MindLab: A Flexible Framework for Training Decision-Making

Laila Frotjold

SIKT AS

Abstract

Decision makers are often faced with insufficient and incomplete information, yet are forced to make decisions on this basis. The result may often be unintended consequences or situations where too few or too many resources have been allocated to solve the problem. Practicing decision making is often realised through live-exercises, which tend to be extremely expensive, or by using table-top games, providing a much lesser amount of realism to the game. MindLab allows for more sophisticated training arenas to a relatively low cost. The idea is to create a simulation model general enough to accommodate different decision making scenarios, accompanied by relatively rich user interfaces and an experiment setting that gives the game a high level of realism. This paper looks into how the MindLab architecture functions, as well as presenting two different simulation models with accompanied user interfaces that are currently being used with MindLab.

Introduction

Decision makers are often faced with insufficient and incomplete information, yet are forced to make decisions on this basis. The result may often be unintended consequences or situations where too few or too many resources have been allocated to solve the problem. Practicing decision making is often realised through live-exercises, which tend to be extremely expensive, or by using table-top games, providing a much lesser amount of realism and feedback to the game.

MindLab is a software project that focuses on sensemaking (Weick, 1995) and decision making in various forms. Its flexible architecture provides for development of training arenas in which the user can play against other users, or face the simulation model alone. It allows for sophisticated training arenas to a relatively low cost. The idea is to create a simulation model general enough to accommodate different sensemaking and decision making scenarios, accompanied by a user interface and an experiment setting that gives the game a higher level of realism.

This paper will first and foremost look into the technical aspects of MindLab, and the ways in which the technology can make it easier for modellers to create rich user experiences from their models, and such improve the availability of the system dynamics methodology. While it is exemplified with real-world experiments and actual models, the focus remains on the possibilities offered by MindLab as a whole.

Background

The MindLab architecture was initially developed as part of an experimentation field for the network centric warfare (NCW) concept. The idea behind NCW is to allow personnel, communication systems, command joints, sensors, weapon platforms and departments to work together – independent of branch, physical location and level in the command chain. The intent is to facilitate fast decisions and rapid reactions as well as using the appropriate resources at the right time and place. A key concept is situation awareness and the importance of sharing the same understanding of any given situation (Forsvarsnett, 2006).

