CHAPTER 4

TEAMS

Overview

• Team organization
• Democratic team approach
• Chief programmer team approach
• Beyond chief programmer and democratic teams
• Synchronize-and-stabilize teams
• Teams for agile processes
• Open-source programming teams
• People capability maturity model
• Choosing an appropriate team organization

4.1 Team Organization

• A product must be completed within 3 months, but 1 person-year of programming is still needed

  • Solution:
    - If one programmer can code the product in 1 year, four programmers can do it in 3 months
  
  • Nonsense!
    - Four programmers will probably take nearly a year
    - The quality of the product is usually lower

Task Sharing

• If one farm hand can pick a strawberry field in 10 days, ten farm hands can pick the same strawberry field in 1 day

• One elephant can produce a calf in 22 months, but 22 elephants cannot possibly produce that calf in 1 month

Task Sharing (contd)

• Unlike elephant production, it is possible to share coding tasks between members of a team

• Unlike strawberry picking, team members must interact in a meaningful and effective way
Programming Team Organization

- Example:
  - Sheila and Harry code two modules, m1 and m2, say
  - What can go wrong
    - Both Sheila and Harry may code m1, and ignore m2
    - Sheila may code m1, Harry may code m2. When m1 calls m2 it passes 4 parameters; but m2 requires 5 parameters
    - Or, the order of parameters in m1 and m2 may be different
    - Or, the order may be same, but the data types may be slightly different

Programming Team Organization (contd)

- This has nothing whatsoever to do with technical competency
  - Team organization is a managerial issue

Communications Problems

- Example
  - There are three channels of communication between the three programmers working on a project. The deadline is rapidly approaching but the code is not nearly complete
  - “Obvious” solution:
    - Add a fourth programmer to the team

Communications Problems (contd)

- But other three have to explain in detail
  - What has been accomplished
  - What is still incomplete

- Brooks’s Law
  - Adding additional programming personnel to a team when a product is late has the effect of making the product even later

Team Organization

- Teams are used throughout the software production process
  - But especially during implementation
  - Here, the discussion is presented within the context of programming teams

- Two extreme approaches to team organization
  - Democratic teams (Weinberg, 1971)
  - Chief programmer teams (Brooks, 1971; Baker, 1972)

4.2 Democratic Team Approach

- Basic underlying concept — egoless programming

- Programmers can be highly attached to their code
  - They even name their modules after themselves
  - They see their modules as extension of themselves
Democratic Team Approach (contd)

- If a programmer sees a module as an extension of his/her ego, he/she is not going to try to find all the errors in “his/her” code.
  - If there is a fault, it is termed a bug.
  - The fault could have been prevented if the code had been better guarded against the “bug.”
  - “Shoo-Bug” aerosol spray

Proposed solution

- Egoless programming
  - Restructure the social environment
  - Restructure programmers’ values
  - Encourage team members to find faults in code
  - A fault must be considered a normal and accepted event
  - The team as a whole will develop an ethos, a group identity
  - Modules will “belong” to the team as a whole
  - A group of up to 10 egoless programmers constitutes a democratic team

Difficulties with Democratic Team Approach

- Management may have difficulties
  - Democratic teams are hard to introduce into an undemocratic environment
  - Democratic teams have to spring up spontaneously
  - An experienced programmer may resent having his/her code appraised by fellow (unexperienced) programmers

Strengths of Democratic Team Approach

- Democratic teams are enormously productive
- They work best when the problem is difficult
- They function well in a research environment

4.3 Chief Programmer Team Approach

- Consider a 6-person team
  - Fifteen 2-person communication channels
  - The total number of 2-, 3-, 4-, 5-, and 6-person groups is 57
  - This team cannot do 6 person-months of work in 1 month

Chief Programmer Team

- Six programmers, but now only 5 lines of communication

Figure 4.3
Chief Programmer Team (contd)

- The basic idea behind the concept
  - Analogy: chief surgeon directing an operation, assisted by
    - Other surgeons
    - Anesthesiologists
    - Nurses
    - Other experts, such as cardiologists, nephrologists

- Two key aspects
  - Specialization
  - Hierarchy

Chief Programmer Team (contd)

