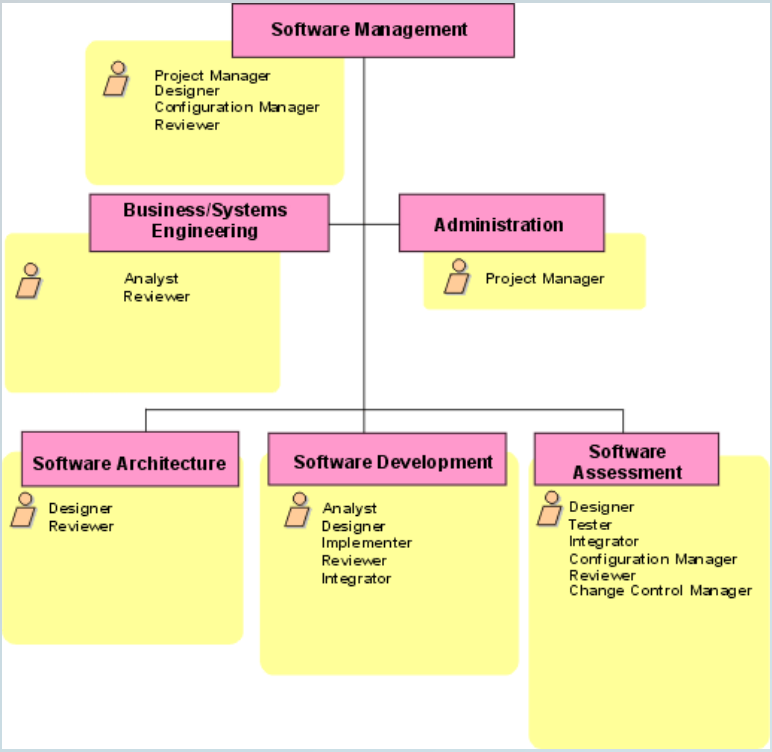
# Team Structures for Software Development Teams

By Tanya Smeltzer

# How do you determine how to structure your team?

- Company needs
  - Will an outside consult be necessary to meet the requirements?
- Software development people available
  - Different levels of skill and experience
- Type of application
  - Is this a large or a small application? How many use cases need to be addressed?

# Default Responsibilities




Suggested organization by UPEDU, as discussed in *Software Project Management: A Unified Framework.* (Walker Royce 1998)

- Shows how project level roles and responsibilities should be assigned to a structure of teams
  - Items shown in yellow boxes do not represent people, rather roles particular teams can fill
  - Size and nature of project will affect how roles are distributed
- More specific decisions on team structures should be guided by some important rules

# Rules (1)

- Small teams typically better (but not too small)
  - More productive
  - Mythical Man Month
    - Remember, adding more people adds more paths of communication
    - Team size in large projects must be balanced against the amount of cross-team interaction to avoid too many communication paths

# Rules (2)

- Avoid deep hierarchies
  - Can lead to longer decision making
    - Require more approval from above
    - Delays cut into development time
  - Compromises communication
    - Too many levels  to communicate decisions and ideas to
    - Individuals high in the hierarchy  may lack the knowledge necessary to make the correct decision

# Rules (3)

- The span of control of any manager or team lead should be limited to 5-9 people
  - Fewer paths of communication
  - All ideas/input more likely to be heard
  - Progress easier to track

# Rules (4)

- Team structure should be driven by the software architecture (not vice versa)
  - Good architecture allows teams to work more effectively in parallel (high cohesion and low coupling between subsystems)
  - Assigning team members to work on different parts of an application without first knowing how they are related can produce unexpected delays or, worse, software that does not meet the requirements

# Rules (5)

- Testing should ideally be performed by a team separate from the development team
  - Other users are more likely to encounter bugs from incorrect use of the software
  - Unit testing should still be performed by the development team
  - May not make economic sense in a very small project

# Rules (6)

- Authorities and responsibilities must be clearly defined.
  - Managers and team leads in the middle should understand their required role in balancing technical and managerial activities.
  - Do the development teams have the authority to change the underlying structure without further approval? How will this affect other components of the application?

