Goal of Software – Provide Value to users

Quality Aspects
- Users Perspective
  - UI/HCI
  - Effectiveness/reliability
  - Security
- Developers Perspective
  - Reliability
  - Understandability
  - Efficiency
  - Maintainability

Communication
- Skills
  - Listen, Prepare, document, collaborate, stay focused, graphics
- Major cost in any system is communication and correcting miscommunication
- Teams—up dynamics
- # of participants > so does communication paths
- More members, more important to control communication

<table>
<thead>
<tr>
<th># team members</th>
<th># Communication paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
</tr>
</tbody>
</table>

Strive for 5-7 members on a team. Why???

- Star configuration
  - Limit communication paths
  - Distribution of information is centralized so all communication must pass through central figure
  - Used when time is critical
  - Ensures team members are not overwhelmed
  - Slows down dissemination of information
Network configuration
- Promotes open communication between all team members
- All artifacts are shared between members whether or not they are relevant
- Increases problem solving because there is more idea sharing
Brook’s Law
• Adding more people to an already late project may make it later
  – More communication paths
  – Time to train new people, learning curve for them and time of existing
    member to train them

Man Power Scheduling
• Common cause of missed deadlines– poor estimation
• Assume all will go well
• No time for unforeseen change
• Man month
  – Men and months are not interchangeable
  – Some parallel, some sequential tasks
  – Human resource planning– number of independent subtasks

Team Structures (pages 579-580)

Democratic Team – Wienberg in 1971
  Network format
  All team members are equals
  Run by group consensus
  Egoless → leader which rotates
  Uses brainstorming
  To be successful → team members must trust each other
  Distribution of skills
  Works better in research than in commercial environment

Chief Programmer – Mills 1971
  Based on surgical team
  Several workers report to a single manager who directly controls tasks and is responsible
  for their work
  Communication is vertical
  Chief programmer – both manager & team member
  Backup programmer – 2nd in command, stand in, equally talented
  Librarian – CM and administrative
Hierarchical

Controlled decentralized communication
Team has a defined leader
Problem solving is a group activity but implementation of solution is partitioned
Control hierarchy communication is vertical while subgroup communication is horizontal
No Silver Bullet

- Essence – inherent difficulties in the nature of software
  - Complexity, Conformity, Changeability, and Invisibility
  - Requirements refinement and rapid prototyping, grow product
- Accidents – difficulties in production
  - High level languages, time-sharing, case tools - programming environments

Need for standards

- Standard – Watts Humphrey
  - A standard is a rule or basis for comparison that is used to assess size, content, value or quality of an object or activity, typically established by designated standards body
- Establish standards for ALL aspects of development
- Give evaluation criteria
- Provide a mechanism for dividing up work
- Allows for interchanging of personnel
- Support SQA
- Establish common support environment
- Promote consistent use of better tools and methods
- Make review work and project understanding easier
- Needed whenever people, procedures and/or tools must co-exist
- Must remain current
- IEEE standards, ISO 9000/9001

Process Models

- Framework for the tasks
  WHAT is to be done
- Paradigm independent (OO or SSAD)
- Leads -- Predictable series of tasks
- Sequence of stages which categorize and control activities
- IEEE
  - A set of activities, methods, practices, and transformations that people use to develop and maintain software and its associated products
- Many different models, no one is correct for all systems, each brings different qualities to the forefront
- Stages
  - Communication, planning, modeling, construction, deployment
  - Linear or Sequential, iterative, evolutionary, parallel
**Figure 2.2** Process flow

(a) Linear process flow

(b) Iterative process flow

(c) Evolutionary process flow

(d) Parallel process flow
Assess Progress

Opt for a phased approach with a number of milestones
Requirements engineering → requirements specification
Design phase → design specification
Implementation → code artifacts
Testing → test report

Agile Process

• Early/rapid incremental delivery of software
• Small self-organizing teams that have control over what they are creating
• Informal methods
• Change represents an opportunity
• Continuous communication between customer and developer
• De-emphasizes the importance of intermediate work products
• XP – Extreme Programming
  – Agile Process Model
  – OO Paradigm
  – Four framework activities
  • Planning
    – User stories
      » Describe the desired functionality
    – Priority Values
    – Acceptance test criteria
    – Iteration plan
      » Group stories into increments
      » Measure called project velocity – number of stories in an increment
  • Design
    – KISS
    – No added functionality
    – CRC cards
    – Prototypes
    – Re-factoring
      » Changing of a software system such that it does not change external behavior yet it improves the internal structure
      » Improving on design after it is written
  • Coding
    – Pair programming
      » 2 to a single machine with separate roles
• Test
  – Unit test
    » Developed before code for each story
  – Continuous integration
  – Acceptance testing

**Measurement**

• Provides objective evaluation

• Software Metrics
  – Assist in estimation, quality control, productive assessment and project control
  – Assess technical quality and assist in tactical decision making

• Measure — collection of data

• Metric — interpretation of the data

• Process Metrics
  – Software quality
  – Organizational performance
  – Measure outputs
    – Errors uncovered before release
    – Errors uncovered by customer
    – Human effort expanded
    – Calendar time
    – Schedule conformance

• Project Metrics
  – Used to adapt project workflow
  – Monitor and control progress
  – Measure input vs. outputs
    – Calendar and effort
    – Production rates
    – Pages of documentation
    – Review hours
    – Function points
    – Delivered source lines

• Product Metrics
  – Quality measures
  – Tracking individual measures
    • Size oriented
      – Normalize quality and/or productivity by considering the size produced
      – LOC
        – Normalized
        – Errors per KLOC (prior to release)
        – Defects per KLOC
– Pages of documentation/LOC  
– Very controversial  
– Function points (page 581-582)  
– Uses functionality delivered in the product as normalizing point  
– To determine function points  
– # user inputs  
– # user outputs  
– # inquiries  
– # files  
– # external interfaces  
– All multiplied by a weighting factor (simple, average, or complex)  

• Quality Metrics  
  – Factors which affect quality  
    • Product operation  
    • Product revision  
    • Product transition  
  – Correctness  
  – Maintainability  
  – Integrity  
  – Usability  

• Example Metric  
  o Defect Removal efficiency (DRE)  
  o \[ \text{DRE} = \frac{E}{E+D} \]  
  o \( E \rightarrow \) errors found before delivery  
  o \( D \rightarrow \) errors found after delivery (defects)