Challenges on Concretisation of Empirical Modelling: a preliminary analysis

Ilkka Jormanainen
Department of Computer Science
Universty of Joensuu
Finland
ilkka.jormanainen@cs.joensuu.fi

Draft of February 16, 2006

The idea of the study Concretisations of Empirical Modelling (EM) is to transfer the EM models that appear on the screen based, to the concrete objects, namely concretisation tools. This is an unique approach, which can be used, for example, when teaching physics, computer science or other sciences. With the approach of the concretisation of EM, it is possible to get rid of the fact that the models in EM approach take place only at the screen but not in the real and concrete world of a learner. This study presents a design for the architecture of the concretisation system. Some technical aspects for the implementation have also been presented. The results of this study indicates that conceptual and fundamental differences between EM and traditional usage of the educational robotics as concretisation tools cause difficulties, which need to be further discussed and resolved. Some solutions for these problems and directions for the future work have been discussed.

Introduction

The educational principles that motivate concretisations are rooted in Jean Piaget's theories of cognitive development (Miglino & Lund, 1999). Seymour Papert built on these theories in his notion of 'constructionist learning'. According to constructionist principles, the active learner is the centre of the learning process. Learners enlarge their knowledge by manipulating and constructing objects (Miglino & Lund, 1999).

Robotics or other concretisation tools can be used to engage the student with a topic to learn and, thereby, foster learning. This target of observations could be some physical phenomena or, for example, some particular algorithm in the field of computer science. A student can create concrete new knowledge and learn in constructionist way by interacting with real world objects (Ben-Ari, 1998). This interaction can also lead more to hands-on learning.

The most direct way for the concretisation is to transform standard algorithmic procedures into explicit 'realworld' routines to be executed by tangible physical devices. Robotics, for example, has been used widely in this way to teach traditional computer science concepts, such as programming, networking and artificial intelligence. A key idea is that robotics can help to motivate students to learn an abstract topic.

Besides fostering motivation, concretisations can also be used to contextualise abstract concepts, such as programming. For example, concrete learning artefacts have been developed to teach basic programming in developing countries. Most of the present ICT tools are based on a cultural norm that has put its primary educational emphasis on symbolism and logic. This does not necessarily correspond to the way of thinking in all cultures and it can not be readily supported using the available resources. This has been the motivation, for example, for teaching programming in the context of Tanzania by using intelligent building blocks, or I-Blocks. These blocks support learning by construction or, more specifically, "programming by building" (Lund & Vesisenaho, 2004).

An alternative approach to the goal of developing interactive artefacts to support experiential learning is offered by Empirical Modelling (EM) developed at the University of Warwick [1]. EM has so far made limited use of the new technologies that are prominent in research on concretisation. It is, however, more radical approach for supporting learning. The EM favours a stance on modelbuilding that gives experiential aspects priority over logic (Beynon, Harfield, & Jormanainen, 2005).

The approach of concretisations of EM and technical solutions behind it are descibed in this article. Target of this approach are concretisations with the contextual qualities of everyday artefacts that also afford rich learning experiences.

This article is organised as follows. Section 2 examines existing systems and tool in the fields of the EM and the concretisation. Section 3 draws outlines of the system architecture and discusses about the implementation of the system. Section 4 presents the results of this developement. Section 5 draws conclusions and discusses the future directions of this work.

[1] EM web pages: http://www.empiricalmodelling.com

Literature review and related work

Overview of the EM tools

The Empirical Modelling is an approach for constructing computer based models that can assist in the understanding of a phenomenon. The approach has an emphasis on experiment, observation and interaction during the developement process (Roe, 2003). Empirical Modelling describes the characteristics and features of a construal with three key concepts: observables, dependencies and agents. This description is made by using a set of definitions, definitive scripts, and in this way, representing state-as-experience. The approach has much similarities with a spreadsheet, where dependencies between cells are recorded by using definitions of the cells' content and their relations.

The most used tool for building models with the EM approach is EDEN intepreter, which exists in three variants intended for text-based interaction, line-drawing and window management, and distributed use.

Concretisation tools

In the few past decades, researchers and industries have developed a number of different robot kits designed to help learning in scientific fields such as mathematics, physics and computer science. These kits typically contain all the parts which are needed to construct a robot: motors, sensors, wheels, gearwheels and belts. Some of the kits (for example LEGO Dacta and LEGO CyberMaster) include cable or radio equipment that make it possible to connect the device to the computer. This allows the user to control the robot. Another approach uses autonomous robots. There is a small computer inside autonomous robots, so they can communicate and move independently according the program the user has constructed. These rather cheap robotic kits have been used, for example, for teaching Java programming to novices (Barnes, 2002). Artificial organisms have also been used for teaching the design and construction of industrial prototypes to engineers (Miglino & Lund, 1999).

