Concurrent Systems Modelling: Interactive Situation Models

0207146

Abstract

This paper will critique Empirical Modelling by comparing its benefits against traditional methods of developing what EM calls Interactive Situations Models; that is models which describe the operation of an artefact and embody its behaviour. Models that can be called ISMs are often used as specifications for real-world artefacts that are embedded into the functionality of a device – for example the behaviours of a digital watch or car locking system. There is a previous body of work within the Empirical Modelling library that has attempted to use tkEden to build and refine such behaviours. By taking these examples, as well as developing my own, this paper shows that Empirical Modelling's suitability to this area of development allows a designer to constantly rework his model, interactively test it, and by doing so capture the semantics of his artefact.

1 Introduction and Background

Since the age of the computer, man has been attempting to observe the real world and model it virtually; whether to further scientific understanding, carry out experiments deemed unsafe for real life, or to provide the user with a virtual existence that bears some similarity to their own into which to escape to. Indeed many examples can be found of these: the modelling of tumour growth for cancer treatment(1), the prediction of the number and types of staff required to effectively respond to major disease or bio-terrorism attack on a given population(2), or the Massively Multiplayer Online Game genre(3) where players interact under the conventional rules of physics and society contained within a giant persistent world.

Such models are attempts by their designers to observe the real world, and via a process of revision and experiment refine the model until the observable behaviour of the model matches the observable behaviour of reality. This process is itself laden with both practical and philosophical subtleties with make the process somewhat less straight-forward than one would expect initially; the seventeenth-century French philosopher Rene Descartes' musings questioning the fallibility of perceptions(4) certainly highlights the problem of observation: if a designer is to observe a real-world behaviour it is only he who can understand whether his model is a correct construal, and could be completely inaccurate in the eyes of another who interpreted the real-world observable differently. It may be the case that the interpretation of a model's behaviour could itself be interpreted differently, and that even the designer cannot be deemed infallible in evaluating its fit. Whilst it may seem futile to the instrumentalist engineer to pursue such philosophical distractions, the fact remains that the interpretation of observation is quite critical – many examples of failed software engineering projects exist, the cause of which being the developers and the customers both sharing the same observable scenario, yet interpreting it differently; upon delivery, it becomes painfully apparent that this 'semantic misalignment' has produced a solution to a problem that does not exist within the customer's problem domain.

It is this 'semantic misalignment' problem which now requires addressing; Computer Science has progressed rapidly in the fields of building software, but is still a potential victim to the *"that's great, but it's not what we wanted"* appraisal pitfall. Indeed it can often be the case that a subset of positive test-cases can lead a model's designer to *believe* his model is accurate, when it later transpires it is not, and thus is a victim of his own 'semantic misalignment'.

2 Conquering the 'semantic misalignment'

It is important to limit ourselves to an acceptable level of reality at this juncture. The teachings of Empirical Modelling often refer to what is known as 'the frame problem': that is, at what point do we factor out the huge permutations of reality from our model? When IBM designed their Large Scale Integration circuits, they did not consider that the unshielded bombardment of cosmic arrays upon the circuitry would cause them to malfunction in high-altitude applications(5). Had they extended their abstraction of reality to include the effects of space radiation, the metaphorical frame bounding their snapshot of reality would have catered for such phenomena; it was only during the implementation of the model that the discrepancy between the real-world and the model became apparent.

In the case of classical behavioural modelling, the frame problem is often shrink-wrapped around the model: no allowance for any deviation in stimuli is catered for, and anything outside the bounds of specified functionality remains undefined. It is this explicit exploration of a model's frame that is unique to the current study of Empirical Modelling, and is the area where classic attempts to model behaviour have quickly splashed in before quietly retreating and leaving the undefined resolution to the tacit understandings of the designer to decide.

In the following sub-sections, methods that have attempted to conquer the problems of frame and understanding are presented. The final subsection will apply such issues to tkEden, a tool developed by the University of Warwick's Empirical Modelling team at the Department of Computer Science, to demonstrate the advantages and disadvantages this relatively newly defined methodology brings to the problem.

2.1 Cognitive modelling

Imagine a real-world artefact such as an aeroplane autopilot, and a person unfamiliar with its operation. This person is attempting to learn how the autopilot functions in a safe environment before being allowed to interact with it in reality, as the ramifications of failing to operate the artefact correctly could result in an outcome so undesirable as to warrant factoring out any possibility of its occurrence. This process of learning how an artefact functions is often referred to by Human Computer Interaction advocates as 'building a mental model'(7). By using what are known as 'positive-learning techniques'[1], in that the exploration of the artefact conveys the way in which it operates, one can build up an understanding of the artefact that is an accurate portrayal of its function.

