JISC e-Infrastructure Programme Meeting
Birmingham, 7 Feb 2008

Agile and Open Development

Neil Chue Hong, OMII-UK
Ross Gardler, OSS-Watch

JISC

Ross Gardler
ross.gardler@oucs.ox.ac.uk

http://www.oss-watch.ac.uk

About OSS Watch

- JISC funded advisory service on open source software (non-advocacy)
- Provide one-to-one consultancy on all issues relating to open source
  - Licencing and IPR management
  - Making your code available as open source
  - Evaluating and using open source
- Produce briefing notes
- We help you understand and apply the JISC Open Source Policy

OMII-UK: Software Solutions for e-Research

omii-uk
www.omii.ac.uk

UNIVERSITY OF Southampton

MANCHESTER 1824

THE UNIVERSITY OF EDINBURGH

EPSRC

JISC

- OMII-UK provides software and support to enable a sustained future for the UK e-Science community and its international collaborators.
  - Core support and development
  - Commissioned Software Programme
  - ENGAGE: improving access to e-Infrastructure
  - Phase II: 2006 – 2009
- Contact me: N.ChueHong@omii.ac.uk

OMII-UK: Adding benefit to e-Science

- More than just the middleware
  - go above the components to provide added value
- Skilled team to help the community
  - putting the right things together, integrating components
  - providing consultancy and support to improve takeup
  - developing, commissioning and improving software

Session Timetable

- Introduction to Agile Development (Neil)
- Assessing your Agility exercise
- Comparing Open Development (Ross)
- Discussion: JISC projects and Agile/Open Development

JISC

What is Agile Development?

JISC

Key points

- Why did Agile Development develop?
- What does it mean to be Agile?
- What are the basic principles of Agile Development?

- Suggested reading:
  - An Introduction to Agile Methods (Hayes, Andrews)
  - Agile Software Development Ecosystems (Highsmith)
  - Agile Alliance website: http://www.agilealliance.org
  - The New Methodology (Fowler)

From Nothing to Monumental

- Most software development is chaotic
  - code like hell (aka code and fix)
  - no underlying plan
  - works initially but hard to scale as system grows
    - hard to add features
    - hard to find and fix bugs
    - long test phases (if at all!)
- Engineering Methodologies imposed discipline
  - improve predictability, improve efficiency
  - but seen as bureaucratic, unpopular
  - reduce the pace of development

From Monumental to Agile

- Developed in reaction to Engineering Methods
  - compromise on amount of process
  - adaptive rather than predictive
  - people-oriented rather than process-oriented
- Why doesn't Monumental work for software?
  - cost of design versus cost of construction in a project
  - impact and likelihood of change
  - formal design models in software are still maturing
  - monumental doesn't always work for engineering!

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it.

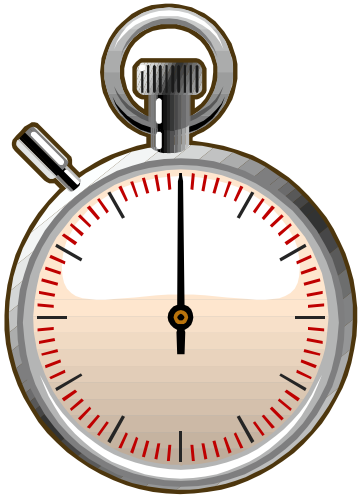
Through this work we have come to value:

- ***Individuals and interactions*** *over processes and tools*
- ***Working software*** *over comprehensive documentation*
- ***Customer collaboration*** *over contract negotiation*
- ***Responding to change*** *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

Core Values of Being Agile

- **Iteration!**
  - not about completing tasks but about completing business functionality and reflecting on what has been achieved
- **Increments!**
  - do the simplest thing that could possibly work and let subsequent iterations build on this
- **Customer-driven!**
  - get your customers involved and exhibit progress each iteration
- **Courageous Communication!**
  - open and honest can often be difficult across levels
- **High quality!**
  - by continuous improvement, not over-engineering
- **United team!**



Agile is not just XP

- … although it is by far the best known

- **XP** - Communication, simplicity, feedback, and courage; use specific technical and collaborative practices
- **Scrum** - Prioritised list of requirements on a product backlog, daily standup meetings, use retrospectives to correct the process
- **Crystal** - Emphasize people, gather techniques from other methods, improve communications, adapt the process itself
- **Lean** - Move closer to customer, shorter cycles, eliminate waste, decide as late as possible, empower the team, build in integrity
- **Dynamic Systems Development Model** - Empower the team to make decisions, frequent product delivery, collaboration between all stakeholders.
- **Feature Driven Development** - Centre development around the feature, create a domain model with domain experts

- Each methodology emphasises a set of mutually supporting techniques, backed by common values.

