DV2573 DECISION SUPPORT SYSTEMS

DSS Model with the Unity Game Engine for Monitoring Dam Performance

Adrian Mikel Maiza (19971230-T531)[1], Florian Dabat (20000807-T219)[2], Gaik Teng Ooi (19990402-T525)[3]

Blekinge Tekniska Högskola, Gräsvik Campus, Karlskrona, Sweden

[1]adma22@student.bth.se,[2]flda22@student.bth.se,[3]gaoo22@student.bth.se

Abstract

The electric power industry often uses 2D visualizations in their monitoring systems to represent different types of data on the conditions of hydroelectric power plants. These methods provide spatial information in a way that could be too complex for unfamiliar users to understand, requiring them to expend more time and energy in deciphering what could be simplified through better visualization. Therefore, this project proposes to develop a 3D prototype simulating a hydroelectric dam using the Unity graphics engine. We also recommend the use of fuzzy logic systems as part of the solution, specifically a mamdani-based decision support system for decision making in high-risk actions such as dam overflow or turbine failure.

1 Introduction

This report describes a Decision Support System (DSS) model, which simulates the management of a hydroelectric dam using the Unity game engine. The prototype is intended to display real-time information about the dam's performance. The DSS is based on a mamdani fuzzy inference system that maps out the recommended course of action based on given variables.

In widely-implemented monitoring systems in the industry, the person in charge of monitoring hydroelectric dams (henceforth referred to as 'the end user') is bombarded with information overload, about different components ranging from the electrical to the mechanical. As such, this could lead to decision fatigue on the part of the end-user as well as a clouded ability to envision the degree of damage to each component and the

consequences of the decisions they will make on the dam's operations.

In this report, we propose a 3d model of a dam system operating based on a mamdani fuzzy logic decision support system, with the model and interface powered by the Unity game engine.

2 Division of labor

We divided our project duties according to our respective experiences and competencies as demonstrated in Table 1.

Surname	Name	Project duties*
Maiza	**Adrian Mikel**	A, B, C, E, F
Dabat	**Florian**	B, C, E, F
Ooi	**Gaik Teng**	C, D, E, F

Table 1: * **A** - Project Director, **B** - Programmer, **C** - Designer, **D** - Researcher, **E** - Report Writer, **F** - Presenter.

3 Planning Phase as Explained through Simon's Model

After deciding on our project, we dedicated the first few weeks to researching academic works related to the topic to understand the challenges in-depth. Our research process can be modeled with the Simon's model of decision making, with there being 5 main phases: intelligence phase, design phase and choice phase. Simon's model has a total of five phases with the last two being irrelevant to our report - Implementation phase, and Feedback and monitoring phase - since our project has not progressed to the point of implementation in the industry.

Intelligence Phase - We began by examining the status quo of modern dam management systems and their challenges. The reality is that there is a need for automated decision support in dam management, as the frequency at which major problems arise can be too high to rely purely on human judgment. The human employee's ability to make decisions is further clouded by the burden of stakeholder's expectations and the existence of multiple uncertainties that would significantly influence dam operations. [1] Such uncertainties include, but are certainly not limited to: weather anomalies, malfunctioning of electrical components and accuracy of forecasts. [2]

We came across the following problems:

- What improvements can we make to current dam management interfaces to convey information more clearly to the reader?
- How can we use this interface to highlight urgent events immediately requiring the reader's attention and decision?
- What decision support system could we use to accurately model the many requirements and subsystems of a hydroelectric dam?

We then came up with the problem statement: *is it possible to combine DSS technology with a 3D game engine for dam management systems?*

Design Phase - We employed the use of online visual platforms like Miro for brainstorming solutions. From there, we came up with a set of basic features we wanted to include in our prototype, which we then established as the following objectives:

1. Monitors and displays energy production for each turbine in real-time
2. Monitors and displays the level of water in the dam in real-time
3. Maintains optimal temperature of the control module
4. Controlling the state of dam gates between "fully open" and "fully closed" depending on the water level

5. Calculate energy storage capacity
6. Monitors and controls water turbine velocity
7. Makes decisions based on energy price forecasts pulled from online forecasts to optimize spending costs
8. Monitors and displays actual power consumption to showcase the dam's efficiency
9. Makes decisions based on the current weather
10. Forecast the prices of energy using machine learning to efficiently reduce the costs
11. Keeps track of the elevations of neighbouring villages and cities to avoid inundations
12. Controls the state of the gates based on streamflow forecast
13. Monitors CO2 releases
14. Monitors the state of the refrigeration gates for the turbines
15. Checks for seismic activities
16. Warns the person in charge in case of need for human intervention
17. Keeps track of the number of employees available when needed
18. Monitors the number of employees with specific skills

We designed the architecture to cover all those requirements (see Figure 1).

