Software Engineering

Part 6. Managing People

- Quality Management
- Process Improvement
- Configuration Management

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr
http://dslab.konkuk.ac.kr

Ver. 1.6

※ This lecture note is based on materials from Ian Sommerville 2006.

Chapter 27.

Quality Management

Objectives

- To introduce the quality management process and key quality management activities
- To explain the role of standards in quality management
- To explain the concept of a software metric, predictor metrics and control metrics
- To explain how measurement may be used in assessing software quality and the limitations of software measurement

Software Quality Management

- Concerned with ensuring that the required level of quality is achieved in a software product.
  - Involves defining appropriate quality standards and procedures, and ensuring that these are followed.
  - Should aim to develop a 'quality culture' where quality is seen as everyone's responsibility.

- Quality ?
  - Means a product should meet its specification.

- Quality problems in software systems
  - There is a tension between customer quality requirements (efficiency, reliability, ...) and developer quality requirements (maintainability, reusability, ...)
  - Some quality requirements are difficult to specify in an unambiguous way.
  - Software specifications are usually incomplete and often inconsistent.

Quality Compromise

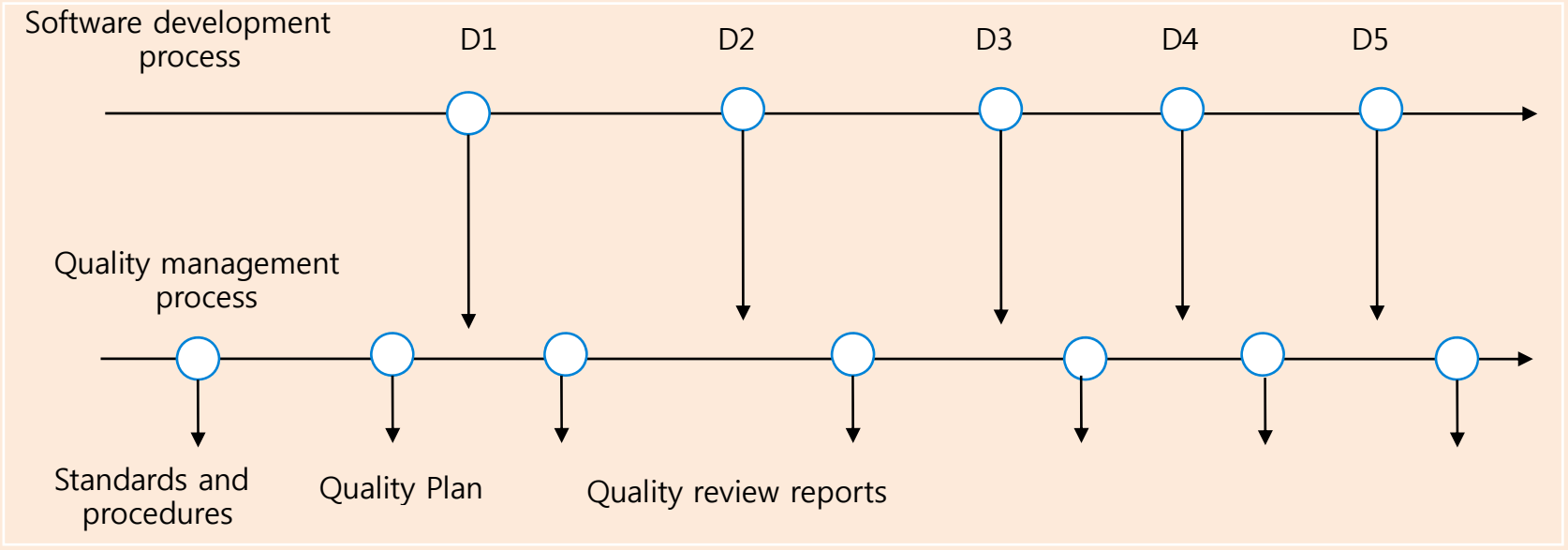
- Cannot wait for specifications to be improved before paying attention to quality management.
- We must put quality management procedures into place to improve quality in spite of imperfect specification.

- Scope of quality Management
  - Quality management is particularly important for large complex systems. The quality documentation is a record of progress and supports continuity of development as the development team changes.
  - For smaller systems, quality management needs less documentation and should focus on establishing a quality culture.

