) Water touches the yellow segments. The depth is too big, and no barrier type can be applied. (b) The perimeter is shrunk by removing the faulty segments. (c) One more segment has to be removed due to overly big water depths. (d) If the important buildings (yellow) cannot be enclosed by the same perimeter anymore, the perimeter is split. Here, the left perimeter degrades to an object protection, while the right one still offers a protection line for a set of buildings.

the algorithm presented by Moreira and Santos [22] with the parameter value $k = 7$. Note that the breach and the initial perimeter are the first two Run Watchers' decisions for every scenario.

3.4 Barrier Placement

At every time step of every simulation run, Run Watchers sample water depths at all important buildings and all present perimeter segments. If water is detected at an important building (water depth > 0), the corresponding simulation run is terminated. Alternatively, if water is encountered at a perimeter segment, Run Watchers make a decision to place a barrier at this segment. A corresponding simulation run should be created originating from the current one. Run Watchers track the current simulation back to find the latest time step when no water was present at the segment. The new simulation run with a barrier placed at the segment will originate from that time step.

If Run Watchers operate in the greedy mode, the original simulation is terminated as the new simulation starts. However, in the normal mode, it keeps running, because a barrier at the considered segment may not be actually needed for protecting the important infrastructure. This can lead to a situation where water appears at a perimeter segment from *inside* the perimeter, as illustrated in Figure 5. In such cases, no action is taken, although this fact is recorded as a reason for *not* placing a barrier.

3.5 Barrier Types

According to the needs of the Flood Protection Center, we consider barriers of four different types. These types are sandbags, AQUARIWA [2], and AquaBarrier [1] plates installed either horizontally or vertically. They differ in a variety of properties. One group of properties defines the capabilities of barrier types. It includes maximum water depth to handle and maximum terrain slope for barrier installation. Another group of properties affects construction times. This group includes loading and unloading times, assembly time, and number of units per cargo. Table 1 shows the characteristics from the first group. For the rest of the barrier type characteristics we refer to related work [33].

Run Watchers decide on the barrier types to use by considering the maximum water depth and maximum slopes at the corresponding perimeter segment. The applicability of each barrier type is checked in the order listed in Table 1. The first barrier type suitable for the given slopes and water depth is tried when the decision to place a barrier is made. If the newly created simulation run reveals that the created barrier is eventually overtopped, the run is terminated. A new run is created instead with the next applicable barrier type from the list. This corresponds to a barrier change decision. Alternatively, if the corresponding barrier cannot be constructed in time, the run is terminated, but there is no need to try the next barrier type in the list. The ordering of the barrier types not only corresponds to increasing water depths, but also to increasing construction times. The construction times computation is analogous to what is described in related work [33]. Generally, if no barrier type from the list can be applied due to time limitations or high water levels, the faulty segment should be removed in a newly created simulation run. The new, smaller perimeter is obtained by shrinking the current perimeter.

Barrier Type	Water Depth	Slope (t)	Slope (n)
Aquabarrier (h)	0.65 m	20 %	20 %
AQUARIWA	0.8 m	20 %	10 %
Aquabarrier (v)	0.95 m	20 %	20 %
Sandbags	1.2 m	-	-

Table 1. Attributes of barrier types: Maximum water level; Maximum slope (tangential to the barrier line); Maximum slope (normal to the barrier line). Aquabarrier plates can be installed either horizontally (h) or vertically (v). Sandbags are usable for any terrain slope

3.6 Perimeter Shrinking

Perimeter shrinking is done by removing one or more segments of the given perimeter. The concave hull subroutine is applied to obtain the new, smaller perimeter. Figure 6 demonstrates an example of perimeter shrinking. Notice that, after a series of segment removals, a single perimeter may split into two or more components. According to the domain experts, it only makes sense to build a protection line if it protects at least one important building. Our algorithm takes care of this by discarding the perimeters containing no important buildings. Additionally, if some important building gets exposed to flood after a segment removal, an object protection is tried for it. Should the object protection eventually fail, we consider the building as impossible to protect. The complete list of causes, decisions, and rules connecting them is given in Figure 7.

3.7 Avoiding Duplicates

In the normal mode, uncoordinated creation of multiple parallel simulations may lead to duplicate barrier configurations in different sce-

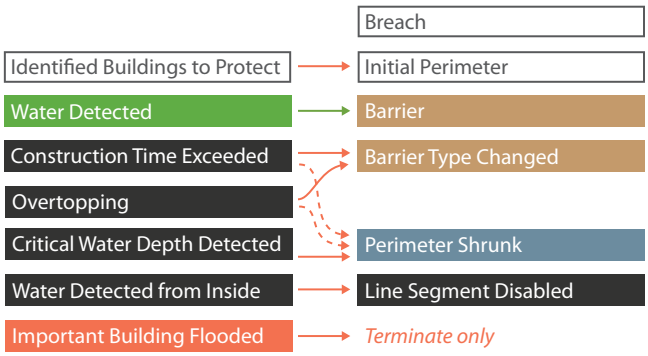

Fig. 7. Run Watcher rules. The left column lists causes that Run Watcher can detect, the right column shows decisions that Run Watchers can make. Arrows indicate the connection between causes and respective decisions. For some causes, multiple decisions can be taken. In this case, a solid arrow corresponds to the preferred decision. Red arrows indicate subsequent termination of the run where the cause has been detected. In case of a green arrow, the original run continues.

narios. If a barrier is placed at a perimeter segment due to a water detection, the original run continues without a barrier. At subsequent time steps, water at this segment will be detected again, and further attempts of Run Watchers to create the same barrier should be blocked. Moreover, creation of identical perimeters should be avoided as well. To handle these issues, Run Watchers analyze the present hierarchy of decisions before creating new simulation runs.