# Rules (7)

- Experience and capabilities are important in assigning responsibilities
  - Teams should be structured for competence in the responsibilities they are assigned
  - A good designer is not necessarily a good developer! If a team is responsible for both, ensure they can do both well.

# Rules (8)

- Team structures should not be rigid
  - Individuals will move between teams over the project's lifetime
  - Team responsibilities will change with the project emphasis from one phase to the next phase
  - This is arguably a bad idea (but more on that later)

# Let’s take a closer look…

So far we’ve been looking at the structure of multiple teams. What about the structure on an individual team level?

- Vertical vs. Horizontal Team Organization
- Agile Approach vs. Traditional Organization
- Agile Team Roles

# Terms

## Specialists

- Those who know a lot about a narrow domain

## Generalists

- Those who know a little about a wide range of topics

## Generalizing Specialists

- Has one or more technical specialties, a general knowledge of software development and the business domain in which they work, and actively seeks to gain new technical and domain area skills in their existing specialties and in other areas

# Vertical Team Organization

- All generalists
- Use cases assigned to individuals or small groups and implemented end-to-end
- Advantages
  - Smooth development on individual use case basis
  - Developers gain wide range of skills
- Disadvantages
  - Typically requires difficult to find and high-paid consultants
  - Typically do not have specific technical expertise required to quickly solve detailed problems
  - May require subject matter experts to work with several groups of developers

# Horizontal Team Organization

- All specialists
- Works on several use cases at once, team members focusing on specific tasks
- Advantages
  - Higher quality of work for each part of the software life cycle
  - Users interact with a small group of specialists who understand their exact needs
- Disadvantages
  - Information required by "back-end" people may not be gathered by the "front-end" people due to the focus on different specialties
  - Difficult to manage (competing priorities)

# The Middle Ground

- Both specialists and generalists
- Generalists focus on single use cases throughout development; specialists work on tasks of various use cases
- Advantages
  - Outside groups interact with a small group of experts
  - Specialists can still focus on their areas of expertise
  - Individual use cases are implemented consistently
- Disadvantages
  - Difficult to manage
  - Generalists are difficult to find

# The Agile Approach

So what does the agile approach aim to do differently from the traditional approach?

- Form teams mostly of generalizing specialists
  - Accomplish a wide range of tasks with consistency
- Form teams with all skills necessary
  - No need to hire outside experts
- Keep teams stable
  - Don’t move people from one team to another for different iterations
  - Consistency is important for progress

# Agile Roles

- Team coach
  - Responsible for facilitating the team, obtaining resources, and protecting team from problems.
- Developer
  - Responsible for the application creation and delivery.  This includes modeling, programming, testing, and release activities.
- Product owner
  - Represents the stakeholders. Responsible for taking care of the prioritized stack of requirements (or product backlog), making decisions in a timely manner, and for providing information in a timely manner.

# Farewell to Traditional Roles

Through use of the agile approach, the traditional need for project managers and business analysts is removed

- Leadership activities of project managers are taken on by the team coach
- Many technical skills previously performed by project managers are taken on by the team through self organization
- The role of business analysts is replaced by the skills of the product owner

# Agile Roles (II)

- Stakeholder
  - Anyone who is potentially affected by the development and/or deployment of a software project.
- Independent tester (optional)
  - Member of a separate team which works in parallel to the development team and validates their work throughout the lifecycle.
- Technical experts
  - Sometimes brought in on a temporary basis to help the team with a difficult problem and to transfer their skills to one or more developers on the team.
- Domain experts
  - Sometimes brought in to work with the team by explaining domain specific details of a requirement