One well-known example of these concretisation tools is Lego Mindstorms robot kit. The history of Lego Mindstorms goes back to 1986 when a research group supervised by Seymort Papert and Mitchel Resnick started to develop Programmable Brick, a small unit capable of connecting to the external world through a variety of sensors and actuators. The brick was designed for the creation of robots and other applications in which a computer might interact with everyday objects (Laverdae, 2001). With Lego Mindstorms kits, it is possible to construct an independent and autonomous robot. The Mindstorms kit includes a RCX unit (Robotic Command Explorer), which is the core of the kit. An advantage of the RCX unit is its flexibility. The unit can be programmed by using a variety of programming languages such as NQC, Java or Visual Basic. The Lego Mindstorms package also contains a programming environment, the RCX Code, which is a RCX specific programming language.

Concretisation with EM principles

Empirical Modelling has so far made limited use of the new technologies that are prominent in research on concretisation. EM is however more radical in another respect, in that it favours a stance on modelbuilding that gives experiential aspects priority over logic. In keeping mind with previous work on concretisation using robotics (Gonzalez, Myller, & Sutinen, 2004), a physical model of a well-known EM model called JUGS, have been constructed to examine the advantages of the software program as well as the benefits of real-world interaction. JUGS is a simple educational program that was originally developed for the BBC Microcomputer in the 1980s to familiarize children with elementary concepts of number theory.

An experiment for this concretisation is deeply described in Beynon et al. (2005). The experiment compares four different concretisations for the JUGS program: a real physical concretisation with real jugs, a softaware with BBC microcomputer, a robotics concretisation and as EM concretisation. The robotics concretisation was build with Lego Mindstorms robotic kit and programming was done with Microsoft Visual Basic (Beynon et al., 2005). The interface for interacting with the jugs build with robotics followed a similar design to the JUGS program, with operations for filling, pouring and emptying. However, no on-screen visualisation was necessary because the physical representation of the jugs enables the student to observe the content level.

The experiment indicates that the electronic counterpart of the engineering problem of manufacturing jugs presents software and hardware challenges that probably confuse student. This means that it is technically hard to build an entirely satisfactory concretisation of this nature in practice with existing tools (Beynon et al., 2005).

Lego robotics have clearly procedural behaviour when using programming languages such as Visual Basic or NQC (Baum, Baum, Gasperi, Hempel, & Villa, 2000). However, with some programming languages, like Java, it is possible to use event triggering and other object oriented programming fundamentals. Empirical Modelling basic concepts (observables, dependencies and agents) implement similar behaviour for EM models. Thus, it would be matter-of-course to select an object oriented programming language as a platform for implementing a concretisation environment with EM principles.

Description of the environment

In this chapter, we present an outline for the framework which allows the implementation of an application which can be used to build concretisations with the EM fundamentals. As a basis for the design, we use a framework originally designed for concretisation of sorting algorithms with LEGO Mindstorms robotics. The framework is described more deeply in Jormanainen (2004a, 2004b).

Overall architecture

The architecture of the framework contains three separate layers according to the original design (Figure 1). Each layer can contain pieces of software and hardware which take care of the responbilities of the layer. For each layer, there is an output which serves as an input for the next layer. Communication between the layers is bidirectional. This means that physical objects can communicate and send information about their states to the application. Figure 1 also presents the idea how this kind of framework should be suitable in different domain areas.

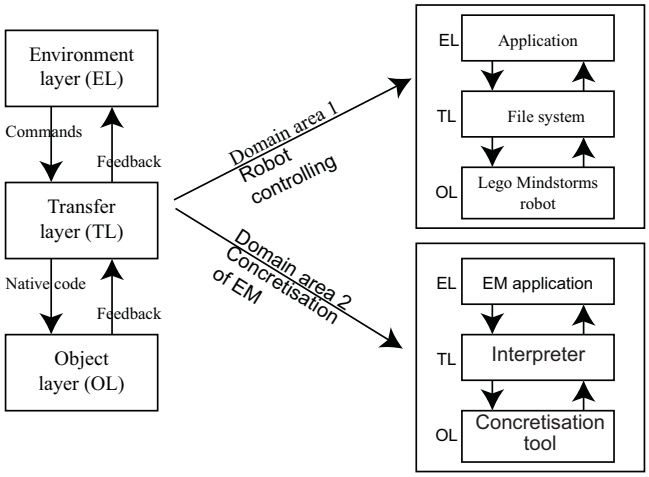

Figure 1. Architecture of the framework.