4 Automated Visualization and Reporting

When searching for successful response plans, Run Watchers make multiple decisions to operate simulation runs. The corresponding decision trees are large and complex. Visualizations are needed to understand causes and effects of the decision making process and to explain them to stakeholders. A fragment of a decision tree in Figure 8 is a generic illustration of Run Watchers' activities. A cause requires Run Watchers to make a decision. Multiple decisions are possible as a reaction to the cause. Any decision eventually leads to another cause.

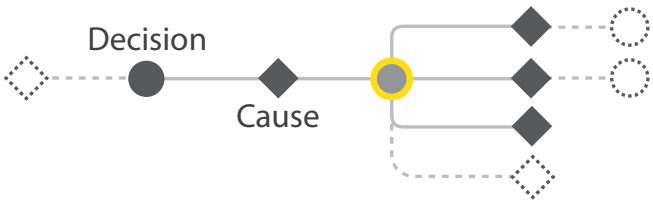

Fig. 8. Decision tree. Decisions (circles) have a unique cause but can have multiple child causes (diamonds). For navigating the tree starting with the highlighted element, the elements displayed in solid are most important, since they are the direct neighbors.

From the perspective of our solution, a decision is coupled with a simulation run consisting of a sequence of time steps. A cause corresponds to a time step satisfying some conditions. In this setup, a vast number of time steps in multiple simulation runs should be traversed to extract relevant causes and decisions. Run Watchers automatically extract causes and decisions relevant to the scenario of interest and generate visualizations and reports to convey them to the user.

4.1 Condensed Decision Clusters

In the context of our solution, time-dependent visualization of the automated decision making process poses several challenges related to scalability. Multiple alternative runs may simulate the same sequence of time steps, hence a vertical stacking of simulation timelines is required to resolve spatial collisions. In case of too many such runs in

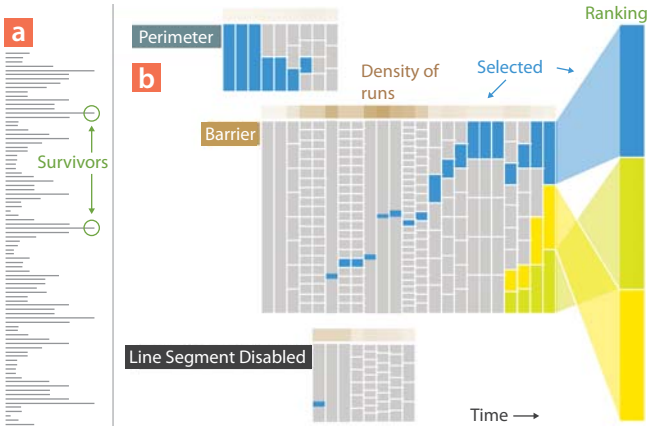

Fig. 9. Condensed Decision Clusters. (a) Plain timelines are not scalable, hard to interact with, and do not convey what has been decided by Run Watchers. Good solutions are the only few survivors on the right, and lots of white space is present there. (b) Decisions are grouped into clusters. Decision clusters rescale time steps of surviving runs vertically. Successful runs are colored according to performance. The ranking bar on the right facilitates the selection of better scenarios.

a stack, thinner timelines are needed for displaying them all. However, for thin timelines, no performance visualization or interaction is possible. Moreover, since the number of simulation runs may vary between time steps, multiple white spaces occur among the timelines (Figure 9a). This becomes even more prominent at the right end of the view, where the few "survivors" can be found. Note that exactly these "survivors" are of interest, since a response plan is considered successful if and only if it managed to protect the important infrastructure through the whole event duration. Finally, stacked timelines as in Figure 9a give no idea of the number and types of decisions made in the course of the search.

We propose a visualization of decisions that solves the above issues. For our application, we identify three classes of decisions: barrier decisions, perimeter decisions, and line segment disabling decisions. The first class contains all decisions to place a barrier (of the first applicable type) or to change a barrier type. The second class contains the initial perimeter decision and all decisions to shrink a perimeter. Finally, the third type contains decisions disabling a perimeter segment in cases when water touches this segment from inside the perimeter. Keeping in mind the tight coupling between decisions and simulation runs, we can view any scenario as a sequence of decisions made by Run Watchers. We cluster these decisions according to the class they belong to. In the view, each cluster occupies a rectangular area called a cluster box (Figure 9b). Decisions of each scenario are distributed over the clusters. In the (horizontal) clusters, they are stacked vertically and aligned relative to the common time axis. At every time step, we rescale the timelines in the vertical direction to always fully occupy the vertical space within the cluster boxes. Color bars above the clusters show the relative densities of decisions in the clusters for each time step. Note that the number of clusters can be arbitrary. It is up to the user to decide which decision classes should be distinguished.

