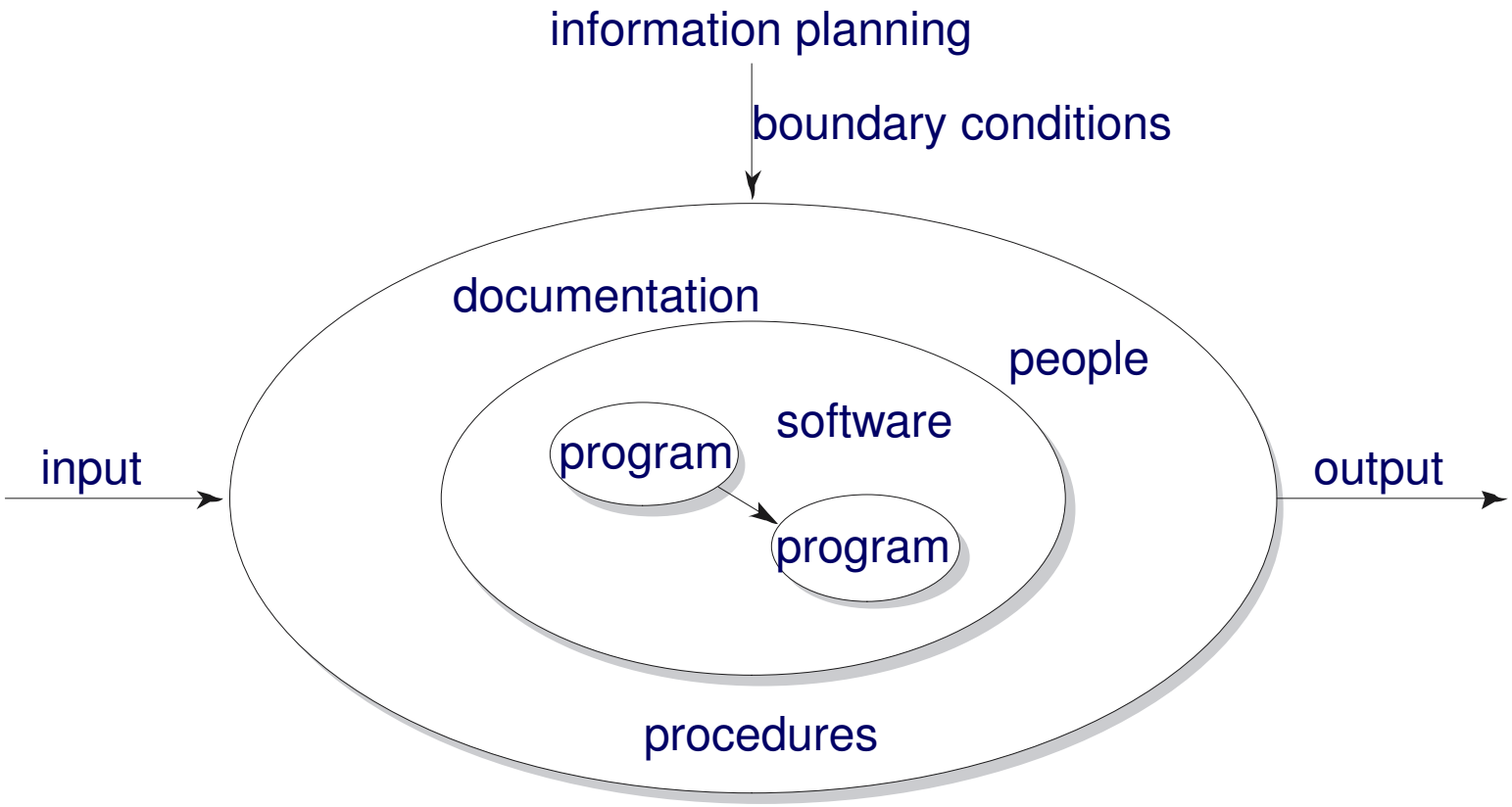
Software Engineering Management



Contents of project plan

- Introduction
- Process model
- Organization of project
- Standards, guidelines, procedures
- Management activities
- Risks
- Staffing
- Methods and techniques
- Quality assurance
- Work packages
- Resources
- Budget and schedule
- Changes
- Delivery

Project control

- **Time,** both the number of man-months and the schedule
- **Information,** mostly the documentation
- **Organization,** people and team aspects
- **Quality,** not an add-on feature; it has to be built in
- **Money,** largely personnel

