Engineering the Object-Oriented Software Process: OPEN and MeNtOR

Presented by
Tony Simons
Overview

- Where have OO Methods come from?
- Where are OO Methods going?
- Beyond Methods........... to a Process Architecture approach to OO development
- Context & Background to OPEN and MeNtOR?
- A Brief Tour of OPEN/MeNtOR
  - Fundamentals of OPEN/MeNtOR
  - Software Engineering Process
  - Software Engineering Process Architecture
  - Summary
- Deploying an OO Process
- Summary
Where Did It All Begin?

- 1960s Simula 67 - OO Concepts
  - class
  - object
  - inheritance
  - relationships
  - polymorphism

- Refined and applied in 1970s with Smalltalk

- Semantic data modelling
Foundational Concepts

Object Oriented Concepts (1960-70s)
What did we do?

- Limited commercial application
  - mainly R&D labs
  - specialised applications
  - small to medium sized programs

- Developed program level support material
  - C++/Smalltalk coding standards and guidelines
  - programming language review checklist of what to look for and what not to do
  - OO design tips and hints

*Methodology = set of programming level standards, tips and hints*
From Concepts to Techniques

- 1980s Booch/Buhr/Seidewitz - OO Techniques
  - Object modelling
  - Interaction diagrams
  - CRC carding
  - Scenario analysis

- Evolution from concepts to techniques that proceduralise activities of OO design

- Limited interaction with the dominant Information Engineering methods
Concepts to Techniques

Object Oriented Concepts
(1960-70s)

Object Oriented Techniques
(1980s)
What did we do?

- Applying OO development to commercial application in certain domains e.g. Telecommunications, CAD/CAM, GIS
- Developed guidelines for design techniques
  - formalised the design documentation
  - formalised the techniques used
  - provided support for the techniques

Methodology = set of design level techniques, guidelines supporting documentation
Techniques to “Methods”?

Early 1990s - OO “Methods”

Whole series of published text book “methods”
  - Booch 1991
  - Rumbaugh et al 1991
  - Jacobson et al 1992
  - Wirfs-Brock et al 1990
Explosion of OO “Methods”

- OOA/OOD: Coad and Yourdon, 1990
- Syntropy: Cook and Daniels, 1995
- OSA: Embley et al., 1992
- ADM3/4: Firesmith, 1993
- Fusion: Coleman et al., 1994
- OBA: Goldberg and Rubin, 1992
- SOMA: Graham, 1992/5
- MOSES: Henderson-Sellers/Edwards, 1994
- Objectory: Jacobson et al., 1992
- Ptech: Martin and Odell, 1992
- Mentor: Object Oriented Pty Ltd, 1993
- BON: Walden and Nerson, 1992
- Synthesis: Page-Jones et al., 1990
- ROOM: Selic et al., 1992
- OMT: Rumbaugh et al., 1991
- Shlaer and Mellor, 1990
- RDD: Wirfs-Brock et al., 1990
- Others
Booch Method


- Object Oriented Analysis and Design With Applications, Benjamin Cummings.

Method summary

- Developed from an Ada Background
- Now oriented to C++ although language-independent
- Supports detailed design issues and some real-time issues
- Process includes Micro and Macro lifecycles
- Documentation through:
  - Scenarios, Class Diagram, State Transition Diagram, Object Diagram
OOSE/Objectory

- Jacobson et al. (1992)
  - Object Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley

- Method summary
  - One of the most mature and complete approaches available
  - Developed from Ericsson Research Labs
  - Supports detailed analysis and design process with complete process documentation
  - Discusses some project management issues
  - Documentation through:
    - Use Cases, Class Diagram, Interaction Diagram, Requirements, Analysis and Design Model
Object Modelling Technique (OMT)

✓ Rumbaugh et al. (1991)
  ✓ Object Oriented Modelling and Design, Prentice-Hall

✓ Methods Summary
  ✓ Developed at GE Concepts Centre
  ✓ Evolutionary approach from SAD
  ✓ Probably currently the most popular method in use
  ✓ Supports detailed analysis issues particularly in regard to the data modelling aspects
  ✓ Documentation through:
    ✓ Scenarios, Class Diagram, State Transition Diagram, Data Flow Diagram
OOA&D - Coad and Yourdon