The described visualization has important advantages. First and foremost, the successful scenarios are easier to find and to interact with. The vast majority of all simulation runs are terminated early, and there are normally only few timelines present on the right edge of decision clusters. They are assigned more display space and thus are easy to select. Performance visualization with a color over these timelines is clearly visible. An interactive ranking bar to the right of the decision clusters allows the user to quickly pick the scenario corresponding to the best response plan. If selected, a scenario is highlighted through the decision clusters (Figure 9b). Second, the better of the failed solutions are also easy to find and select, since they are close to the successful ones. These solutions can be useful for explanations

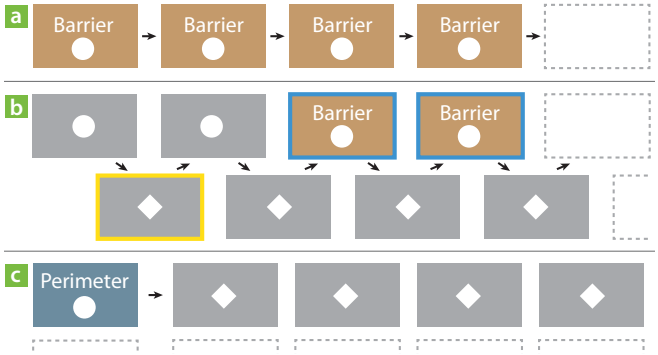

Fig. 10. Different layouts for the storyboard view. (a) Action plan. Only barrier decisions are shown that belong to the actual response plan. (b) Decisions justification. Shows all decisions relevant to the active scenario in the top row, and associated causes in the bottom row. Parts of the action plan are highlighted. (c) Hard decisions. Perimeters that could have potentially protected a building of interest are shown on the left. Next to it the causes are given demonstrating why particular solutions based on this perimeter could not be applied.

and comparisons. Third, since the clusters show only the computed time steps, the user can observe the real-time progress of the simulation and see how many promising scenarios are being simulated at any time by looking at the colors corresponding to the ongoing runs. Finally, the user can comprehend the number of decisions of different types from the densities in the clusters.

4.2 Storyboards

Run Watchers are capable of generating visualizations and reports of their decision making activities. Three target groups are taken into account, for which the information is provided from different perspectives (Figure 10). In the first mode, the detailed action plan is extracted from the scenario. This mode targets response personnel working onsite. Exact material amounts and timings are presented along with barrier locations and types. The second mode is meant to be used by flood managers. In this mode, Run Watchers present the whole set of causes and decisions that led to a particular scenario. This is needed when flood managers justify their decisions post factum to the administration. The third mode is useful for explaining hard decisions to stakeholders. Run Watchers provide a set of reasons for *not* protecting particular buildings in the region.

Action plan In the action plan mode, Run Watchers generate a storyboard showing the sequence of actions to be taken by the response personnel. An action plan for a selected scenario is presented by a horizontally oriented sequence of story blocks, where the left-to-right direction corresponds to the chronological order (Figure 10). For our current application, each story block represents a barrier placement. A story block displays a thumbnail image with a 3D view of the actual barrier location. A toggle button in the block header allows for switching perspectives. Three perspectives are available for each action, they range from an overview perspective showing the surroundings to a close-up perspective where the details of the barrier placement can be seen. The relevant barrier is highlighted in the 3D view. The thumbnail image is linked to the main view, where the user can interactively navigate the 3D rendering. Another button enables the detailed textual report on the action. The report is automatically generated by Run Watchers and includes the barrier type, construction duration, barrier length, and GPS coordinates of the barrier. Additionally, every story block of the action plan has a label showing the time until which the action should be completed. For further details like local barrier height variations, the user can consult the 3D view.

Decisions justification In this mode, the storyboard shows the chronological sequence of causes and decisions that led to a selected scenario. Story blocks in this view are aligned in two rows. The upper row corresponds to decisions made by Run Watchers, and the lower row contains causes for those decisions (Figure 11). The causes are aligned with the gaps between decisions, thus representing the cause-and-effect relation. Decisions belonging to the actual action plan are highlighted in yellow, and causes that led to a simulation run termination are marked in red. Toggled textual reports display information relevant to the given cause or decision, e.g., the water depth that caused a barrier overtopping or perimeter segments removed while shrinking. Thumbnail views highlight the actual details.

Hard decisions After a catastrophic event, flood managers may have to explain the implemented response plan from a different perspective. Stakeholders might want to know why their building was not protected. In this case, flood managers should be able to convey the underlying tradeoff between protecting the important infrastructure and protecting that building. Generating a strict proof of the inability to protect a building would be an extremely complex problem. However, Run Watchers can aid the user by providing a set of examples of what would have happened if the building of interest had been protected. In the corresponding view, example perimeters are shown that could protect the building of interest. For every such perimeter, Run Watchers traverse the related decisions, pick the reasons why all response plans based on this perimeter failed, and present them as a list of story blocks. This approach can be seen as applying a certain type of a query to the set of generated results, where the search criterion is the building of interest being protected.

4.3 Linked Navigation through Decisions and Causes

Here we describe a mechanism for navigating through the complete decision tree. It is based on the selection of causes and decisions in the condensed decision clusters and involves multiple, coordinated views. If the user clicks on any time step in the decision clusters, a corresponding scenario is selected and highlighted (Figure 12). The storyboard view presents the scenario, and the main 3D view shows a perspective for the selected time step. In addition, the decision clusters display the local neighborhood in the decision tree, mentioned in Figure 8. The current decision is shown by a thick dot, as well as the previous decision. The cause for the current decision is displayed by a diamond and visually related to it by an arrowed bar. In addition, the next cause in the selected scenario is shown by another diamond. Number labels specify the absolute ordering of causes and decisions in the scenario. Clicking on a cause diamond displays alternative decisions for this cause and shows the corresponding story box in the storyboard view and a 3D rendering in the main view. Selecting a decision is equivalent to selecting the alternative scenario where this decision is implemented, so the storyboard is updated correspondingly. The control elements described above are necessary and sufficient for navigating the whole decision tree. Additionally, for a selected scenario, navigation back and forth through the related decision chain is possible by means of the slider control.