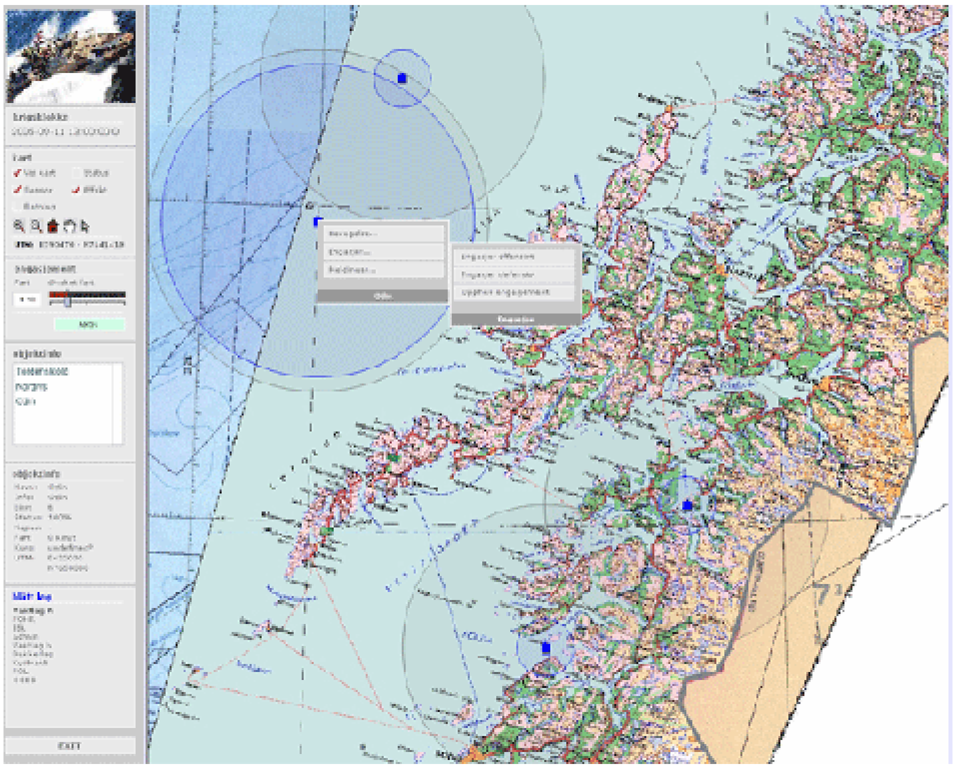


Figure 1: Screenshot of a MindLab user interface

The earliest versions of MindLab were tailored to this purpose, its aim being to serve as a practice arena for NCW. A Vensim simulation model allowed for different scenarios to be played, and a client interface was developed in Macromedia Flash. As Figure 1 shows, the interface is based on a map, in which different resources are placed. Specific information about user interface elements is found in the menu bar to the left. The simulation model keeps track of the different teams, players and units, as well as their belonging properties (such as team colour or maximum speed for a unit). Each player is typically represented by a group of users, in which the users need to collaborate in order to decide the actions of that player. The users receive different information depending on the team to which they belong and what the team's data collecting sensors cover. Through interaction with the interface, the users can also manipulate the underlying model, for example by moving a unit or engaging other units. Part of the idea with this concept is that some players (usually at higher decision levels) cannot directly manipulate units, they need to communicate their intentions through other players, usually introducing delays and miscomprehensions.

In this early version, the client, the server and the model were all tightly coupled, and could not function as stand-alone, interchangeable applications. As experiments showed that the training arena provided by MindLab was highly appropriate for the problem domain, and as the potential of the concept became clearer, MindLab was rewritten and refined to a far more general structure, able to accommodate any kind of simulation model and any kind of user interface.

Methodology

MindLab can support wide scoop of learning theory, since the instruction depends on the simulation model and the user interface. For example, the single-user model developed for MindLab is rooted in trivial constructivist (von Glaserfeld, 1990) learning theory, which emphasizes the interlacing of content, context and understanding, centred on individual construction of knowledge. This kind of constructivism seeks to achieve learning through active exploration rather than traditional textbook environments (Norman & Spohrer, 1996). Simulations and microworlds are explicitly studied and found suitable for stimulating learning within this discipline.

The multi-user model on the other hand, embraces the concepts of social constructivism (Vygotsky, 1978) and situated learning, where learning is seen as a function of activity, context and culture. Lave & Wenger hold that knowledge needs to be presented in an authentic context, and that learning requires social interaction and collaboration (Lave & Wenger, 1991, at Psychology.org, 2006). This is exactly what the multi-user model seeks to achieve. The authentic context is given partly by the scenario applied to the model, and partly by the user interface, which presents the user with a map containing sensors and effectors, much similar to a real military information display. The social interaction takes place both within the groups that represent each player and between the groups, who need to communicate by means of e-mail or other available communication channels.

MindLab Architecture

MindLab consists of four main components: a simulation model, a database, simulation server architecture, and the user interface. Different simulation models can be used, the only requirement is an implementation of a general interface for communication with the server. Similarly, different clients can be used, given that they adhere to the xml-based communication protocol defined by the server. The use of a database is optional, but typically provides a convenient way to initialise the model with different parameters. This way, one can easily apply different parameter sets to different games. Figure 2 illustrates the concept. The current database also contains other data, such as logging of user activity and results obtained by the different users. The applicability of these features naturally depend on the model in question and on the interests of the model designer.