Imagine that the learner's 'mental model' could be somehow extracted from their mind and represented in a standard manner. It is not inconceivable that if this were possible, it would prove an interesting test of this model's semantic capture of an artefact to compare it against the actual model of the artefact, as the latter would have been represented in a standard manner during development. The inputs and outputs of each model could be compared to ensure that the user's understanding was indeed the correct interpretation of the artefact and that no 'semantic misalignment' existed; indeed if the underlying model contained states, every aspect of the phases of execution could be compared – which would ensure that the frame of the user and the frame of the artefact matched. It may seem more like a concept from Science Fiction, but can indeed be done: if a user themselves models their understanding then it can be tested. Descartes' issues aside, if the user can accurately express their understanding in a model there is no barrier to comparing the two, and by doing so highlight any 'semantic misalignment' and make the process of exploring the frame of the model redundant.

John Rushby did exactly this in his paper 'Using model checking to help discover mode confusions and other automation surprises'(8).

> *'Complex systems are often structured into "modes" (for example, an aircraft flight management system might have different modes for cruise, initial descent, landing, and so on), and their behaviour can change significantly across different modes. "Mode confusion" arises when the system is in a different mode than that assumed by its operator; this is a rich source of automation surprises, since the operator may interact with the system according to a mental model that is inappropriate for its actual mode.'*

Rushby specified these models using a language known as Murϕ (pronounced "Murphy"), which uses some specified *state variables* and a series of *rules* that define the movement of these states dependant

[1]An illustration of positive and negative learning: As a schoolchild, and one ungifted in Chemistry, I was placed in the lower set to learn the basics whilst the elite chemists of my year presumably pursued academic enlightenment. Whence the topic of electrolysis came up for study, the complex explanations of atomic interaction were presented as *"On the diagram, the flow of electricity always goes from left to right"*. When I questioned about the flow of electricity if the apparatus was observed from the other side of the desk, I was reassured that every single GCSE examination follows this trend, and that the true explanation was both above my understanding and irrelevant in obtaining the answer. Whilst my teacher was correct that all papers conformed to this model, it was a fine example of negative-learning: the way in which I was understanding electrolysis matched how it worked in the examination scenarios, but presented a useless ground for further understanding of anything related to electrolysis – more so than if I knew nothing of it as I would have to unlearn my understanding (as hard as unlearning to ride a bike). Had my teacher explained the involvement of the cations and anions then I would have received positive-learning as my means of deriving the outcome was through my correct understanding of the phenomena and thus could be built on in future if I were to, for example, become a manufacturer of hydrogen. I got an A.

upon the firing of the rules. Murϕ explores these states using the declarations to find all reachable states (in his case *behaviours*) and linking the two models together with an invariant that, if both models exhibited the same *behaviour* at the same time, would return *true*. By observing when this invariant became *false* the points where the mental model and that of the artefact differed would highlight problem areas that the developers of the artefact should address. By putting this into practice, Rushby used his solution to highlight the exact cause of an altitude deviation observed during a NASA study when twenty-two airline crews flew realistic two hours missions inside DC-9 and MD-88 aircraft simulators.

The method of asking the user to model the artefact and then comparing it to the actual one is a guaranteed way of discovering whether the user has considered all the influences acting on an artefact and defining its frame. Clearly this is limited to situation where the artefact's model is available and machine-parsable; outside of these man-made artefacts the frame problem is still very much alive. How could one model a situation, for example, when there was no designer overlooking it (spirituality and persuasion aside) or record of the creation available? The things most interesting in modelling tend to be things we do not fully understand.

2.2 Formal Methods

Rushby's solution could not escape the confines of man-made artefacts, or ones that were so completely understood by man to the point of being codified. His capture of observations falls under the category of Formal Methods – ways in which designers attempt to abstract away from the world the construal will exist in and concentrate only in defining precisely and with no ambiguity the operations, percepts and outputs of models. Tools such as Z and B(6) allow a modeller to sit down and specify these, and by doing so obtain a mathematical proof that the inner workings are correct.

Whilst such validations of operation are essential, it is the application of the model that may fail due to the affliction of the frame problem. Consider a simple Entry/Exit schema that keeps track of who has entered a building, whether they are employes and how many people in total there are. Whilst a mathematical proof may indeed confirm that the system works within its own frame of existence, it cannot be expected to keep track if someone decides to leave via a window, or blowing a hole in the wall. That might seem farcical, but so would be the suggestion that cosmic radiation is interfering with your network connection. And what if the Entry/Exit system was implemented in a prison?