5 Implementation

The parallel data-flow architecture implemented in our solution makes it possible to realize the concept of Run Watchers. Our system features two types of parallelism. On the one hand, modules of different types can be executed in parallel, e.g., those providing simulation parameters and boundary conditions; those carrying out the simulation proper; or those analyzing the simulation data. On the other hand, multiple scenarios can be simulated in parallel, taking into account their hierarchical dependencies. In this setup, a Run Watcher itself is a module authorized to control other modules. The data-flow is asynchronous, meaning that the Run Watcher can create new scenarios while others are being simulated and analyzed in parallel. We use a GPU-based flood simulation based on the shallow-water method [7]. No terrain roughness is taken into account in the numeric scheme, which corresponds to the absence of friction. This can be seen as the worst-case scenario where the water spreads the fastest.

A crucial part of our approach is the perimeter segment removal. Multiple segments may need to be removed at once if the corresponding barriers fail at the same time step. Let P be a perimeter obtained

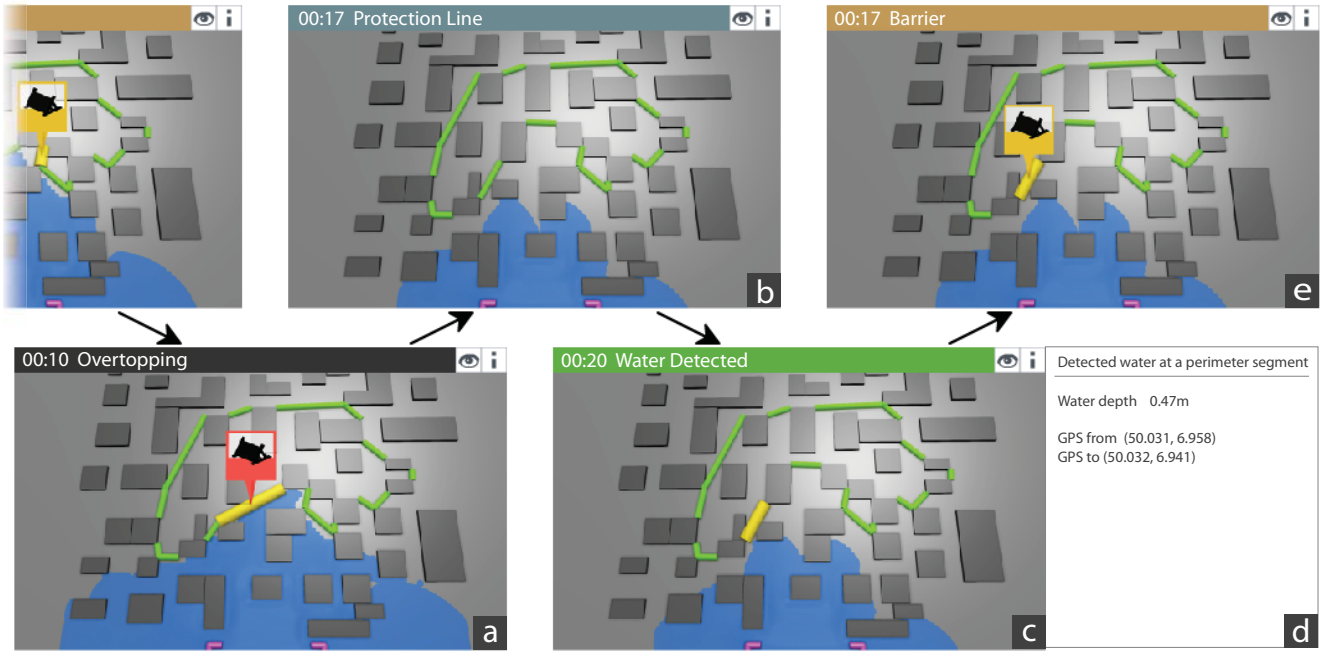

Fig. 11. Storyboard for decisions justification. (a) A barrier is overtopped causing (b) a perimeter shrinking. (c) Water is detected at a segment (yellow). (d) Details about the local water depth and the segment position are reported. (e) The detected water causes the creation of a barrier.

from the set of vertices V of the "footprint" polygons, and let p be a segment to remove. Assuming that v_1 and v_2 are the start and end points of p, the new perimeter P' is obtained from the set of vertices $V' = V/\{v_1, v_2\}$. The algorithm takes care that the new perimeter will not have a segment passing through the same pair of buildings as that of v_1 and v_2. Several other conditions and edge cases are accounted for, which we omit for brevity. The approach can be generalized to the removal of multiple segments. As a result of a segment removal, a perimeter is obtained that encloses a proper subset of the original set of buildings. In certain cases, a perimeter is split into multiple ones after a series of segment removals. This is naturally achieved if the result of the concave hull algorithm no longer encloses all the important buildings at hand. In this case, the algorithm runs again for the remaining buildings. Our algorithm drops the perimeters containing no important buildings. If, after a segment removal, some important building is left outside the perimeter, an individual protection is assigned to it, surrounding this building only. Should the individual perimeter eventually fail, we consider the building as unprotectable.