Quality Management Activities

- Quality assurance
  - Establish organisational procedures and standards for quality.
- Quality planning
  - Select applicable procedures and standards for a particular project and modify these as required.
- Quality control
  - Ensure that procedures and standards are followed by the software development team.
- Quality management should be separate from project management to ensure independence.

Quality Management and Software Development



Process and Product Quality

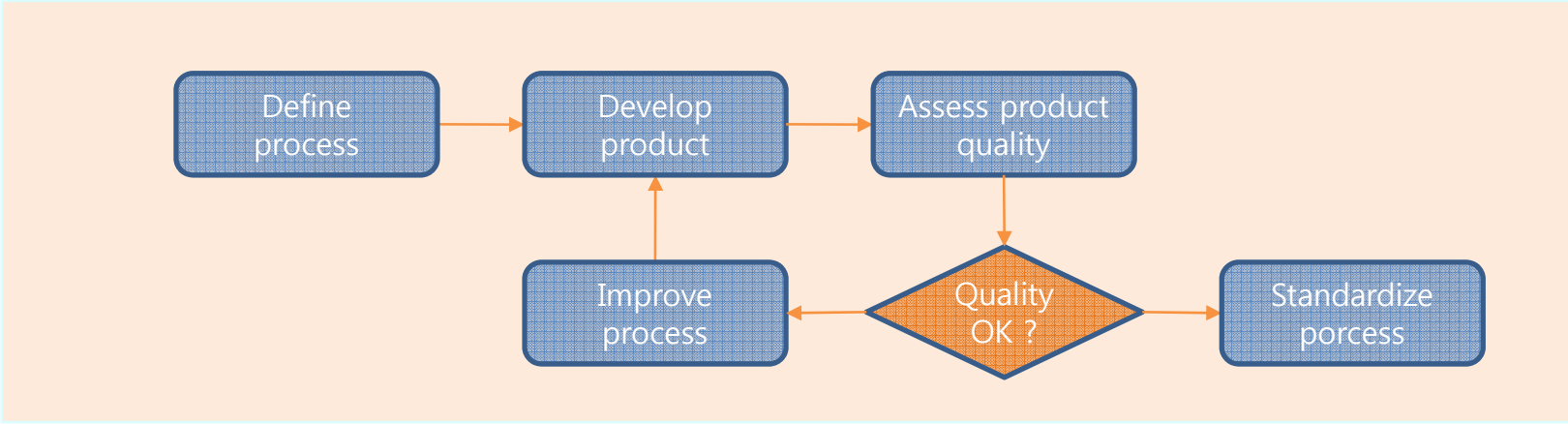
- The quality of a developed product is influenced by the quality of the production process.
  - Important in software development as some product quality attributes are hard to assess.

- However, there is a very complex and poorly understood relationship between software processes and product quality.

Process-based Quality

- There is a straightforward link between process and product in manufactured goods, but more complex for software.
  - Application of individual skills and experience is particularly imporant in software development.
  - External factors such as the novelty of an application or the need for an accelerated development schedule may impair product quality.

- Must be careful not to impose inappropriate process standards.



Quality Assurance and Standards

- Standards are the key to effective quality management.
  - May be international, national, organizational or project standards.
  - Encapsulations of best practice

- Product standards
  - define characteristics that all components should exhibit, e.g. a common programming style.
- Process standards
  - define how the software process should be enacted.

Product and Process Standards

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of documents to CM
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Problems with Standards

- They may not be seen as relevant and up-to-date by software engineers.
- They often involve too much bureaucratic form filling.
- If they are unsupported by software tools, tedious manual work is often involved to maintain the documentation associated with the standards.

Standards Development

- Standard development involves practitioners in development.
  - Engineers should understand the rationale underlying the standard.
- Standard should be reviewed regularly.
  - Standards can quickly become outdated and this reduces their credibility amongst practitioners.
- Detailed standards should have associated tool support.
  - Excessive clerical work is the most significant complaint against standards.

ISO 9000

- An international set of standards for quality management.
  - Applicable to a range of organizations from manufacturing to service industries.

- ISO 9001
  - Applicable to organisations which design, develop and maintain products.
  - A generic model of the quality process that must be instantiated for each organization using the standard.