Layers of the architecture

In the general framework, an *environment layer* (EL) takes care of the communication between the user and the programming environment. Results of these actions are sent to a *transfer layer* (TL), which takes care of transferring and converting the code produced by the environment layer, to an *object layer* (OL). This layer represents physical objects, which can be for example LEGO Mindstorms robots or some other tools for concretisation. When using the framwork in the context of concretisation of EM, the following content of the layers can be identifed:

Environment layer: This layer contains an EM application which can be used to define the model, that is observables, dependencies and agents. The application is based on EDEN interpreter. With EDEN, it is possible to use different definitive notations, which give new possibilities for modelling. For example, *DoNaLD* and *Sasami* are definitive notations which allow usage of 2D and 3D graphics with EM models. In this work, we will define a definitive notation for controlling and defining behaviour of the concretisation tool.

Transfer layer: Due to the remarkable differences between the natures of EM models and procedural behaviour of the concretisation tools, for example the robotics, it is necessary to build an abstraction layer which works as a wrapper between definitive notations and procedural commands of the concretisation tool. In the context of the concretisation of EM, the transfer layer contains this wrapper. Technically, the wrapper is an intepreter, which turns the definitive notations to the procedural program code. The result of this layer is an *intermediate code*, for example Java, in the case that LeJOS environment is used with LEGO Mindstorms. This intermediate code is passed to the *native compiler*, which compiles the code to the native language of the concretisation platform, for example assembler or Java byte code. Beside these compilers, this layer may contain some devices for transferring native code to the concretisation platform. This device could, be for example, an infrared transmitter or an USB interface.

Object layer: According to Jormanainen (2004a), the objective of an object layer is to represent the physical objects which perform the actual concretisation. The layer takes the native, procedural program code as an input. An output should be a concretised model based on the EM principles, made by user. EM principles emphasis also concurrency: one model can contain several objects and these objects are capable to communicate with each others. By sending and receiving messages, objects can affect each others state (position, movement, activity). Therefore, it is an essential requirement that also concretisation platform has a possibility for communication between units (if several units are presented in the model under investigation). The object layer takes care of this communication procedure. Obviously, the user has defined this communication with the EM notation in the environment layer.

Each of these layers must take care of that its output is passed succesfully to the next layer (excluding the input to the environment layer which is done by the user, and the output from the object layer, which appears to the user). In the case of a failure, the layer must pass an error message (`ERR`) to the previous layer. Otherwise, the layer sends an acknowledge message (`ACK`) to the layer where it got its input. For example, if compilation of the EM definitions fails, the transfer layer must pass an `ERR` message to environment layer. If the compilation is succesfully, the transfer layer passes a `ACK` message to the environment layer. The environment layer, in turn, passes these messages to the user. Table 1 summarise input and output specifications for the layers.

Table 1
Summary of input and output specifications for the layers.

Layer	Input	Output
Environment	User input	EM definitions
Transfer	EM definitions	Native code
Object	Native code	Concretisation

Evaluation

When evaluating the structure of the layers presented in the previous section, keeping in mind the previous work in the field (Gonzalez et al. (2004) and Beynon et al. (2005)), two kind of challenges can be indicated with the actual implementation. The following technical challenges can be in-

dicated: a reliable communication between items of the concretisation platform, limitations of working in a physical environment, tracking the positions of the concretisation objects, and a large diversity of possible concretisation platforms.

Technical challenges

The concrete objects of the platform used for prototyping so far (LEGO Mindstorms robots) communicate with each other via an infrared transmitter/receiver. Therefore, there have been noticed some problems during the communication. For example, a message can get lost if robots can not ”see” each other (Gonzalez et al., 2004). Therefore, there is a need to develop a strong communication protocol for the inter-object communication in the object level.

In this approach of the modelling, the concretisation objects operate in a concrete world. This fact causes problems due the limitations of working in a physical environment. For example, the movement of the robot may vary on different kinds of surfaces. Also the voltage level of the power supply may cause different speeds between similar objects if they are tend to move. Light conditions may change radically during the execution of the concretisation and this may cause some strange results if light sensors have been used.

According to the EM fundamentals, objects in the model are aware of their state and they can also provide this information to other objects. However, when using physical objects, it might be hard to track the position of the object. Thus, the object may not be aware of its position related to each other. This may lead into situation where objects are not aware of their internal states in the model.