6 Case Study

In the case study, we considered a 14 m wide breach in the protection walls in the urban area of Cologne. The in-flow hydrograph at the breach was set to a constant hydraulic discharge of 40 m^3/s and a constant water level of 11.9 m. Four important objects were identified in the surrounding region, namely, a kindergarten, two schools, and a subway entrance. Around each important building, a radius of 200 m was picked for the initial perimeter derivation. The guaranteed response time was chosen to be 20 min, and the whole event duration to be 2.5 hrs. Two resource depots were picked in a distance of approximately 0.5 km from the region of interest. We assumed one truck per depot and one person for every two barrier meters to construct.

In order to save simulation time in the normal mode, we adopted a "non-greediness" setting. That is, for the value k of the non-greediness, each perimeter is allowed to have at most k simulation runs where some barrier is not placed despite the detected water. This limits the number of alternatives tried. The problem of protecting the important buildings was solved on a single desktop PC (*Intel Core i7-3770, 16 GB RAM, GeForce GTX 680*) with a simulation grid consisting of 700×600 cells of 3 m size. We made two attempts: one in the greedy mode, and the other one with a non-greediness $k = 1$. The greedy attempt delivered the successful response plan in approximately 20 min, after trying 38 decisions overall. The normal version took approximately 45 min to solve the problem. The result was 6 response plans with slightly varying performances and 415 decisions tried during the search. A simulation run through the whole event duration took approximately 5 min. Therefore, an exhaustive exploration of 415 decisions would already take 415 runs $\times$ 5 min $\approx$ 34.5 hrs of simulation with one PC. However, a successful response plan is likely to consist not of a single decision, but of some combinations of decisions. Extending the exhaustive search to combinations would greatly increase the required computation time.

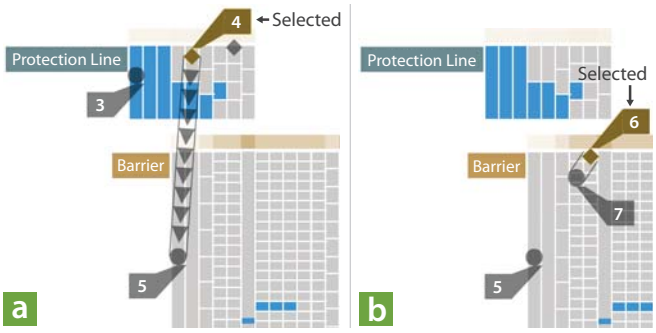

Fig. 12. Navigating a decision tree. The complete decision tree can be navigated within decision clusters. The clusters display the neighborhood of the selected cause or decision (see also Figure 8). Number labels are shown according to the path through the decision tree leading to the selected plan. The navigation step (a) is followed by the navigation step (b). Linked views (storyboard, main 3D view) display information about the selected element.

Figure 13 shows parts of the storyboard presenting the reasoning chain for the normal case. The first decision (a) corresponds to the breach. The yellow frame indicates that this story box is coordinated with the other views. Based on the breach simulation, the important