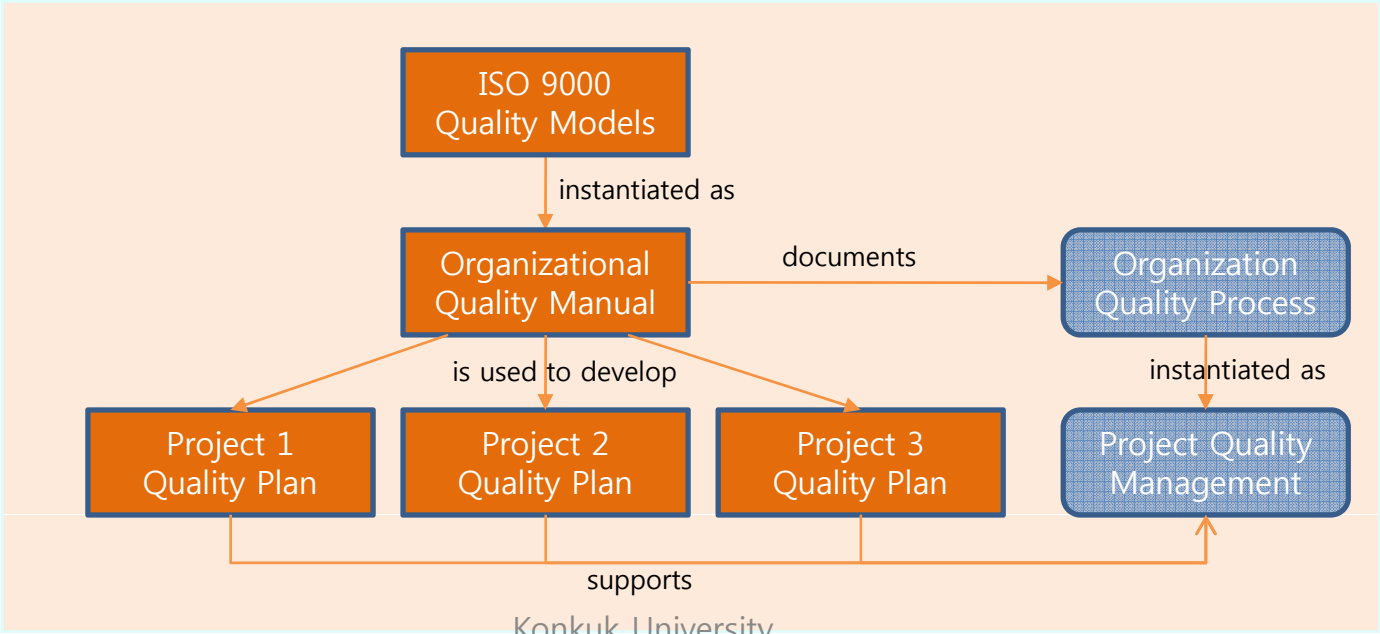
ISO 9001

Management responsibility	Quality system
Control of non-conforming products	Design control
Handling, storage, packaging and delivery	Purchasing
Purchaser-supplied products	Product identification and traceability
Process control	Inspection and testing
Inspection and test equipment	Inspection and test status
Contract review	Corrective action
Document control	Quality records
Internal quality audits	Training
Servicing	Statistical techniques

ISO 9000 Certification

- Quality standards and procedures should be documented in an organizational quality manual.
- An external body may certify that an organization's quality manual conforms to ISO 9000 standards.

