L. Shyamal, cc-by-2.5, http://en.wikipedia.org/wiki/File:Bug_aggregation.jpg

The Open University
Centre for Research in Computing

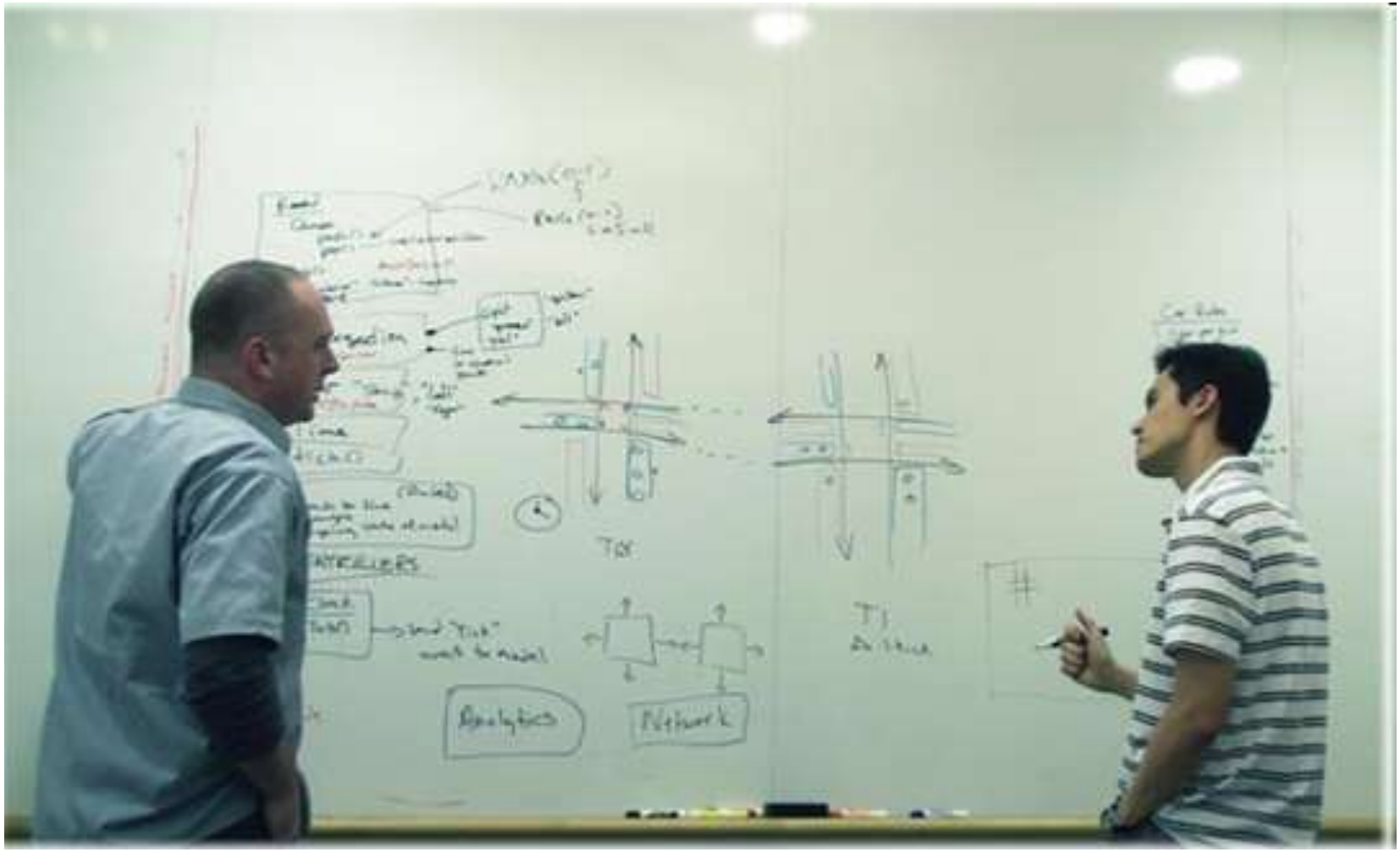
Marian Petre

Doug Lennox

Helen Sharp

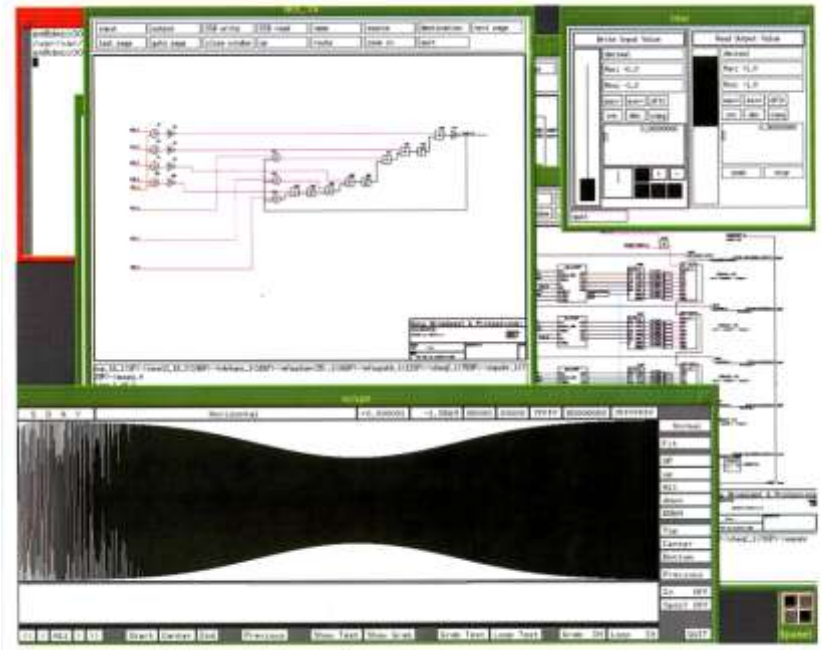
Helen Sharp

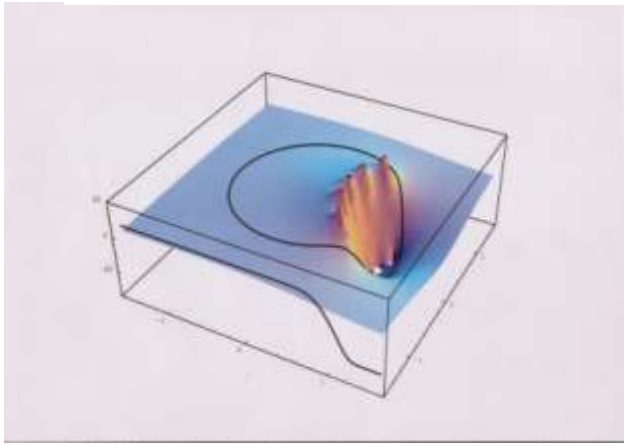
Hugh Robinson

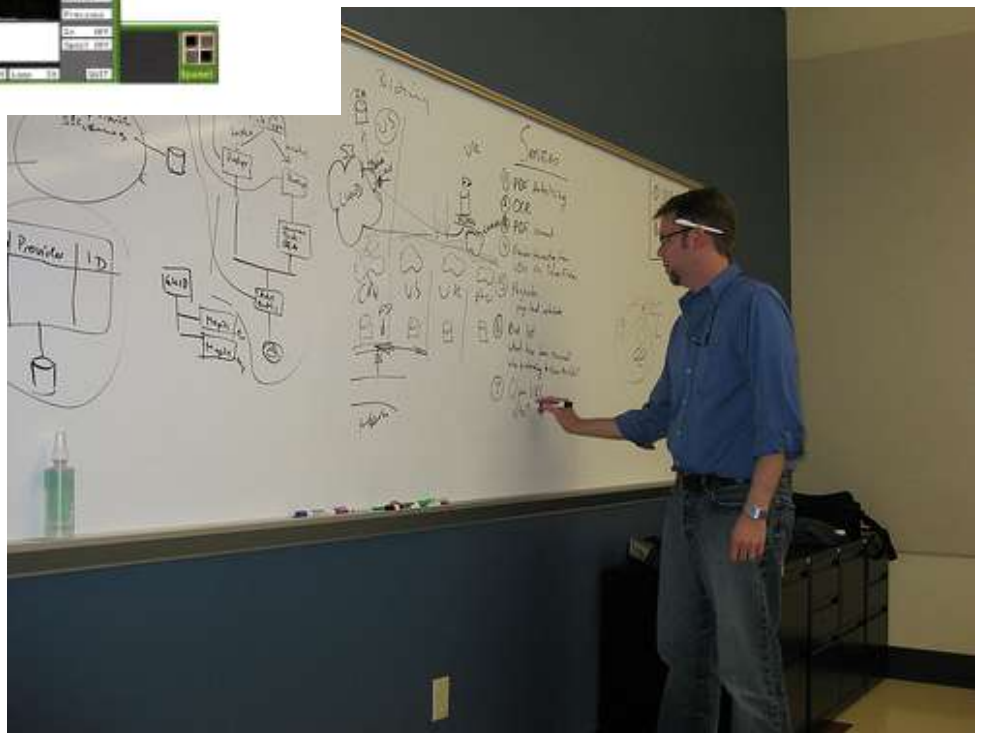


Studying Professional Software Design workshop

Max Patte – Reflection – Te Papa







Martin Kalfatovic: MOBOT

The Open University

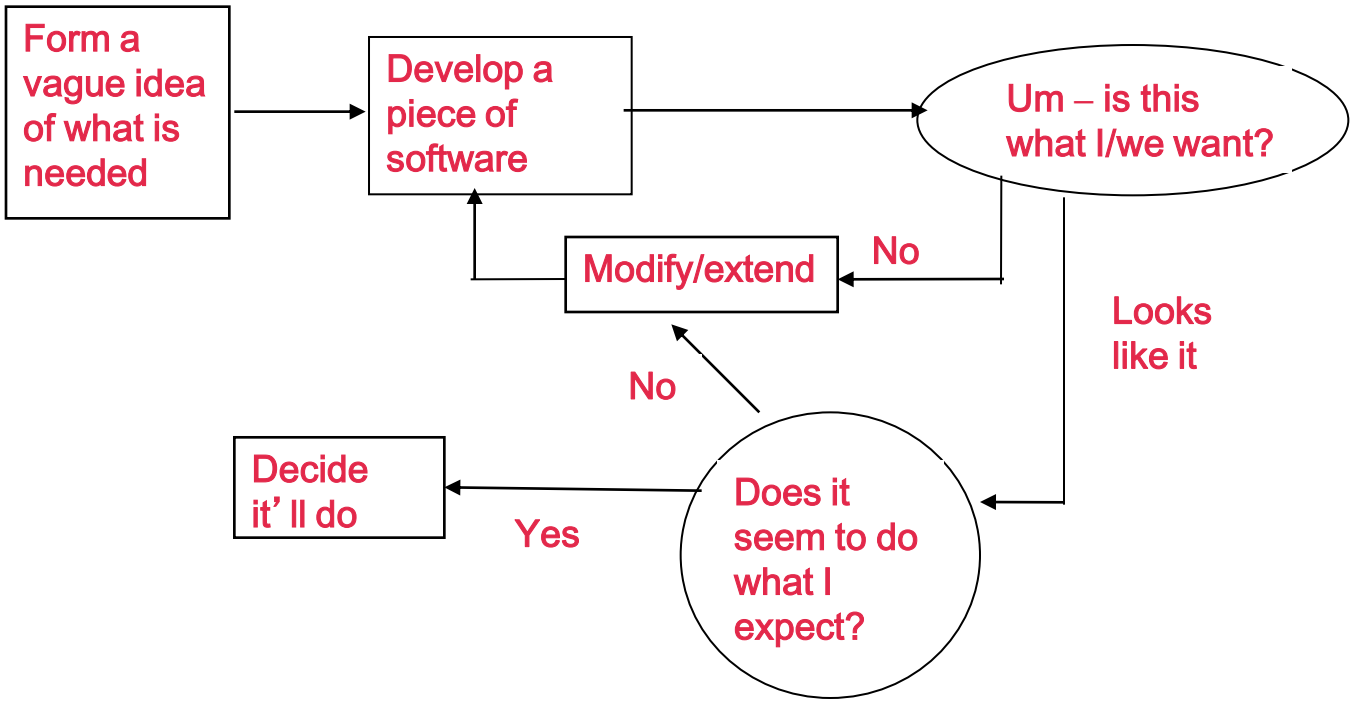
Centre for Research in Computing

some typical characteristics of scientific software *development practices*:

- heavily iterative: try it and see
- requirements are not fully understood, either by the users or the developers