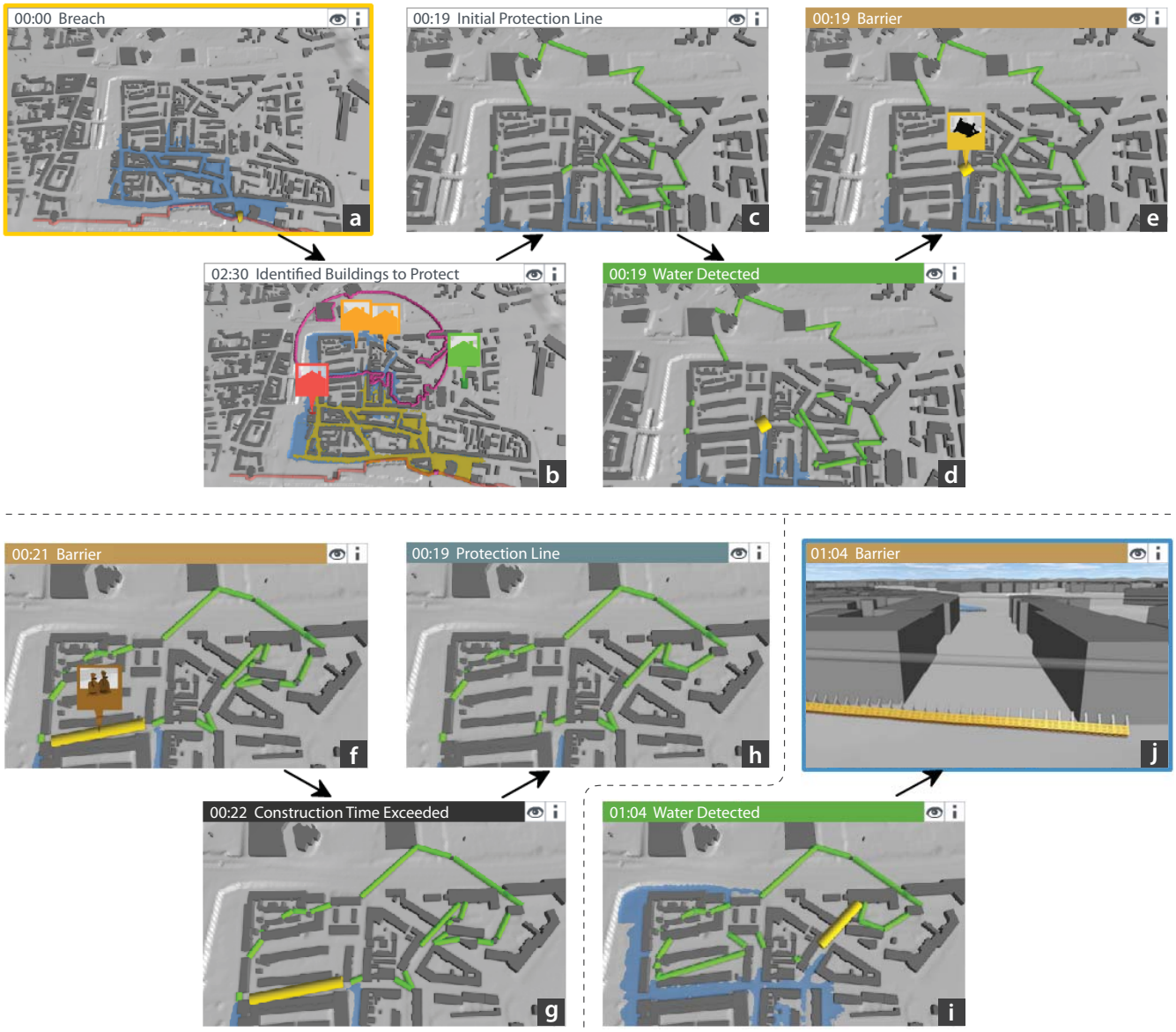

Fig. 13. Storyboard displaying the reasoning for the case study. The detailed description can be found in the text.

buildings are classified (b). The subway entrance cannot be protected (red), one of the two schools does not require protection (green). The kindergarten (orange, left) and the other school need protection. The system derives the initial region (violet) using the initial wavefront (yellow), the final wavefront (blue), and the important buildings to be protected. The decision is made (c) for the initial perimeter (green line). When water is detected (d) at one of the segments, highlighted in yellow, Run Watchers place there a barrier (e) of horizontally oriented Aquabarrier plates. The next shown episode happens later in time. Sandbags are tried at one of the segments of the modified perimeter (f). This barrier fails due to the construction time (g), hence the perimeter is again shrunk (h). Further on, as water is detected at a perimeter segment (i), Run Watchers decide to use Aquabarrier there (j). This decision is a part of the actual action plan, hence the blue frame around the story box. In the story box, the detailed perspective is selected.

Figure 14 shows an example of a hard decision explanation. The building of interest, labeled with yellow, could have been protected with the given perimeter (left), but then the important building, labeled with red, would be flooded (right). In Figure 15, the resulting decision clusters are shown. The six successful plans are displayed on the right. The third best plan is selected and the corresponding scenario highlighted through the decision clusters.

7 Evaluation

At the evaluation session, we presented our solution to the consortium of domain experts from the Flood Protection Center. Here we describe the results of the real-time demonstration and follow-up discussion. After a quick recap of the main questions (Q1-Q3), we explained the notion of protection perimeters and gave examples of segment removal with multiple important buildings at hand. The concept was found to be very intuitive and required little explanation. The same holds for the idea to search for response plans by placing barriers on demand and modifying perimeters when necessary. The experts concurred with the whole approach, considering it reasonable to have larger perimeters first and to shrink them gradually as the impossibility of protecting the current area becomes evident. Although the way our system arranges barriers between buildings was not always considered optimal, the experts agreed that it was indeed sufficient to answer the questions Q1 and Q2. The response personnel working on-site is always encouraged to make local optimizations in the barrier placement. Nevertheless,

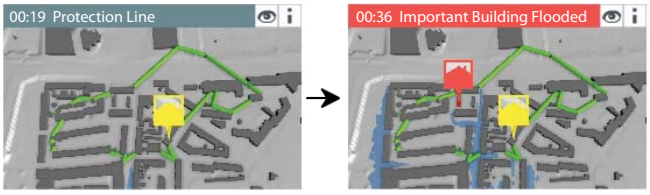

Fig. 14. Justification of a hard decision. A perimeter on the left could have protected the building (yellow label), but then the important building would have been flooded.

this is a direction for future improvements.

The consortium welcomed how little user input was required to set up the scenario. They found rational the way to derive the initial perimeter from the flood wave and the important infrastructure locations. Moreover, the experts highly valued the fact that no sketching of barrier locations was needed. However, they pointed out that the guaranteed response time and availability of construction personnel differ between daytime and nighttime. If the breach happens at night, it may take more time to get people ready for a response, and there may be considerably less people available for construction. Therefore, they suggested that in the preparation phase the search should be executed at least twice with different parameters.

The decision clusters were considered next. After a demonstration of branching timelines coming from multiple simulation runs in a simple, non-clustered view, the scalability issues were well understood. The way of clustering simulation runs according to decision types was not received well though. This was found neither intuitive enough nor useful. However, the experts agreed that vertical rescaling of timelines in the clusters according to their local density makes the visualization clearer. They particularly welcomed the fact that only successful plans were present and shown at the end of the time range, because for flood managers mostly those scenarios are of interest. The ranking bar was found useful for quickly picking the best response plans. In general, the experts mentioned that it was good to have the whole variety of simulated scenarios shown in the view. From this they could have an idea of how many scenarios were simulated and, more importantly, explore some failed plans for documentation purposes. According to the experts, failed plans were relevant for explaining particular decisions.

