Quality 24
Process Improvement 26
Real processes

- Cleanroom
- RUP
- XP

Figure 24.3 Process-based quality

Product Quality

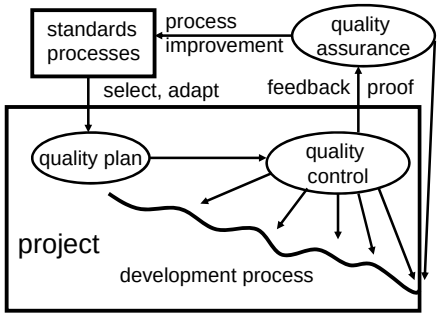
- Acceptable: usable, learnable, compatible
- Efficient: response time, memory use
- Dependable: safe, reliable, secure
- Maintainable: documented, structured

[Fig. 1.2, 24.2]

Quality Management

- Organization
  - defining standards
  - defining processes
- Project
  - quality plan
  - checking outcome

Quality Management

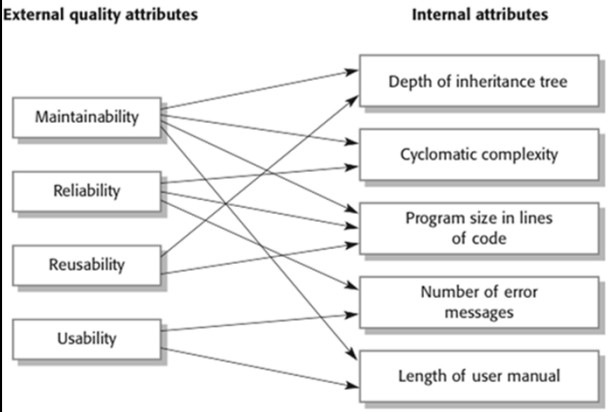


Quality Plan

- Product Quality Requirements
  - safety/reliability analysis
  - too good and over budget ≠ quality
- Development plan
  - standards, documentation
  - required validation and verification actions
  - release plan, config. management
- Process risks

Quality assessment

- External attributes
  - what is required
  - measurable after completion (maybe)
  - cannot be measured during development
- Internal attributes
  - can be measured during development
  - predict external attributes (maybe)
    - learn from experience



Software metric	Description
Fan-in/Fan-out	Fan-in is a measure of the number of functions or methods that call another function or method (say X). Fan-out is the number of functions that are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability. I discuss cyclomatic complexity in Chapter 8.
Length of identifiers	This is a measure of the average length of identifiers (names for variables, classes, methods, etc.) in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if-statements are hard to understand and potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value of a document's Fog index, the more difficult the document is to understand.

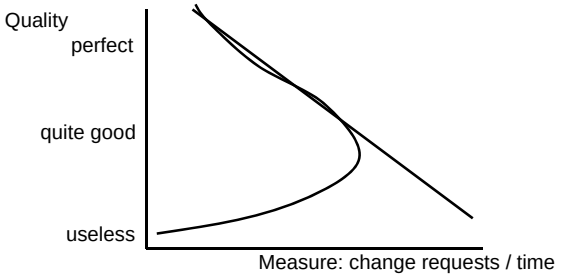
Figure 24.12 The CK object-oriented metrics suite

Object-oriented metric	Description
Weighted methods per class (WMC)	This is the number of methods in each class, weighted by the complexity of each method. Therefore, a simple method may have a complexity of 1, and a large and complex method a much higher value. The larger the value for this metric, the more complex the object class. Complex objects are more likely to be difficult to understand. They may not be logically cohesive, so cannot be reused effectively as superclasses in an inheritance tree.
Depth of inheritance tree (DIT)	This represents the number of discrete levels in the inheritance tree where subclasses inherit attributes and operations (methods) from superclasses. The deeper the inheritance tree, the more complex the design. Many object classes may have to be understood to understand the object classes at the leaves of the tree.
Number of children (NOC)	This is a measure of the number of immediate subclasses in a class. It measures the breadth of a class hierarchy, whereas DIT measures its depth. A high value for NOC may indicate greater reuse. It may mean that more effort should be made in validating base classes because of the number of subclasses that depend on them.
Coupling between object classes (CBO)	Classes are coupled when methods in one class use methods or instance variables defined in a different class. CBO is a measure of how much coupling exists. A high value for CBO means that classes are highly dependent, and therefore it is more likely that changing one class will affect other classes in the program.
Response for a class (RFC)	RFC is a measure of the number of methods that could potentially be executed in response to a message received by an object of that class. Again, RFC is related to complexity. The higher the value for RFC, the more complex a class and hence the more likely it is that it will include errors.
Lack of cohesion in methods (LCOM)	LCOM is calculated by considering pairs of methods in a class. LCOM is the difference between the number of method pairs without shared attributes and the number of method pairs with shared attributes. The value of this metric has been widely debated and it exists in several variations. It is not clear if it really adds any additional, useful information over and above that provided by other metrics.