Choice Phase - Based on the existing knowledge and skill sets of our group, we considered using heuristics and intelligent systems to solve the problems. We settled for an approach relying on fuzzy logic, which has already been done in previous academic works using an ANFIS model (adaptive network-based fuzzy inference system) [3].

We also first attempted to model a hydroelectric dam system with the Defold game engine as per the initial proposal but switched to Unity instead. This was due to Unity has more documentation online, the ease of importing assets from Blender, and a more forgiving learning curve.

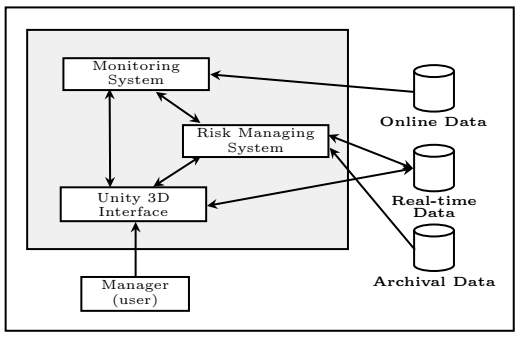


Fig. 1: Designed architecture.

4 Implementation - Decision Support with Fuzzy Logic

Hydro turbine health monitoring - Fully functioning hydro turbines are integral for dam operations, considering that inefficient turbines lead to a loss of power which would incur unnecessary costs. In worst case scenarios, they could even lead to accidents if turbines overheat to the point of causing turbine fires due to mechanical and electrical malfunctions. Hence we elected to implement a turbine health monitoring system on the basis of a fuzzy logic system. The input variables chosen to monitor the hydro turbine's health are turbine temperature, turbine blade rotation speed and efficiency.

The temperature of the turbine is a reliable indicator of the turbine's health. When it overheats, one can say with certainty that it is approaching its failing point. The rotation speed of turbine blades is defined in units of rotations per minute (rpm). Efficiency is the rate of useful power output to the power input, which decreases along with the turbine's health due to excessive heat loss, for instance.

There are multitudes of factors that determine the ranges of each variable, like the

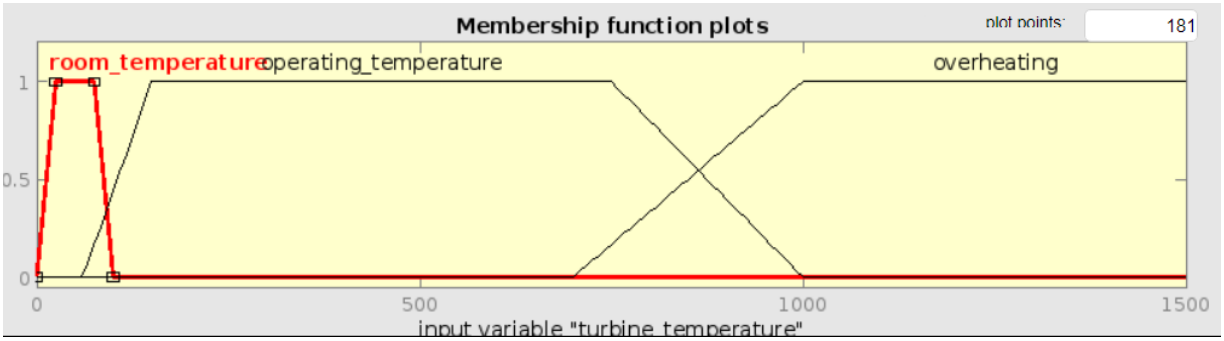


Fig. 2: Function plot for the input variable turbine temperature.

blade length, blade material, how deeply the fan is nestled in the turbine's outer casing, and so on so forth. For this project, we chose an arbitrary range of values for each variable, with the assumption these ranges are unique to the turbine used in our Unity project - these ranges may not be suited to monitor the health of real-world turbines accurately.

4.1 Defining input variables

The temperature was mapped from the lowest being 20°C (the temperature of a turbine in 'resting' state, or when it's turned off) and the highest being the turbine blades' melting point (1000°C), although turbines can still operate at much higher temperatures. [4]
We defined the following fuzzy sets for each input variable:

$$T_{turbine} = [(T_{room}), (T_{operation}), (T_{overheating})] \quad (1)$$

with the following ranges of values for each element in the turbine temperature set:

$$T_{room} = (0, 0), (1, 25), (1, 75), (0, 100) \quad (2)$$

resulting in the following function plots (see Figure 3).