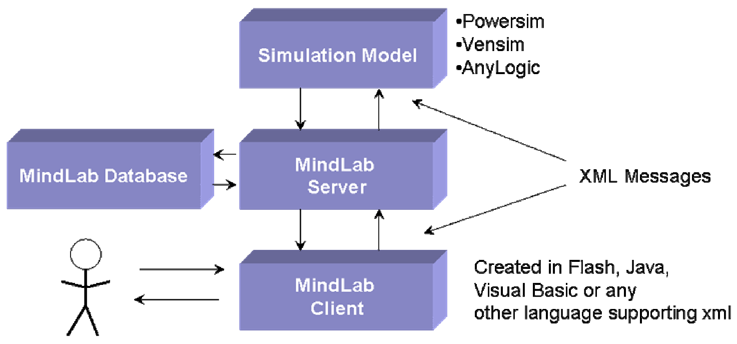


Figure 2: MindLab architecture

The communication with the clients is based on a tailored version of the command pattern (Freeman et al.). A command object is simply an object representing a command, that is, an action that the client is allowed to perform on the server. This could be anything from a login-action to a request of changing a parameter in the simulation model. The objects are communicated via XML, and the client is thus required to implement functionality for parsing and using XML.

The client will have access to different commands at different stages of the game. For example, once it connects to the server, it gets access to a login command. If it logs in successfully, a new list of commands is sent to the client, including retrieval of sessions, creation and joining of sessions etc. When the client has joined a session, and implicitly, a game, it gets access to commands for manipulating the simulation model. This way the server has full control over the actions each client is allowed to make at any time. It also makes it easy to extend the behaviour of the client, simply by adding command objects to its “allowed-list”.

A simulation model used with MindLab needs a minimum set of methods to provide information to the server and the client. These methods have been gathered in a general interface containing the most important methods for setting initialisation parameters as well as retrieving XML data. Different simulation software allows different method calls, and different simulation models require different data to be updated. This fact has been acknowledged through the method “callModelMethod” in the interface:

```
public void callModelMethod(String methodName, List parameters)
```

This method allows the client to specify the method and the according parameters. For a Powersim simulation model for example, the method name could be “setValue” and the variable name and new value would be listed as parameters, and the method could then be executed via Powersim’s COM-interface. For an AnyLogic model, the method can be more specific, for example “moveUnit”, providing the unit ID and the new position in the parameter list. In any case, the server does not need to know what method calls the model accepts.

Another feature recently included in the server part of the architecture is a questionnaire component, a feature that allows modellers to pop up questionnaires to the user at specific times of model execution. The answers provided by the user are then stored in the database.

Database

The database can contain any initialisation parameter used by the simulation model, and is not a required part of the MindLab architecture. Using a simulation model without connecting to a database is of course not a problem, however, a database can provide more flexibility with respect to policy testing and storing of different sets of parameters. Another advantage provided by the database, is the use of the server’s logging functionality, which is useful for analytic purposes. Events at any detail level can be logged, a feature which gives the modeller a convenient tool for learning about the way in which the model is used. The simulation models currently used with MindLab both use the same scenario database, given their scenario-oriented nature. A general skeleton with information regarding each scenario, information about the user interface, as well as user data, forms the base information. In addition to this, the database has been extended with data relevant for the two simulation models used this far. Some of this more specific data is overlapping for the two models, and some data only applies to one model. This setup has proven to be flexible and allows the administrator to compose different scenarios from the same base objects, as well as adding new objects as needed.

