An investigation of Empirical Modelling of artefacts and its advantages over traditional modelling for real-world activities

Yu-Yang Chen
Computer Science, University of Warwick, Coventry CV4 7AL, UK

Abstract

The objective of this paper is to illustrate the characteristics of Empirical Modelling (EM) in real world situations, and make a comparison with a traditional modelling paradigm. In order to observe activities within the real world domain, the wiimote is used as the mediating device between models and physical experiences. One of the limitations in modelling is building an artefact so that it reflects the physical environment as closely as possible. While EM caters for a part of the problem by allowing users to modify the model based on their cognition, the accuracy and efficiency of such method is still unreliable, but we are still able to see a clear difference in the model limitation between EM artefacts and traditional modelling representation.

Introduction

"Empirical Modelling describes the characteristics of a construal with reference to three key concepts: ***observables, dependencies*** *and* ***agents****. A construal can be seen as embodying patterns of observables, dependencies and agency that are characteristics of the situation that is being construed"* [1].

By modelling with EM techniques, we may obtain an open-ended modelling environment. The underlying EDEN engine will take care of the dependences between observables within the EM model, while the model is being modified. To date, EM models are mostly based on modellers' construal, and placing the appropriate agency within the model performs actions, but this is seldom being tested along with the complexity of real world situation. To do so, a sensing device is needed in order for the EM model to comply with the real situation. Antony Harfield, formerly a PhD student in the Department of Computer Science, University of Warwick, has developed a tool which can connect the wiimote with EDEN for modelling activities.

Models

To make the comparison, two different models need to be built; one with a classical programming paradigm, and the other as an EM artefact. Both of them are enabled to take direct input from the wiimote, for real world domain observation. These models are illustrated below:

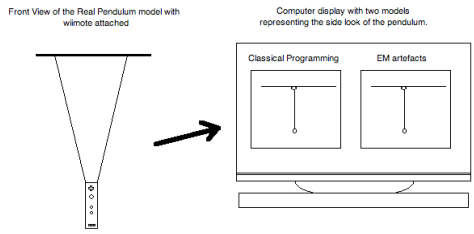

Figure.1: Illustration of the models

The real object being modelled is a simple pendulum. Both models are capable of representing movement of the pendulum on computer screen, which means both of them are capable of modelling the external environment, within the limiting set of constraints (e.g. rope length, weight of the wiimote, pivot position …etc).

To represent the complexity and uncertainty of real world domain, we need to make some state change that is outside the scope of the behavioural model pre-specified by an input-output transformation function. The simplest thing to do is to change the pendulum behaviour so it no longer fits the pre-specified input-output function; the step taken here is to interrupt the pendulum swing with a stick:

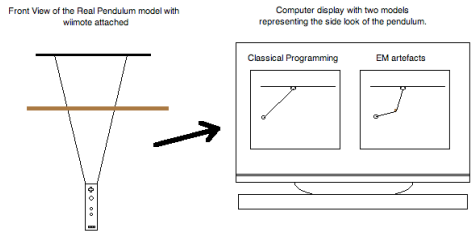

Figure 2: The difference between conventional programming and EM artefact

Neither model has any information about the newly added stick. The initial models are no longer appropriate for their purposes; with EM, as long as we understand the dependencies between the stick and all the pre-existing observables, we can add the stick as an observable into the model to enhance its behaviour; we may also need to add, or modify some reactive agent so the state monitoring and manipulating are performed in correct manner. With the Classical programming model, due to the pre-specified and fixed model behaviour, the only way to alter the model is to redefine the limitations on its scope so that the constraint set accounts for the stick object.