Myths of Agile Development

- ***Agile is new and untested***
- ***Requirements not documented***
- ***No architect(ure) = chaos***
- ***There is no design stage***
- ***Risks are being ignored***
- ***Devs / Customers will hate it***

omii-uk
www.omii.ac.uk

Is Agile for my project?

- Agile isn't always appropriate
- It doesn't guarantee success
- Must be adaptive!
  - spot deficiencies and correct process
- It can be difficult to apply particular techniques to distributed teams
  - to be covered later
- *Many projects have benefitted from committing to an agile methodology*
- Are you on the way to being agile already?

Assess your Agility!

- Look at the sheet, consider your project, and answer the questions truthfully!
  - If you aren't sure, give yourself zero points
- Total up your scores for each of the five sections
- Write them on the sheet on your table
  - Anonymous exercise to protect the innocent!
- Should take about 15 minutes

JISC

Open Development

- What is open development?
- How does it relate to agile development?
- How does it relate to JISC projects?
- How do we improve the “typical” JISC development approach?
- How can OSS Watch and OMI-UK help?

JISC

SS Watch

Open Source: More than a Licence

- It is a development methodology
- Key attributes include:
  - User engagement
  - Transparency
  - Collaboration
  - Agility
- But, Open Development is not the same as Agile Development

JISC

OSS Watch

What are the differences?

- Some agile processes require co-location of developers and customers
- Open development requires that anyone can participate regardless of their location
  - NOTE: this does not mean that anyone has the **right** to modify open source code in the core repository

JISC OSS Watch

In what ways are they similar?

- Many agile practices evolved from or alongside open development, e.g.
  - Collective code ownership
  - Incremental design and architecture
  - Real customer involvement
- Other agile practices are so “obvious” they are already found in open development, e.g.
  - Version Control
  - Trust

JISC

Why should JISC projects care?

- Agile development, when done right, is repeatedly shown to be the most effective way of producing software.
- JISC projects are usually either:
  - Too small to follow all agile process, and/or
  - Too widely distributed to follow all agile processes
- Open Development is more suited to a typical JISC project
  - Regardless of licence choice

JISC

So is there a defined Agile Process for Open Development?

No :-(

JISC

Can we create an Agile Process for Open Development?

Yes :-)

At least I believe so

JISC

SS Watch

A Mapping from XP to Open Development

The following slides map eXtreme Programming* to our proposed Open Development process.

We will look at each category in the questionnaire and briefly discuss the main practices within it.

After this section we will discuss these practices in the context of JISC projects. Which do we feel are appropriate and which are not.

* as described in The Art of Agile Developmet by James Shore and Shane Warden, published by O'Reilly

JISC

Thinking: Mindfullness Vs. Experts

- Root-Cause Analysis
- Pair Programming

- Root-Cause Analysis
- (Peer Review)

JISC

OSS Watch

Collaborating: Clear Communication

- Trust
- Ubiquitous Language
- Coding Standards
- Iteration Demo
- Reporting

- Trust
- Ubiquitous Language
- Coding Standards
- Snapshot Releases and Screencasts
- Reporting

JISC

OSS Watch

Releasing: Demonstrating Potential Value

- "Done Done"
- No Bugs
- Version Control
- Ten-Minute Build
- Continuous Integration
- Collective Code Ownership
- Documentation

- "Done Done"
- No Bugs
- Version Control
- Ten-Minute Build
- Continuous Integration
- Collective Code Ownership
- Essential Formal Documentation

JISC

SS Watch

Planning: Plan for change

- Vision
- Release Planning
- The Planning Game
- Risk Management
- Iteration Planning
- Stories

- Vision
- Release early, release often
- Issue Management
- Risk Management
- Iteration Planning
- Story Issues

JISC

OSS Watch

Developing: High Quality = Low Cost

- Incremental Requirements
- Customer tests
- Test-Driven Development
- Simple Design
- Incremental Design and Architecture
- Spike Solutions

- User Engagement
- User Tests
- No Test, No Commit
  Simple Design
- Evolutionary Design
- Snippets

JISC

So how does this process score?

- This mapping is inspired by my experience in The Apache Software Foundation.
- This is merely my own experience
- The ASF process scores as follows:
  - Thinking: 89
  - Collaborating: 85
  - Releasing: 96
  - Planning: 99
  - Developing: 89

JISC

OSS Watch and OMII-UK can help in what way?