Measurement process

- Choose measurements
  - Goal - Question - Metric (GQM)
- Select components
- Measure
- Identify anomalous values
  - compared to normal product / company values
- Analyse anomalous components

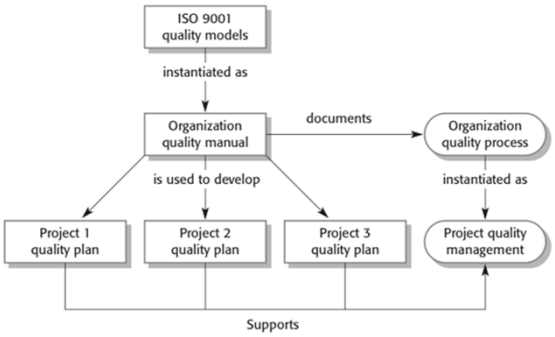
Interpretation of metrics



Quality Standards

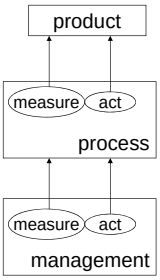
- Save work
  - no need to reinvent
  - supported by tools
- Capture ”good practice”
- Can define levels of quality
- Improve communication
- Product / Process / ISO 9001

ISO 9001



Process improvement

activities quality attributes standards



- maintainability
- usability
- reliability

- variable names
- user documents
- version mgm

- understandability
- visibility
- robustness

- inspection proc.
- milestones
- change control

- certification

- ISO 9001
- CMMI

Process Quality Fig. 26.2

- People actually follow it:
  - acceptable, usable, learnable
- It delivers:
  - efficient, in time, acceptable product quality
- Manageable
  - visible, robust to problems, reliable, adaptable
- Supportable
  - documented, structured, measurable

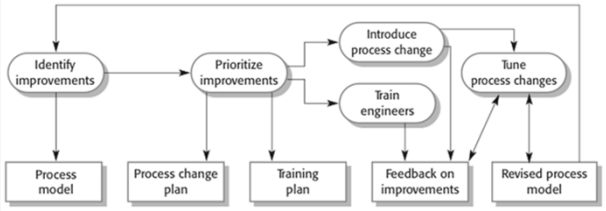
Process measurement

- Resources required
  - time
  - money
- Occurrence of events
  - failed system builds
  - missed deadlines
  - missing process documents

Process analysis

- Questionnaires, interviews
  - honesty is the best policy?
- Ethnographic studies
  - observe the tribe
    [insert Dilbert cartoon]

Implementing change



Capability Maturity Model 26.5

- Process areas (22 in CMMI)
- Maturity levels
  0. incomplete
  1. performed
  2. managed
  3. defined
  4. quantitatively managed
  5. optimizing

initial

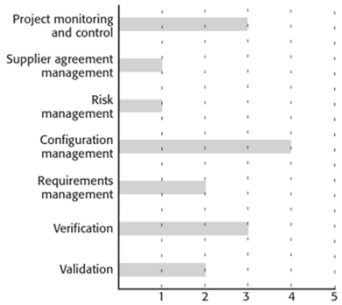
Figure 26.7 Process areas in the CMMI

Category	Process area	Category	Process area
Process management	Organizational process definition (OPD)	Engineering	Requirements management (REQM)
	Organizational process focus (OPF)		Requirements development (RD)
	Organizational training (OT)		Technical solution (TS)
	Organizational process performance (OPP)		Product integration (PI)
	Organizational innovation and deployment (OID)		Verification (VER)
Project management	Project planning (PP)		Validation (VAL)
	Project monitoring and control (PMC)	Support	Configuration management (CM)
	Supplier agreement management (SAM)		Process and product quality management (PPQA)
	Integrated project management (IPM)		Measurement and analysis (MA)
	Risk management (RSKM)		Decision analysis and resolution (DAR)
	Quantitative project management (QPM)		Causal analysis and resolution (CAR)

Various CMMs

- Staged CMM
  – classify organisation
- Continuous CMM
  – classify each practice (next slide)
- The ”CMM Principle”
  – P-CMM 22.2 People management processes

Figure 26.11 A process capability profile



What is the company goal?