Managing time

- **Measuring progress is hard ("we spent half the money, so we must be halfway")**
- **Development models serve to manage time**
- **More people $\Rightarrow$ less time?**
  - Brooks' law: adding people to a lae project makes it later

Managing information

- **Documentation**
  - Technical documentation
  - Current state of projects
  - Changes agree upon
  - …
- **Agile projects: less attention to explicit documentation, more on tacit knowledge held by people**

Managing people

- Managing expectations
- Building a team
- Coordination of work

Managing quality

- Quality has to be designed in
- Quality is not an afterthought
- Quality requirements often conflict with each other
- Requires frequent interaction with stakeholders

Managing cost

- Which factors influence cost?
- What influences productivity?
- Relation between cost and schedule

Configuration management

- identification and definition of **configuration items,** such as source code modules, test cases, requirements specification
- managing changes and making configuration items available during the software life cycle, usually through a **Configuration Control Board** (CCB)
- keeping track of the **status** of all items (including the change requests)

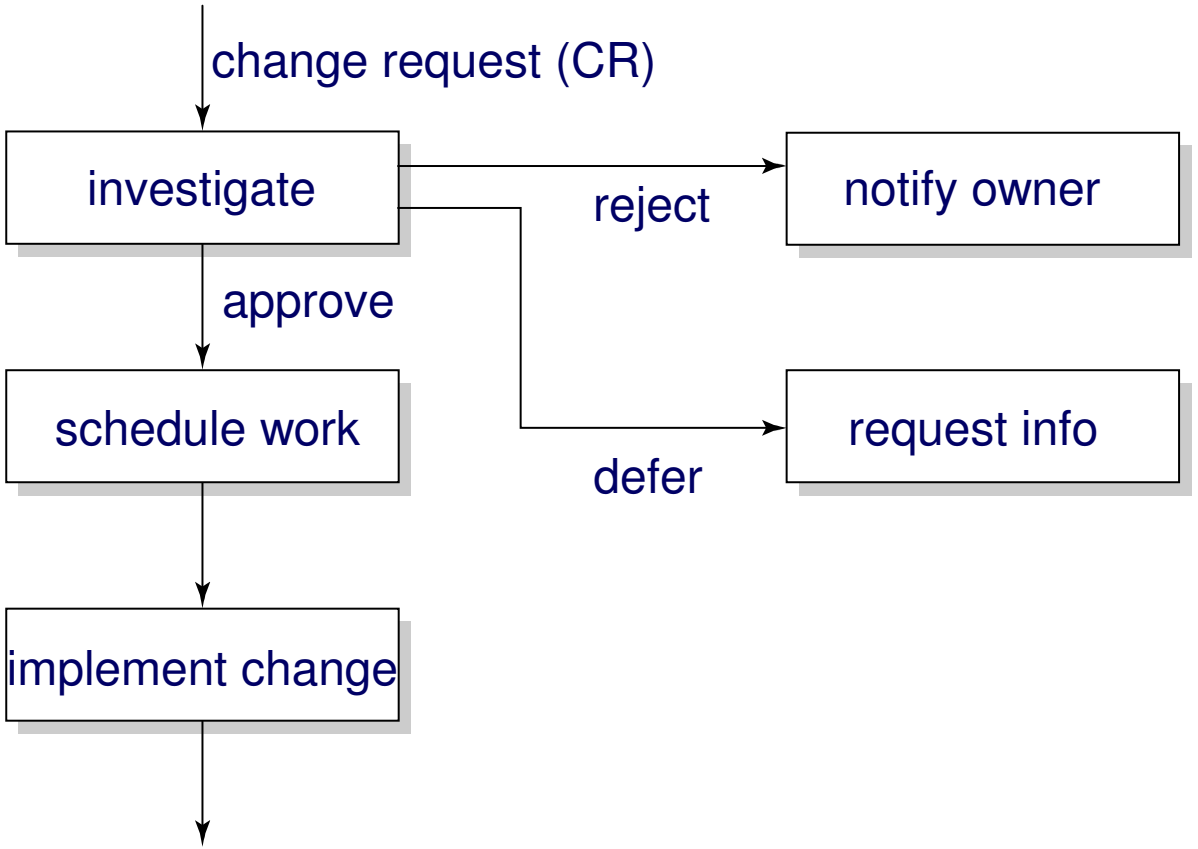
- **crucial for large projects**

Configuration Control Board

- ensures that every change to the baseline (change request - CR) is properly authorized and executed
- CCB needs certain information for every CR, such as who submits it, how much it will cost, urgency, etc
- CCB assesses the CR. If it is approved, it results in a work package which has to be scheduled.