- Agile/Open Development requires a culture change in your project team
- Finding a good mentor is an important part of learning how to adopt open/development

JISC

omii-uk
www.omii.ac.uk

SS Watch

omii-uk
www.omii.ac.uk

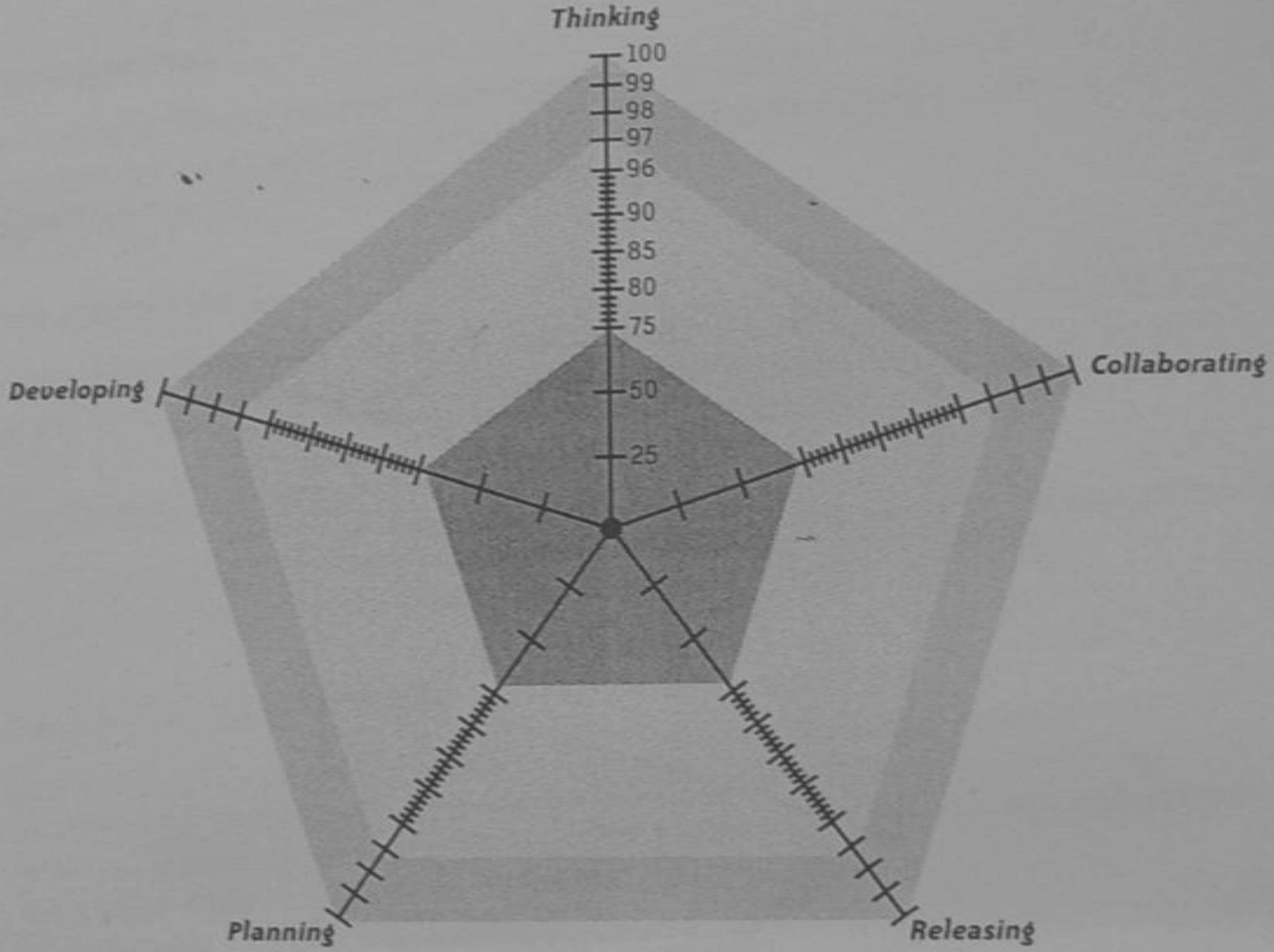
- We have staff experienced in both open and agile development
- We can mentor your team

- info@oss-watch.ac.uk

(or a bar near here)

- Help on software development and infrastructure
- Work with projects to improve sustainability of software
- Consultancy (and some funding) available

- info@omii.ac.uk


Thinking
100
99
98
97
96
90
85
80
75
50
25
Collaborating
Developing
Planning
Releasing