More about Cadence for EM

Going beyond classical programming ….

Characteristics of tools to be introduced in the module ...
they are concerned with modelling in which we

- *observe meaningful things*
- *adopt a **constructivist** stance*
- *exploit an **empirical** approach*

that we wish to reconcile / can be reconciled with the more abstract, rationalist, theoretical framework that characterises classical computer science

Reconceptualise by **introducing the human dimension** ... key shift in emphasis towards questions such as:

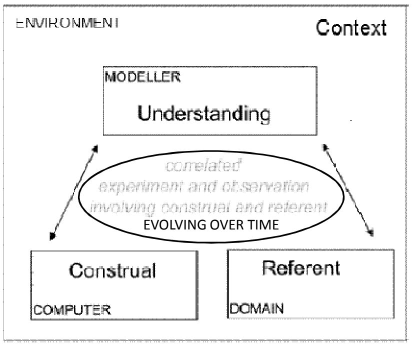
*? what is the **experience** of the people engaging with Turing computation, procedural programs, functional programs etc.*

Consider people's experience ('programmers', 'users', 'modellers' or 'analysts' etc.) with reference to

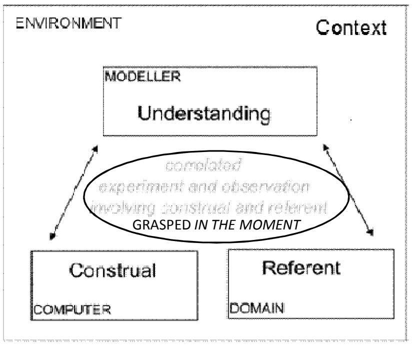
* What are the significant things that they observe?
* How are they able to interact and manipulate?
* What is the context for their interaction and interpretation?

when they are engaged in some variety of programming / model-building activity.

EM as *an experience of construction*



EM as *construction of an experience*



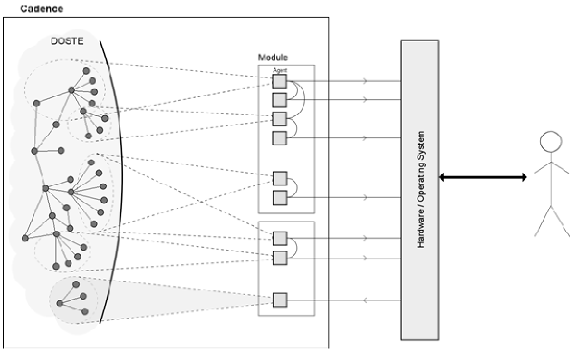
The word **experience** is a key word in Empirical Modelling, and is being used in a very distinctive way e.g. the financial modeller

experiences the current financial situation
“Phwor! I might be about to make a lot of £££s here …”

experiences the numerical patterns in the cells of the spreadsheet
“ … that number looks a bit big for that cell”

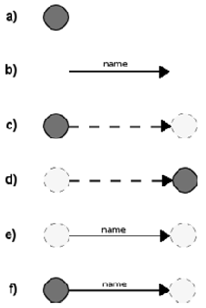
experiences the current context
“… now I can buy my boyfriend a Ferrari … “

experiences the connection between all these experiences
“figures on screen / money in world / boyfriend ecstatic”



The CADENCE environment and the DOSTE engine

Terminology for describing a DOSTE structure



a) Node

b) Edge (directed and labelled)

c) Context, or in cases where there is a more concrete interpretation it may be called an object.

d) Value

e) Property (or quality)

f) Observable (requires a context and property). If the context is interpreted as an object then this may be called an attribute.

Aspects of Stargate featured in the Cadence model

- components - chevrons / symbols
- attributes – where chevron is, whether lit up
- behaviours – process-like observables that are changing
- visual characteristics – geometry and textures
- states of the Stargate – with or without the puddle
- viewpoint perspective – e.g. from moving spaceship?
  with smoke in the cabin?
- orientation of the Stargate
- status of human agent – Cadence model-builder vs gamer
- context for the human agency
  - is the Stargate in motion?
  - are we trying to correct a flaw in the visualisation?
  - are we assessing the model aesthetically or technically?

All state and every kind of manipulation of state in the Stargate model is expressed by framing definitions

- assigning a value to an observable (=)
- establishing a dependency relation (is)
- introducing a process-like observable (:=)

These account for meaningful interaction in whatever observational context, whether directly experienced by a human agent or projected on to an automated agent

Contrast with traditional software development, where **requirements**, **design**, **implementation**, **testing** distinct

Contrast with traditional software development, where ***requirements, design, implementation, testing*** *distinct …*

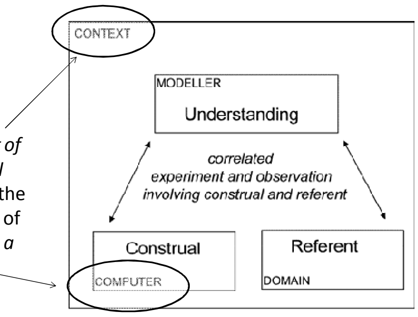
In Cadence, all these activities can be supported within one and the same environment …

The guiding principle for the development is at all times **matching** *what is as-of-now observed in the referent* **to** *what is as-of-now embodied in the construal*