Simulation model

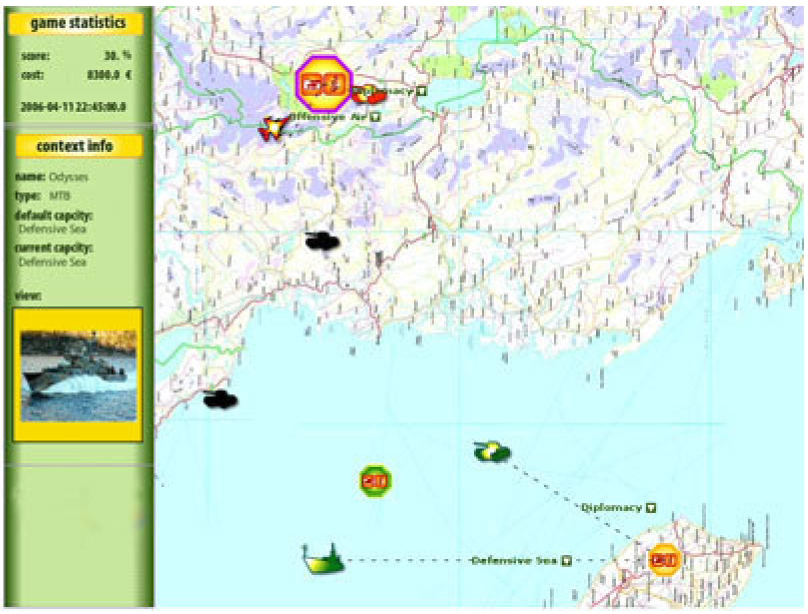
The server currently only supports use of AnyLogic simulation models. Support for Powersim and Vensim models are planned, and such support can be implemented as the need for it arises. In order to make a simulation model adaptable to the system, it needs to implement a generic interface that the server can use for communication. As AnyLogic is Java-based, this implementation is rather straight-forward, because the model itself can implement the interface. For other simulation technologies however, a Java-based communication-layer needs to be constructed.

Given some basic initialisation data (such as a scenario identifier), the models are expected to perform their own initialisation. For the current models this means connecting to the database and retrieve relevant data. The initialisation procedure is likely to vary from model to model, and this responsibility has thus been placed with the model itself.

Once initialised, the server can perform basic model-control actions, such as advancing the model one timestep. For each advance, it retrieves the updated model data and passes it to the clients. The updated data needs to be presented in XML format, other than that, the model is free to produce whatever update it wishes. This way, it is a matter between the model and the client (user interface) which data are to be transmitted, and how the XML is to be structured. The server knows nothing about these data.

Specific models

As stated above, MindLab can accommodate any simulation model as long as it complies with the communication protocol defined by the game server. The two models currently used have different structures and serves different purposes. The multi-user model provides predominantly a framework consisting of teams, players and units, where the tasks and dilemmas are partly presented through an external scenario brief, and partly generated by the players themselves. Communication delays and scarce resources are often the main sources for discussions and prioritising performed by the players.



score: 30. %
cost: 8300.0 €
2006-04-11 22:45:00.0
name: Odysses
type: MTB
default capcity: Defensive Sea
current capcity: Defensive Sea
view:

Figure 3: The user interface of the single user model

The single-user model on the other hand presents the tasks as inherent elements of the model, and the user interface can thus represent the dilemmas visually. The tasks have pre-defined requirements for being solved, and can escalate through several degrees of severity before either timing out or being solved by the user. A resource can also be required for a longer period of time in order for it to take effect. The challenge for the user is thus to find the optimal combination of resources to apply to each task, and to prioritise among the present tasks (some tasks may be less important or require zero resources, indicating that it disperses by itself and disappears).

The user interface

Two different user interfaces have been developed to work with the MindLab architecture. They are both implemented in Macromedia Flash, though any technology can be used as long as it supports socket connections and can parse and generate XML. Figure 3 shows a screenshot of the single-user client, which uses a map to create physical dispersion, and displays tasks and units as symbols placed in the map. As with the multi-user game (Figure 1), extra information about units and tasks can be found in the menu to the left. Although the main concept is the same for the two models (a map and a context menu), the functionality in the two interfaces is very different. One has a scrollable and zoomable map, with the possibility of using an ArcIMS map service, whereas the other uses a static, fixed-sized bitmap. One portrays units with sensors and effectors, whereas the other only implicitly includes sensors and effectors, but then again has the possibility of allocating different capacities to different units and groups of units. One interface has the possibility of visualizing tasks, whereas the other leaves the concrete task much to the imagination of the users. One makes use of the server's questionnaire functionality, the other has yet to make use of this feature.