❖ Coad and Yourdon (1991)
  ❖ Object Oriented Analysis, Prentice-Hall
  ❖ Object Oriented Design, Prentice-Hall

❖ Methods Summary
  ❖ Information Engineering background
  ❖ Language-independent
  ❖ Simple approach
  ❖ Stronger in analysis, emphasises data component
  ❖ Documentation through:
    ❖ OOA Model, State Transition Diagram,
    ❖ Message Service Chart
Responsibility Driven Design (RDD)

Wirfs-Brock et al. (1990)

Designing Object Oriented Software, Prentice-Hall

Method summary

- Developed at Tektronix
- Smalltalk-oriented
- One of the earliest approaches with quite widespread appeal
- Documentation through:
  - Hierarchy Diagrams, Collaboration Graphs
What Did Industry do?

- Industry began to embrace new OOA&D techniques for client-server, GUI, and PC development
- Many new industry sectors e.g. Banking, Insurance, Health
- Applied new OO concepts and languages
  - C++
  - Smalltalk
- Applied new OO design techniques
  - CRC carding
  - Use case analysis
  - Interaction diagramming
What Did We Learn?

- The textbook “methods” are not enough
  - limited scope
  - focus on design
  - no single approach is complete
  - don’t deal with the “hard” issues of project management, quality assurance, project practicalities

- Needed to develop organisation-specific OO methods
  - Integrating OO techniques
  - Integrating to traditional methods
  - Make OO techniques commercially robust

Methodology = client specific methods that covered the lifecycle requirements of the particular project
Object-Oriented Methods

The well-known OO methods are:

- Booch
- OMT
- RDD
- Coad and Yourdon
- OOSE

Most well-known “OO methods” are not really methods at all but rather a set of techniques.
Overview

☒ Where have OO Methods come from?
☒ Where are OO Methods going?
☒ Beyond Methods......... to a Process Architecture approach to OO development
☒ Context & Background to OPEN and MeNtOR?
☒ A Brief Tour of OPEN/MeNtOR
  ☒ Fundamentals of OPEN/MeNtOR
  ☒ Software Engineering Process
  ☒ Software Engineering Process Architecture
  ☒ Summary
☒ Deploying an OO Process
☒ Summary
Convergence of Techniques

Basic agreement on:
- Concepts
- Terminology
- Techniques

Waging of Peace in the notation wars with activities such as the UML/OPEN notation
- Booch and Rumbaugh merging notation
- Jacobson joining the approach
- OPEN initial involvement

Common Object oriented Methodology MetaModel Architecture (Henderson-Sellers et al 1994)
Maturing of the Basics

- Initial Techniques
- Explosion of Ideas
- Convergence of Techniques

Time

Number of techniques
Coalescence of Techniques
Increasing Focus on Process

- Move from concepts and techniques to lifecycles and processes
  - Jacobson et al
  - Shlaer/Mellor
  - OPEN and MeNtOR

- Move to consider larger scale issues
  - Organisation structure
  - Reuse strategies
  - Component based development
  - Costing models
Overview

様々 Where have OO Methods come from?
ocy Where are OO Methods going?
ocy Beyond Methods......... to a Process Architecture approach to OO development
ocy Context & Background to OPEN and MeNtOR?
ocy A Brief Tour of OPEN/MeNtOR
ocy Fundamentals of OPEN/MeNtOR
ocy Software Engineering Process
ocy Software Engineering Process Architecture
ocy Summary
ocy Deploying an OO Process
ocy Summary
What is a Method?

- Models
- Procedures
- QA
- Metrics
- Techniques
- Tools
- Guidelines
- Roles
- Lifecycle process
- Deliverables
- Representation
- Project Management
- Coding Standards
- Procedures
- Method
What is the Role of a Method?

