# Models of scientific software development

Judith Segal

Empirical Studies of Software Development Group,
Centre for Research in Computing
The Open University
UK MK7 6AA
j.a.segal@open.ac.uk

## Background

One feature which distinguishes scientific software development is the poorly understood and complex domain:

- The scientists have to be deeply involved with the development
- Often, the scientists are the developers, ’professional end user developers’, or they develop the software in partnership with software engineers

## My ***creed***:

In order to identify those software engineering techniques and tools which might best benefit developers of scientific software, we have to understand

- What they do when they develop software
- The context in which it is developed.

## My field studies

### Concerned the following scientists

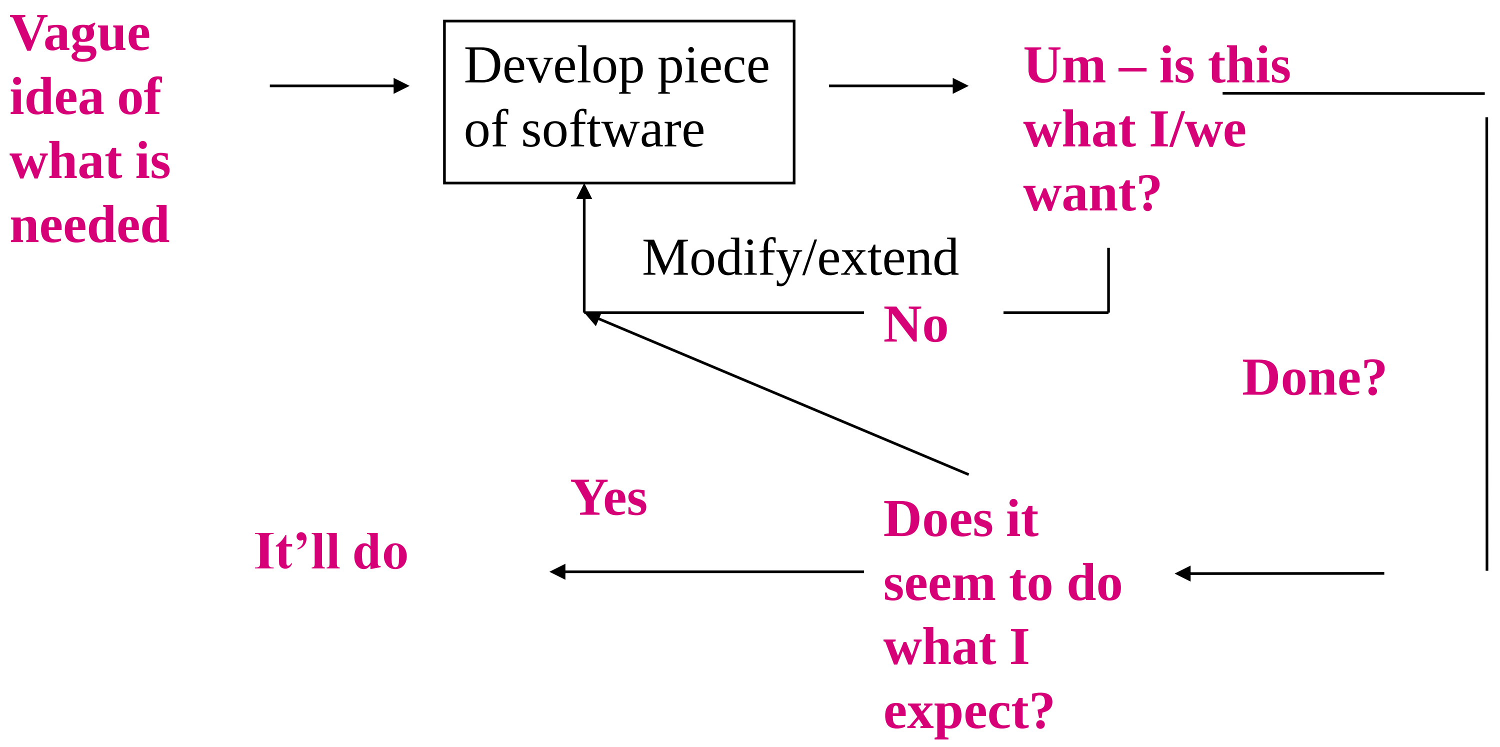
- Financial mathematicians
- Earth and planetary scientists
- Biologists

### Developing

- Models of financial markets
- Software to drive instruments
- Software to store, manage and analyse data

Sometimes scientists develop software on their own, sometimes in partnership with software engineers.

