Uses of Empirical Modelling in driver education

0824599

Abstract

Empirical Modelling is a novel paradigm of thinking about learning based on constructivist ideas. Through observation a modeller can build a computerised construal that represents their understanding of a real-world artefact. Through the modelling process and experimentation on an object a modeller or user of a model will gain knowledge. This paper looks at the issues with EM and related tools and how they can be used to educate. A specific modelling study of a roundabout scenario is proposed where aspiring drivers could experience a roundabout on the computer and experiment with the modifications of certain observables. The driver may then feel more confident when they take to the road as they already have gained some experiential knowledge in this particular domain.

1 Introduction

There are many domains where modelling is used, from scientific observation to assisting in the making of economic and business decisions. These models often lend to being somehow inserted into a computer. There are popular tools for modelling within specific domains, however, these are not always useful when encountering a new unknown domain. Empirical Modelling aims to remedy the limitations that occur with traditional approaches and provide a more frictionless approach to extracting an understanding from the mind and putting it into a form that is manipulable both by a user, as they are gaining an understanding of a given artefact, and by other people who may learn from the other person's understanding.

This paper will begin by describing what Empirical Modelling is, how it came about, the principals of EM and how it may be used when learning. This will then be followed by a study on the creation of a model that can be used to help a person learn about the use of a roundabout. Finally, the paper will conclude by analysing the challenges that exist with Empirical Modelling at the moment and a discussion about the possible directions in which this research area is heading.

1.1 Limitations of Traditional Approaches

The more traditional approaches to modelling, such as building a mathematical model, have a disadvantage in that they are too rigid (Russ, 1997). This leads to problems when the creator of the model has incomplete knowledge of the domain in which they are modelling as mathematical models are not necessarily easily revisable. Another problem with the more traditional approaches to modelling is that there is a clear separation between the model and the user's head and the model that is being created. This separation can create a friction in the modelling process as the user is less likely to be bold and make changes that will not be easy to correct.

1.2 Overview of EM

EM presents a new paradigm that takes a more constructivist approach at learning, rather than the "instructionist" approach that is taught in classical methods (Beynon and Harfield, 2007). As a result knowledge is created in an experiential rather than declarative manner.

1.2.1 Fundamentals

The central focus is put upon a *construal* which is developed over time through observation of some real-world artefact. There are three fundamental concepts of EM: *observables*, *dependency* and *agents*.

An observable is a property of the environment being modelled.

Dependency is a relationship between one or more observables. For a model to work it is important to ensure that dependency chains remain acyclic. The principle of dependencies is that when an observable has been changed so do the dependent observables.

Finally, an agent is an entity who imparts some change on the observables and dependencies. Such agents may or may not be autonomous.

Additionally, the concept of *agency* exists which can be thought of as an agent's ability or permission to observe an observable or change an observable.

1.2.2 Definitive Notations

Definition-based notations, now referred to as *definitive notations* exist to aid in the creation of a model. They remove some of the effort that exists in traditional programming by removing the necessity to explicitly define dependencies and the functions that manage them. A definitive system consists of two types of thing: *variables* which contain data and *definitions* which define the dependencies between objects. A definition typically targets a single variable and is defined in terms of dependency on one or more variables. The target is then computed using this definition.

The notation that will be used throughout this paper is the Evaluator of DEfinitive Notations (EDEN). Following sections shall contain more discussion of the usage of definitive notations and why they are useful in this context.

1.2.3 The Construal

A construal is consists of the artefacts that make up the modeller's understanding of an environment. The construal is built up through experimentation and interaction with the environment.

2 Computers in Learning

It will first be considered how computers can be used in education and where EM may have a place. Thanks the decreasing cost of computer hardware computing have become ubiquitous. Most houses and places of education have access to a computer and therefore a computer can be a vital tool in the delivery of educational material. Despite this availability and the initial optimism surrounding the usage of computers in education, educational usage has seen a disappointing uptake (Beynon, 2007).

Computer applications could provide the user with a richer experience than existing pen and paper methods as multimedia content could be made available. Also, computers could afford the learner with the ability to experiment with the data given to them to aid their learning. Within a computing environment there are usually few ill-consequences for making mistakes which could encourage learning, if only the correct resources were available.

The description of a generalised spread-sheet type application is frequently cited as a positive example of a successful application of computing in learning (Russ, 1997). A key problem with traditional programming that is used in computing today is that modifications cannot be made "live" as the programme is running. This is counter-intuitive when taking a constructivist approach to learning as the way human beings learn through manipulation of artefacts and observations of the consequences. These consequences may lead to further experimentation to gain and reinforce knowledge about the artefact. Of course spread-sheets address the problem of modifying an application while running, but they are not particularly visual applications and the domain in which they can be used is therefore too limited to be used as general purpose learning software.