- Organisation Standards
  - Process
  - Deliverables

- Guidance & Support
  - Techniques
  - Guidelines

- Monitoring & Control
  - Project Management
  - Quality

- It is NOT a recipe book
  - Access to business knowledge is mandatory
  - Creativity in design is still essential
A method should thus provide a standard, yet flexible, framework for developing systems, that blends engineering rigour with engineering creativity.

View a method as a street directory providing the traveller with a guide to a successful outcome.

Using a method permits success to be repeated (and failures to be avoided).
Process Maturity Levels

**Capability Maturity Model (CMM)**

1. **Initial**
   - Basic Management Control

2. **Repeatable**
   - Process Definition
   - Process Measurement

3. **Defined**
   - Process Control
   - Basic Management Control

4. **Managed**
   - Optimising
Overview

- Where have OO Methods come from?
- Where are OO Methods going?
- Beyond Methods........ to a Process Architecture approach to OO development

- Context & Background to OPEN and MeNTOR?

- A Brief Tour of OPEN/MeNTOR
  - Fundamentals of OPEN/MeNTOR
  - Software Engineering Process
  - Software Engineering Process Architecture
  - Summary

- Deploying an OO Process

- Summary
Context

OPEN = Object Oriented Process, Environment and Notation

OPEN is:
- a Public Domain Object Oriented Software Process
- the open (freely available) “lite” version of MeNtOR

MeNtOR is an industrial strength, commercial, object oriented software process
OPEN Consortium

OPEN is a group of OO professionals and researchers committed to the development of OO software processes

Consists of:

- B. Henderson-Sellers
- I.M. Graham
- D. Firesmith
- M. Page-Jones
- E. Yourdon

Object Oriented Pty Ltd

- Object Oriented Pty Ltd is Australia’s first - and leading - consulting organisation specialising exclusively in Object Technology

- Object Oriented Pty Ltd is the supplier of MeNtOR products and services
OPEN and MeNtOR

- Provide a complete and disciplined process for object-oriented software engineering
- Embody the best of Booch, OMT, RDD, OOSE, MOSES, SOMA and others
- Are largely notation, language and tool independent
- Support iterative and incremental development
- Promote Software Process Engineering (i.e., engineering the software process)
OPEN

OPEN is the public domain version of MeNtOR - an industrial strength object oriented software process

MeNtOR is a commercial implementation of OPEN

OPEN is a level 2 methodology suitable for trialling OO methods and for small projects
The Development of OPEN-MeNTOR

- Booch, OMT, RDD
- Objectory
- MOSES
- SOMA
- MeNTOR v1
- MeNTOR v2
- OPEN formed
- MeNTOR v3
- OPEN-MeNTOR

1997/8
History of OPEN

- COMMA project (funded late 1994) which, in turn, encouraged the merger of MOSES and SOMA and later Martin/Odell October 1995 OOPSLA
- 1995/6 interest from other major methodologists (e.g. Firesmith, Page-Jones, Reenskaug, Selic, Yourdon). Other eminent OT gurus join OPEN collaborative.
- First publication -- IEEE Computer, April 1996
The Development of *Mentor*

- *Mentor* was initially an internal handbook used by OOPL consultants.
- *Mentor* has been developed based on OOPL’s extensive experience with clients and our software house over the past 6 years.
- Embodies 30 person-years effort
Overview

- Where have OO Methods come from?
- Where are OO Methods going?
- Beyond Methods.........   to a Process Architecture approach to OO development
- Context & Background to OPEN and MeNTOR?
- A Brief Tour of OPEN/MeNTOR
  - Fundamentals of OPEN/MeNTOR
  - Software Engineering Process
  - Software Engineering Process Architecture
  - Summary
- Deploying an OO Process
- Summary
Overview

 куди have OO Methods come from?
 куди Where are OO Methods going?
 куди Beyond Methods.......... to a Process Architecture approach to OO development
 куди Context & Background to OPEN and MeNtOR?
 куди A Brief Tour of OPEN/MeNtOR
  куди Fundamentals of OPEN/MeNtOR
  куди Software Engineering Process
  куди Software Engineering Process Architecture
  куди Summary