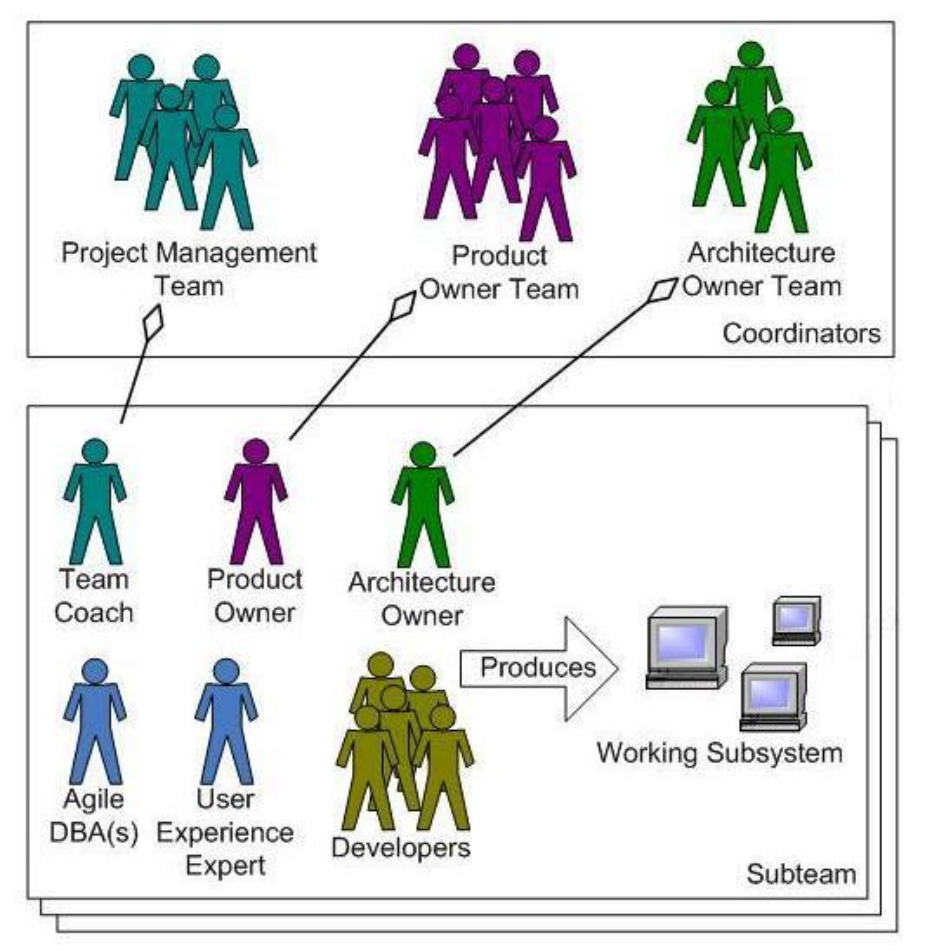
# What about larger projects?

- Multiple teams, or sub-teams, are created to focus on one or more sub-systems of the overall application
- These teams consist of the same roles as smaller agile teams, as well as a couple of additional roles
- Three coordinating teams are created
  - Composed of product owners, team coaches, and architecture owners from the sub-teams
  - Coordinate project management, technical and architectural issues, requirement and product ownership issues, and system integration

# The Additional Roles

- Architecture owner
  - Responsible for facilitating architectural decisions on a sub-team
  - Do not have sole responsibility of creating the architecture
  - Part of the architecture owner team (responsible for overall architectural direction of the project)
- Integrator
  - Responsible for building the entire system from its subsystems
  - Can be one or more people in this role

# So what does this look like?




As integrators, independent testers, and technical and domain experts are not part of the main teams, they are not shown here.

# Questions?

# Resources


Unified Process for Education (UPEDU). http://www.yoopeedoo.org/upedu/

Scott Ambler, “Web services programming tips and tricks: How to organize a software development team,” November 2000. [Online]
http://www.ibm.com/developerworks/webservices/library/ws-tip-team.html

R. Srinivasan, “Well-defined hierarchy speeds up decision making,” May 2002.
http://www.icmgworld.com/corp/news/Articles/RS/may_0202.asp

Scott Ambler, “Roles on Agile Teams: From Small to Large Teams,” 2005-2009.
http://www.ambysoft.com/essays/agileRoles.html