- so, configuration management is not only about keeping track of changes, but also about workflow management

Workflow of a change request



Tool support for configuration management

- if an item has to be changed, one person gets a copy thereof, and meanwhile it is locked to all others
- new items can only be added to the baseline after thorough testing
- changes in the status of an item (e.g. code finished) trigger further activities (e.g. start unit testing)
- old versions of a component are kept as well, resulting in versions, like X.1, X.2, …
- we may even create different branches of revisions: X.2.1, X.2.2, … and X.3.1, X.3.2, ...

Functionalities of Software Configuration Management (SCM) tools

- Components (storing, retrieving, accessing, …)
- Structure (representation of system structure)
- Construction (build an executable)
- Auditing (follow trails, e.g. of changes)
- Accounting (gather statistics)
- Controlling (trace defects, impact analysis)
- Process (assign tasks)
- Team (support for collaboration)

Models of configurations

- **version-oriented:** physical change results in a new version, so versions are characterized by their difference, i.e. **delta**
- **change-oriented:** basic unit in configuration management is a logical change

- **identification of configuration becomes different: "baseline X plus fix table bug" instead of "X3.2.1 + Y2.7 + Z3.5 + …"**

Evolution of SCM tools

- Early tools: emphasis on product-oriented tasks
- Nowadays: support for other functionalities too. They have become a (THE) major tool in large, multi-site projects
- Agile projects: emphasis on running system: *daily builds*

Configuration Management Plan

- **Management section: organization, responsibilities, standards to use, etc**
- **Activities: identification of items, keeping status, handling CRs**

People management

- People have different goals
- People and productivity
- Group processes
- Coordination of work
- Importance of informal communication

Mintzberg's coordination mechanisms

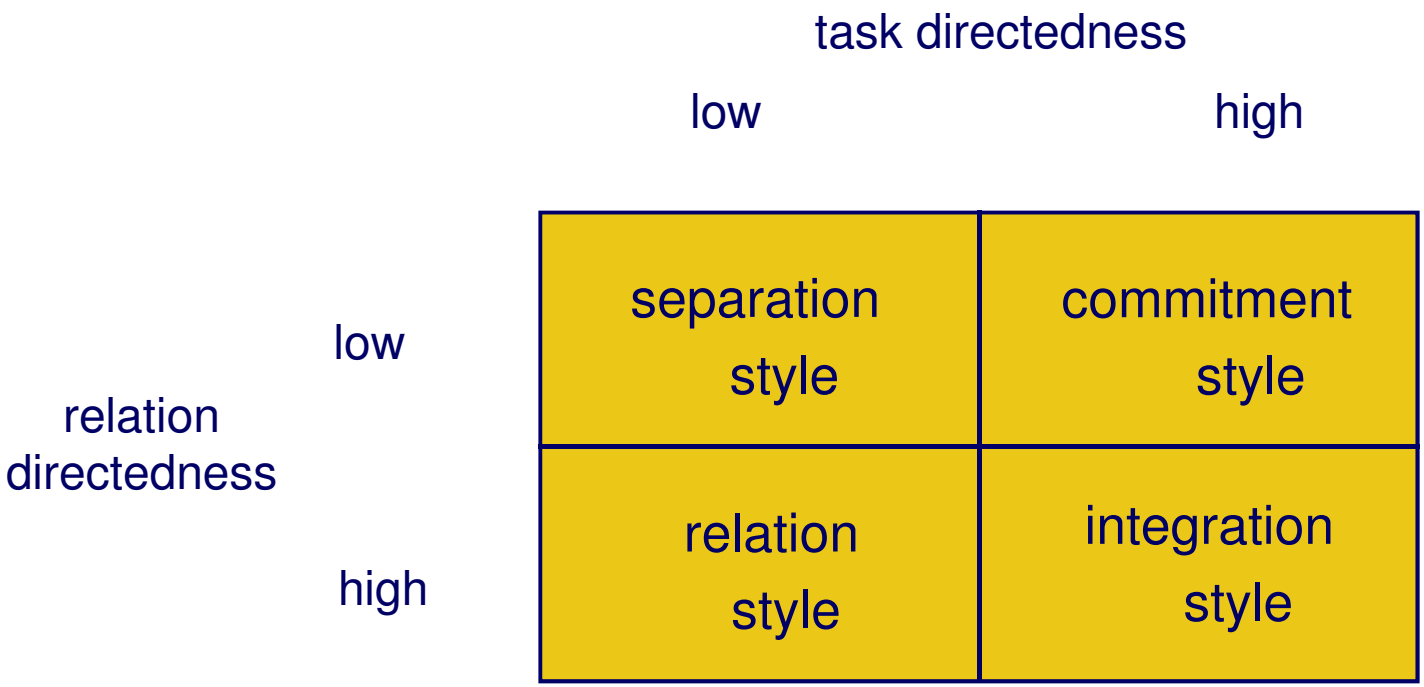
- Simple: direct supervision
- Machine bureaucracy: standardization of work processes
- Divisionalized form: standardization of work products
- Professional bureaucracy: standardization of worker skills
- Adhocracy: mutual adjustment