This gives uniformity and consistency to the experience of the modeller quite unlike traditional programming as classically conceived …

Empirical Modelling as *Construction*

Cadence has exceptional qualities in respect of *management of observational contexts* and the development of construals *on a computer*



Many interpretations of construal …

Compare and contrast …

- making sense of Stargate
- creating a radically new software system
- making a financial spreadsheet
- devising a walk
- discovering electromagnetism

All involve engaging with the world and making some form of construal of a referent

Specialising the context for construal …

Simplifying assumptions, such as in spreadsheets:

- context not too rich (cf. sheets in ss)
- autonomous behaviour is limited
  (cf. ss doesn't change by itself)
- perceived structure is primitive / embryonic
  (cf. Faraday's experimental world)
- modeller not an expert in software development
  (cf. ss interfaces for input / visualisation)
- modeller has domain-specific expertise
  (cf. ss functions average/sum and pie charts)

… basis for older EM tools more limited than Cadence

The EDEN tool …

Simplifying assumptions, such as in spreadsheets:

- context not too rich (» *use scripts of definitions*)
- autonomous behaviour is limited
  (» *only use 'is' and '=' definitions*)
- perceived structure is primitive / embryonic
  (» *have a flat space of observables*)
- modeller not an expert in software development
  (» *modeller frames defns / programs functions*)
- modeller has domain-specific expertise
  (» *supply familiar types, operators, depictions*)

… *basis for older EM tools more limited than Cadence*

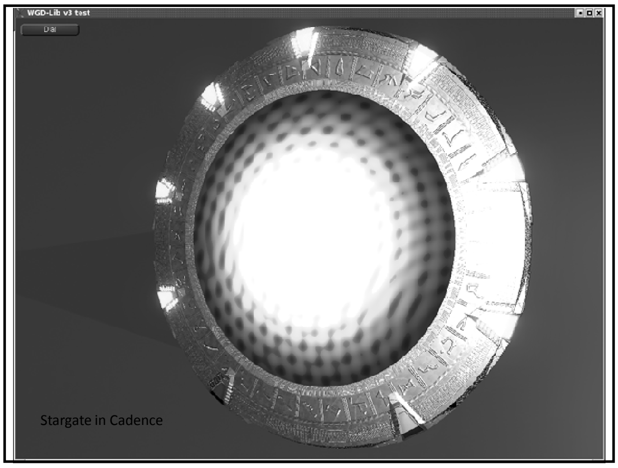
Several motivations for studying EDEN …

Cadence is still a research prototype in development

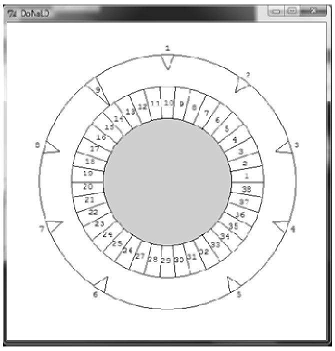
Cadence is potentially powerful and expressive, but there is a 'science' to using it for Empirical Modelling: cf. the 'scientific method', good engineering design

Experience with EDEN clarifies the principles underlying the use of Cadence for EM, and informs the design of interfaces for the use of Cadence by non-programmers

Many EDEN projects developed over twenty years:
`http://empublic.dcs.warwick.ac.uk/projects/`



Stargate in Cadence



Stargate in the 'definitive notation for line drawing' DoNaLD in EDEN

Extracts from the `cartoonstargate` script

```
%donald
point centre
int outrad, midrad, innrad
circle innc, midc, outc
```

```
centre = {500,500}
innc = circle(centre, innrad)
midc = circle(centre, midrad)
outc = circle(centre, outrad)
```

```
innrad = 200
midrad = 300
outrad = 400
```

```
point inn1, …,  inn39
point out1, … , out39
line spoke1, …, spoke39
spoke1 = [inn1, out1]
```

```
inn2 = centre + {innrad @ theta2}
out2 = centre + {midrad @ theta2}
theta2 = theta1 + (2 - 1) * 2 * pi div 39
```

```
point chev9L, che9R, chev9V
line chev9LV, chev9RV
chev9LV = [chev9L, chev9V]
chev9RV = [chev9R, chev9V]
```

```
real chevsubtend
chevsubtend = 0.1
real chevangle
chevangle = 8 * pi div 39
```

```
boolean locked9
int radchev9V
locked9 = false
```

```
chev9L = centre + {outrad @ ((pi + chevsubtend) div 2 - (2-1)*chevangle)}
chev9R = centre + {outrad @ ((pi - chevsubtend) div 2 - (2-1)*chevangle)}
chev9V = centre + {radchev9V @ (pi div 2 - (2-1)*chevangle)}
radchev9V = if (locked9) then midrad else (midrad+outrad) div 2
```

More about Cadence

- Nick Pope (forthcoming 2011). *Supporting the Migration from Construal to Program: Rethinking Software Development*. Draft PhD Thesis (submitted), Computer Science, University of Warwick, Coventry, UK.
- Beynon and Pope, *Cadence and the Empirical Modelling conceptual framework: a new perspective on modelling state-as-experienced*, Warwick CS-RR-#447