 куди Deploying an OO Process

 куди Summary
Software Engineering Process

SEP is a time sequenced set of activities which transform a user’s requirements into software

= Method

Provides a tested and well defined approach to developing object-oriented software systems
An OPEN/MeNtOR SEP

Project initiation

Requirements elicitation

Model refinement

Project planning

Evolutionary development: OOA, OOD, OOP, Testing

User review

Consolidation

Programme planning

Resource planning

Domain modelling

Other projects

Build

Evaluation

Implementation planning

Bug fixing

Use of system

Use of system
Mentor’s
Software Engineering Process

Programme Lifecycle

Overall Programme Definition

Development Programme

Project 1

Project 2

Project 3

Post Programme Review

Product Lifecycle

Major Software Release

Minor Software Release

Decommissioning

Patch Software Release

Patch Software Release

Project Lifecycle

Project Initiation

Business Investigation

System Definition

Development

System Acceptance

Deployment and Review

Overall Programme Definition

Development Programme

Project 1

Project 2

Project 3

Post Programme Review

Major Software Release

Minor Software Release

Decommissioning

Patch Software Release

Patch Software Release

Project Initiation

Business Investigation

System Definition

Development

System Acceptance

Deployment and Review
Mentor’s System Definition Phase

System Definition

Problem Definition
System Qualification
System Finalisation

Project Initiation
Business Investigation

Development
System Acceptance
Deployment & Review

Process Legend:
SLM - Software Lifecycle Modelling
RM - Requirements Modelling
SM - System Modelling
UI - User Interface Modelling
PM - Project Management
QA - Quality Assurance
AE - Alternative Evaluation
RpM - Repository Modelling
PT - Prototyping
AT - Acceptance Testing
PD - Programme Development

Object Oriented Pty Ltd
Version 1.0: May, 1996

OPEN Consortium
Slide 44
Seamlessness

Object Model Sequence

- Business Knowledge
  - Business object identification
  - Task Object Model
- Systems Knowledge
  - Language mapping
  - System Object Model
- Implementation Object Model
  - Requirements capture
  - Logical design
  - Analysis
  - Physical design
  - World
  - System
The leap from world model to the system model

The Rubicon
Overview

- Where have OO Methods come from?
- Where are OO Methods going?
- Beyond Methods........ to a Process Architecture approach to OO development
- Context & Background to OPEN and MeNtOR?
- A Brief Tour of OPEN/MeNtOR
  - Fundamentals of OPEN/MeNtOR
  - Software Engineering Process
    - Software Engineering Process Architecture
  - Summary
- Deploying an OO Process
- Summary
Software Engineering Process Architecture

- Defines the *constructs* that combine to form a Software Engineering Process:
  - Reusable *Process Units*
  - Software *Lifecycle Models*

- Defines the *rules* that govern how these constructs may be combined
The Big Picture

Software Engineering Process Architecture (Organisation-Level)

used to produce

Software Engineering Process 1 (Project 1)
M-SEP1

Software Engineering Process 2 (Project 2)

Software Engineering Process 3 (Project 3)

Mentor v2.0
A Process Unit

- Defines a set of related *activities* which are performed during a project
- Defines the *inputs* to generate the *outputs* (called *deliverables*) through the use of a series of *activities*
Mentor’s Process Units

- Acceptance Testing
- Alternatives Evaluation
- Component Modelling
- Concept Exploration
- Installation
- Programme Development
- Project Management
- Prototyping
- Quality Assurance
- Requirements Modelling
- System Modelling
- Subsystem Modelling
- Repository Design
- Post Deployment Review
- User Interface Modelling
Processes, Activities, Tasks & Techniques

- Process Unit
  - Activity 1
    - Task 1
      - Technique 1
    - Technique 2
  - Activity 2
    - Task 2
      - Technique 1
      - Technique 2
Process Units Have Activities

Diagram: System Modelling
- Architectural Analysis uses
- Architectural Design uses
- Solution Domain Modelling uses
- Development Planning uses
- Subsystem and Component Specification uses

- System Architect
- System Modeller
- Mentor
Activities

Each activity is defined in terms of a series of tasks which are the smallest unit of work subject to management accountability.

