Implementing Empirical Modelling principles for evolving software

Meurig Beynon, Nicolas Pope and Steve Russ

Software evolution has three interrelated aspects. The roles of the human agents, the computing environment, and the context for the interaction and interpretation of software are all prone to evolve. Empirical Modelling (EM) [1] is developing new concepts and tools that promise to address all three aspects of software evolution in a coherent way, and to blend development with evolution in a conceptually seamless manner.

EM places its emphasis on the fundamental role of experience in constructing knowledge, in keeping with the practical principles illustrated in model-building with spreadsheets. Prototype tools to tackle each of the three aspects of software evolution have been applied in a wide range of projects [2]. Our current research is integrating these prototypes to do fuller justice to what is now a mature conceptual framework for studying software evolution [3:#114] respectively underpinned in its three aspects by William James's radical empiricism [7], Gooding's notion of construal [5] and Latour's constructivist ideal [8].

Where some approaches explore how far software evolution can be automated, EM emphasises the essential need for human comprehension of software in relation to its context. This accords with Naur's view (as cited in [3:#105]) of the key role of intuition in software development, and of the importance of 'theory' in the sense introduced by the Ryle that equips us "[to know] how to do certain things and in addition can support the actual doing with explanations, justifications, and answers to queries, about the activity of concern". EM constructs interactive artefacts ("construals") that mediate such knowledge experientially, exploiting the capacity of computer-based technologies to generate rich experiences.

The key concepts of EM are observables, dependencies and agency. In EM, the apprehension of particular configurations of observables, dependencies and agency is seen as characteristic of intelligent human interaction within an environment. In broad terms: an *observable* is an element of our experience to which an identity and current status can be ascribed; a *dependency* is a perceived causal link between a change made to one observable and a contingent change that then occurs to another; an *agent* is any collection of observables that can be viewed as a corporate entity to which state change can be attributed.

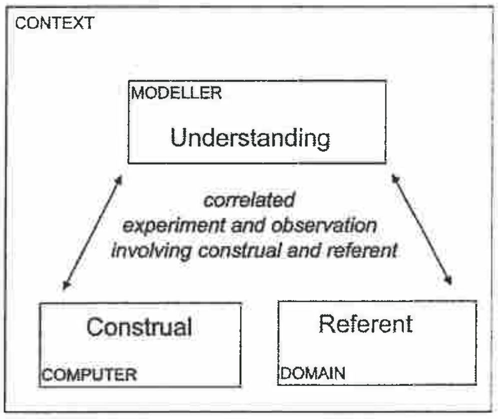

CONTEXT
MODELLER
Understanding
correlated experiment and observation involving construal and referent
Construal
COMPUTER
Referent
DOMAIN

Figure 1: Making construals in Empirical Modelling

Making a construal involves personal crafting of a correspondence between interaction with the computer artefact and interaction with its referent that is ideally so intimate as to be immediately apprehended by the developer.

In EM, every aspect of software development and evolution is interpreted with reference to observables, dependencies and agency. The activity depicted in Figure 1 is adapted so as to capture the perspective of any state-changing agent acting in the software environment, whether a human agent such as a user or developer, or an automated agent such as a computer or device. This involves projecting the MODELLER perspective on to the interface between any such agent and its environment, and characterising its interactions with reference to its 'observables' and 'dependencies'.

In the early stages of development, especially in radical design [6], EM takes the form of concurrent systems modelling to identify the relevant human and automated agents and the ways in which their interaction is mediated

[3:#087]. To this end, the observables associated with an agent are classified into those that are directly apprehended ('oracles'), those conditionally under the control of the agent ('handles') and those defined by a dependency ('derivates'). The developer devises construals of the interactive context surrounding the individual agents that can then be integrated into a systemic view where the MODELLER in Figure 1 interacts as an objective external observer. Experiment and observation play a fundamental role throughout, and the ongoing crafting of observables, dependency and agency drives both the open-ended initial development and subsequent evolution.

EM principles and tools for concurrent systems modelling were first developed in collaboration with British Telecom [3:#017]. Managing the evolving human perspective, taking account of learning and adaptation, has been extensively studied using a special-purpose interpreter (EDEN) for formulating and maintaining networks of dependencies ("definitive scripts"). Software development using definitive scripts was initiated with support from IBM [9] and later applied by Cartwright in resolving challenging problems of digital media portability at the BBC [4]. Modelling with scripts demonstrates rich scope for evolving software on a small canvas (cf. Keer's simulation of a navigating cataglyphis ant [3:#110]) but does not address the essential need to make a realistic model of the computing technology itself - the processes, networking, and management of processors and peripheral devices - and to scale up using object-oriented principles.