In the context of decision justification, the storyboard visualizations were considered as a good complement to the decision making algorithms. Automatically derived action plans with the timing and order for constructing barriers were found very useful. According to the domain experts, textual descriptions are more important for the response phase, whereas the detailed 3D perspectives are more suitable for conveying decisions to administrators after a catastrophic event. Furthermore, the detailed perspectives are useful for proper stacking of sandbags or for correctly orienting Aquabarrier plates towards the flood wave. The experts requested a representation that distributes the story boxes along the timeline so that periods of higher workload are clearly visible [12, 28]. We take this as a possibility for improvement. Finally, the domain experts approved the approach to explain why particular buildings could not be protected by providing a set of relevant examples. They considered it rational and relevant for their purposes.

Summing up the evaluation, the consortium agreed that our solution provides algorithms and visualizations to answer the questions Q1-Q3. The overall positive assessment was strengthened by expressing a desire for further collaboration on a commercial basis.

8 Conclusions and Future Work

In this paper, we demonstrated an approach to explore a complex, multidimensional parameter space of flood response plans. It makes use of an automated simulation monitoring and steering according to domain-specific rules. Unlike exhaustive approaches, our solution does not require plenty of computation time. Unlike in manual steering approaches, little user interaction is needed. Moreover, due to the automatic generation of visualizations and reports, the user is relieved

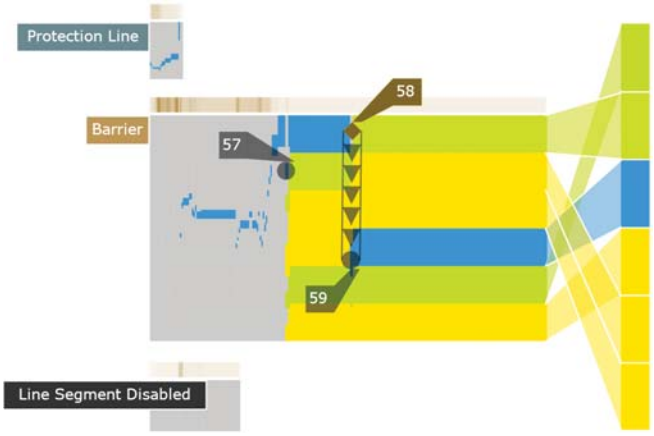

Fig. 15. Resulting decision clusters for the case study. Six successful response plans have been found and ranked.

of the burden to extract the relevant knowledge from a large number of decisions and causes underlying the search.

Despite the specificity of the field chosen for demonstration, the presented approach is rather generic and applicable in other areas as well. For example, one could think of planning an evacuation from large crowded spaces, with an agent-based pedestrian simulation in use. In general, any simulation-based search where simulation steps are rather expensive and simulation states are reusable could benefit from our approach. The approach could become even stronger with a mechanism of weighing the decisions in order to figure out more promising ones and give them more priority in the computational resources allocation. A prominent issue spotted out by the evaluation concerned the decision cluster visualization. One approach to make navigation through the decision tree more intuitive could be to split the visualization into multiple views, e.g., to show a non-temporal decision tree separately from simulation timelines.

In the field of decision support for flood management, there are still multiple ways to improve. First of all, the solutions delivered by our system are educated guesses and thus not optimal, they should be rather used for guidance. One example is an exact barrier placement between buildings. This may need to be adjusted by the on-site personnel during the response time. The system could be more flexible in arranging barriers and account for more factors while doing that. An interesting improvement could be to make the system account for terrain elevations which could constitute a natural protection. Additionally, the approach could make use of ensembles in order to search for response plans applicable to multiple breach scenarios at once.

The hard decisions justification in our solution is based on executing a search over the set of computed results. Further interesting search criteria could be added to answer important "what-if" questions decision makers often ask, e.g., "What if we want to protect this area?" or "What if we use only that particular barrier type?". Accounting for the uncertainty in simulation results is on our list of future work as well. Finally, our tool would greatly benefit from a practical evaluation with flood managers during a planned exercise. By running it in the simulated conditions set up by the exercise developers, we would get valuable insights into problematic areas and identify the most important directions for further improvements.

Acknowledgments

This work was supported by grants from the Vienna Science and Technology Fund (WWTF): ICT12-009 (Scenario Pool), and from the Austrian Science Fund (FWF): W1219-N22 (Vienna Doctoral Programme on Water Resource Systems) and P24597-N23 (VISAR). We thank the Stadtentwässerungsbetriebe Köln, AöR.

REFERENCES

[1] Aquabarrier Systems Limited - Complete flood defence solutions. http://www.aquabarrier-systems.com/ (last visited on March, 31st 2014).

[2] AQUARIWA - das innovative Hochwasserschutzsystem. http://www.aquariwa.de/home/ (last visited on March, 31st 2014).

[3] D. Azócar, M. Elgueta, and M. C. Rivara. Automatic lefm crack propagation method based on local lepp–delaunay mesh refinement. *Advances in Engineering Software*, 41(2):111–119, 2010.

[4] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *Visualization, 2007. APVIS'07. 2007 6th International Asia-Pacific Symposium on*, pages 133–140. IEEE, 2007.

[5] M. Booshehrian, T. Möller, R. M. Peterman, and T. Munzner. Vismon: Facilitating analysis of trade-offs, uncertainty, and sensitivity in fisheries management decision making. *Comp. Graph. Forum*, 31(3pt3):1235–1244, June 2012.

[6] A. Borrmann, P. Wenisch, M. Egger, C. van Treeck, and E. Rank. Collaborative Computational Steering: Interactive collaborative design of ventilation and illumination of operating theatres. *ICE08, Plymouth*, 2008.

[7] A. R. Brodtkorb, M. L. Sætra, and M. Altinakar. Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers & Fluids*, 55:1–12, Feb. 2012.

