

3C05: Software Development Team Structures

© 2001, Zuhke Engineering

1

Unit 6: SW Development Team Structures

Objective:

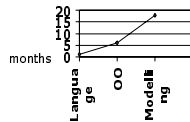
- To discuss the different roles involved in large-scale software engineering projects
- To show the qualifications and capabilities for team members adopting these roles
- To review how teams are composed and projects are staffed.

© 2001, Zuhke Engineering

2

Creating an OO Team

- OO Learning Curve
 - 1 month to learn language syntax
 - 6 to 9 months to become proficient in OO
 - 12 to 18 months to become moderately proficient in modelling and methodology



© 2001, Zuhke Engineering

3

How to Kick-Start an OO Project

- Mentoring!
- Seed project with experienced people
- External/internal consultants at key stages
 - Planning
 - Project start up
 - Regular design and code reviews
 - Post-project review



© 2001, Zuhke Engineering

4

Case Study

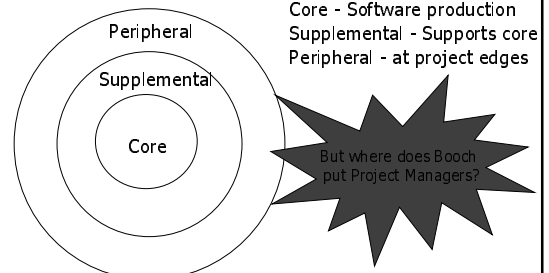
- A company took a group of non OO programmers and over a period of one month trained them in C++ and an OO methodology. They then launched them straight into a full-blown OO Project. Naturally the Project failed badly. How did this happen? Management did not understand that object technology is different to conventional software development.

© 2001, Zuhke Engineering

5

The OO Project Team

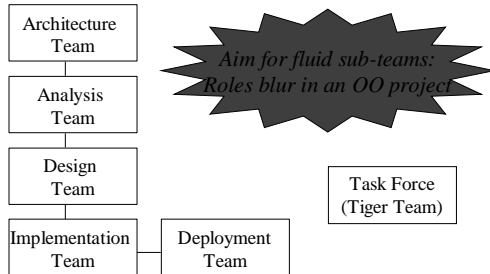
According to Booch!



© 2001, Zuhke Engineering

6

Sub-teams



© 2001, Zuhke Engineering

7

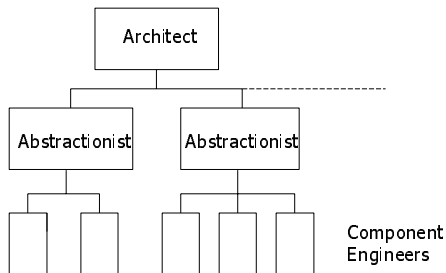
Abstractionists

- To simplify our discussion, we introduce the role of *Abstractionist*
- Abstractionists embrace the following USDP roles:
 - Use-Case Specifier
 - Use-Case Engineer
 - System Analyst
 - System Integrator
- These USDP roles are often done by the same person anyway!

© 2001, Zuhke Engineering

8

Core Team: Structure

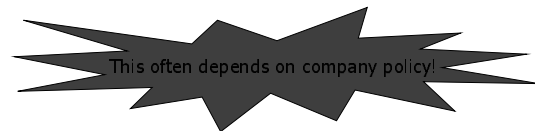


© 2001, Zuhke Engineering

9

What about testing?

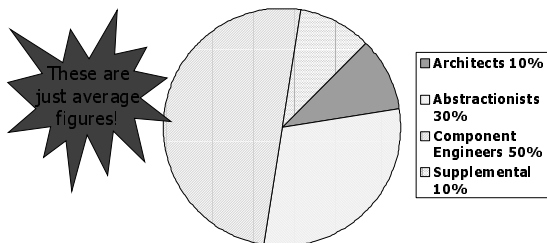
- Testers can be a member of an Abstractionists team (just like a Component Engineer)
- Testers may belong to a separate Test Team



© 2001, Zuhke Engineering

10

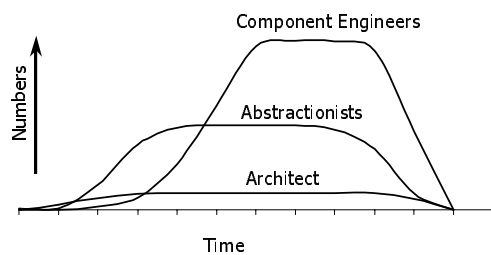
Staffing



© 2001, Zuhke Engineering

11

Staffing Profiles



© 2001, Zuhke Engineering

12

Core Team: Roles

- Architect
 - System architecture and vision
- Abstractionist
 - Micro-architectures
 - One Abstractionist per class package
- Component Engineer (programmer)
 - Implementing abstractions

© 2001, Zuhke Engineering

13

Architect: Responsibilities

- System Architecture
- Assess technical risks
- Define content of successive iterations
 - Help in planning
- Consultancy
- Marketing
 - Future product definition

© 2001, Zuhke Engineering

14

Architect: Skills

- Experience
 - Problem domain
 - Software engineering
- Vision
- Leadership
- Communication
- Proactive and goal-oriented
- Risk taker

© 2001, Zuhke Engineering

15

Abstractionist: Responsibilities

- Identify classes, packages, subsystems, mechanisms, frameworks
- Define interfaces
- Direct implementation and (possibly) testing
- Advise and support the Architect
- Mentor and lead Component Engineers

© 2001, Zuhke Engineering

16

Abstractionist: Skills

- Experience
 - Must know how to find abstractions
 - Strong programming skills
- Leadership
 - Ability to manage a small team of developers
- Communication
 - Able to express complex ideas simply
- Proactive and goal-oriented

© 2001, Zuhke Engineering

17

Component Engineer: Responsibilities

- Implement scenarios, mechanisms and classes
- Tactical class design
- Class-level testing
- Advise abstractionist about tactical risk
- Participate in Task Forces and code walkthroughs

© 2001, Zuhke Engineering

18

Component Engineer: Skills

- Good coding skills and likes to code!
- Perhaps has specialisations e.g. GUI
- Familiar with OOA/OOD principles

Myth of the replaceable programmer

- Some Project Managers view programmers as the "lowest form of life". They are just replaceable parts
- This ignores the fact that a good programmer may be up to 10 times more productive than a bad programmer
- Good programmers are very valuable and need to be encouraged and rewarded

OO as an Amplifier

- Object orientation acts like an amplifier - it makes the best programmers much better, and the worse programmers much worse!
- The same is true for Abstractionists !

The Supplemental Team

- Project Manager
- Integrator
- Quality Assurance Engineer
- Documentor
- Toolsmith
- System Administrator
- Librarian

Project Manager: Responsibilities

- Oversee the Project's deliverables
- Establish and drive schedules
- Staffing
- Work break down
- Budgeting
- Co-ordinate with patrons and user community

Project Manager: Skills

- Experience
 - Leadership
 - Proactive
 - Goal oriented
 - Communication
- Pragmatic
- Risk-averse
- Politically aware

The Peripheral Team

- Patron
 - Champions the Project
- Product Manager
 - Manages a product line
 - Manages marketing, training, support
- End user
 - Client of the Project
- Technical support

Key Points

- The key to successful operation of the USDP or any other OO lifecycle is to organise into small flexible teams
- There should be a "chain of responsibility" and continuity of ownership for artefacts from requirements down to code
- A good policy is to give responsibility for whole chains to individual teams