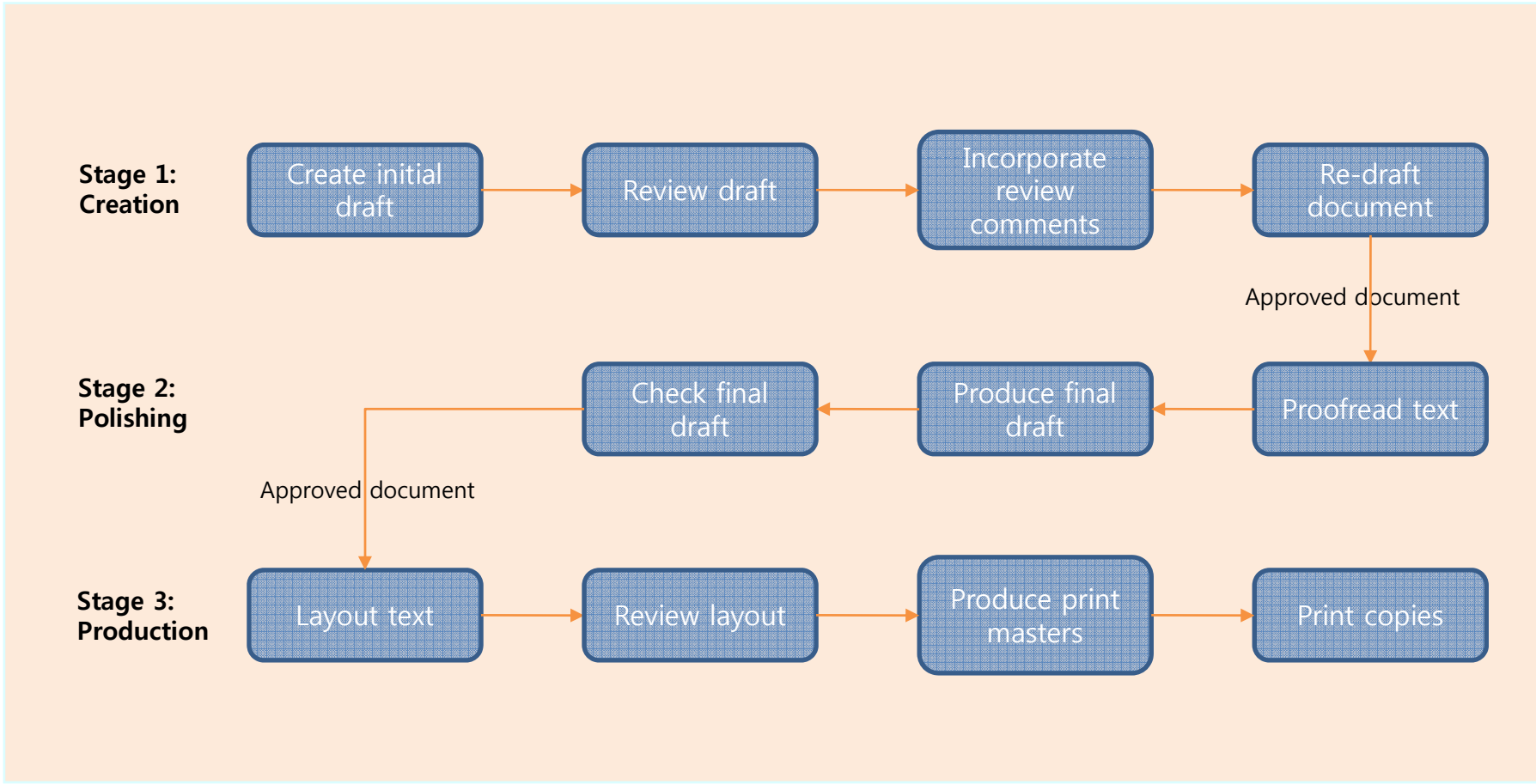
- Some customers require suppliers to be ISO 9000 certified.



Documentation Standards

- Particularly important - documents are the tangible manifestation of the software.
- Documentation process standards
  - Concerned with how documents should be developed, validated and maintained.
- Document standards
  - Concerned with document contents, structure, and appearance.
- Document interchange standards
  - Concerned with the compatibility of electronic documents.

Documentation Process



Document Standards

- Document identification standards
  - How documents are uniquely identified.
- Document structure standards
  - Standard structure for project documents
- Document presentation standards
  - Define fonts and styles, use of logos, etc.
- Document update standards
  - Define how changes from previous versions are reflected in a document.

Document Interchange Standards

- Document interchange standards allow electronic documents to be exchanged, mailed, etc.
  - Needed to define conventions for their use e.g. use of style sheets and macros.

- Need for archiving.
  - The lifetime of word processing systems may be much less than the lifetime of the software being documented.
  - An archiving standard may be defined to ensure that the document can be accessed in future.

