

Module OOP5 OPEN's Activities (plus summary of Tasks)

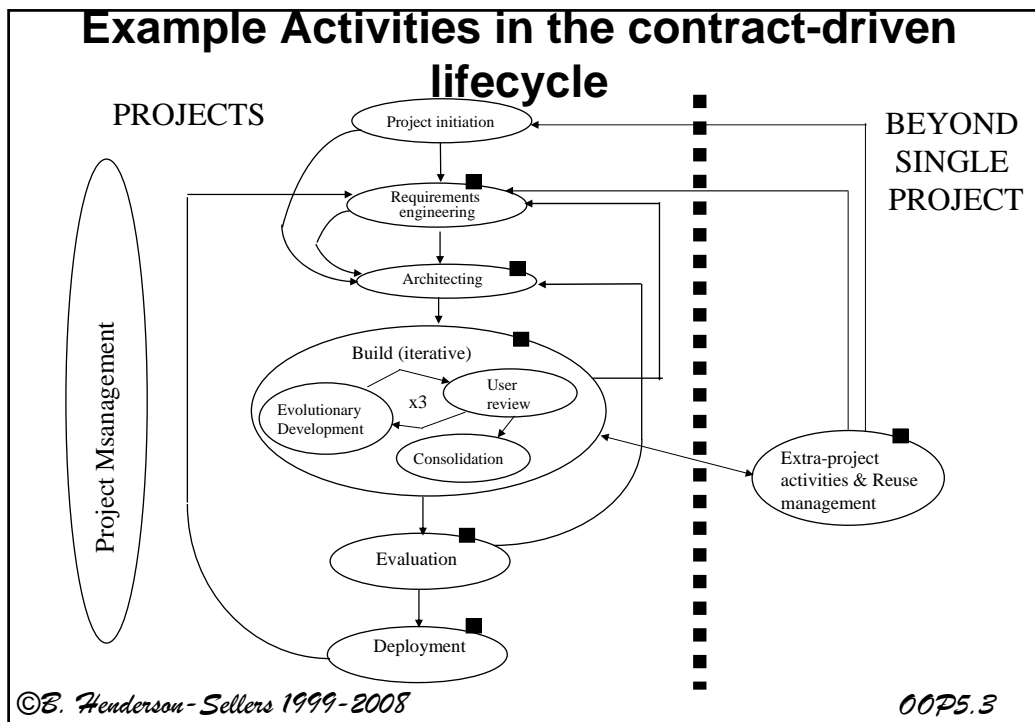
©B. Henderson-Sellers 1999-2008

OOP5.1

- Activities represent largescale, heterogeneous goals - things to be done
- Tasks represent smallscale, homogeneous goals - commensurate with project management

©B. Henderson-Sellers 1999-2008

OOP5.2



Project Activities in OPEN may include, for example:

- Project initiation
- Requirements engineering
- Architecting
- Build
- Evaluation
- Deployment
- Project management

Project initiation

- Business planning
- Organizational strategy
- Business proposal made leading to need for requirements engineering

Requirements engineering

- Identifying source(s) of requirements
- Gathering requirements (elicitation)
- RAD workshop may be useful
- One on one discussions/interviews
- Identify stakeholders, internal actors, support systems and largescale functionality required of system
- Commence building Business Object Model

- Complexity added by culture of organization and people in it.
- Possible use of Object-Z and Soft Systems methodologies or RAD
- Identification of task scripts/use cases

Architecting

- produce a formally documented consensus between stakeholders regarding overall structure and major mechanisms
- maximize quality of the architecture
- maximize productivity of architecture teams

Build

- The “Modelling Process”
- Design technical/system object model and system build
- Evolutionary Development: OOADP + User Review + Consolidation
 - develop object model structure incrementally (plan, URS, OOAD, OOP, review)

OOAD

- combination of analysis and invention
- identify CIRTs (“objects” in system)
- identify responsibilities, services and associated business rules (pre and post conditions)

- identify CIRT interactions
- use abstraction techniques
- identify behaviour
- model “dynamic” behaviour (STDs, event diagrams)
- encompass problem domain and solution domain

- identify generalization
 - after first iteration
 - possible future reusable classes
- integrate library classes
- design class internals
- continual refinement of model(s)
- create subsystems/packages
- evaluate subsystem quality
- testing (against scenarios)

Implementation (OOP)

Switch from external (interface) to internal (code) view

- internal algorithmic design
- look for places where existing patterns will assist
- hide all “properties” now physically stored as data
- structured design of algorithms (5 SLOCs typical of good OO code)

- OOPL-specific details
- detailed inheritance structures
- use existing library classes
- code- style important
- code testing
- possible componentization

User Review

- package/subsystem/component testing
- evaluation of documentation
- assessment of progress towards objectives
- assessment of quality
- evaluate reuse potential

Review undertaken by peers, management and users

Consolidation

occurs after a number of OOAD/P/Testing iterations

- cf. previous timeboxes
- consider potential reuse
- metrics and usability assessments OK?
- documentation high quality?
- conversion plans satisfactory?

Evaluation

- Defect analysis
- Comparison of result with intention (formal review)
- Inspections (Fagan or Gilb and Graham)
- Reuse possibilities

Deployment

- Change management
- Hardware and software resources
- Environment and locations
- Education of users
- Technical and user support

Project management

- covers a wide variety of issues including project planning, contract management, customer relationship management, human resource management, schedule management and policy establishment

There are other Activities that may be useful for development (but not discussed further here)

- Configuration management
- Risk management
- Metrics engineering
- Quality engineering
- Integration
- Training
- Process engineering
- Environment engineering
- Website development
- Design
- Component selection
- Testing

Other Activities focus on the programme not the project

- Programme planning
- Resource planning
- Domain modelling
- Reuse engineering (incl. possible role of outsourcing/COTS)

Programme Planning

- Consider interaction between projects
- Resources to be balanced across all department
- Concurrent and distribution issues

Resource Planning

Resources = time + money + people + scope +
tools & techniques - in the context of quality

- role of reuse
- metrics
- units of work (tasks)
- deliverables and milestones
- team structure
- education and training plan

Domain Modelling

- Extensions beyond single project (reuse focus)
- Basis of BPR efforts
- Repository management
- Use of existing components

Reuse Engineering

- Strategy for reuse
- Costs and amortization
- Create a reuse mindset
- IP issues for consulting companies
- Impact of frameworks and components on reuse strategy

NOW

Maintenance = enhancement

- Error correction vs. evaluation
- New requirements, minor changes
- Minimal impact at class or subsystem level
- Close relationship between developers and users leads to higher quality systems which require less maintenance
- Metrics collection important

OPEN Tasks

Task is a unit of work commensurate with
good project management

Tasks can be loosely grouped in

- project management
- modelling
- reuse engineering
- metrics management
- component engineering

These will be discussed in detail in Modules P6, P7,
P8, P9 and P10 respectively

... and

**There are other task groupings not
discussed in this subject**
(for details see App. E of course text)

These are

- architecture
- business modelling
- configuration management
- database-related
- deployment
- design
- environment engineering
- implementation
- integration

... and

- process engineering
- programme management
- quality engineering
- requirements eng.
- risk management
- testing
- training
- transition
- user interface
- website management

SUMMARY

- Contract-driven and tailorable lifecycle
- Activities as objects in the lifecycle (across technical and management domains)
- Activities have associated Tasks, which are actioned in turn by Techniques