# SOFTWARE PROJECT MANAGEMENT AND COST ESTIMATION

# Communication

- Training
- Intercommunication
- Effort increases as:
- $n(n - 1)/2$
- 3 workers require three times as much pair-wise intercommunication as 2; 4 workers need 6 times as much as 2.

# Improving communication

- Use hubs to cut down communication overhead
- Examples
  - Specification/design documentation
  - WiKi
  - Development meetings
- For all these the communication overhead goes up as N not N^2

# Brooks Experience

- 1/3 planning
- 1/6 coding
- 1/4 component and prototype testing
- 1/4 system test (all components in hand)

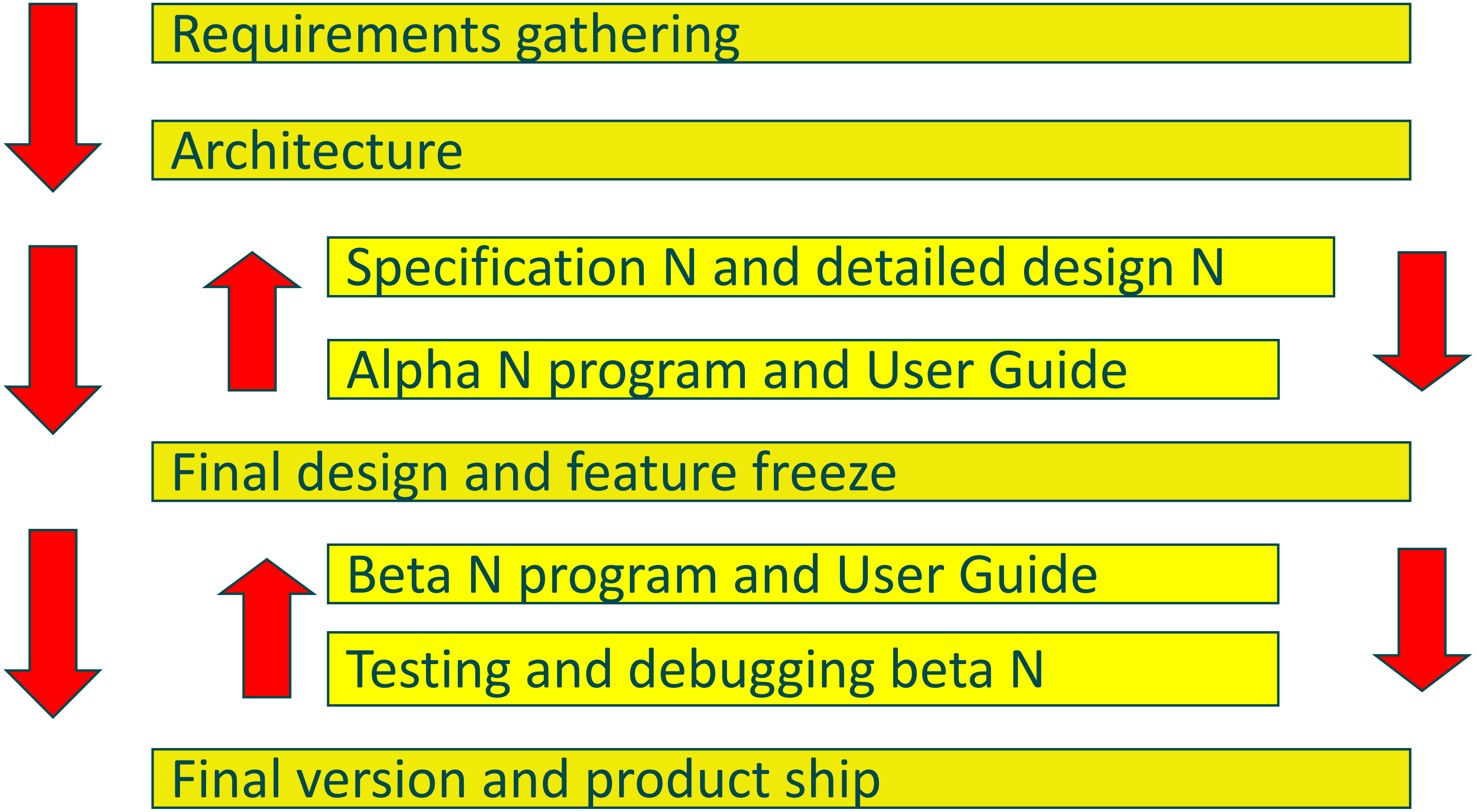
# Brookes law

- "Adding manpower to a late software project makes it later."
- Exceptions
  - Applies to projects late already only
  - Using modern development techniques reduces communications overhead:
    - Continuous integration, test first design, design patterns
  - Highly decoupled projects with clear modular specifications