Formal methods presents us with a very neat way of coping inside a construal's frame. Stretching it however may often require an entire rethink of the construal's structure, and could possibly introduce non-determisinsm to the point of working with the Universal Set – that is, the set of absolutely everything ever, past present and future, and possibly more. The definition of an operation's preconditions could be an infinite task.

Formal Methods is like a paper aeroplane made of crisp, sharp edges and brilliant white paper, each fold defining the placement of the next and weakened if corrected, that when flown in the classroom sails across the stationary air and belies a story of the simple elegance of man-made flight. Only when it is taken outside and flown in the wind and rain does it concede its inability to model man-made flight as only a sodden, flaccid and translucent piece of paper can when so far outside the frame in which it was designed. What is needed instead is a light-weight malleable material with which to work, to knead and to shape, that can be felt and experienced, smoothed and reworked by hand until it is correct.

2.3 Empirical Modelling

To best demonstrate this, I will introduce a car locking and alarm model:

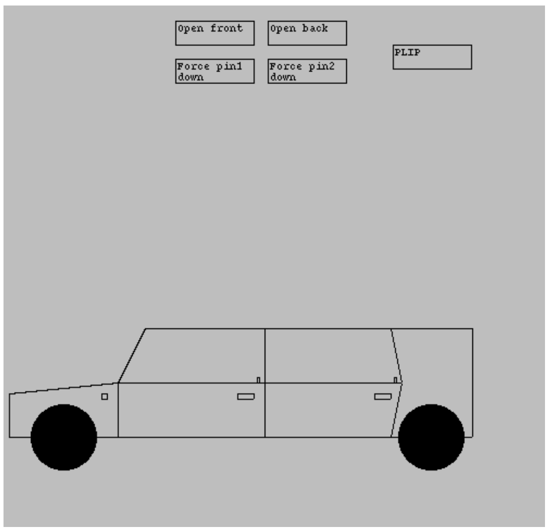

Figure 1: tkEden Car lock and alarm system

I encourage you to play around with it. Try locking and unlocking it, try opening a door if the locking pins are down, try sitting inside it when locked and pulling up the locks – you'll set the alarm off. Try unlocking it, getting in and locking the doors by hand. See if you can break into it without setting the alarm off, or finding a way to silence it. The point of this is that it *feels* like a locking system.

2.3.1 How does this help us in relation to the frame problem?

This model was inspired by the time I went to my car and attempted to unlock it using the keyfob. It looked to unlock, but in fact the passenger side door had not unlocked, and as I opened the driver's door the alarm went off. It transpired that this was because if all the doors did not unlock, the car would consider itself still locked, even though some of the doors had succeeded in unlocking. Now this seems like a silly design flaw, but we shall think a little deeper about it.

Consider a car, with two doors and a central locking system. One of the doors is locked, the other open. The careless owner has left it all winter without a drive, and the battery has died. When the battery is replaced, the car exists where before it was just an object; now it has state and knows itself to have a locking system and, for all it knows, for the first time ever. It then hears the keyfob 'plip' asking it to lock or unlock – what should it do? One door is locked, maybe the other should follow suit? Maybe it should unlock the other one, but what if this door was forced to be unlocked and it was someone attempting to break in to the other side? Without having a state to start from, it is hard to know what state to go to. If you try this in the tkEden script, you shall see that it always defaults to unlock and disable the alarm; $broken1$ and $broken2$ take an $\%eden$ value of 1 or 0 to indicate if the front or back door has a broken lock motor or not. It turns out that my real car knew it was last told to lock, so did an unlock but left the alarm on as the owner could then get in it and lock it (thus silencing the alarm) and still drive the vehicle, whereas a would-be thief would not be able to unless they had the driver's key also, and would have simply locked themselves inside someone else's car[2]. The designer of a similar system may not have defined such defaults, and it is by his interactions with his model that allow him to experience and discover these issues and rectify them if necessary; it would be far too late to discover these mode confusions(8) if the first opportunity to experience the model was after implementation.

[2]The behaviours of locking systems and alarms vary between manufacturers.

I was still left with a seized locked door to fix. I set aside the weekend with a replacement motor to fit, but this could not be done in situ. The door had to come off. Try removing the door in tkEden with:

%eden $door1Remove = 1;$

Notice that the alarm does not go off; should it? The door is still locked as is the rest of the car – the original frame of the problem never considered that the door might actually be physically removed. The key point here is that the alarm sounding is a dependancy of the alarm being armed (which relies on the car being locked) and that at least one of the locking pins indicates the door is open. In order to cope with the door being removed we have to break this dependancy, and not derive the status of the car's security from the door pins alone.