[8] 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.

[9] P. Caymes-Scutari, A. Morajko, T. Margalef, and E. Luque. Scalable dynamic monitoring, analysis and tuning environment for parallel applications. *Journal of Parallel and Distributed Computing*, 70(4):330–337, 2010.

[10] T. Eickermann, W. Frings, P. Gibbon, L. Kirtchakova, D. Mallmann, and A. Visser. Steering UNICORE applications with VISIT. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1833):1855–1865, 2005.

[11] F. González, M. Á. Naya, A. Luaces, and M. González. On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics. *Multibody System Dynamics*, 25(4):461–483, 2011.

[12] O. Hoeber and J. Gorner. Browseline: 2d timeline visualization of web browsing histories. In *Information Visualisation, 2009 13th International Conference*, pages 156–161. IEEE, 2009.

[13] M. Jern, J. Rogstadius, T. Astrom, and A. Ynnerman. Visual analytics presentation tools applied in html documents. In *Information Visualisation, 2008. IV'08. 12th International Conference*, pages 200–207. IEEE, 2008.

[14] M. Kapadia, S. Singh, B. Allen, G. Reinman, and P. Faloutsos. Steerbug: an interactive framework for specifying and detecting steering behaviors. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 209–216, New York, NY, USA, 2009. ACM.

[15] D. Ling, L. Bu, F. Tu, Q. Yang, and Y. Chen. A finite element method with mesh-separation-based approximation technique and its application in modeling crack propagation with adaptive mesh refinement. *International Journal for Numerical Methods in Engineering*, 2014.

[16] A. Lu and H.-W. Shen. Interactive storyboard for overall time-varying data visualization. In *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific*, pages 143–150. IEEE, 2008.

[17] P. Lundblad and M. Jern. Visual storytelling in education applied to spatial-temporal multivariate statistics data. In *Expanding the Frontiers of Visual Analytics and Visualization*, pages 175–193. Springer, 2012.

[18] K.-L. Ma, I. Liao, J. Frazier, H. Hauser, and H.-N. Kostis. Scientific storytelling using visualization. *IEEE Computer Graphics and Applications*, 32(1):12–19, 2012.

[19] K. Matkovic, D. Gracanin, M. Jelovic, and H. Hauser. Interactive visual steering - rapid visual prototyping of a common rail injection system. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1699–1706, 2008.

[20] 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.

[21] A. Morajko, T. Margalef, and E. Luque. Design and implementation of a dynamic tuning environment. *Journal of Parallel and Distributed Computing*, 67(4):474–490, 2007.

[22] A. Moreira and M. Y. Santos. Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. 2007.

[23] J. D. Mulder, J. van Wijk, and R. V. Liere. A survey of computational steering environments. *Future Generation Computer Systems*, 13, 1998.

[24] B. Nairouz, M. Hoepfer, N. Weston, and D. Mavris. Investigations for time step settings in a dynamic system co-simulation environment. In *Electric Ship Design Symposium*, 2009.

[25] H. Ribičić, J. Waser, R. Fuchs, G. Blöschl, and E. Gröller. Visual analysis and steering of flooding simulations. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):1062–1075, 2013.

[26] P. Ruponen. Adaptive time step in simulation of progressive flooding. *Ocean Engineering*, 78:35–44, 2014.

[27] E. Santos, J. Tierny, A. Khan, B. Grimm, L. D. Lins, J. Freire, V. Pascucci, C. T. Silva, S. Klasky, R. Barreto, and N. Podhorszki. Enabling advanced visualization tools in a web-based simulation monitoring system. In *eScience*, pages 358–365, 2009.

[28] C. Stab, K. Nazemi, and D. W. Fellner. SemaTime - timeline visualization of time-dependent relations and semantics. In *Advances in Visual Computing*, pages 514–523. Springer, 2010.

[29] J. E. Swan II, M. Lanzagorta, D. Maxwell, E. Kuo, J. Uhlmann, W. Anderson, H.-J. Shyu, and W. Smith. A computational steering system for studying microwave interactions with missile bodies. In *Proceedings of the conference on Visualization'00*, pages 441–444. IEEE Computer Society Press, 2000.

[30] C. Tominski, P. Wollgast, and H. Schumann. 3D Information Visualization for Time Dependent Data on Maps. In *Proceedings IEEE Symposium on Information Visualization 2005 (InfoVis 2005)*, pages 175–181, 2005.

[31] T. Torsney-Weir, A. Saad, T. Möller, H.-C. Hege, B. Weber, and J.-M. Verbavatz. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1892–1901, Dec. 2011.

[32] 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.

[33] J. Waser, A. Konev, B. Sadransky, Z. Horváth, H. Ribičić, R. Carnecky, P. Kluding, and B. Schindler. Many Plans: Multidimensional Ensembles for Visual Decision Support in Flood Management. *Computer Graphics Forum*, 33(3):281–290, 2014.

[34] M. Wohlfart and H. Hauser. Story telling for presentation in volume visualization. In *Proceedings of EuroVis 2007*, pages 91–98, 2007.

[35] S. Yau, V. Karamcheti, D. Zorin, K. Damevski, and S. G. Parker. Application-aware management of parallel simulation collections. In *ACM Sigplan Notices*, volume 44, pages 35–44. ACM, 2009.

[36] Y. Zhang, Y. Wang, M. Wang, and J. Jiang. Co-simulation of Flexible Body Based on ANSYS and ADAMS [J]. *Journal of system simulation*, 17:004, 2008.