CS405 Introduction to Empirical Modelling - 2013-14

Empirical Modelling ("EM") is a body of principles and tools concerned with computing activity that is based on observation and experiment (hence 'empirical') ...

... the principal theme of CS405 is alternative ways to think about "computing-in-the-wild"
- with special attention to software development, computer programming and computer science

---

Introductory notes for Lab 1

Orientation

Key questions ...

1. *To what extent have the problems that underlie the 'software crisis' been resolved?*
2. *Can current principles and tools for software development meet the additional challenges that contemporary use of computers present?*

Will argue that:

1. *The problems underlying the software crisis are unresolved.*
2. *Current principles and tools for software development are inadequate to meet the challenges of contemporary use.*

Background resources

Distinguished IBM software engineer and computer scientist **Fred Brooks (1987) No Silver Bullet** is a good source for reflections on the software crisis

- argued that none of the sw development techniques (programming paradigms, object orientation, formal methods, methodologies etc) of the time was resolving the essential difficulties.
- suggested that software should be 'grown' rather than constructed.
- identified *conceptual integrity* as a key problem, and advocated that software should be developed *as if* by a single designer.

Revisited this topic in 1995, and suggested that his conclusions remained valid.

Response to Brooks by **David Harel (1992) Biting the Silver Bullet** emphasised the importance of taking human capacities such as vision into account (cf. "visual formalisms" like statecharts). Later led to the Play-Engine (Harel and Marelly, 2003).

Michael Jackson (2006) What Can We Expect from Program Verification? emphasised the importance of taking account of empirical understanding of the domain in complex sw system development.

Common thread is focusing on modelling activities that have an empirical aspect, even though our objective may be to describe a program that is logical and formal in character. Compare the aspirations in:

- object-oriented programming - seeing programming as modelling
- agile development approaches: emphasis on *evolving* specifications and programs

CS405 draws attention to the semantic / philosophical issues raised by making these moves - e.g. how these fit with the traditional story about specification, and program/PL semantics (cf. **Winograd and Flores's Understanding Computers and Cognition (1986)**, **Dave West's Hermeneutic Computer Science (1997)** and **Object Thinking (2004).**)

Key concerns to be addressed in tackling Brooks's agenda: How to adapt to uncertain and evolving requirements? How to accommodate domain learning when developing software? How to support the communication and collaboration between developers?

In contemporary applications, have even more demanding pressures - e.g. to support end-user development, or distributed participatory design, in live programming environments (such as arise from the Web, mobile technologies and real-time control and monitoring of embedded systems). We will motivate this in Lab 1 with reference to recent online materials from Bret Victor.

Tools

Will be making use of two related tools for EM in the course of the module:

- the EDEN interpreter, a well-established tool that has been used for over 25 years
- the JS-EDEN web-enabled variant of EDEN, which has been in development over the last 2-3 years

Some familiarity with EDEN will be essential, as many of the illustrative models to be discussed in the module make use of it. In the assessed work, students can make use of either EDEN or JS-EDEN. JS-EDEN exploits JavaScript in its implementation. It was conceived and initially developed by Tim Monks as the subject of his MSc dissertation, and has since been further developed by Nick Pope and Ant Harfield.

In this first lab, some of motivating ideas and background concepts in EM will be introduced through a "guest appearance" of Cadence, a research prototype that was developed by Nick Pope in connection with his EM-related doctoral work. You will not be expected to understand Cadence in detail, but to appreciate the quality of the software development at which it is directed.

How tools are used in the module

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.

Key notions

In Empirical Modelling, we build *construals* rather than programs. A construal is an artefact you can interact with which helps to make sense of something else. In this lab, the computer model we study is a construal of the Stargate - interacting with it helps to understand what a Stargate is, whether or not you already have some idea of this.

In building construals rather than programs, we need different concepts. The key concepts we introduce are:
observable - dependency - agent ("ODA")

- an **observable**: - something perceived as having an identity and being subject to change (e.g. a chevron in the Stargate)
- a **dependency**: - a relationship that is perceived to connect a change to one observable to a change to another (e.g. when the orientation of the Stargate changes, so does the orientation of the "puddle" within it)
- an **agent**: - something that can be deemed to initiate change (e.g. the person who presses the "dial" button in the Stargate)

A key issue is that these notions are *subjective* - they are relative to an observer and a mode of observing and interacting.

About the assessed work

There are variety of different ways in which this can relate to the module ...

- can choose your own topic for the assessment - which comprises a paper + a model
- can assign a weighting to the written and practical work in the assessment in ratio from 70:30 to 30:70
- can contribute through
    - model development (using a variety of tools available)
    - extension / commentary / analysis of existing models
    - extension of the tools

The module is wide-ranging in scope, covering many different themes (to include non-logicist foundations, educational technology, concurrent systems modelling as motivated by the challenges of software development), and spanning technical, methodological and philosophical issues. You will not be expected to understand everything in depth, or to master all the techniques. Note that you will be able to do well in the assessed work without any background knowledge of conventional software development or advanced programming skills.

Questionnaire to help us gauge the audience ...