- Chief programmer
  - Successful manager and highly skilled programmer
  - Does the architectural design
  - Allocates coding among the team members
  - Writes the critical (or complex) sections of the code
  - Handles all the interfacing issues
  - Reviews the work of the other team members
  - Is personally responsible for every line of code

Chief Programmer Team (contd)

- Back-up programmer
  - Necessary only because the chief programmer is human
  - The back-up programmer must be in every way as competent as the chief programmer, and
  - Must know as much about the project as the chief programmer
  - The back-up programmer does black-box test case planning and other tasks that are independent of the design process

Chief Programmer Team (contd)

- Programming secretary
  - A highly skilled, well paid, central member of the chief programmer team
  - Responsible for maintaining the program production library (documentation of the project), including:
    - Source code listings
    - JCL
    - Test data
  - Programmers hand their source code to the secretary who is responsible for
    - Conversion to machine-readable form
    - Compilation, linking, loading, execution, and running test cases (this was 1971, remember!)

Chief Programmer Team (contd)

- Programmers
  - Do nothing but program
  - All other aspects are handled by the programming secretary

The New York Times Project

- Chief programmer team concept
  - First used in 1971
  - By IBM
  - To automate the clippings data bank (“morgue”) of the New York Times

- Chief programmer — F. Terry Baker
The New York Times Project (contd)

- 83,000 source lines of code (LOC) were written in 22 calendar months, representing 11 person-years
- After the first year, only the file maintenance system had been written (12,000 LOC)
- Most code was written in the last 6 months
- Only 21 faults were detected in the first 5 weeks of acceptance testing

Why Was the NYT Project Such a Success?

- But, after this fantastic success, no comparable claims for the chief programmer team concept have been made

F. Terry Baker
- Superprogrammer
- Superb manager and leader
- His skills, enthusiasm, and personality "carried" the project

Strengths of the chief programmer team approach
- It works
- Numerous successful projects have used variants of chief programmer teams

Impracticality of Chief Programmer Team Approach

- The chief programmer must be a highly skilled programmer and a successful manager
- There is a shortage of highly skilled programmers
- There is a shortage of successful managers
- The qualities needed to be a highly skilled programmer are unlikely to be found in a successful manager, and vice versa

Why Was the NYT Project Such a Success?

- Prestige project for IBM
  - First real trial for PL/I (developed by IBM)
  - IBM, with superb software experts, used its best people
- Extremely strong technical backup
  - PL/I compiler writers helped the programmers
  - JCL experts assisted with the job control language
4.4 Beyond Chief Programmer and Democratic Teams

- We need ways to organize teams that
  - Make use of the strengths of democratic teams and chief programmer teams, and
  - Can handle teams of 20 (or 120) programmers

- A strength of democratic teams
  - A positive attitude to finding faults

- Use chief programmer teams in conjunction with code walkthroughs or inspections

Beyond Chief Programmer and Democratic Teams (contd)

- Potential pitfall
  - The chief programmer is personally responsible for every line of code
    - He/she must therefore be present at reviews
  - The chief programmer is also the team manager
    - He/she must therefore not be present at reviews!

- It is easier to find a team leader than a chief programmer
- Each employee is responsible to exactly one manager — lines of responsibility are clearly delineated
- The team leader is responsible for only technical management

- Budgetary and legal issues, and performance appraisal are not handled by the team leader
- The team leader participates in reviews — the team manager is not permitted to do so
- The team manager participates in regular team meetings to appraise the technical skills of the team members
Larger Projects

- The nontechnical side is similar
  - For even larger products, add additional layers

Beyond Chief Programmer and Democratic Teams (cont'd)

- Decentralize the decision-making process, where appropriate
  - Useful where the democratic team is good
  - One programmer draws up the test cases, the other tests the code

4.9 Choosing an Appropriate Team Organization

- There is no one solution to the problem of team organization
- The "correct" way depends on
  - The product
  - The outlook of the leaders of the organization
  - Previous experience with various team structures

Choosing an Appropriate Team Organization (cont'd)

- Exceedingly little research has been done on software team organization
  - Instead, team organization has been based on research on group dynamics in general

- Without relevant experimental results, it is hard to determine optimal team organization for a specific product