full paper 165 M. Khoury

Data · November 2016

3 authors:

Mehdi Khoury
University of Exeter

Mark S. Morley
KWR Watercycle Research Institute

Dragan Savic
University of Exeter

Serious Game Approach to Water Distribution System Design and Rehabilitation Problems

Mehdi Khoury, Mark S. Morley, Dragan Savić

Centre for Water Systems
University of Exeter, EX4 4QF, Devon (United Kingdom)

Correspondence to: Mehdi Khoury (m.khoury@exeter.ac.uk)

Abstract. This paper presents a multi-player online Serious Game developed to investigate end-user behaviour when faced with complex water distribution system (WDS) design and rehabilitation problems. The Game was evaluated through an event during which 24 postgraduate students were grouped into 8 teams competing to obtain the best WDS design solution. TB-SeGWADE (Team-Based Serious Game for WDS Analysis, Design & Evaluation) couples an innovative and visually attractive interactive front-end with a server-side modelling engine handling real-time hydraulic simulation. A multiplayer online game infrastructure is implemented allowing interactions with a model to be instantly broadcast to both collaborating users and competing teams. The interactions are recorded and can subsequently be replayed or analysed through the interface to explore the decision making process in detail. Role-playing games have the concept of a "Dungeon Master" (DM), being the actor setting scenarios for different problems that the users have to solve as a team. A game variant has been developed with a special interface with which it is possible to introduce environmental events and incidents for the players having to cooperate to solve problems.

Game types that can be configured include: (i) Design-related tasks - constructing networks so as to minimize future problems where interactions are compared to a metric so as to evaluate how compliant it is with relevant engineering standards; (ii) Post-incident tasks: diagnostic of problem(s) and mitigation actions; and (iii) Optimisation tasks: single-user or multi-user optimisation. In the case of multi-user optimisation tasks, several users can have conflicting priorities.

A simplified version control system is implemented within the game engine to allow collaborating users to save modifications in the "team space" in a fashion that can be recorded and analysed for future evaluation. The Serious Game engine is designed to be extensible and can be reconfigured dynamically to run different EPANET (Rossman, 2000) input files permitting a range of games to be developed independently. In addition, the front-end employs adaptive graphics that can switch between HTML5 canvas and WebGL rendering technologies, depending on the client hardware capabilities, in order to deliver the best user experience. The evaluation of the game during a five days-long event has brought valuable insight regarding design and implementation procedures to use when building and running such a complex software, and has shown that this type of Serious Game has a lot of potential as a teaching and a collaborative tool for Water Distribution Systems due to the high degree of user engagement displayed by participants.

1 Introduction

Serious games are an emerging type of cognitive tools designed to train and teach in an entertaining and interactive manner (Michael et al., 2006; Rayburn et al., 2007). Although serious games have already been used in numerous domains, only a handful of computerised game simulations have been applied to the water sector (Medema et al, 2016). Water Distribution Systems (WDS) problems are technically complex and require a great deal of expertise and skills to solve, making it an ideal experimental ground for building serious games where engineers can approach problems by designing solutions in a collaborative fashion as well as competing between teams to compare and improve existing solutions. Most game applications in the water sector are primarily educational and largely focus on river basin management (Savic et al., 2016). The level of simplification is quite high and the user interfaces are usually fairly basic: there are generally no 3D visualisation, and, most importantly, at this stage, to our knowledge, there exists only one Serious Game for the technical design of water infrastructures (Morley et al, 2016; Savic et al., 2016), and that game did not allow collaborative work between players. SeGWADE (Serious Game for Water distribution systems Analysis, Design & Evaluation) is a second iteration built on the game cited above, and a conceptually simple, computationally elaborate game for WDS analysis that aims at filling this gap. As it is the first time that a multi-player online Serious Game with both collaborative and competitive elements is applied to technical problems relevant to WDS, new design solutions had to be implemented to respond to emerging problems. Although we provide some preliminary results regarding user engagement and team performance, the primary goal of this paper is to provide an analysis of the design and implementation problems encountered, as well as the solutions and lessons learned.