2.1 Constructivism

The idea of constructivism existed many years before modern-day computers were invented. As a result there are many different schools of thought on constructivism. The most very basic principal of constructivism is that knowledge is constructed in the mind of a learner. An expansion of this is that people learn by looking for meaning and patterns in the events that are going on in their environment and that full information about the environment is not necessary to construct this knowledge (Watzlawick, 1984).

The constructivist approach to teaching lends well to the use of computers in education as the experiential element of learning can be exploited, given the correct tools. Bodner (1986) remarks that the constructivist model of learning has significant overlaps with the influential scientist and philosopher, Kuhn's analysis of science.

Empirical Modelling, with its base in constructivism, exploits the principals of constructivism.

2.2 EM in Education

A significant focus of EM is put on using it as a teaching tool. As has already been stated, EM aims at providing a more experiential approach to learning. Constructivist theories of learning suggest that the learning process consists of the correlation of experiences with some artefact and that lessons are learned through manipulation of the artefact (Beynon, 1997).

(Beynon, 2009) states that computing is ubiquitous and that computers can be used in many applications. It therefore does not necessarily make sense to impose computing domain-specific constructs upon users. While formal methods and formality have their place in computing, in most applications they can exist in the background without loss of functionality. Many artefacts that one may wish to model have nothing to do with formal computing methods and therefore a more generalised form of programming may

be more appropriate to the application. This is where Empirical Modelling comes in to the picture.

3 Modelling Study

A model of a roundabout has been created to demonstrate a number of principals of EM that have already been discussed in Sections 1.2 and 2.2.

3.1 Aims

The model has been created for two explicit purposes. The first purpose is to demonstrate how an understanding of an artefact develops during through the process of building an Empirical Model. The second purpose is to demonstrate how this may be used by a second person to also develop an understanding of this artefact.

It is loosely inspired by a model created for the first Warwick Electronic Bulletin by an anonymous contributor. Where possible parts of the model were directly translated into the JS-EDEN notation. This was done to both demonstrate the differences between the various tools that are available and where any limitations lie.

The model itself is a simplification of road system incorporating a roundabout. The aim of the model is to help the user develop an understanding of how a roundabout works, potentially before being taken onto the road by an instructor. An agent can gain an insight

3.2 Roundabout Model

The model was developed gradually from the roundabout, to the exit roads, the addition of vehicles and finally modelling the vehicles as agents; one of which can be controlled by the user of the model.

This procedural generation of model was natural and frictionless due to the usage of EM principals.

3.3 Modelling in JS-EDEN

Initially the model started as a single definition of a circle which marked the centre of the roundabout (Figure 1). Through the gradual addition of observables and dependencies using the EDEN definitive notation, a model was built up. In

The Final model can be seen in Figure 2 we see a model where all of the lines within the drawing have `centreX` and `centreY` ,which correspond to the centre of the circle in Figure 1, at the root of their dependency chains. As a result changing the values

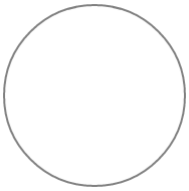

Figure 1: The first stage of modelling

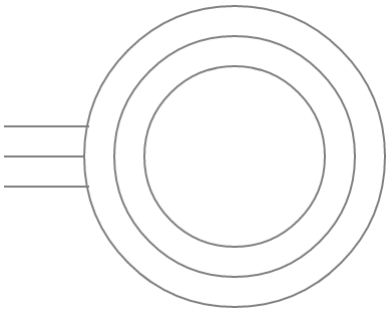

Figure 2: A model with dependencies

of either `centreX` or `centreY` will result in the translation of the whole model in its current form.

A collision detection algorithm was implemented by calculating the line intersects between the car and the road environment. This, of course relies heavily on dependencies. An example of the collision detection can be seen in Figure 3. The roundabout changes red in response to the driver's error providing vital feedback.

3.4 Observables

Within the roundabout model, where possible, everything has been expressed as an observable rather than being hard-coded values within the drawable observables. As a result it is possible to develop meaningful

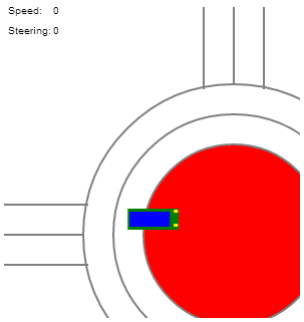

Figure 3: A misguided driver

interdependencies between the observables. These observables can then be modified by the agent to aid in their learning of the environment, as is discussed in Section 3.5.