## Scientists developing software: a model of scientific software development (SSDS)


Vague idea of what is needed
Develop piece of software
Um – is this what I/we want?
Modify/extend
No
Done?
Yes
It'll do
Does it seem to do what I expect?

## Characteristics of this model

- Requirements emerge rather than being specified upfront
- Requirements identification and evaluation co-occur
- Testing is cursory

## It works (apparently)! But in a very specific context…

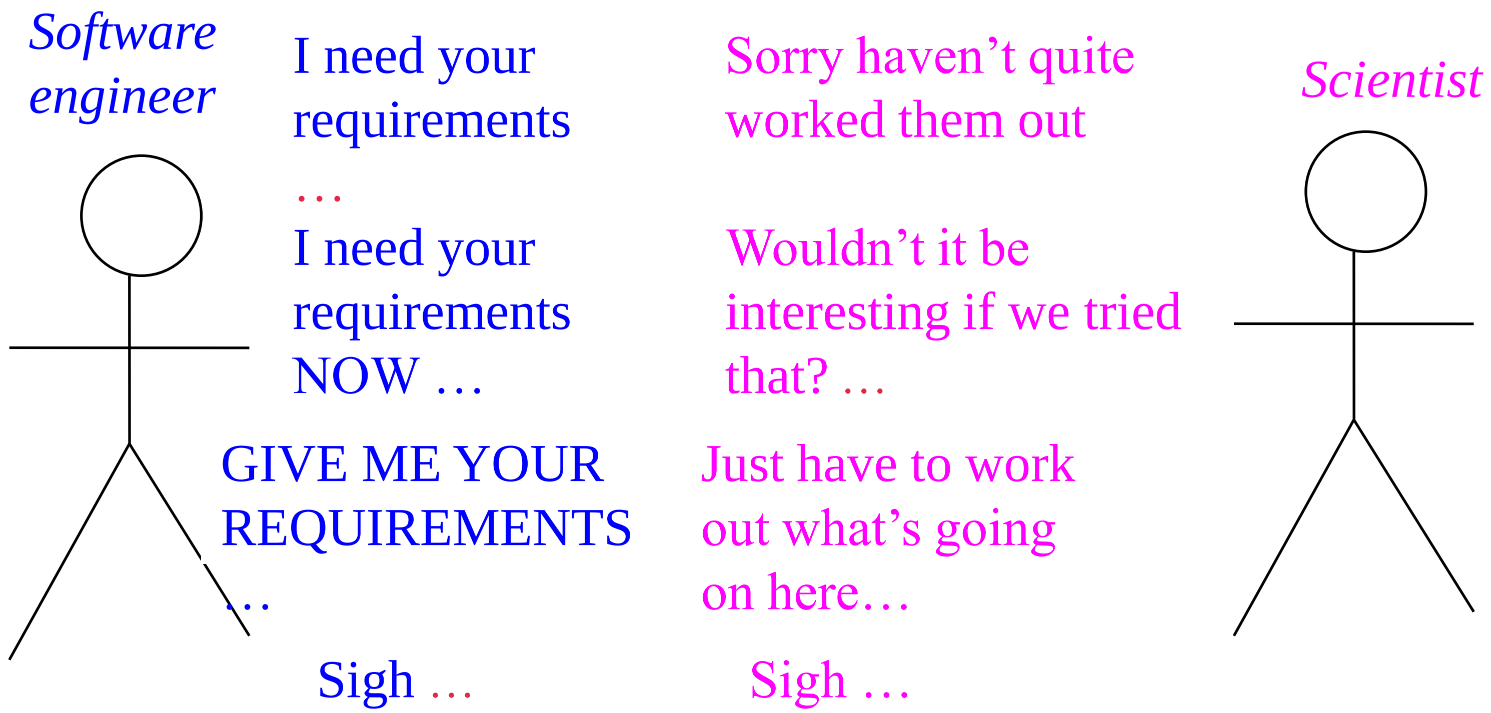
- Developer is embedded in the user community
- Software not overly complex nor safety-critical
- The developer is truly representative of the user community (so the community is coherent).

What happens when the context changes and software engineers become involved?