The same process was applied to the ranges of the variables 'turbine efficiency' and turbine blade rotation (here written as 'turbine velocity'). Turbine efficiency was on a scale of 0% to 100%, while turbine blade rotation had the range of 0 to 1000 rpm (rotations per minute).

4.2 Defining the output variable

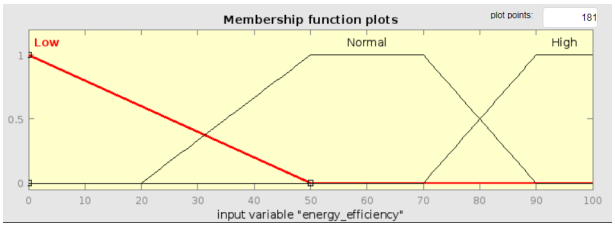
The resulting output variable is the state of the turbine's health. The states were defined as 'off', 'normal' and 'damaged'. Since there is no conceivable way of quantifying it, the range is given as 0 to 1 as increasing severity of damage to the turbine (see Figure 4).

4.3 Rule definition

As the basis for all decisions made by the mamdani system, a set of rules must be defined. Using the rule editor in MATLAB, we defined a set of rules stating the preconditions for the turbine to be in each state. For example, the first rule in the ruleset is as follows:

If (turbine temperature is overheating) and (turbine velocity is fast) and (energy efficiency is low) then (urgency for repairs is damaged). (3)

Based on each rule, the fuzzy logic system comes up with a fuzzy set, visually illustrated by MATLAB's fuzzy logic designer system as a 3D mesh, which is implemented in the Unity prototype's GUI to guide the



(a) Turbine efficiency.

(b) Turbine rotation velocity.

Fig. 3: Membership plot functions for input variables.

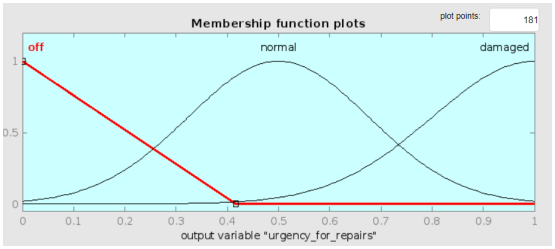


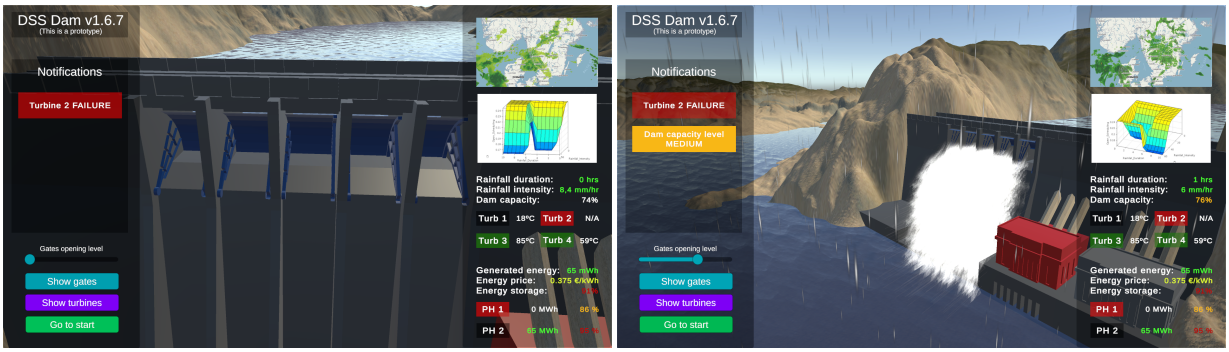
Fig. 4: Membership plot function defining the output variable, stages of turbine health.

user on how and why they should schedule repairs for the turbine based on its output.