3.5 Dependencies

As dependencies are a fundamental component of EM it would be apt to have a discussion thereof. Within the roundabouts model relationships between observables have been modelled as dependencies as much as possible to aid extensibility. For example, modifying the width of the roads would modify the co-ordinates the centre of the approach roads. Due to the inability to group objects in JS-EDEN the car is, too, modelled as a series of various dependencies. These dependencies mean that one could add new objects into the environment with their own dependencies; and modification of an observable, such as the road width discussed, will result in the modification of the observables down the chains of dependency.

3.6 Agency

The agents within this model are the driver and the other drivers. Additionally a ”town-planner” agent could be defined as the agent who has agency over observables relating to the roads. For the purpose of this model the driver may also take on the role of the town planner.

Within the JS-EDEN environment the user has agency over all observables through the observables menu to the left of the page. Despite this, the user explicitly has agency over a number of observables including the angle that the exit roads have with each other, the width of the road and the width of the car. This allows the user to experiment with different ratios. Agency is explicitly defined through the usage of sliders, text boxes and buttons that can be interacted with by the user. The car is moved my modifying the acceleration and steering observable. Within a real-world situation, of course, these would be the only observables that the driver would have agency over. These observables are modified using a keyboard input. This input is captured through the injection of JavaScript which then modifies the observables.

3.7 Analysis of EM Tools

During the modelling exercise two environments were used. The model that inspired the creation of this one was the desktop EDEN interpreter. The model itself was then implemented within JS-EDEN using its own syntax.

3.7.1 Desktop Tool

For the purpose of this discussion ”desktop tool” means the tool that runs standalone without the need for an internet connection, the EDEN interpreter. It was created in a time when network connectivity was less prevalent and there was a greater expectation on developers to provide standalone rather than in-browser applications. The interpreter is quite robust with many features for creating graphical interfaces in addition to the ability to create models. This, however is a mixed blessing as the syntax for creating user interfaces is commonly mixed in with the EDEN syntax, sometimes causing confusion.

A version of the EDEN interpreter that can send its output to a remote web browser has been created but when multiple people attempted to access the server it encountered problems with overloading leading to slow responses.

The tool works across platforms (with some tweaks), which is good, but was not very easy to set up a fresh installation on a machine. Usability is quite an important factor when thinking about a modelling environment and this installation process may ”turn off” a user to the idea of using the computer for education.

In summary, the tools are useful but a more robust and

3.7.2 JS-EDEN

The original model was created using a combination of DONALD and EDEN notations. A few parts of the EDEN notation were able to be directly ported, but much of the code had to be rewritten from scratched based on personal experience of the model. This was especially true in the case of DONALD notation which remains largely unimplemented in JS-EDEN. A big feature that’s currently absent in JS-EDEN is the ability to group objects into some shape. This resulted in potentially unnecessary duplication of code which made managing observables much more tricky. Additionally the version of JS-EDEN that was used in the development of this model, emile, was quite slow even on a computer with a multi-threaded quad-core CPU using one of the latest web browsers. This presented a serious challenge when implementing the model as the environment simply crashed when dependency chains became too long and when the observable table became too big. There are also issues with cross-browser support with one of the most

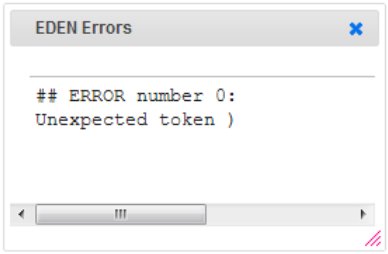

Figure 4: An unhelpful error message

prevalent browsers, Internet Explorer by Microsoft, not being supported. Some of these issues have been addressed in a later version which was released just after the completion of this report. However, these issues will remain a challenge as long as the eden environment is implemented within the current technologies that are included within a browser.

Currently many functions, such as keyboard input are not directly supported by JS-EDEN which necessitates the injection of JavaScript ”hacks”. Although this is reasonable to manage in the case of a relatively small model such as the roundabout model, it is likely to become harder to understand what the model is going to do. This returns to the previous argument about multiple notations. An additional problem is that JavaScript does not follow the EM principals as it is one of the procedural languages that we are trying to move away from when programming using this paradigm.

A final criticism of JS-EDEN is that the error messages are not very descriptive (Figure 4). As a programmer, one may be able to find errors using the vague information in the error window. However a normal user, who JS-EDEN is supposed to be appealing to, may become frustrated and give up after a few futile attempts at debugging.