The final technical problem which can be indicated, is the diversity of the concretisation platforms. There are a large selection of applicable concretisation platforms, for example educational robotic sets, available. Basically, each of these platforms uses its own method for controlling the objects (programming approach and language, transfer methods etc.). When developing concretisation platform for EM, this issue must be taken into account. The framework itself provides a partial solution for this problem, since one fundamental of the framework presented in Jormanainen (2004a), is a requirement that each layer can be replaced easily with a new one. This means that interfaces between layers are general and layers are not aware about the internal functionality of the other layers.

Fundamental challenges

The implementation of the system contains also fundamental challenges. These challenges have been partially noticed also in the previous studies in the field: for example Beynon et al. (2005) argue that ”developing artefacts that can support the highly imaginative and speculative processes of tentative construction and reflection that underlie the constructivist ideal, the challenge is to make such development as accessible, natural and inviting as possible”. Current programming and controlling environment for the educational robotics are based on the procedural programming, and they necessarily do not meet this requirement. Therefore there would be a need for the environment which use higher abstraction level for controlling the construction.

Conclusions and Future work

In this article, we have presented an architecture for the concretisation environment based on Empirical Modelling fundamentals. The environment is based on the three-layer framework presented in Jormanainen (2004a). The evaluation based on the previous work in this field (for example Gonzalez et al. (2004) and Beynon et al. (2005)) indicates that several technical and conceptual problems must be overcome in order to achieve a meaningful result with an environment which would be ready to use in learning situations. These challenges include for example the following issues:

- A reliable communication protocol between the concretisation objects
- Reliable, independent movement of concretisation objects
- Conceptual difficulties to turn the EM models to the procedural programs needed by the concretisation objects
- A large diversity of concretisation platforms
- The fundamental differencies between the EM definitions and procedural programming languages

Some of these problems are possible to overcome with the technical choices when developing the system. For example, reliable communication protocol would be possible to implement with Bluetooth instead of IR. In this way, it would be possible to get rid of constraints of IR message passing. According to the web pages of the LEGO Group, there will be a new LEGO Mindstorms set available in August 2006. This set will contain some useful improvments, for example Bluetooth communication. Furthermore, the new Mindstorms set will include servo motors with rotation sensors, which could allow an accurate measurement of movement of the concretisation objects. LEGO Mindstorms platform has showed its strength as a prototyping platform so it would be a natural choice for the further developement.

However, this new platfrom does not solve fundamental and conceptual problems with the approach. These problems must be discussed within a scholarly community working in this field.

References

Barnes, D. J. (2002). Teaching introductory Java through LEGO MINDSTORMS models. In *Proceedings of the 33rd sigcse technical symposium on computer science education* (pp. 147–151). ACM Press.

Baum, D., Baum, D., Gasperi, M., Hempel, R., & Villa, L. (2000). *Extreme Mindstorms: an Advanced Guide to LEGO MINDSTORMS*. APress.

Ben-Ari, M. (1998). Constructivism in Computer Science Education. *SIGCSE Bulletin, 30* (1), 257–261.

Beynon, M., Harfield, A., & Jormanainen, I. (2005). Varieties of concretisation: an illustrative case study. In T. Salakoski (Ed.), *Kolin Kolistelut - Koli Calling 2005. The Fifth International Conference on Computer Science Education* (pp. 153–156). Turku Center for Computer Science (TUCS).

Gonzalez, J. L., Myller, N., & Sutinen, E. (2004). Sorting out sorting through concretization with robotics. In *Proceedings of the working conference on advanced visual interfaces* (pp. 377–380). ACM Press.

Jormanainen, I. (2004a). A Visual Approach for Concretizing Sorting Algorithms. In L. Malmi (Ed.), *Kolin Kolistelut - Koli calling 2004. Fourth Annual Finnish/Baltic Sea Conference on Computer Science Education* (pp. 141–145). Helsinki University of Technology, Department of Computer Science and Engineering.

Jormanainen, I. (2004b). *A Visual Interface for Concretizing Sorting Algorithms*. Master's thesis, Department of Computer Science, University of Joensuu, Finland.

Laverdae, D. (Ed.). (2001). *Programming Lego Mindstorms with Java*. Rockland, MA, USA: Syngress Publishing.

Lund, H. H., & Vesisenaho, M. (2004, September). I-Blocks for ICT Education Developement - Case Iringa Tanzania. In *Proceedings of 33th International Symposium on IGIP IEEE / ASEE* (pp. 364–371). Switzerland: University of Applied Science of Western Switzerland.

Miglino, O., & Lund, H. H. (1999). Robotics as an Educational Tool. *Journal of Interactive Learning Research, 10*(1), 25–47.

Roe, C. (2003). *Computers for Learning: An Empirical Modelling perspective*. Doctoral dissertation, Department of Computer Science, University of Warwick.