External and Internal forces

- Example context: a complex software development project in a new, not yet explored area, within a government agency

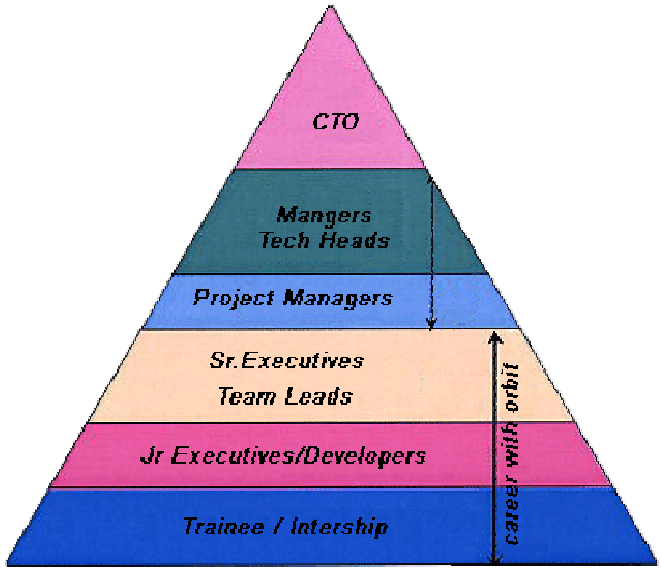
- External force: the bureaucratic context is likely to want to push a bureaucratic type of organization, with bosses, and hierarchical decision procedures
- Internal force: the project really requires a more democratic, consensus-based type of organization

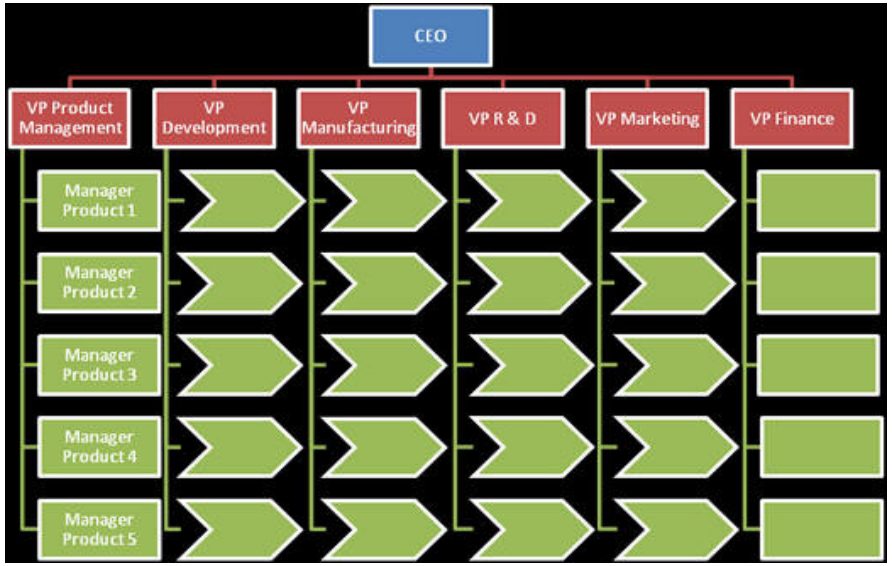
Reddin's management styles



Team Organization

- Hierarchical organization
- Matrix organization
- Chief programmer team
- SWAT team
- Agile team/Extreme Programming (XP)
- Open Source
- Development