2 Methodology

The TB-SeGWADE (Team-Based) game structure builds upon the structure of its predecessor CP-SeGWADE (Competing Players) by adding collaborative elements to the multiplayer online competing game initially developed. As such, it has a modular architecture where clients (the players using a web browser from their machine) visit a web site (the server), log in, and play the game collaboratively. The first application of this game architecture is to the rehabilitation of an existing WDS by the optional installation of duplicate pipes in order to alleviate a pressure deficiency, minimise water leaks, water age and overall cost. The second application is to react as a part of a team to an incident caused intentionally by the Dungeon Master (DM) where the connection to one of the reservoirs in the network is intentionally disabled and to deal with the consequences. Finally, the third application is a multi-objective optimisation task as users controlling different WDS areas may have diverse and sometimes conflicting priorities, while having to compromise for the overall design of the network. The first application of the game (CP-SeGWADE) was about system rehabilitation by selecting the diameters of the duplicate pipes from a predetermined list of commercially available pipe sizes. The present application (TB-SeGWADE), builds on this by introducing two additional objectives to the existing minimisation of cost: the minimisation of water age and leakage, while collaborating with the rest of the team so as to reach a satisfying global solution. In the present game, changing existing default pipes with newer pipes will reduce leakage, and changing the diameters of these pipes will have consequences on both water pressure, flow, and therefore water age. The user can select diameters for each available pipe interactively and then submit the resulting model for hydraulic simulation. At that point the resulting pressures are evaluated to determine whether the model meets the minimum pressure requirements, while leakage, age and cost are computed. Each player acts as part of a team of three, every of them controlling a different portion of WDS. The area controlled by each team member represent roughly a third of the network. When a new "best" solution (i.e., the lowest sum of the best rankings for cost, leakage, and water age) is identified, the game banner on each of the currently connected clients is updated to show the identity the team and the score that they have achieved – adding a competitive element to the exercise. At the end of the game, it is possible to summon a high-score table on each of the teams' browsers to display their relative performance.

The presence of a collaborative game between teams does not seem significant from a methodology point of view, but it does introduce complex challenges from the point of view of design and implementation.

3 Architecture

The game engine is designed using the well-known MVC (Model-View-Controller) Client/Server paradigm with a few modifications so as to make it slightly more efficient for the purposes of gaming over the internet. As in a standard Model-View-Controller pattern, the model organises the application from the point of view of the data, logic and rules, independently from the user interface. The view has the user interface in which the information is visually represented. The Controller acts as an intermediary that takes input from the user and converts it to commands for the model or view. In order to reduce the communications overheads and load on the web server, the MVC loop is short-circuited by using the angular.js (Angular, 2016) library so that relatively trivial operations, such as input data validation, are performed on the client-side and not sent to the server-side model.

The game engine part of the application is written entirely in JavaScript on both server and client side. This provides an unmatched flexibility from a development point of view. The Node.js (Node.js, 2016) and Socket.io (Socket.io, 2016) libraries are used for connectivity to handle instant messaging between clients and server during game time. Sails (Sails, 2016) being the most popular MVC (Model–view–controller) framework for Node.js, is being used for deployment. The code is hosted as open source on GITHub at https://github.com/gentr1/water-serious-game. End users access the game with their web-browsers and is thus compatible with a variety of platforms, having been tested on Windows, Linux, iOS, and Android. Users having access to WebGL capabilities can switch between a medium and a high graphic detail mode which offers a more aesthetically pleasing immersive 3D visualisation (see Figure 2) and which is better able to scale in size for rendering large networks. In order to accommodate the gamut of hardware capability that might be experienced on diverse target platforms, the graphic rendering is designed to fall-back to HTML Canvas technology in the event that WebGL rendering is unavailable. Interaction with the network takes place by selecting the element of the network that is to be modified. This pops-up a selector that allows the player to change the properties of the network element directly. For the example presented in this paper the options are limited to changing pipe diameters. However, it is possible using the game engine developed to manipulate pumps, valves and other network components in a similar fashion, according to the needs of the game problem.