5 Implementation - Final 3D Prototype in Unity

Advantages over Defold - We first began to implement the model using the Defold game engine as said previously. However, we came across some problems with this engine. Firstly, when we tried to import the 3D model of the dam, we noticed that Defold did not support Blender files and required further modifications, which was a time consuming task for us as it was too far outside our scope of knowledge and we wanted to focus on investing more time in fine-tuning prototype in the middle-to-late stages. Furthermore, although it is possible to do 3D projects with it, Defold seems more suited for 2D games as there are limited amounts of built-in tools and assets for 3D modelling compared to other engines. This could also be because this engine uses the Lua language that is less common than some others. This posed a tough challenge since learning this language from scratch might compromise our focus on the quality of the prototype. For these reasons, we made the decision to use another engine. The one we chose is Unity as it seemed to be beginner-friendly and there was abundant documentation for it online. C# is the programming language used C# which is easier for us because we had experience with it or really similar languages such as Java. This saved us significantly more time in comparison to coding in Lua. The biggest advantage of this solution is that Unity supports more file extensions and it allows direct imports of Blender files. It still required a few modifications such as the rotation axis for the flooding gates, but overall the amount of work needed was much less than before and the tool was much easier to work with. Some other built-in tools let us add more professional-looking touches, like the fully customizable particles system for realistic-looking water- and rainfall animations.

Available features - The Unity 3D simulation is supposed to help the end-user have a better understanding of the current status of the dam. For that reason, the prototype provides real time information in 3D and 2D. The simulation offers the end user the possibility to check the status of each of the turbines and powerhouses from all directions, as well as the degree to which each gate is cur-



(a) Gates closed

(b) Gates opened

Fig. 5: 3D Visualization of the status of the dam's gates. (a) shows closed gates and (b) shows opened gates with the respective particle system that shows the amount of water going through the gates.

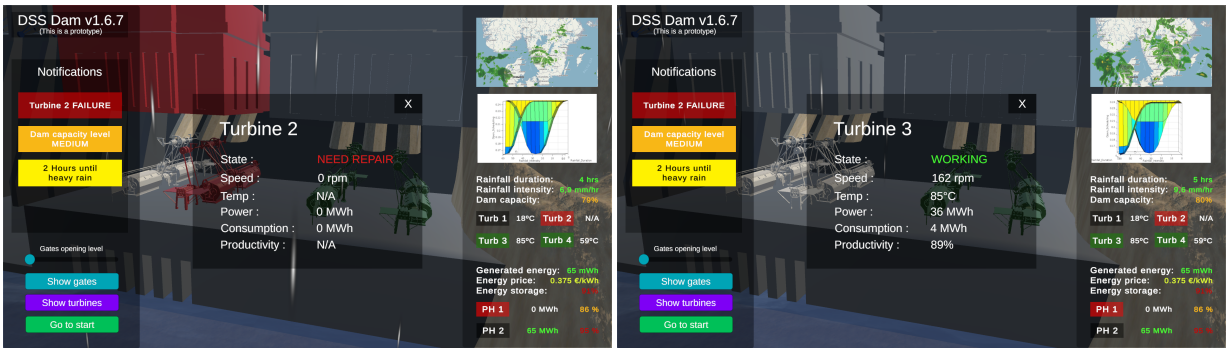
rently open. This opening level of the gates is controllable through the simulation itself and shows the amount of water going through the gates with a particle system. As a more visual way of displaying information, it allows the user to understand the current status of the dam on a more instinctive level(see Figure 5).

Also, the simulation offers a 2D GUI that displays various types of information.The most pressing events are displayed in a notification panel to the left of the screen. Below it, a slider grants control over the water gates (specifically their degree of opening), allowing the user to open and close them depending on how close to overflowing the mamdani system decrees the dam is. Below it, three buttons are available: one for moving the camera to the turbines, another for moving the camera towards the gates and the last one to go back to the main view of the dam. These buttons were thought necessary to be able to get a better view of some specific parts of the dam. On top of that, when checking on the turbines, the powerhouses will be slightly elevated so that the turbines can be observed in more detail.

On the right panel, starting from top to bottom, we have actual satellite meteorological information, the resulting surface view of the fuzzy logic system, live information about rain measurements (rainfall duration and rainfall intensity) and the status of the dam's capacity, the status of the turbines and their temperature, live energy information (generated energy, energy price and energy storage) and the status of the powerhouses.

When clicking on one of the turbines, additional information will be displayed in the middle of the screen in the form of a modal window that can be seen in Figure 6b. The state of the turbine is here clearly stated and not just represented by a color, and the temperature is still present. On top of that, the rotation speed of the turbine is given, as well as data regarding the energy production: power generated, power consumed (i.e. not converted in energy), and the productivity that is deduced from the two previous metrics.