Some general rules

- Use fewer, and better, people
- Fit tasks to people
- Help people to get the most out of themselves
- Look for a well-balanced team
- If someone doesn’t fit the team: remove him

Managing Software Quality

- Quality of the product versus quality of the process
- Check whether (product or process) *conforms to* certain norms
- Improve quality by improving the product or process

Approaches to quality

	Conformance	Improvement
Product	ISO 9126	‘best practices’
Process	ISO 9001 SQA	CMM* ISO 15504 /SPICE**

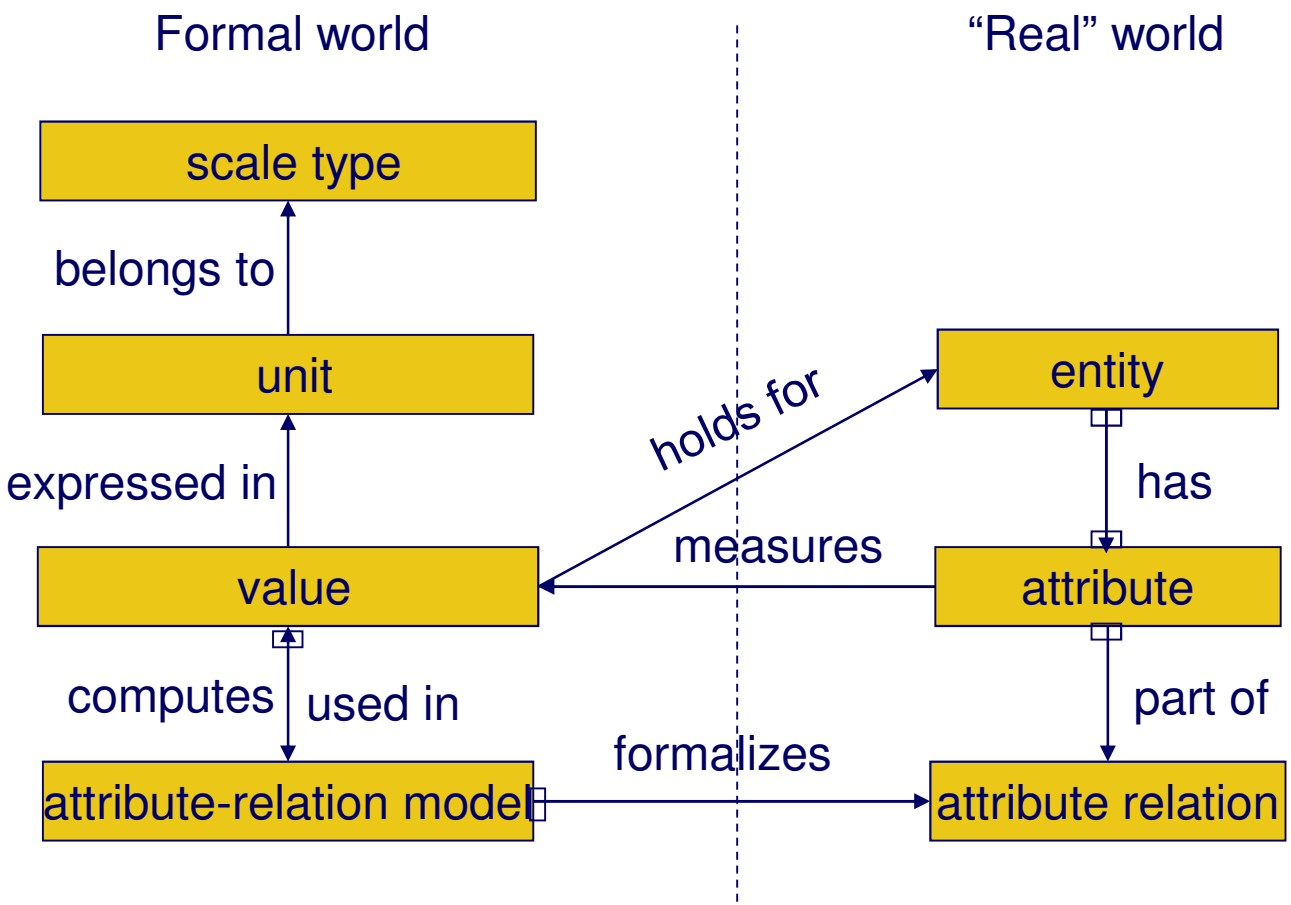
* Capability Maturity Model

**Software Process Improvement and Capability Determination

How to measure “complexity”?