In the game, a simplified "version control" interface became necessary in order to handle users' actions with respect to their team, and then save and retrieve different solutions. When a user wishes to evaluate a change without necessarily changing the water network configuration for the rest of the team, he can click on an "evaluate" button that will return a result as if he had some private "drawing board". When a player wants chosen modifications to impact the network for other members of the team, then a "commit" button can be pressed, that will immediately impact the displays other team members can see. Every "commit" will also save the configuration of this user portion of the water distribution network to the history of the team actions. This user configuration can then be retrieved to go back to earlier choices if the user wishes it. If loading an earlier configuration, the user will need to press either "evaluate" or "commit" to see the full extent of the consequences derived from merging an earlier choice with other team users' latest configurations. If every user separately load their first choice, and commit in turn, then the game will revert to its original state.

Communication between server and clients, done through socket.io, allows messages to be instantly broadcasted between users at game time. The introduction of a collaborative element to the game tends to amplify problems linked to network connectivity. The messaging system as seen in Figure 3 is part of the game and makes collaborative work possible even with poor connectivity. The key technical change was the insertion of a job queue and a looping process that adds a new job every time a user commit. Once the hydraulic simulator produces an output, the server broadcasts the result of the computation to all clients. If the client does send an acknowledgement of receipt back, then the job is taken off the queue, and the next job is processed. If there is no such acknowledgment, the job is left on the queue to process the next time somebody commits a result.

Underpinning the game is a hydraulic simulation engine based on EPANET (Rossman, 2000) which includes the ability to simulate hydraulic models using a pressure-driven approach (Morley et al, 2008). Adopting a pressure-driven model allows more meaningful results to be shown to the user in the event of an infeasible network solution being proposed.

4 Case study

The case study network employed sets the problem of rehabilitating a failing water distribution system which is experiencing pressure shortages and excessive levels of leakage. The model employed is based on the Modena network of Bragalli et al. (2012) and models the system-wide leakage using the approach applied to the BBLAWN competition model (Giustolisi et al., 2015). This network model was selected to offer sufficient level of complexity that the network could be divided into three sub-networks of sufficient size. It also provides enough redundancy to make it an interesting challenge for the users to collaborate in solving the problem as a whole whilst being confined to working on their own specific areas.

The problem requires the users to robustly strengthen the network to remedy the pressure shortfalls, whilst simultaneously:

1. Minimizing leakage
2. Minimizing maximum water age in network

3. Minimizing cost of infrastructure interventions undertaken to achieve (1) & (2)

In order to achieve these goals each team can elect to close or replace any of the existing pipes in the system from a catalogue of 13 commercially available pipe diameters. Closed pipes no longer contribute to leakage in any way as they are hydraulically isolated. As well as improving leakage, installing a new pipe in the system reduces the interior roughness of the pipe and consequently the head loss experienced by the pipe. To be considered valid, a solution must have no nodes that are hydraulically disconnected and a minimum pressure requirement of 20m at any node with demand had to be observed.

5 Results and lessons learned

The results obtained demonstrate a high degree of player engagement with the game with an average number of 555 submissions per user. This is quite substantial, especially if the considerable amount of team based communication needed to lead to a compromise for a satisfying common solution is taken into account. The charts in Figure 5 represent the performance of the eight teams (marked with 'X'), the Genetic Algorithm optimiser ('+'), and the Pareto optimal front ('O' marks) obtained through GA optimization of the problem. As can be seen the GA clearly outperforms the students when considering water age and leakage. However, it is apparent that the students managed to achieve much lower cost solutions than the GA at the expense of the other two objectives. This suggests that the students either prioritised the cost objective or found that cheap, feasible solutions were easier to come by. It would be interesting to determine whether any prioritisation of the cost objective was down to the students' better understanding of the effects of the decisions they were making on the costs in contrast to the less direct influence of pipe replacement on water age and leakage. The process of fine-tuning the design of the game is ongoing following players' feedback from the past two events.

The principal lessons taken from challenges encountered while running the game and suggestions received by users, are:

i. To take into account internet connectivity of variable quality in the design of the game – the inclusion of an extra loop as described in Figure 2 is pretty much a requirement.

ii. To factor the complexity resulting from the necessary "version control" aspects of saving and retrieving choices of users while making sure that the overall team network configuration results from merging different players' choices in a consistent fashion. The introduction of an "evaluate" and a "commit" buttons working in conjunction with the behaviours described in section 3 was a major solution.

iii. From an organisational point of view, in order to maintain an efficient reactivity during the event to correct possible problems with the game, a physical separation between the person in charge of dealing with players during the event, and the person in charge of server maintenance and debugging located was found to be more effective.