Display of important information - For the users to have a clear overview of the current state of the dam, the prototype takes a variety of information that is displayed into account. As the amount of information might be a bit overwhelming at first, the decision to display the most relevant one with specific colors to indicate a particularly good or bad



(a) Turbine failure

(b) Working turbine

Fig. 6: Important information display. (a) displays the information about a turbine with failure status and (b) displays the information about a working turbine.

value was taken. As such, a gradient from green to red was used to indicate the current state of the turbines, or the capacity of the dam. These choices, though easy to understand, might still not be enough since it is necessary to select a specific turbine to have access to more information about it, so we thought that having only a small red text on the screen may take too much time to be noticed. For this reason, additional colors were added for both the turbines and the powerhouses, once again using green to indicate a good state and red to indicate that a problem has occurred. For the turbines, though, differentiation between a non-functioning one and a damaged one was required and so a white color was adopted to that end. Additionally, problematic turbines and the powerhouse containing it will blink in red so that it is immediately brought to the user's attention. As previously stated, a notification center was also added to show that information along with more subtle ones, such as the amount of water in the dam when it reaches a high enough value. All these features in terms of information visualization can be seen in Figure 6.

Trouble with the implementation of the model - The original plan was for the system to allow the user to directly see the consequence of a specific action, so that they are convinced that they made the best decision. The link between the MATLAB-produced DSS and the Unity project was, however, much more complex than expected and we lacked the experience to combine them with one another. This was coupled with the fact that we had to switch from Defold to Unity in the duration of the project. As such, some of the data used in the mamdani fuzzy inference system isn't taken into account yet in the prototype. As it stands, it is ,for now, better to look at both the software and the model as separate entities.

6 Personal reflections

Adrian Mikel Maiza - The beginning of the project was a little bit overwhelming since the topic was new to me and the technologies that we were about to use were also new. The learning curve at the beginning was slow because using Defold for our 3D project was not straightforward and we lost a lot of time figuring out how to use it. Once we switched to Unity, things started to go better. I learned a lot about 3D modelling with Blender and Unity and how to use them for modelling a Decision Support System. Regarding time for designing and developing, maybe a little bit more time to figure out

how to model a good DSS would have been appreciated. But anyways, the project overall was a good opportunity to learn and work together with my group mates, which I already knew beforehand.

Florian Dabat - This project was for me a good opportunity to learn about 3D modelling, but the time given to do it was a bit lacking for me to be able to have satisfying results. Considering that we didn't have any experience with these tools, a longer period of time would have been appreciated as both decision support systems and 3D modelling were completely new to me so I had to spend a lot of time on it to do the simplest things. Adding to that the fact that we started with a software not really beginner-friendly, a lot of time was spent for little results at the beginning. Luckily, a lot of documentation was available for Unity which helped us a lot. The fact that we chose to work on dams, which we didn't know anything about, didn't help on that regard even though it was interesting to research it. Overall it was still a good experience and we were able to get through most of the hardships we encountered.

Gaik Teng Ooi - As a former physics student, the topic on dams was very interesting for me to research about. To my surprise, there was a significant number of research projects on decision support systems for dam monitoring operations, and using the most in-depth reports as references for this project helped us get off to a decent start. I proposed the idea to use fuzzy logic systems after seeing a slide in Lecture 7 mention it briefly, and I had already learned about it from a course in my home university.

Having the option to create our own groups worked particularly well for us, since it's the middle of the semester and I already knew who I would have the most working synergy with.

7 Conclusion

Decisions made based on the information provided by our model prove to be built on more solid foundations as users absorb information more efficiently and can make them with more confidence. Additionally, with the help of fuzzy logic mamdani systems, users are provided with understandable guidelines for making decisions, being more instinctive to use and free of information overload. Therefore, we believe that the prototype we propose in this work can be useful in real-world applications. To this end, it would be convenient in further works to integrate the mamdani fuzzy systems in the same prototype and continue working with different criteria to add more functionalities, since at this stage we have only focused on the risk control of dam overflows and turbines failures.

References

1. Ahmad, S. K., Hossain, F.: A web-based decision support system for smart dam operations using weather forecasts. Journal of Hydroinformatics **2**(5), 687–707 (2016)
2. Zhu, F., Zhong, PA., Sun, Y., Yeh, W.: Real-Time Optimal Flood Control Decision Making and Risk Propagation Under Multiple Uncertainties. Water Resources Research
3. Ranković, V., Grujović, N., Divac, D., Milivojević, N., Novaković, A. : Modelling of Dam Behaviour Based on Neuro-Fuzzy Identification. Engineering Structures Volume 35 **2**(5), 107-113 (February 2012)
4. Han, JC. : Fundamental Gas Turbine Heat Transfer. J. Thermal Sci. Eng. Appl. Jun 2013