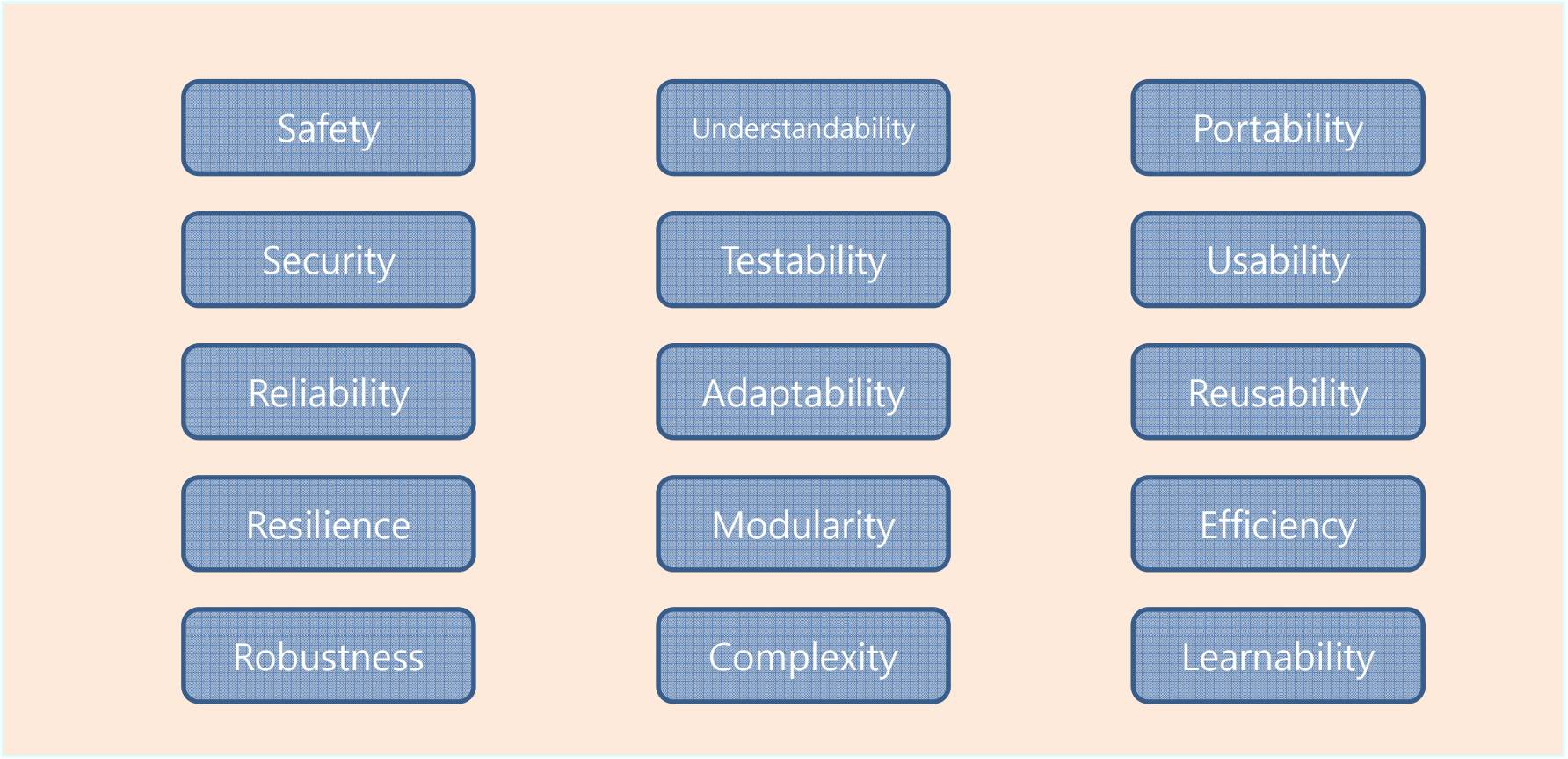
Quality Planning

- Quality plans
  - Set out the desired product qualities and how these are assessed.
  - Defines the most significant quality attributes.
  - Should define the quality assessment process.
  - Set out which, where and when organizational standards be applied.

- Quality plan structure
  - Product introduction
  - Product plans
  - Process descriptions
  - Quality goals
  - Risks and risk management

- Quality plans should be short, succinct documents
  - If they are too long, no-one will read them.

Software Quality Attributes



Quality Control

- Quality control involves checking the software development process to ensure that procedures and standards are being followed.

- Two approaches to quality control
  - Quality reviews
  - Automated software assessment and software measurement

Quality Reviews

- Quality reviews are the principal method of validating the quality of a process or of a product.
- A group examines part or all of a process or system and its documentation to find potential problems.

- Different types of review with different objectives
  - Inspections : for defect removal (product)
  - Reviews : for progress assessment (product and process)
  - Quality reviews (product and standards).