Try redefining the alarm to include a noise sensor:

$alarmSounder\, is\, (alarm + pin1Up + pin2Up + noise);$

We can push the frame even further by making the operation of the plip dependant on the power contained in its battery; if the keyfob cannot send a plip then the entire system will fail to operate regardless. We consider the keyfob to be an 'agent' in the system and now extend that to be dependant upon another agent, the battery. This can be though of in LSD notation(9)(10) as:

```
agent keyfob() {
        oracle batteryPower;
        state (bool) sendingPlip, pressed;
        protocol pressed and batteryPower > .5 −> sendingPlip=true;
}
```

This will allow the keyfob to see the status of the battery, and will then be constrained to operate if and only if the battery is observably sufficient.

We might find that the frame is still too restrictive for practical purposes. We might find that the keyfob button seems a little stiff and a press does not entail sending a plip; again we can introduce a new oracle to a finger agent that can provide a certain force to the keyfob, which can then use this value to determine whether the button is pressed or not. The LSD notation of the finished keyfob agent is written in eden as:

```
proc doPlip {
        if (battery > .5) {
                if (finger > 4) {
                        writeln(”The user is sending a plip”);
                        plip = 1;
                        eager();
                        plip = 0;
                }
                else {
                        writeln(”Button requires 5+ netwons of force”);
                }
        }
        else {
                writeln("battery less than half a volt");
        }
}
```

You can interact with the additional constraints of the keyfob by altering the values of $battery$ and $finger$ in $\%eden$.

3 Conclusion

This paper has shown how Empirical Modelling allows a designer to shape the meaning and function of his model whilst he is building it. By doing so, he is confirming that the semantic meaning of the construal matches the artefact and that any semantic misalignment has been factored out during shaping. In contrast, using Formal Methods limits any tangential exploration of the frame, as was done introducing the battery and then finger pressure to the behaviour of the plipper, by allowing self-contained dependancy modifications which leaves the rest of a construal unchanged.

The paper has introduced some of the philosophical issues that surround the process of observation and interpretation, and that one can never concisely define what an construal's frame may be as each person's interpretation of reality can not be confirmed or denied by anyone; an argument that Descartes' took to the extreme by questioning whether everything is just a dream in the mind of the reader.

Methodologies that can cope with conforming to a certain frame, such as Formal Methods, are often inflexible in their ability to adjust the bounds of the frame when required, and that the model itself is hard to experience and validate outside of applying mathematical proofs.

Possibly the most important concept to highlight is that the Empirical Modelling process has itself confirmed the operation of the construal to within the scope of its frame, and that this frame could easy slide and increase to incorporate the semantic meanings required of it. Not only have the modeller's interpretations of the observation of the artefact been interpreted into a construal, but that the modeller has also potentially gained a deeper understanding of the artefact by trying to emulate it; using EM as a learning tool is certainly an active area of current research for this reason.

References

[1] ScienceDaily.com, 2006. 'Tumor Growth Computer Model Sets Stage For Customized Cancer Treatment'. Available at <http://www.sciencedaily.com/releases/2006/12/061201180429.h
Accessed 20th December 2006.

[2] United States Department of Health and Human Services, 2005. 'Computer Staffing Model for Bioterrorism Response'. Available from <http://www.ahrq.gov/research/biomodel.htm>. Accessed 20th December 2006.

[3] Wikipedia.com, 2006. 'Massively multiplayer online game'. Available at <http://en.wikipedia.org/wiki/Massively_multiplayer_online_gam
Accessed 20th December 2006.

[4] Cress, D., 1999. *Discourse on Method and Meditations on First Philosophy*. 4th Ed. Indianaplois: Hackett Publishing Co Inc .

[5] Zeigler, J.F., 1994. IBM Journal of Research and Development, *'Terrestrial cosmic rays and soft errors'*, Vol 40, No. 1, 1998. Available at <http://www.research.ibm.com/journal/rd/401/curtis.html>. Accessed 20th December 2006.

[6] The World Wide Web Virtual Library, 2005. 'The Z notation'. Available at <http://www.zuser.org/z/>. Accessed 20th December 2006.

[7] Dove, L., 1999. 'Mental Models and Usability'. Available at <http://www.lauradove.info/reports/mental

[8] Rushby, J., 1999. Reliability Engineering and System Safety, *'Using model checking to help discover mode confusions and other automation surprises'*, Vol. 75, No. 2, pp. 167-177, 2002. Available at <http://citeseer.ist.psu.edu/507284.html>. Accessed 20th December 2006.

[9] Beynon, M., 1986. 'The LSD Notation for Communicating Systems'. Available from Department of Computer Science, University of Warwick, England.

[10] Beynon, M. and Norris, M. and Slade, M, 1988. 'Definitions for Modelling and Simulating Concurrent Systems'. Available from Department of Computer Science, University of Warwick, England.