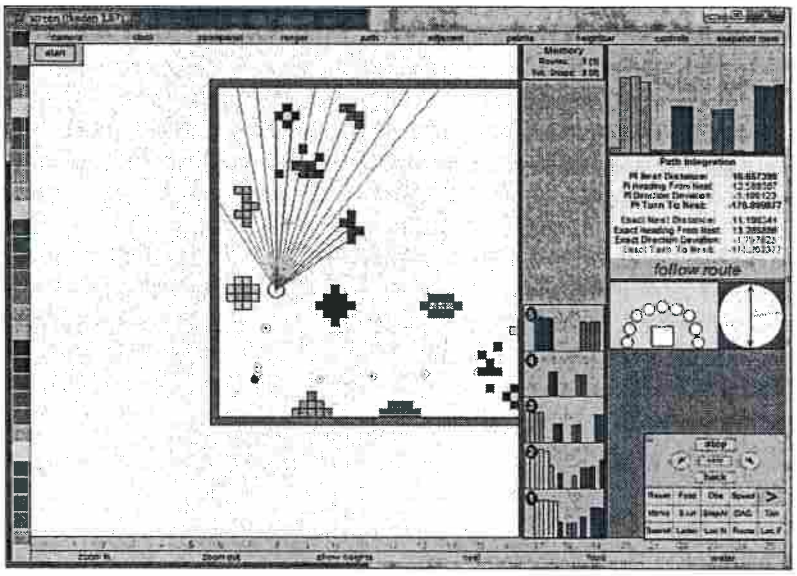
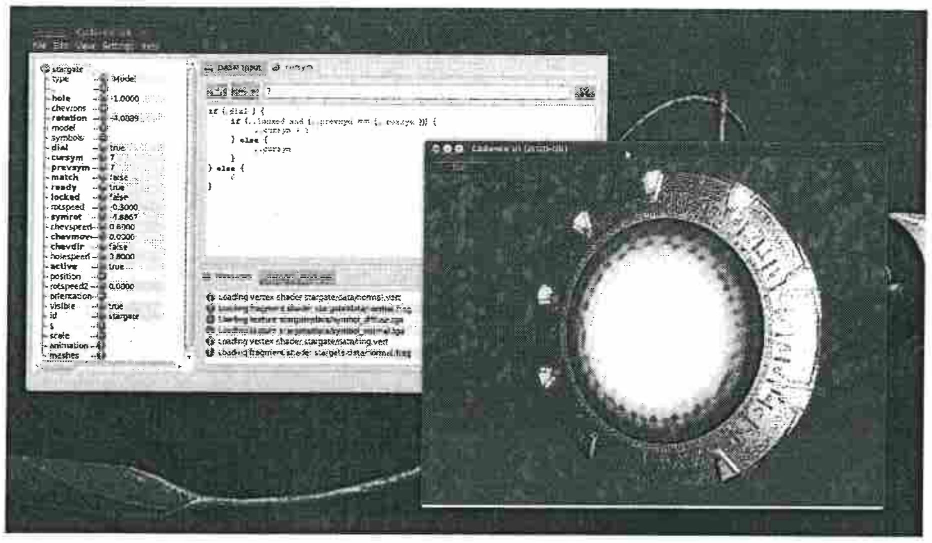


Figure 2: Keer's EDEN construal of ant navigation

These limitations have been addressed in a new tool (CADENCE) recently developed by the second author [3:#113] that promises to enable a full integration of all three aspects of software evolution consistent with the overall vision of EM. CADENCE supports process-like observables that can be directly connected to devices, both analogue and digital, and linked by dependency. In this way, it has been used in prototype applications such as on-the-fly computer game design, a video wall implemented on a multiprocessor architecture, a music synthesiser, and shared browsers over a distributed network. Future work will be directed at exploiting CADENCE to re-engineer and integrate the prototypes that have been developed to support EM for evolving software, which include the Abstract Definitive Machine [3:#010], EDEN and its distributed and web-enabled variants [3:#053,10,11].



```
stargate
- type      Model
-
- hole      1.0000
- chevrons
- rotation  -3.0889
- model
- symbols
- dial      true
- cursym    7
- prevsym   7
- match     false
- ready     true
- locked    false
- rotspeed  -0.3000
- symrot    -1.8667
- chevspeed 0.8000
- chevmove  0.0000
- chevdir   false
- holespeed 3.8000
- active    true
- position
- rotspeed2 0.0000
- orientation
- visible   true
- id        stargate
- $
- scale
- animation
- meshes
```

```
if (.dial) {
    if (..locked and (..prevsym == (..cursym))) {
        ..cursym + 1
    } else {
        ..cursym
    }
} else {
    0
}
```

```
Loading vertex shader stargate/data/normal.vert
[illegible]
[illegible]
[illegible]
Loading vertex shader stargate/data/ring.vert
Loading fragment shader stargate/data/normal.frag
```

Figure 3: Pope's Stargate construal in CADENCE

Links:

1. The Empirical Modelling website: http://www.dcs.warwick.ac.uk/modelling
2. The Empirical Modelling archive: http://empublic.dcs.warwick.ac.uk/projects
3. Empirical Modelling papers: as indexed at http://www2.warwick.ac.uk/fac/sci/dcs/research/em/publications /papers/
4. Richard Cartwright et al on Portable Content Format: http://www.dvb.org/documents/newsletters /DVB-SCENE%20Issue%2014%20Final.pdf
5. David Gooding on construals: http://www2.warwick.ac.uk/fac/sci/dcs/research/em/thinkcomp07 /gooding2.pdf
6. Michael Jackson on radical design: http://mcs.open.ac.uk/mj665/PFrame10.pdf
7. William James on radical empiricism: http://www.brocku.ca/MeadProject/James/James_1912 /James_1912_toc.html
8. Bruno Latour on constructivism: http://www.bruno-latour.fr/articles/article/87-CONSTRUCTIVISM.pdf
9. Paul Ness on 'Creative Software Development': http://www2.warwick.ac.uk/fac/sci/dcs/research /em/publications/phd/pness/
10. The Web EDEN interpreter: go.warwick.ac.uk/webeden
11. The JsEden interpreter: http://www.dcs.warwick.ac.uk/~empublic/js-eden