Analysis

In the paper "The interpretation of States: a New Foundation for Computation?" [2], Beynon and Russ mention the two factors of programming addressed by *B. C. Smith* :

1st factor: *"What must be directly realised in a physical substrate if a program is to do any work"*

2nd factor: *"What the symbols in the program mean, what they are about"*

The comparison between the ordinary and EM models can be seen analogous to the distinction between one-person programming and reactive systems. When we are constructing/programming the pendulum model using one-person programming, we assume that the physical substrate (i.e. the computer) and the underlying theory (e.g. SHM), are well understood and reliable. With the effect of physical constraints, mathematical specification methods can be applied for the prediction of the state change. This is done without much consideration of the external interpretation, and can be seen as an example of one-person programming, when the requirements are simple and involve less external reference. The programmer conceives the functional relationship between input and output states, and develops a program for the computer to perform the state transformation. This is reliable as long as the state machine contains a finite number of states (i.e. FSM). At this point, all the possible state changes within the scope of a simple pendulum swing are realised and handled reliably; but what if there are states outside the programmer's consideration?

"It is widely appreciated that focusing on paradigms for giving instructions to the computer has limited value – the real challenge in building large software systems lies in capturing the system requirements and arriving at the specification of the processes that are to be implemented in software." [3]

With conventional programming, all the possible states are considered to be preconceived at the start of the development, but this is not appropriate for reactive systems; a reactive system involves concurrent state change by different agents (such as the stick). Without a clear reference to the 2nd factor of programming, conventional computation only deals with the particular kind of state change where only the initial and final observations (input – output) are significant. For this reason, the interpretability of the program is poor. In EM, the user is able to observe the pendulum activity through the screen, without any knowledge of the internal state changes or theoretical base beyond the externally accessible observables (i.e. the graphical representation on the screen).

In [1], observation of external accessible states is considered to be the fundamental ingredient of computation. When exploring the possible state change in a reactive system, an understanding of its computational interpretations is crucial if the user intends to identify the dependencies between observables. In EM, an agent can either be represented by a human, physical substrate, or

computational process. Within an EM artefact, one way to change the internal state is by adding extra observables into the system and interacting with them. This is strongly related to the modeller's understanding of the system, as the addition of new observables also requires new dependencies to be added into the artefacts; in another words, if we want to add new observables into an EM artefact so that it represents the construal associated with our referent, we first need to investigate the relevant observables involved in the interaction.

Within the pendulum model, by experimenting with the artefact, the modeller can develop their construal of the system. (E.g. a changing the length of the string would also change the oscillation period). This experimental interaction is only available within the EM environment; conventional programming does not allow redefinition of the program parameters dynamically. A special-purpose notation called LSD can also represent the agent-oriented model of the system. This process can be described as identification of the agents. The EM artefact should be capturing the behavioural model of the real pendulum, which includes the dependencies and agencies. This is the key for the transition between EM and real domain; without a correct behavioural model, the EM domain will not be in direct correlation with the real domain.

The initial pendulum artefact has no information about the stick and its interaction profile, but we are free to add extra observables and dependencies into the system through the definitive script invocation. The agency and dependency related to the stick can also be added definitively, providing another degree of freedom in software engineering. The observing agents can make changes to the observables dynamically, so that the externally accessible state-change can directly affect the agents' construal of the system, hence referent of the domain.

Conclusion

The conventional programming model has some shortcomings when designing a reactive system. Empirical Modelling could be providing a possible perspective for more suitable solution. In this pendulum experiment, we have demonstrated the limitation of the conventional computation, and how it can be improved through the paradigm of modelling with definitive script. From the EM perspective, the emphasis is on *making sense* of the computation, and *thinking with the Computer*. This emphasis is most appropriate when a more complex programming model and enriched interaction between agents are dominant influences on the computation. By leading the pendulum model to an unexpected state, we simulate the inconsistency in the real-world domain, and illustrate how sense making activities help human agents in constructing the EM observables. The wiimote did serve as a reliable observing/agency device in this experiment, but the result from the accelerometer is inconsistent due to its irregular shape. This can be improved with the method mentioned in [4].

Reference

[1] W.M.Beynon and Chris Roe *"Enriching Computer Support for Constructionism"* 2006

[2] W.M.Beynon and S.B.Russ. *"The Interpretation of States: a New Foundation for Computation?"* 1992.

[3] W M Beynon, R C Boyatt, and S B Russ *"Rethinking Programming"* 2006

[4] Maurizio Vannoni and Samuele Straulino *"Low-cost accelerometers for physics experiments"* 2007