- evolutionary: requirements emerge as understanding of science and the software evolve
- lack of testing, much less systematic testing procedures
- little ‘community of practice’
- programming *per se* is not valued

From empirical work by Judith Segal

A model of scientists developing their own software



Judith Segal

contexts vary …

- developing one's own thinking - using the code as a thought prompt
- immediate solutions to problems at hand
- developing a library of components – for use by a larger group
- developing substantial models or simulations to be used by a community
- developing infrastructure software for scientific facilities

“Faster chips and more sophisticated algorithms aren’t enough—if we really want computational science to come into its own, we have to tackle the bottleneck between our ears.”

Greg Wilson

Time to solution is determined by:

how long it takes to write a program	how long it takes that program to run
human time	machine time

Every language makes a tradeoff between these

Python	Java	Fortran
MATLAB	C#	C

Greg Wilson
Software Carpentry

experts can:

- identify what is relevant and important and to ignore the unimportant;
- match strategies to tasks;
- recognise resonances across domains;
- have and use strategies for dealing with intractable problems by recognising analogies or transforming them into simpler problems;
- understand the consequences of design decisions, to encompass both abstraction and detail;
- handle conflicts among constraints or principles.

Autodesk

Helen Sharp

disciplines of innovation

- ways to maintain the knowledge base
- ways to change perspective
- ways to expand the search space

Many of their strategies concern
expansion of the design space,
not just convergence to a solution.


Jennie Lee Building
Health & Safety

tolerance of error

- understanding by breaking
- leveraging insight
- ecologies of bugs
- tolerance within context
- deferral

domain knowledge

Experts:

- think hard about the problem domain – before constructing solutions
- know where the domain knowledge resides in the program
- recognise a change of purpose

doing certain actions in the right order:

e.g.:

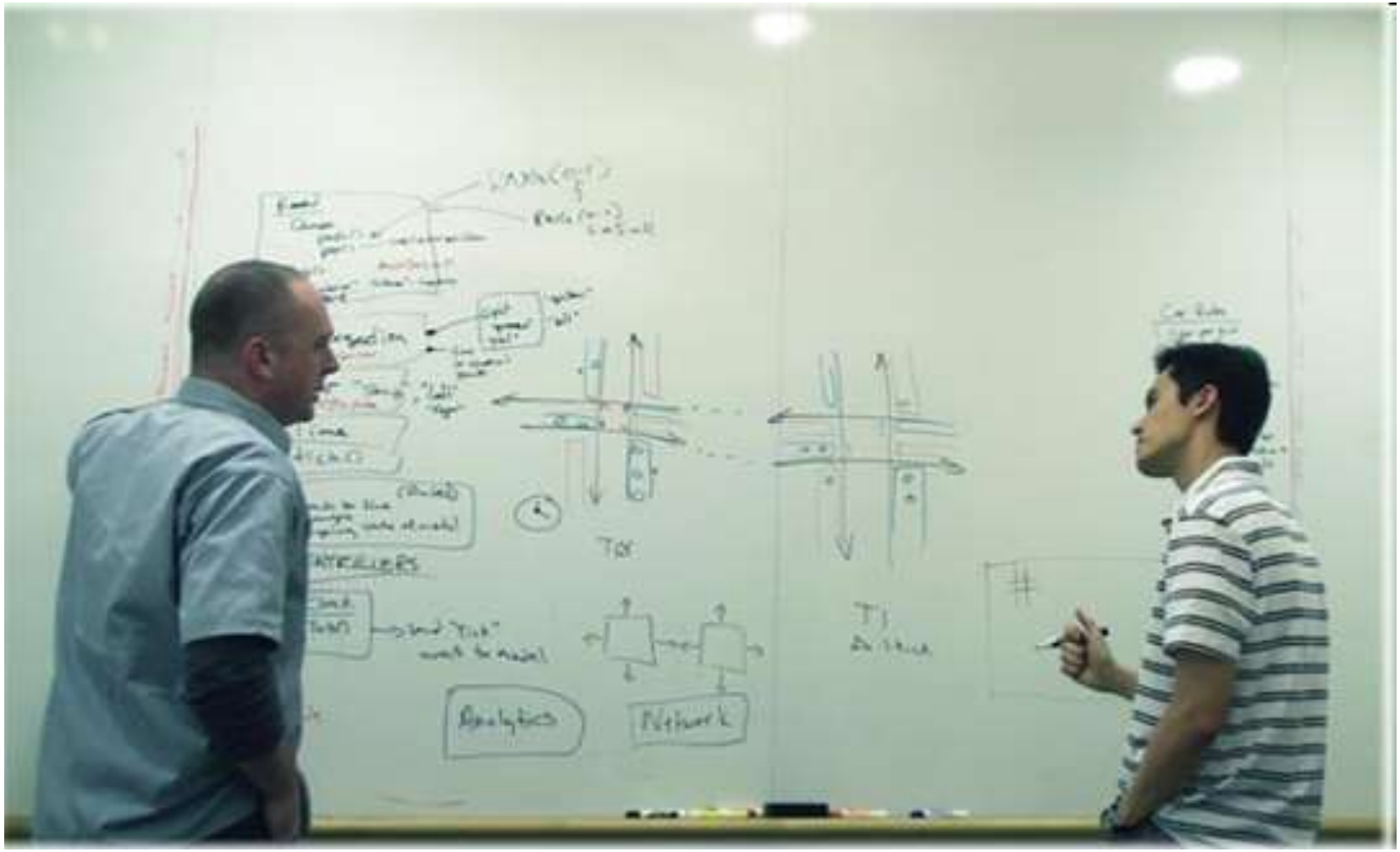
> “Setting goals before taking action
>
> Understand problems before generating solutions
>
> Designing before writing design documents
>
> Validating designs before investing in code
>
> Steak before sizzle”

John Schrag, Autodesk

http://dux.typepad.com/dux/2009/07/values-in-software-design-practice-.html

experts keep track of…

- provisionality: awareness of which decisions are firm, and which are exploratory
- rationale: why key decisions are made
- provenance: how a particular result came to be – with a trail in the code and output



Studying Professional Software Design workshop

Experts use systematic practices …

- testing, debugging, code reviews
- daily discussions
- building tools to suit practice
- disciplines of innovation
- tinkering, play, bricolage

… that are socially embedded and reinforced

- pair debugging
- reliance on team to catch slips
  → freedom to experiment
- rewarding success
- roles related to skills

"heroic programming"
the guy w/ the anorak
who developed s/w that
worked for as long as he
stayed at the company and
stopped the day after

reflective practice

- systematic efforts to alter perspective
- deliberate changes of representations, of paradigms
- cultivating an awareness of alternatives
- reviews of experience
- tolerance for error

why bother?

Because …

good practices can save you time and pain;

bad practices can damage your science.

“Sound methodology can empower and liberate the creative mind; it cannot inflame or inspire the drudge.”

(Fred Brooks)