Experiments and usage

The multi-user model has been used by the Norwegian defence for over a year, with different groups of users and different settings. The single-user model has not yet been used in formal experiments, and is likely to go through another development phase before it is used as part of the training at the defence's schools. The multi-user experiments have been conducted at different locations, with users at different levels of expertise, yet usually following the same pattern. The common purpose has been to train the users' abilities to communicate and develop a common understanding of a complex situation, often with limited means of communication. It has also been an aim to train the collaborative skills of the users and to encourage to reflection around the ways in which decisions are made.

In general, local intranets have been set up for the purpose of the game. Different players have been located in different rooms, equipped with one or two computers and the available means of communication. A minimum level of communication is e-mail messaging between the players. Sometimes, depending on the scenario and the premises, the players have been allowed to use VHF or pretended video conferences realized through meetings in the hall. Usually, one player in the game is represented with several persons who need to cooperate in order to decide the actions of the player. This way cooperative skills are trained both within the groups and between the groups. The number of players present varies from scenario to scenario, but six or seven players are not unusual. The game stab, representing civil and/or enemy units, is located in a separate room, and generates situations based on the actions carried out by the players. This way the temperature can be adjusted by introducing unexpected events if the given scenario does not produce enough decision dilemmas.

Before the game starts, the players are given a scenario brief, often a text document accompanied by oral instructions. They need to create a statement of mission and establish rules of engagement. As the game starts they all go to the different rooms, and from that point on, the means of communication are limited.

After the game, de-brief is normally conducted, where the different player groups describe their situational reports, and the players comment on what they learned, what they could/should have done differently, what went wrong etcetera.

Conclusion

While findings and observations from the experiments are interesting and indeed could be elaborated upon, the main interest of this paper is to look into the possibilities that MindLab offers. Several development iterations with experiments along the way have helped define the product and identify the most important features required by such a tool. Looking at MindLab as it presents itself today, it is clear that its first and foremost advantage is the ease with which it lets the modeller turn the model into a game with an interface that can reach an audience beyond the system dynamics circle. The additional possibilities of using logging and questionnaires allows for quite sophisticated data collection for analytical purposes.

In addition to these advantages, MindLab decouples the user interface from the model, rather than bundling the model with the interface. This allows for the realisation of multi-user sessions as well as different user interfaces for each model. It further allows the same, or more or less the same, user interface to be used with different models.

In conclusion, we believe MindLab to be a powerful tool in both academic and commercial contexts. It serves both areas as a tool for creating learning environments for system dynamics models, and academic circles in particular by offering the modeller to learn about the ways in which the models are used through logging and questionnaires.

Future work

MindLab is under continuous development, and among the desired functionality are an integrated messaging system, web-camera facilities and a more holistic learning environment, presenting the user with a schedule consisting of games to play, information to read and questionnaires to complete. In addition to this, modules for communication with other simulation software than AnyLogic are planned. Each development iteration aims to leave MindLab a little more general, flexible and robust.

References

Freeman E., Freeman E., Sierra K., Bates B. (2004) *Head First Design Patterns.* O'Reilly Media, United States of America

Forsvarsnett (2006). http://www.mil.no/fu03/start/info/nettverk/ 2006: Norwegian Defence Official Website [Accessed: 28. Februar, 2006)

Norman, D. A., & Spohrer, J. C. (1996) Learner-Centered Education. *Communications of the ACM*, 39(4), 24-27.

Psychology.org (2006). http://tip.psychology.org/lave.html. 2006: Psychology.org's website. [Accessed: 28. Februar, 2006]

Vygotsky L.S. (1978) Mind in society. Cambridge, MA: Harvard University Press.

Von Glaserfeld, E. (1990) An exposition of constructivism: Why some like it radical. In R.B.Davis, C.A.Maher and N.Noddings (Eds), Constructivist views on the teaching and learning of mathematics (pp 19-29). Reston, Virginia: National Concil of Teachers of Mathematics.

Weick, K. E. (1995) Sensemaking in organisations (Thousand Oakes, CA: Sage)