- The length of the program?
- The number of goto’s?
- The number of if-statements?
- The sum of these numbers?
- Yet something else?

A measurement framework



Scale types

- Nominal: just classification
- Ordinal: linear ordering (>)
- Interval: like ordinal, but interval between values is the same (so average has a meaning)
- Ratio: like interval, but there is a 0 (zero) (so A can be twice B)
- Absolute: counting number of occurrences

Quality attributes (McCall)

- **Product operation**
  - Correctnessdoes it do what I want?
  - Reliability — does it do it accurately all of the time?
  - Efficiency — will it run on my hardware as well as it can?
  - Integrity — is it secure?
  - Usability — can I use it?
- **Product revision**
  - Maintainability — can I fix it?
  - Testability — can I test it?
  - Flexibility — can I change it?
- **Product transition**
  - Portability — will I be able to use it on another machine?
  - Reusability — will I be able to reuse some of the software?
  - Interoperability — will I be able to interface it with another system?

Taxonomy of quality attributes (ISO 9126)

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

ISO 9001

- Model for quality assurance in design, development, production, installation and servicing
- Basic premise: confidence in product conformance can be obtained by adequate demonstration of supplier's capabilities in processes (design, development, …)
- ISO registration by an officially accredited body, re-registration every three years

Capability Maturity Model (CMM)

- Initial level: software development is ad-hoc
- Repeatable level: basic processes are in place
- Defined level: there are *standard* processes
- Quantitatively managed level: data is gatheread and analyzed routinely
- Optimizing level: stable base, data is gathered to improve the process

Initial $\Rightarrow$ repeatable level

- Requirements management
- Project planning
- Project monitoring and control
- Supplier agreement management
- Measurement and analysis
- Process and product quality assurance
- Configuration management

Repeatable $\Rightarrow$ defined level

- Requirements development
- Technical solution
- Product integration
- Verification
- Validation
- Organization process focus
- Organization process definition
- Organizational training
- Integrated project management
- Risk management
- Decision analysis and resolution

Software Process Improvement (SPI)

- Formulate hypotheses
- Carefully select metrics
- Collect data
- Interpret data
- Initiate improvement actions

- Iterate

Lessons w.r.t. data collection

- Closed loop principle: result of data analysis must be useful to supplier of data
- Do not use data collected for other purposes
- Focus on continuous improvement
- Only collect data you really need

Software Cost Estimation

- Quantitative models ($E = 2.5\ KLOC^{1.05}$)
- Qualitative models (e.g. expert estimation)
- (Agile cost estimation)

- Relate cost to development time

On productivity

- **Even if quantitative models are not that good, the cost drivers of these models learn us about productivity:**
  - Writing less code helps
  - Reuse helps
  - Quality of people is important
  - Tools help
  - …

Walston-Felix

- One of the early algorithmic models (1977)
- Many factors (29 out of 51 projects)
- Only three alternatives (high, medium, low) per factor

- Its form influenced many later models

COCOMO (COnstructive COst MOdel)

- **Very well-documented (Boehm book, 1981)**
- **Basic form: $E = bKLOC^c$, where b (~3) and c (1+ε) depend on the type of project (mode):**
  - Organic: relatively small and well-known
  - Embedded: inflexible environment with many constraints
  - Semidetached: somewhere in between
- **More complex form: takes into account 15 multiplicative cost drivers**

Function Point Analysis (FPA)

- **Size (=cost) is based on number of data structures used:**
  - I: number of input types
  - O: number of output types
  - E: number of enquiry types
  - L: number of logical internal types
  - F: number of interfaces
- **Then, magically, UFP = 4I + 5O + 10E + 4L + 7F**
- **Unadjusted FP (UFP) Count**