iv. Periodic game server resets were required when a Windows operating system was used for the machine hosting the game. Due to its stability, a Linux operating system seems more suitable alternative for hosting as game server.

v. Due to the complexity of programming a multi-player online game architecture, if at game time, a code bug reveals itself and relates to the fact that players' history might contains erroneous data, it is better, after correction, to erase completely all players' history and restart the game from to ensure that future playing sessions will behave correctly.

vi. At game time, operational resiliency can be achieved by having an identical backup server where it is possible to rapidly transfer all latest player and team data and scores to a new database so as to continue the game in the same fashion, but on a brand new working platform.

vii. Finally, the use of a version control system is practically mandatory. Updating the code of a multi-player-online games requires distant components talking to each other need to be constantly updated in a synchronised fashion.

From a user feedback perspective, the main feature that was seen as missing was the ability to easily locate and modify pipes in batch via a tabular format similar to an interactive, searchable, sortable and editable spreadsheet. In short, players would have appreciated on top of the gaming visual interface an extra tabulated editor more oriented toward engineering problem solving. Similarly, the ability to swap the legends with the names of the different nodes and pipes with information regarding individual elements leakage, age, and pressure was also suggested as a desirable feature.

6 Conclusions & Future Directions

A multi-player online Serious Game has been developed to allow users to work in competing teams to optimize a water distribution system problem in which network reinforcement is undertaken by selecting pipes to be duplicated to meet a minimum pressure requirement across the network as well as minimising resulting costs, leakage and water age.

The evaluation of the game took place through a five days-long event during which valuable lessons were learned regarding the design and implementation procedures to follow when building and running such a complex piece of software. Furthermore, this type of Serious Game shows a lot of potential as a teaching and a collaborative tool for Water Distribution Systems due to the high degree of user engagement displayed by participants.

The game engine has been designed to be extensible and will be deployed on diverse optimization problems. It is envisaged that these problems will involve multiple and sometimes conflicting objectives and will be undertaken by teams where users representing different stakeholders will have different priorities in terms of the objectives and how they might be achieved.

More playing sessions will be needed to focus later on an in depth analysis of the learning effects of the game on the players, as we have but skimmed the surface of all the possibilities offered by such a rich problem solving tool.

The game as presented in this paper can be installed and played from the source code available on GitHub at https://github.com/gentr1/water-serious-game.

References

Angular.js - Link: https://angularjs.org (accessed 21 June 2016)

Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A. & Toth, P.: On the Optimal Design of Water Distribution Networks: a Practical MINLP Approach. *Optim. Eng.* 13, 219-246, 2012.

Giustolisi, O., Berardi, L., Laucelli, D., Savic, D. & Kapelan, Z.: Operational and Tactical Management of Water and Energy Resources in Pressurized Systems: Competition at WDSA 2014. *J. Water Resour. Plann. Manage.*, 10.1061/(ASCE)WR.1943-5452.0000583, C4015002, 2015.

Medema, W., Furber, A., Adamowski, J., Zhou, Q. & Mayer I.: Exploring the Potential Impact of Serious Games on Social Learning and Stakeholder Collaborations for Transboundary Watershed Management of the St. Lawrence River Basin. Water, 8(175), 24, 2016

Michael, D. & Chen S.: Serious games: games that educate, train and inform. Thomson Course Technology, Boston, Massachusetts, USA. 312, 2006.

Morley, M.S. & Tricarico, C.: Pressure Driven Demand Extension for EPANET (EPANETpdd) – Technical Report 2008-02, Centre for Water Systems, University of Exeter, UK. 11, 2008.

Morley M.S., Khoury M., Savić D.: Serious Game Approach to Water Distribution System Design and Rehabilitation Problems, 18th Water Distribution Systems Analysis Conference, 2016.

Node.js - Link: https://nodejs.org (accessed 21 June 2016)

Rayburn, E.M.: Applying simulation experience design methods to creating serious game-based adaptive training systems. Interacting with Computers, 19(2), 206–214, 2007.

Rossman, L.A.: EPANET 2, Users' Manual, United States Environmental Protection Agency, Washington, USA, 200, 2000.