## Problems! (i) inappropriate use of software engineering models

The context:

- Software engineers developing a library of components to drive a laboratory instrument in a spacecraft
- Software engineers following a waterfall-like model of development advocated by the European Space Agency
- Scientists used to developing software to drive instruments in the lab
- Scientists working on a lab model to derive the requirements.



*Software engineer*

I need your requirements …

I need your requirements NOW …

GIVE ME YOUR REQUIREMENTS …

Sigh …

*Scientist*

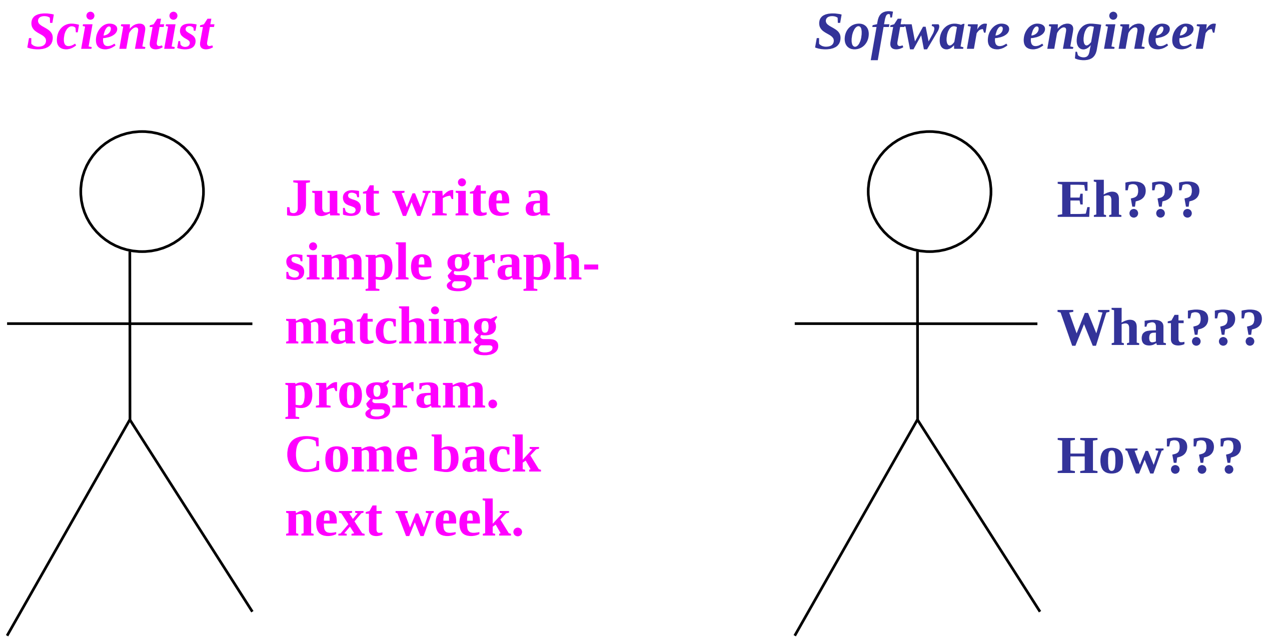
Sorry haven't quite worked them out

Wouldn't it be interesting if we tried that? …

Just have to work out what's going on here…

Sigh …

## More problems: (ii) inappropriate use of the SSDS model


*Scientist*
Just write a simple graph-matching program. Come back next week.
*Software engineer*
Eh???
What???
How???

## So…some software engineering techniques are less appropriate to scientific software development…

- Upfront requirements specification

## Some are more …

Testing

[Testing] has also been something new for me. Testing just didn't happen. .. One assumes that if the numbers came out of the other end, they were right. And that is in hindsight, an embarrassingly stupid assumption… the concept of writing code that is testable has been a very useful one.'

## BUT which are more appropriate and which less, and in which contexts?

Software engineering has a huge variety of software engineering techniques and tools

Scientific software development takes place in a huge variety of contexts

HPC or not

Software engineers involved or not

Test oracle or not

Purpose of software

- Theory testing
- Data management
- Analysis
- Orchestration of workflows
- Simulations

## A possible research agenda

1. Need to identify contexts of scientific software development
2. Need to identify possibly relevant techniques
3. Need to identify a matching between 1 and 2 (including validation)

- AND

4. Need to make scientists aware of our results.

Thank you for listening.

Any questions?