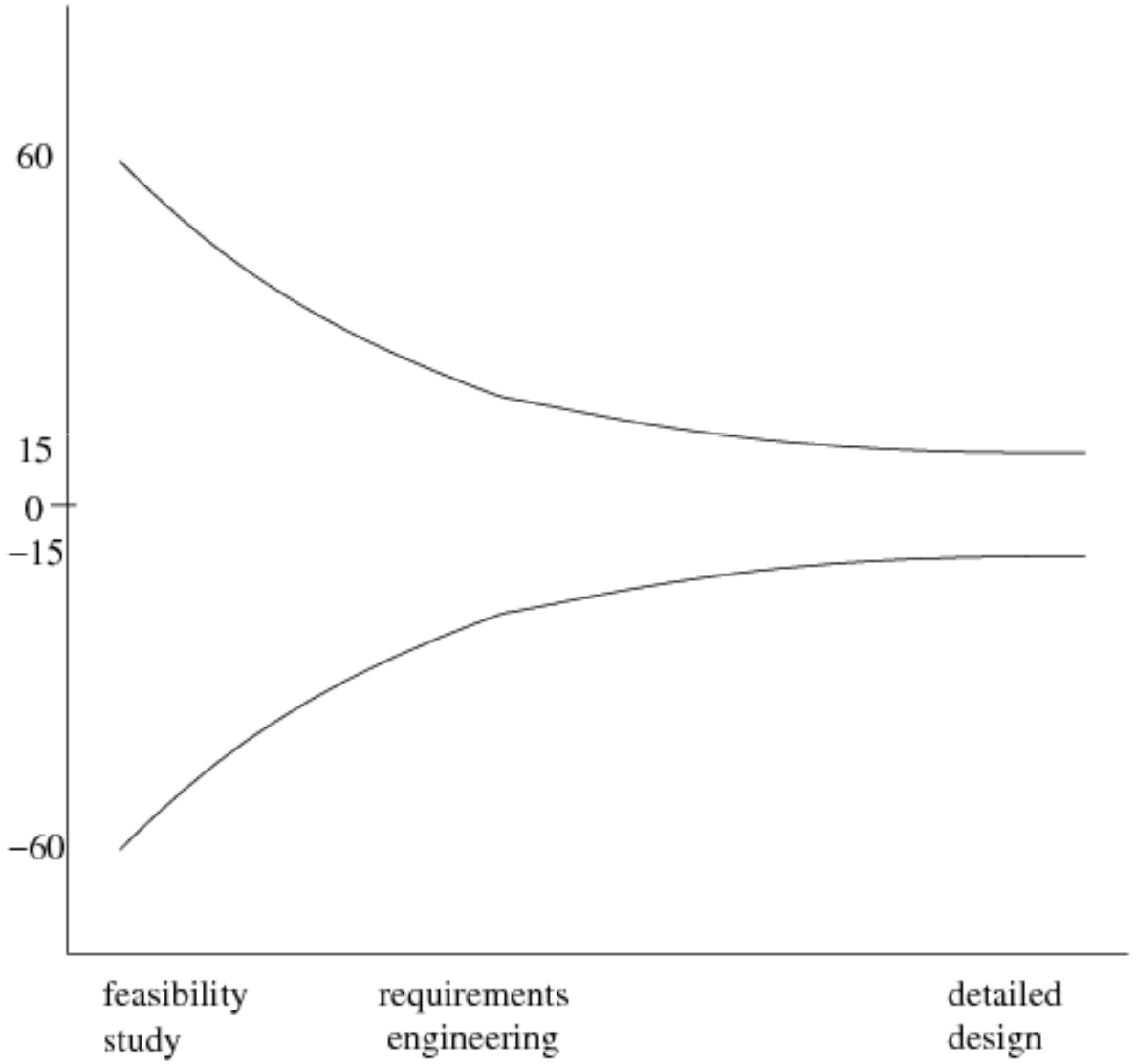
Use Case Points

- **FPA-like model, starting from use cases**
- **Counting depends on the use case**
  - How many steps in success scenario
  - How many classes in the implementation
  - Complexity of the actors involved
- **Next, corrections for the technical and environmental complexity**

Difficulties with applying these models

- **People do not collect numbers, so:**
  - This project costs the same as the last project
  - We have 6 months, so it will take 6 months
  - …and other political estimates
- **These models require calibration**

Cone of uncertainty



From total effort to $\Rightarrow$ number of months

- **A lot of consensus between models: $T \cong 2.5E^{1/3}$**
- **Compressing this value has a price:**
  - Team larger $\Rightarrow$ more communication
  - New people first slows down productivity
- **Combined: Brooks' law:**

Adding manpower to a late project makes it later

Impossible region