Tasks are accomplished by the use of “well-known” OOA&D techniques, such as:

- scenario analysis
- CRC carding
- object and class modelling
- and many more..
Activities Have Tasks

Solution Domain Modelling

- Identify Candidate Classes
- Undertake a CRC Walkthrough
- Develop Class Specifications
- Establish Relationships
- Develop System Scenarios
- Identify Subsystem Structure

System Modeller

uses

uses

uses

uses

uses

uses
Activities Produce Deliverables

Acceptance Testing

- Test Model Creation
- Test Environment Establishment
- Test Planning
- System Testing
- Test Execution
- User Testing
- Requirements Model
- System Model
- Acceptance Test Model

Development Team Representative
Test Manager
User Representative
Tester

User Representative
Test Manager
Development Team Representative

Object Oriented Pty Ltd
Version 1.0: May, 1996

OPEN Consortium
Slide 56
Activities

- Pre and post-conditions are part of the contract
- Deliverables and testing are part of post-condition
- Flexibility -- tailor able process
### Activities and Tasks

Tasks say what is to be done

<table>
<thead>
<tr>
<th>Activities</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For each activity/task combination we will recommend five levels of probability from Always to Never
Examples of Tasks

- Create and/or identify reusable components
- Design and implement physical database
- Design user interface
- Develop and implement resource allocation plan
- Evaluate quality
- Identify user requirements
- Map roles onto classes
- Test
- Undertake feasibility study
- Write manual(s)
Tasks say what is to be done in order to satisfy post-condition of each Activity; they do not say HOW the Task can be accomplished.

This is the role of the technique. The developer chooses their own toolbox of techniques from the wide range (well over 150) provided -- part of the tailoring process.
## Tasks and Techniques

Techniques suggest how it is to be done

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Tasks</th>
<th>Tasks</th>
<th>Tasks</th>
<th>Tasks</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For each task/technique combination we will recommend five levels of probability from Always to Never.
Examples of Techniques

- Context modelling (BPR)
- DBMS selection
- DCS architecture specfn
- Hierarchical task analysis
- Metrics collection
- Power types
- Project planning
- Role modelling
- Rule modelling
- System event modelling
Deliverables Have Release States
Mentor Deliverables are Housed in Workbooks

- Programme Workbook
- Project Workbook
- Subsystem Workbook
- Developers Workbook
- Requirements Workbook
- Component Workbook
- Prototype Workbook
- System Workbook
- Test Workbook
**Scenario:** Reply to a Message  
**Index Number:** BS002  
**Description:** Allows the user to reply to a message  
**Actors:** Mail User  
**Authors:** C. Brown  
**Preconditions:** None  
**Scenario Text:**  
1. Read Message  
2. Create a reply  
*uses:* Compose a Message  
3. Send the reply  
**Alternatives:** None  
**Extends:** None  
**Cross Validation:** None  
**Maturity Level:** Red  
**Questions/Notes:** Needs more detail  
**Modification:** First version 23/05/95

**Scenario:** Compose a Message  
**Index Number:** BS003  
**Description:** Allows the user to create a message  
**Actors:** Mail User  
**Authors:** C. Brown  
**Preconditions:** None  
**Scenario Text:**  
1. Record the response  
2. Indicate that the response has been recorded  
**Alternatives:** None  
**Extends:** None  
**Cross Validation:** None  
**Maturity Level:** Red  
**Questions/Notes:** Needs more detail  
**Modification:** First version 23/05/95
Processes are Interactive

User Interface Modelling

- Design an Interface
- Develop Guidelines & Standards
- Storyboarding

Requirements Modeller

UI Modeller

RM
Requirements Modelling

PT
Proof of Concept

SM
System Modelling
Software Lifecycle Models

- Are a *framework* that specifies the way in which a project may be run
- Are a general description, or template for software projects
- Provides structure to the Software Engineering Process through:
  - Phases
  - Stages
  - Milestones - internal and external
Software Lifecycle Model