- Usually level 3 is good enough
  – unless you’re NASA
  – or a new company in India
- The lower end of the scale
  -1. We don’t care about quality
  -2. Let’s do all the paperwork - but not the job
  -3. Quality is for the weak.
  Real programmers need no documentation

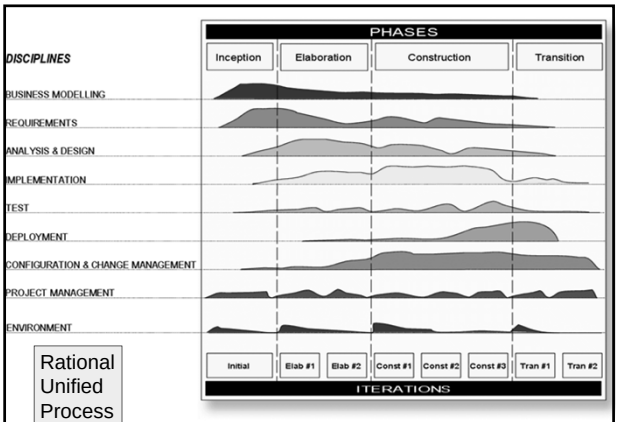
Real Processes

- A combination of ”best practices”
- Need to be adapted
  - start with a standard
  - adapt
  - introduce
  - monitor - change
- Supported by tools and standards

Cleanroom (1987) 15.1

- Goal: zero-defect software
- Ingredients

– formal specification	Teams specification
– incremental development – structured programming	development
– static verification (inspections - no tests!)	
– statistical system testing	certification



Rational Unified Process (RUP)

Rational Unified Process

- Goals:
  - realistic
  - reuse
  - tool support
- Incremental and iterative
- OO, UML, visual models, components
- Quality: support processes, transition

Traditional vs. Agile .

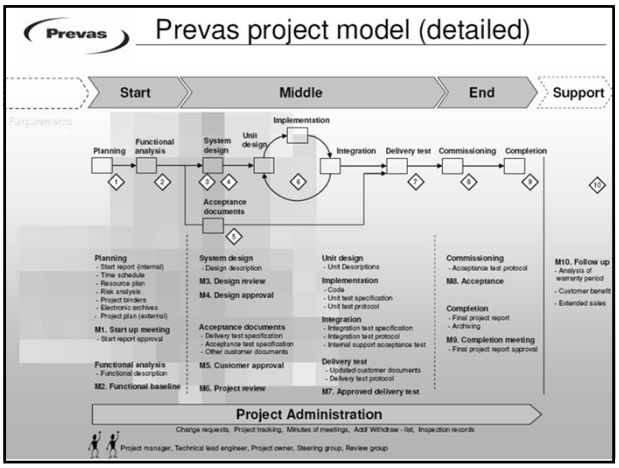
- Follow a plan
- Change costs
- Frozen requirements contract
- Documentation
- Deliverables at a deadline

- People
- Embrace change
- User stories, tests, customer involvement
- Working software
- Time-boxed

incremental

- smaller increments

Agile principle	Scrum	Extreme Programming
Incremental planning and development	Sprints Sprint backlog Planning poker	Implement user stories Story cards Planning poker
Customer involvement	Product owner Demo at end of sprint	Customer representative in development team
People, not process	Scrum meetings Sustainable pace (time-boxed)	Pair programming Collective ownership of code Sustainable pace
Embrace change	Change occurs from one sprint to the next	Continuous integration and release Test-first development
Maintain simplicity	Refactoring No anticipation of future requirements	Refactoring No anticipation of future requirements


Prevas project model (detailed)
Start
Middle
End
Support
Planning
Functional analysis
System design
Unit design
Implementation
Integration
Delivery test
Commissioning
Completion
Acceptance documents
Planning
- Start report (internal)
- Time schedule
- Resource plan
- Risk analysis
- Project binders
- Electronic archives
- Project plan (external)
M1. Start up meeting
- Start report approval
Functional analysis
- Functional description
M2. Functional baseline
System design
- Design description
M3. Design review
M4. Design approval
Acceptance documents
- Delivery test specification
- Acceptance test specification
- Other customer documents
M5. Customer approval
M6. Project review
Unit design
- Unit Descriptions
Implementation
- Code
- Unit test specification
- Unit test protocol
Integration
- Integration test specification
- Integration test protocol
- Internal support acceptance test
Delivery test
- Updated customer documents
- Delivery test protocol
M7. Approved delivery test
Commissioning
- Acceptance test protocol
M8. Acceptance
Completion
- Final project report
- Archiving
M9. Completion meeting
- Final project report approval
M10. Follow up
- Analysis of warranty period
- Customer benefit
- Extended sales
Project Administration
Change requests, Project tracking, Minutes of meetings, Add/ Withdraw - list, Inspection records
Project manager, Technical lead engineer, Project owner, Steering group, Review group