Types of Review

Property	Principal Purpose
Design or Program Inspections	To detect detailed errors in the requirements, design or code. A checklist of possible errors should drive the review.
Progress Reviews	To provide information for management about the overall progress of the project. This is both a process and a product review and is concerned with costs, plans and schedules.
Quality Reviews	To carry out a technical analysis of product components or documentation to find mismatches between the specification and the component design, code or documentation and to ensure that defined quality standards have been followed.

Quality Reviews

- Quality reviews carefully examine part or all of a software system and its associated documentation.
  - Code, designs, specifications, test plans, standards, etc. can all be reviewed.
  - Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management.

- Any documents produced in the process may be reviewed.
- Review teams should be relatively small and reviews should be fairly short.
- Records should always be maintained of quality reviews.

Review functions

- Quality function
  - Part of the general quality management process
- Project management function
  - Provide information for project managers.
- Training and communication function
  - Product knowledge is passed between development team members.

Review Results

- Comments made during the review should be classified
  - No action : No change to the software or documentation is required.
  - Refer for repair : Designer or programmer should correct an identified fault.
  - Reconsider overall design : The problem identified in the review impacts other parts of the design. Some overall judgement must be made about the most cost-effective way of solving the problem.

- Requirements and specification errors may have to be referred to the client.

Software Measurement and Metrics

- Software measurement is concerned with deriving a numeric value for an attribute of a software product or process.
  - Allows for objective comparisons between techniques and processes.

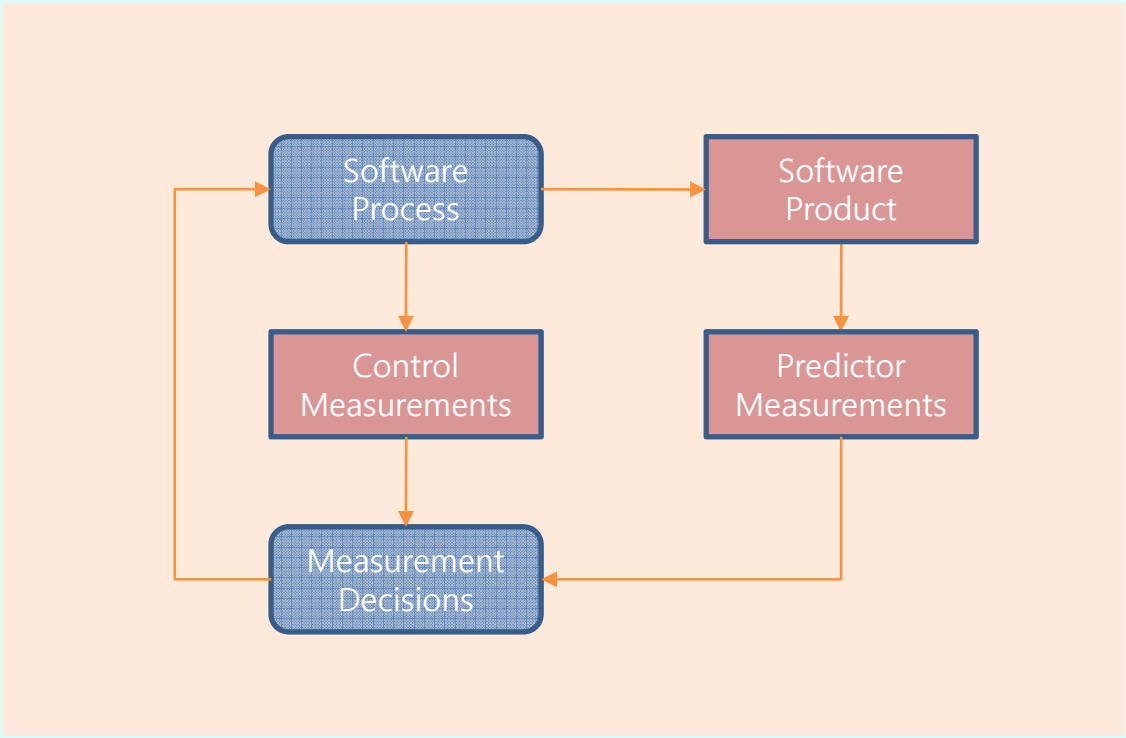
- Although some companies have introduced measurement programs, most organizations still don't make systematic use of software measurement.
- Few established standards in this area.