Examples include:

- Waterfall
- Iterative
- Incremental
- Incremental/Iterative
- Spiral
- Fountain
Phases and Milestones

External Milestones (EM) (Project Reviews)

EM1  EM2  EM3  EM4

Phase 1  Phase 2  Phase 3  Phase 4

Internal Milestones (IM)
Phases, Stages and Processes

Phase X

Software Process 1
Software Process 2
Software Process 3

Stage 1
Stage 2
Stage 3

Release state 1
Release state 2
Release state 3
External Milestone
Constructing a Software Engineering Process

Software Lifecycle

Phase

Stage

RM
SM
PD
CM

AT
IN
Processes Support Many Configurations

System Definition Phase

Problem Definition

System Qualification

Development Phase

System Finalisation

Component Development

Scenario 1

Scenario 2

Scenario 3
OPEN and MeNtOR’s Software Engineering Process Architecture

 méthodology

Ability to develop additional SEPs optimal for individual project types

Provides an organisation with flexibility and consistency
Overview

✔ Where have OO Methods come from?
✔ Where are OO Methods going?
✔ Beyond Methods......... to a Process Architecture approach to OO development
✔ Context & Background to OPEN and MeNtOR?
✔ A Brief Tour of OPEN/MeNtOR
  ✔ Fundamentals of OPEN/MeNtOR
  ✔ Software Engineering Process
  ✔ Software Engineering Process Architecture
  ✔ Summary
✔ Deploying an OO Process
✔ Summary
Summary

OPEN and MeNtOR provide
- a complete and disciplined process for object-oriented software engineering
- consistency with flexibility
- Promotes Engineering of the Software Process
**OPEN Support**

- **Books and papers (precursor methods):**
  - OPEN/SOMA: Migrating to Object Technology (I.M. Graham), Addison Wesley, 1995
  - OPEN/Firesmith: Object-Oriented Requirements Engineering and Logical Design, Wiley, 1993
OPEN Support

Books and papers:

- The OPEN-MeNtOR methodology, Object Magazine, Nov 1996, 6(9), 56-59
- OPEN project management, ObjectExpert, Jan/Feb 1997, 2(2), 30-35
- The OPEN Process Specification, Addison-Wesley, July 1997, in press
- OPEN’s Toolbox of Techniques, Addison-Wesley, September 1997 (approx).
OPEN Support

CASE tools

ObjectMaker, Simply Objects, SOMATiK plus (likely soon) MetaEdit, Graphical Designer, LBMS, Paradigm Plus

Training worldwide from several third-party companies e.g.

Tower, KSC, Thomsen Due, FourFront, COTAR, Vayda, OIG, Genesis
MeNtOR Support

- Object Oriented Pty Ltd
  - Formal Training
  - Workshops
  - Process and Design Consulting
  - Online Manuals & Templates

- CASE tools
  - LBMS, StP, Simply Objects
  - Paradigm Plus, Object Team...
“Open” Research Issues

- Incorporation of Formal techniques (Object Z/FOOM) [Swinburne and UQ]
- Screen layout algorithms [Newcastle University]
- Reverse engineering [Monash University]
- Empirical testing [Dow Jones Telerate]
- Requirements Engineering [Discovery/Simons UK, IDIS/Winder UK]
Discovery: Task Modelling

- Interviews
- Requirements Document
- Task Model
- Form Contract
- Narrative Model
- Linguistic Analysis
- Vocabulary
- Task Scripts
- Task Priority

Linguistic Analysis

- Interviews
- Requirements Document
- Task Model
- Form Contract
- Narrative Model
- Linguistic Analysis
- Vocabulary
- Task Scripts
- Task Priority
Discovery: Object Modelling

- Vocabulary
- Task Scripts
- Task Priority
- Storage Analysis
- Control Analysis
- Behaviour Analysis
- Control Model
- Data Model
- Responsibility Cards
- Design Patterns
- Delegation Analysis
- Interaction Model
Discovery: System Modelling

