

Classical programming ...1

Behaviour is derived from a pre-specified conception of function and purpose ...

... based on interactions whose outcomes are reliable and for which the mode of interpretation is determined in advance

...motivates declarative approaches

Classical programming ...2

... motivates declarative approaches:

```
output=F(input)
```

... problematic to deal with a dynamic input, as in playing a game

... hence add "lazy evaluation" to model as

```
stream_of_output=F(stream_of_input)
```

Significance of interpretation ...

Miranda *can* be viewed as a definitive notation over an underlying algebra of functions and constructors

BUT this interpretation emphasises

program design as a state-based activity

NOT

declarative techniques for *program specification*

Illustrative example

... a version of 3D OXO written in the functional programming language Miranda

... to be compared with *oxoJoy1994* which was in some respects 'derived' from it

Two experimental systems!

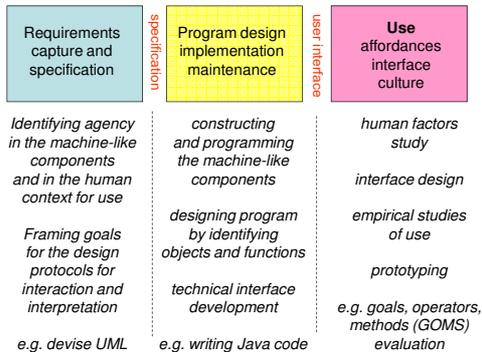
A definitive Miranda ("admira"): definitive notation with general functional programs and types as operators & data structures

The Kent Recursive Calculator (KRC): developing functional programs by framing definitive scripts

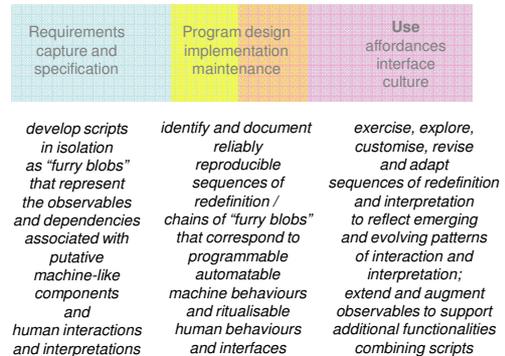
Two emphases

- Empirical Modelling encourages us to consider programming in a holistic way, using similar principles to deal with the entire process of development from conception to customisation and use
- It also has a means to represent the specific activity that is captured by a traditional program (a "pseudo-program")

Traditional programming



Empirical Modelling

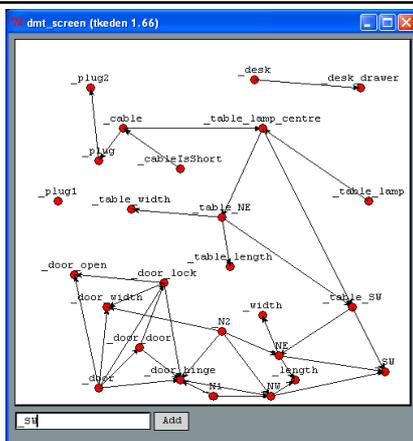


Rethinking programming ...

... formal specification from an observation-oriented perspective

Objects and dependencies

- An **object** corresponds to a particular way of associating observables: grouping together observables according to whether they exist concurrently
- A **dependency** links observables according to how they are linked in change: whether making a change to the value of one observable necessarily entails changing others



Object model vs. account of observation

An account of observation is a more primitive concept than an object model: it entails fewer preconceptions about what might be observed ...

“Definitive scripts are neutral wrt agent's views & privileges”

Object model vs. account of observation 2

Definitive script expresses different agent views and privileges to transform (cf. subject-oriented programming)

“What architect can do vs what user can do”

... highlights how the script affords *views of* and *access to* possible transformations

Objects vs observations 1

A definitive script

represents the atomic transformations of a geometric symbol

DoNaLD room can be transformed through redefinition in ways that correspond ‘exactly’ to the observed patterns of change associated with opening a door, or moving a table

Objects vs observations 2

Thesis:

- set of atomic transformations of a symbol captures its semantics [cf. Klein's view of a geometry as “the study of properties invariant under a family of transformations”]
- Illustration via a geometric pun (demo)

Is the DoNaLD room an object in the class-based OOP sense? 1

Can view each room transformation as a method for the object

BUT

definitive script is an object specification

only if

set_of_transformations_performed_on_room is **circumscribed**

Is the DoNaLD room an object in the class-based OOP sense? 2

Circumscription creates objects

BUT

a definitive script merely reflects observed latent transformations

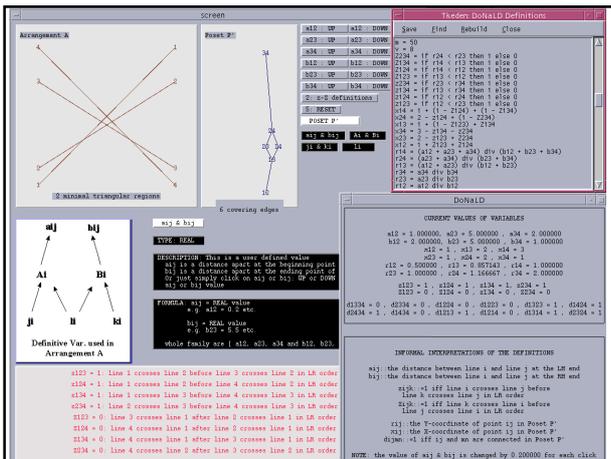
Comprehending / designing an object = knowing / determining everything we can do with it

BUT

definitive script doesn't circumscribe the family of transformations that we can apply

From logic to experience

- the computer enables us to use logical constructs to specify relationships that admit reliable interpretations and support robust physical realisations
- human skill and discretion plays a crucial role in crafting ritualisable experiences
- NB classical computer science doesn't take explicit account of robust physical realisations or ritualisable experience

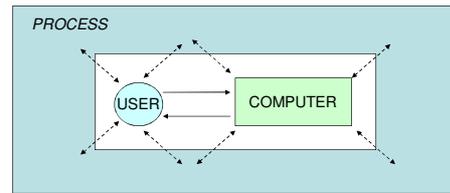


Features of the lines model ...

- directly accessible external observables: $z123 = 1$ means that line 1 crosses line 2 before line 3 crosses line 2 in L-to-R order
- the ideal geometry as associated with a mode of interaction with the model (subject to being able to enhance the accuracy of arithmetic indefinitely on-the-fly)

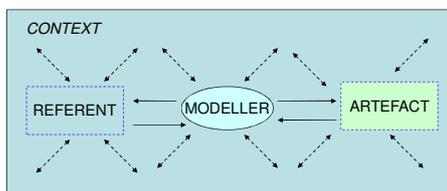
Programming from two perspectives

- a program is conceived with reference to how its behaviour participates in a wider process with functional objectives: states emerge as the side-effects of behaviours
- a computer artefact is developed so as to reflect the agency within an environment: the artefact and environment evolve until (possibly) program-like processes emerge



Conventional programs as embedded in processes of interaction with the world

Programs are understood in relation to processes in their surrounding environment



Artefacts and their referents as sculpted out of open interaction with the world

States of the referent and the artefact are connected through experience of interacting with the referent and the artefact

An EM perspective on programming some problematic issues

In focusing on current state-as-experienced, we have some problems to resolve:

- Behaviour raises questions about agency: what is the status of a "computer" action?
- How do we deal with state-as-experienced in semantic terms?
- How do we make science of activities in which human interpretation is so critical?