How do you scale agile methods?

- How do you make a big piece of software?
- Build it out of big pieces
Alan Cameron Wills

- Mentor and presenter in software development process
  Joint author of *Objects, Components and Frameworks – the Catalysis approach*
  Clients in many fields, both sides of the Atlantic

- Technical Director of **Trireme International Ltd**
  Source of **process and architecture mentoring** in UK, Europe, …
  [http://www.trireme.com](http://www.trireme.com)

- Director of **FastnLoose Ltd**
  **Software development** using agile methods and components
  [http://www.fastnloose.com](http://www.fastnloose.com)

---

**Effective improvements in productivity**

- **Agile Development**
  Customer involvement, incremental development, rigorous testing
  Convergence on real requirement

- **Component based development**
  Building from pre-existing parts
  Rapid assembly of candidate solutions
Agile Development

Principles

- Emphasis on helping people work together, rather than tools
- Incremental delivery, iterative development, strong testing

Benefits

- Convergence on real current requirement
  - cf what they thought was required, at time of signing
- Early delivery of a working solution

Examples in this space:

- eXtreme Programming (www.xpdeveloper.com)
- Crystal (www.crystalmethodologies.com)
- DSDM (www.dsdm.org)

Moving Requirements

- take aim ...
- no steering
- moving target

- control
- guided: continual course corrections
- moving target
Incremental development

- Emphasis on customer feedback during development

Fast Software Inc

Client

requirements

Developer

Software

incremental delivery

Features tested & demonstrated as soon as developed

Minimal investment in travelling in the wrong direction

Agenda

Agile development

Component Based Development

pluggability

software product lines

model driven architecture

Incremental CBD
“Component Based Development”

CBD has (at least) two meanings:

.NET, J2EE/EJB, …
Separation of business logic from container

Product Line Architectures
Family of end-products from a kit of pluggable components

Kit of pluggable components

Java Beans [-like] example:
Plug in more bits from the kit...

Components unchanged, but rewired:

- Few connector types compared to number of components

Connector specifications

Connector definitions are fundamental to pluggability, whether the connectors are pure interface standards, or have an implementation as a software framework.
Software Product Lines

Families of products
made from kits of components

Product Line Architecture =
definition of connectors

Component connectors

Connector definition can include:
- interface mechanisms – e.g. CORBA, RMT, TCP/IP, …
- basic syntax – e.g. XML
- Platform Independent Model

* = required

- classes of objects communicated on the connector
- what you can do with them
- business rules applying to them
Model what the components talk about

Different components may have different internal models, but must speak a common language.

These items appear as parameters of the actions between the components.

Business rules

Each component should observe the rules about what's allowed at least when communicating with other components.

Static constraints — written informally or in OCL/C++/...

e.g., an Order shall be charged to an Account of the same Customer to whom it is to be delivered.
**Dynamic rules**

Rules about the allowed transitions

<table>
<thead>
<tr>
<th>UK System</th>
<th>French System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Order</td>
</tr>
<tr>
<td>request item</td>
<td>request item</td>
</tr>
<tr>
<td>(requested)</td>
<td>(ordered)</td>
</tr>
<tr>
<td>pay</td>
<td>deliver</td>
</tr>
<tr>
<td>ordered</td>
<td>delivré</td>
</tr>
<tr>
<td>deliver</td>
<td>pay</td>
</tr>
<tr>
<td>done</td>
<td>finis</td>
</tr>
</tbody>
</table>

What happens if these systems pass orders to each other to fulfill?

**Testing generic software**

Component designers don’t know who they’re talking to

Supply components and frameworks with rigorous tests for plug-ins

We supply this framework

We must supply these tests

Mock Objects

We supply this framework

Someone else plugs in a component here

And }

-> 'Mock Objects'

Agile Components

Alan Cameron Wills
Activities in Product Line Development

**Rapid Assembly**

- end-product construction
- component construction
- connector definitions

**Deep Thought**

---

**Agenda**

- Agile development
- Component Based Development
- Incremental CBD
The incremental development loop

Incremental delivery

tight loop of feedback between user and developer

Users
requirements, feedback

Developers
incremental delivery

Product

Incremental CBD

requirements, feedback

Product Builders

Component Designers

Product family

Component kit

incremental delivery
Incremental CBD

**Low ceremony**
Rapid assembly of products driven by customer needs

**Higher ceremony**
More model-driven
More focus on best solution

Family planning: Proactive population

Populate kit with components needed for product family
=> analysis of family and variations upfront
Lazy population

- **Products**
  - quickly-created specialised solutions built from reusable Assets

- **Assets**
  - created by generalising from designs of known Products not from bright ideas about what might be useful

- **Protect & cultivate assets**
  - devote resources to maintaining ‘design capital’
  - Planned evolution of framework
  - Framework separately resourced from products

Software Development Framework

- **Framework**
  - **Process**
    - Management – how to run the project
    - Technical – what to do on the project
  - **Architecture**
    - design guidelines and patterns
  - **Software Components**
    - coherent structure and components that can be specialised and assembled to form project
  - **Tools**
    - and platforms
SDF Team Works on Projects

- SDF material is generalised from work on Products
- Members of SDF Team work part time on Product projects
  - Provide mentoring about SDF process, architecture, components
  - Get feedback on the SDF
  - Monitor Products for ideas that can be incorporated into SDF
  - Spend time enhancing SDF after experience on products

Summary

- Develop a coherent product line built from a component framework
  - not disparate miscellany of components
- Rapid product assembly; more ceremonious component design
- Generalise framework incrementally from delivered products
- Rigorous regression tests: supply them with components
- Treat tools, methods, and patterns as part of the component framework

alan@trireme.com

TriReme International Ltd http://www.trireme.com