With this criticism of JS-EDEN it may appear that one is a detractor of its use. This is far from the truth and the criticisms exist primarily to highlight some areas where change is needed. These issues are outweighed by the fact that JS-EDEN provides one single notation with a consistent syntax. This prevents a number of errors that can occur when having to switch between (sometimes vastly) different syntaxes.

Despite these issues, it was fairly straightforward to create the model within the limitations of the language itself. The issues encountered can be forgiven as the new software is in its infancy and currently has a relatively small number of developers.

4 Further work

Initially it was intended that the modelling of blind spots would be included in the model. This, unfortunately, proved to be non-viable due to memory constraints in the JavaScript sandbox on the development machine. In the future, when JS-EDEN has matured to a greater extent it would be useful to add this into the model. Thanks to the way in which Empirical Modelling works this should be a fairly simple exercise as the blind spots could be added as dependencies on existing observables. This would require some degree of transparency to be added to the drawing objects.

It may be noticed that only one light moves properly in correspondence to the car. This is due to a bug that exists where occasionally an element of an array observable cannot be read which causes the whole model to fail to load in JS-EDEN. If I were to have more time I would have sought the source of this bug and then either resolved it or reported it so that it could be fixed in due course.

Some functionality has been implemented but disabled due to JS-EDEN graphical constraints. When transparency is implemented these functions can too be reimplemented. It may have been possible to add an additional JavaScript ”hack” in an attempt at resolving this transparency problem, but that was beyond the scope of this model and would have resulted in time constraint problems.

Because of the above limitations this model could be considered to be a work in progress. When the limitations of the definition language are resolved future users could add their experiences of agency and dependency into the model or build a new model from scratch based upon my experiences as I have noted down in the model.

In the current model the centre of steering is the last point in front of the car, directly between the wheels. This could be made more realistic by researching the geometry of a turning vehicle.

5 Conclusions

Empirical Modelling is still a fairly immature paradigm, despite the relatively long time period that it has existed.

There are some conflicting ideas about the direction of the paradigm and in the past this led to some fragmentation in terms of tool design and notations. Although there may be a need for different notations to fit different domains, differing notations sometimes wildly differ in terms of syntax.This

has largely been addressed by the introduction of JS-EDEN which utilises a more consistent notation language and, importantly, a single syntax. Should JS-EDEN supersede the various other tools, and become the only tool, there will hopefully be more agreement and convergence of ideas amongst researchers in the field.

The introduction of Web Eden made the sharing of EM models more accessible to a wider range of people who may not wish to install special purpose software. As Web Eden relied heavily upon back-end server power it suffered from problems with scalability. JS-EDEN has again solved much of this problem. A future improvement would be to allow people from across the world to collaborate on a single instance of a model with changes propagating to all viewers of the model. This, of course, presents a potential issue of concurrency which will hopefully be addressed in future research.

Throughout the modelling process it was noted that additions of new elements to the computerised version is remarkably easy due to the properties of definitive notations.

Empirical Modelling has yet to gain widespread usage outside of the research group. This is not because the principals of EM are discredited but likely due to the difficulty in changing people's attitudes when it comes to the modification "tried and tested" teaching methods. In the future, as the EM tools become more accessible; mature and become more comprehensively documented, more educationalists will become interested in this new paradigm of computational thinking. As the government has recently been pushing for the improvement of IT education in schools now is a better time than ever to develop and push these methods.

Acknowledgements

I would like to thank Dr. Meurig Beynon and Dr. Steve Russ for pioneering this fascinating new paradigm and giving their time to teaching interested students.

References

M Beynon. Computing technology for learning - in need of a radical new conception. *Educational Technology & Society*, 10:94–106, 2007.

M Beynon. Constructivist computer science education reconstructed. *HEA-ICS ITALICS*, 8:73–90, 2009.

M.B Beynon. Empirical modelling for education technology. In *Cognitive Technology*, pages 54–68, 1997.

M.B Beynon and A Harfield. Lifelong learning, empirical modelling and the promises of constructivism. *Journal of Computers*, 2(3):43–55, 2007.

G.M Bodner. Constructivism: A theory of knowledge. *Journal of Chemical Education*, 63:873–878, 1986.

S Russ. Empirical modelling: the computer as a modelling medium. *The Computer Bulletin*, 39(2):20–22, 1997.

P. Watzlawick. *The Invented Reality: How Do We Know What We Believe We Know?: Contributions to Constructivism*. Norton, 1984. ISBN 9780393017311.