- Design Patterns
- Responsibility Cards
- System Layering
- Collaboration Graph
- Library Review
- Reusable Frameworks
- Reusable Components
- Application Scavenging
- Coupling Analysis
- Interaction Model
- Delegation Analysis
Discovery: Language Modelling
Overview

- Where have OO Methods come from?
- Where are OO Methods going?
- Beyond Methods ........... to a Process Architecture approach to OO development
- Context & Background to OPEN and MeNtOR?
- A Brief Tour of OPEN/MeNtOR
  - Fundamentals of OPEN/MeNtOR
  - Software Engineering Process
  - Software Engineering Process Architecture
- Summary

Deploying an OO Process

Summary
Deployment Process

The deployment of an OO method involves a number of phases and should be treated as any other managed project.

Phases include:
- Selection
- Project Establishment
- Project Driving
- Review and Improvement
Selection Phase

- Review the process:
  - Identifying if it is a *Technique, Method or Process*
  - Exploring the coverage of the *concepts*
  - Exploring the form of *representation* supported
  - Exploring the coverage of the *lifecycle requirements*
  - Identifying how it deals with *pragmatic aspects* of undertaking projects
  - Identifying the *CASE tools* supported
  - Identifying the *languages* supported
Selection Phase (cont)

**Review:**
- The availability of *training* services
- The availability of *mentoring* services
- The *CMM level* for which you wish to be certified (Processes and Methodologies only)
- The requirements of the *Quality Management System* that you currently have in place
Project Establishment Phase

- The Project Establishment phase is where the initial setting up and planning occurs

- Activities of the phase include:
  - Install the process material
  - Undertake project planning sessions
    - Define SEP
    - Undertake an initial tailoring/integration
  - Formal Training
    - 1-3 weeks
    - concepts, languages, processes
  - Process automation
    - tool selection
    - tool integration
Project Establishment Phase (cont)

Identify a Software Process Group or Core Team

- “Own and maintain the method”
- Act as mentors to future team members
- Be committed to process improvement

Train and mentor this team on:

- object oriented techniques
- process improvement techniques
Project Driving Phase

 '('The Project Driving phase is where the project proper starts and the team applies the process ')' Activities of the phase include:

 - Mentoring
 - Workshops
 - Process Refinement
 - Reviews

('Ensure correct application of process')
Review and Improvement Phase

The Review and Improvement phase is where the project is reviewed regularly to ensure the process is being applied and to document any improvements.

Activities

- Project Reviews
  - Assisting with the “finer points” of process
- Process Audits
  - Reports on the application of the process and areas for improvements and refinement
- Post Project Review
  - Lessons learnt and recommendations made
Some Key Lessons Learnt

_successful_ Process Deployment Requires:

- Consistent Application
- A Pragmatic Approach
- Proactive Management
- One Message
- Visible Results and Feedback Loops
- Strong Leadership Based on Experience
- A Desire to Change
- Medium Term Investment Horizon
Overview

- Where have OO Methods come from?
- Where are OO Methods going?
- Beyond Methods......... to a Process Architecture approach to OO development
- Context & Background to OPEN and MeNtOR?
- A Brief Tour of OPEN/MeNtOR
  - Fundamentals of OPEN/MeNtOR
  - Software Engineering Process
  - Software Engineering Process Architecture
  - Summary
- Deploying an OO Process

Summary
Summary

✦ Hope to have provided you with
  ✦ An overview of where OO methods have come from and where they are going
  ✦ An overview of the OPEN/MeNTOR object oriented process architecture
  ✦ A possible deployment strategy for process in your organisation

✦ Process is a critical element for effective software engineering - software engineering with object technology demands an investment in an OO software process
Contacts

Brian Henderson-Sellers
Director COTAR
School of Computer Science and Software Engineering
Swinburne University of Technology
Hawthorn, Victoria 3122
brian@csse.swin.edu.au


Julian Edwards
Software Process Group
Object Oriented Pty Ltd
PO Box 528,
North Sydney, NSW, 2089
j.edwards@oopl.com.au
mentor@oopl.com.au