Sails - Link: http://sailsjs.org (accessed 21 June 2016)

Savic, D.A., Morley, M.S. & Khoury, M.: Serious Gaming for Water Systems Planning and Management, Hydroinformatics and Urban Water Systems, *Water* (submitted for publication), 2016.

Socket.io - Link: http://socket.io (accessed 21 June 2016)

Figure 1: STREAM CDT, 11-14 July, Torquay – groups of users compete against other teams sitting on different tables. The TB-SeGWADE 3D interface shows the network and an information box with the chosen diameter for a pipe colored in white. The green portion of the network is the part that the player can control.

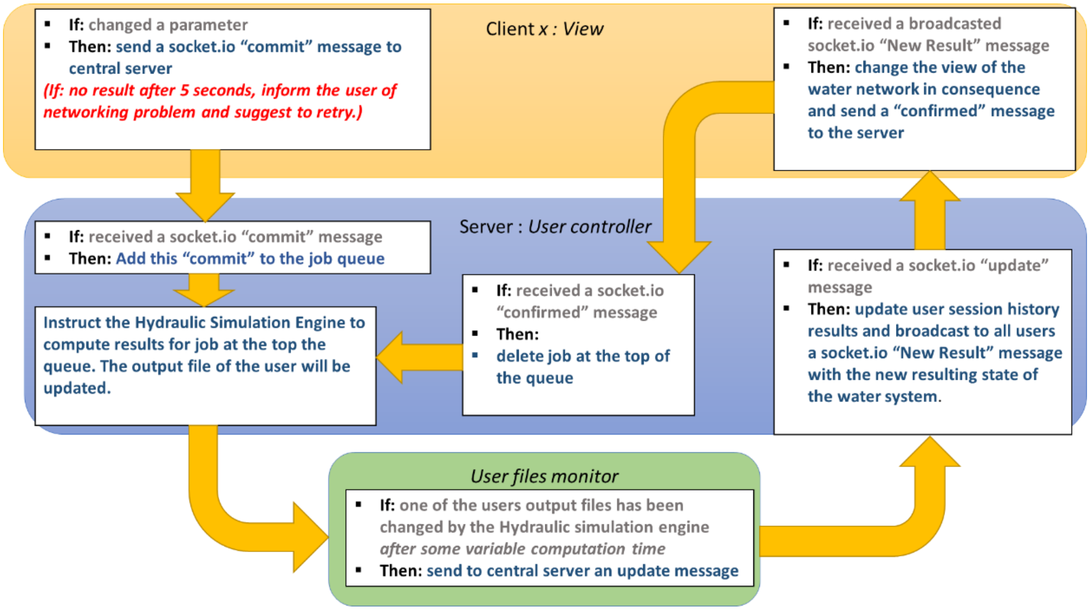

Figure 4: Network lapse tolerant Socket.io messaging system between clients and server developed as an improved strategy.

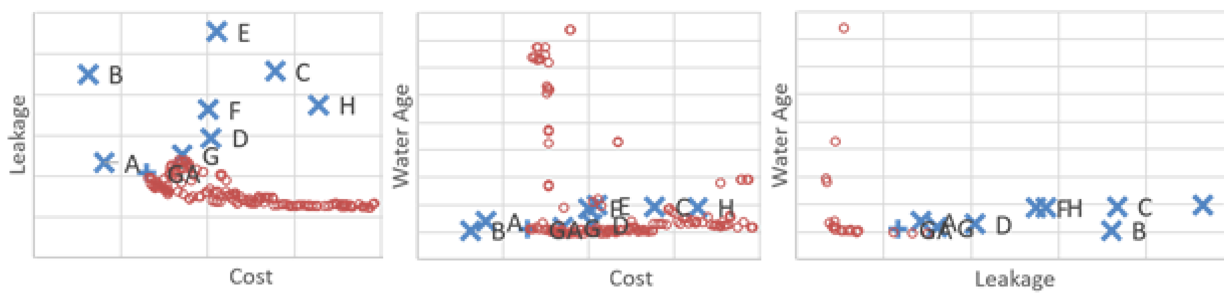

Figure 5: Performance of the height teams (X marks), the Genetic Algorithm optimiser (+ mark), and the Pareto optimal front (O marks) obtained through GA optimization of the problem.