# Experience/techniques for smaller projects

- Rudy Rucker teaches software engineering computer games at SJSU
- He advises
- Estimate how long planning will take
- Multiply that time by 3
- Use the “Inventer Lifecyle” to generate an Alpha 1 by ~½ way through the project

# Inventor Lifecyle (one/two person projects)



Requirements gathering

Architecture

Specification N and detailed design N

Alpha N program and User Guide

Final design and feature freeze

Beta N program and User Guide

Testing and debugging beta N

Final version and product ship

Rucker (2003) p39

# Cost Estimation Research

- 100s of papers since the 1960s
- As development techniques improve e.g. OO, cost estimation has to adapt
- Move from
  - Coding estimation
- Move to
  - Functional estimation

# Cost estimation approaches

- Expert estimation
  - Planning poker
  - WBS
- Formal estimation
  - COCOMO
- Combined methods
  - Each formal estimation technique has a expert phase anyway

# Other approaches

- Case based reasoning
  - Using many previous projects/coding efforts to estimate this project
- Lexical analysis of requirements specifications

# Planning poker

1. Each member of planning team given pack of cards with numbers on
2. Project manager introduces project
   - Team clarifies assumptions
   - Discuss risk
3. Each member picks a card as estimate
4. Lowest and highest estimation members given change to justify decision
5. Discuss, then go back to 3, until consensus reached

# Planning poker benefits

- Reduces anchoring
  - Low anchor
    - “"I think this is an easy job, I can't see it taking longer than a couple of weeks”
  - High anchor
    - “I think we need to be very careful, clearing up the issues we've had in the back end could take months”
- Studies
  - Molokken-Ostvold, K. Haugen, N.C.

# Software productivity metrics

- Measures of size
  - Lines of (source) code per person month: (LOC/pm)
  - Object code instructions
- Document pages
- Measure of function
  - Function points
  - Object points

# Lines of code  (KLOC)

- Easy to measure
- Difficult to estimate
- As productivity measure?
  - Code quality
  - Project delivery
  - Language dependency

# Estimating lines of code

- Structural decompose project into separate modules
- Get programmer to produce 3 figures for each module
  - pessimistic estimate of LOC for module
  - average estimate of LOC for module
  - optimistic estimate of LOC for module
- Use weighting factor based on previous estimation performance

# System Development times

| | Analysis | Design | Coding | Testing | Documentation |
|---|---|---|---|---|---|
| **Assembly code** | 3 | 5 | 8 | 10 | 2 |
| **High level language** | 3 | 5 | 4 | 6 | 2 |

| | Size | Effort | Productivity |
|---|---|---|---|
| **Assembly code** | 5000 lines | 28 weeks | 714 lines/month |
| **High level language** | 1500 lines | 20 weeks | 300 lines/month |

# Function points

- Estimates of the program feature elements
  - External input and output
  - User interactions
  - External interfaces
  - Files used

# Calculating Function points

## x Weighting factor

| | Simple | Average | Complex | |
|---|---|---|---|---|
| User input count | 3 | 4 | 6 | |
| User output count | 4 | 5 | 7 | |
| User inquiries | 3 | 4 | 6 | |
| Number of Internal logical files | 7 | 10 | 15 | |
| External Interface files | 5 | 7 | 10 | |
| Count total → | | | | |

# Function point analysis

- Internal logic file
  - tables in a relational database
  - Xml files used in application
  - Complexity : record types, data element types (e.g. surname, post code)
- External interface file
  - Same as ILF but not maintained by application
- User input
  - Usually user input screen
  - Complexity : data element types and file type referenced (e.g. count of tables updated)

# Function point analysis

- External outputs
  - Data presented to the user
  - Some mathematics or derived data obtained
  - Complexity measure:
    - data element types and file type referenced (e.g. count of tables updated)
- External enquires
  - Data presented to the user
  - No maths or derived data involved
  - Complexity measure : see external output

# Function point estimation including the value adjustment factor (VAF)

**FP = count-total x (0.65 + 0.01 $\sum$Fi)**

**F1 = Reliable backup and recovery (1-5)**

**F2 = Data communications (1-5)**

**F3 = Distributed functions (1-5)**

**F4 = Performance (1-5)**

**F5 = Heavily used configuration (1-5)**

**F6 = Online data entry (1-5)**

**F7 = Operational ease (UI) (1-5)**

**F8 = Master file updated online (1-5)**

**F9 = Complex interface (1-5)**

**F10 = Complex processing (1-5)**

**F11 = Reusability (1-5)**

**F12 = Installation included (1-5)**

**F13 = Multiple sites (1-5)**

**F14 = Facilitate change  (1-5)**