Software Metric

- Any type of measurement which relates to a software system, process or related documentation
  - Lines of code in a program
  - The Fog index
  - Number of person-days required to develop a component
  - etc.

- Allow the software and the software process to be quantified.
  - May be used to predict product attributes
  - May be used to control the software process.
  - Can be used for general predictions.
  - Can be used to identify anomalous components.

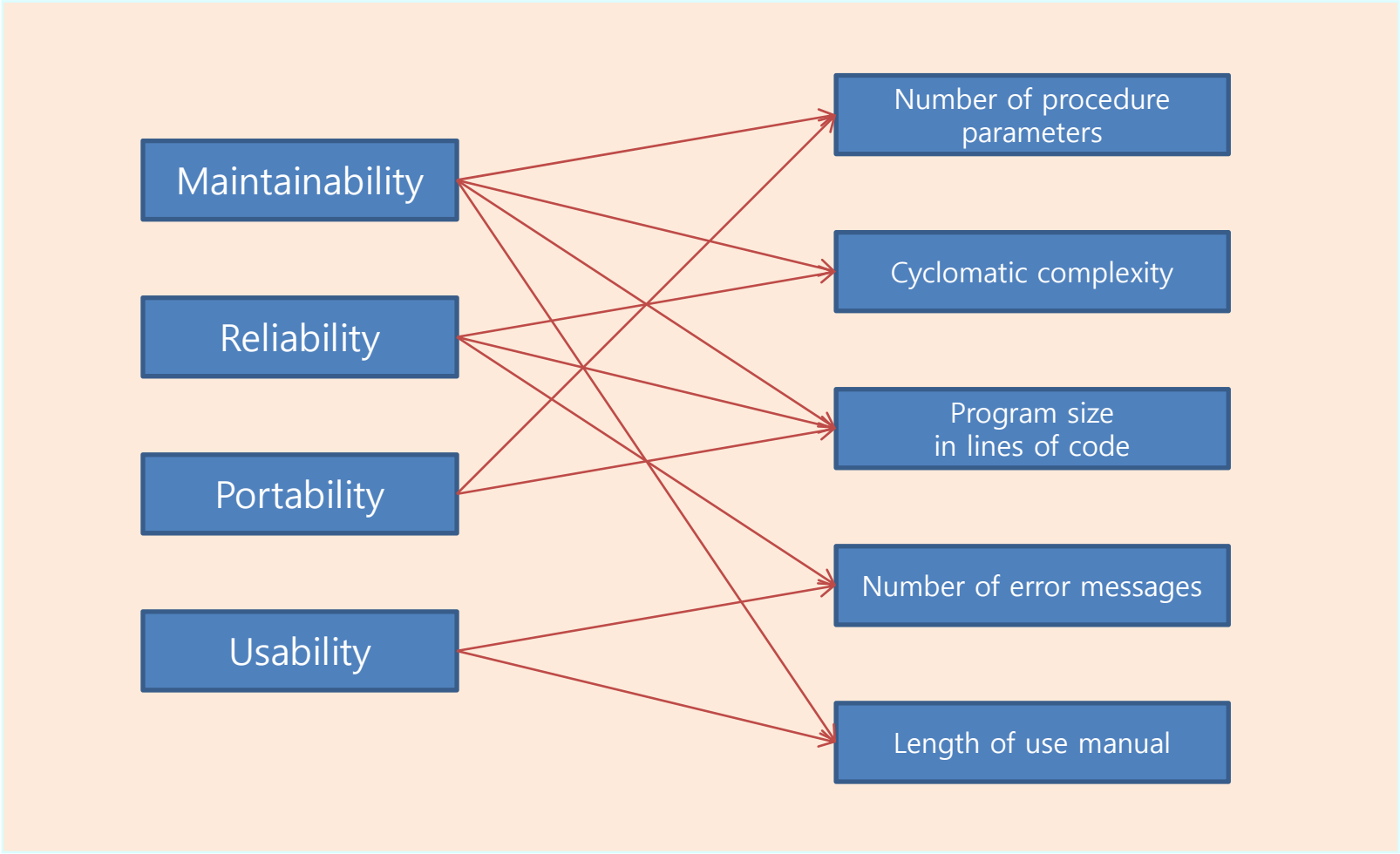
Predictor and Control Metrics



Metrics Assumptions

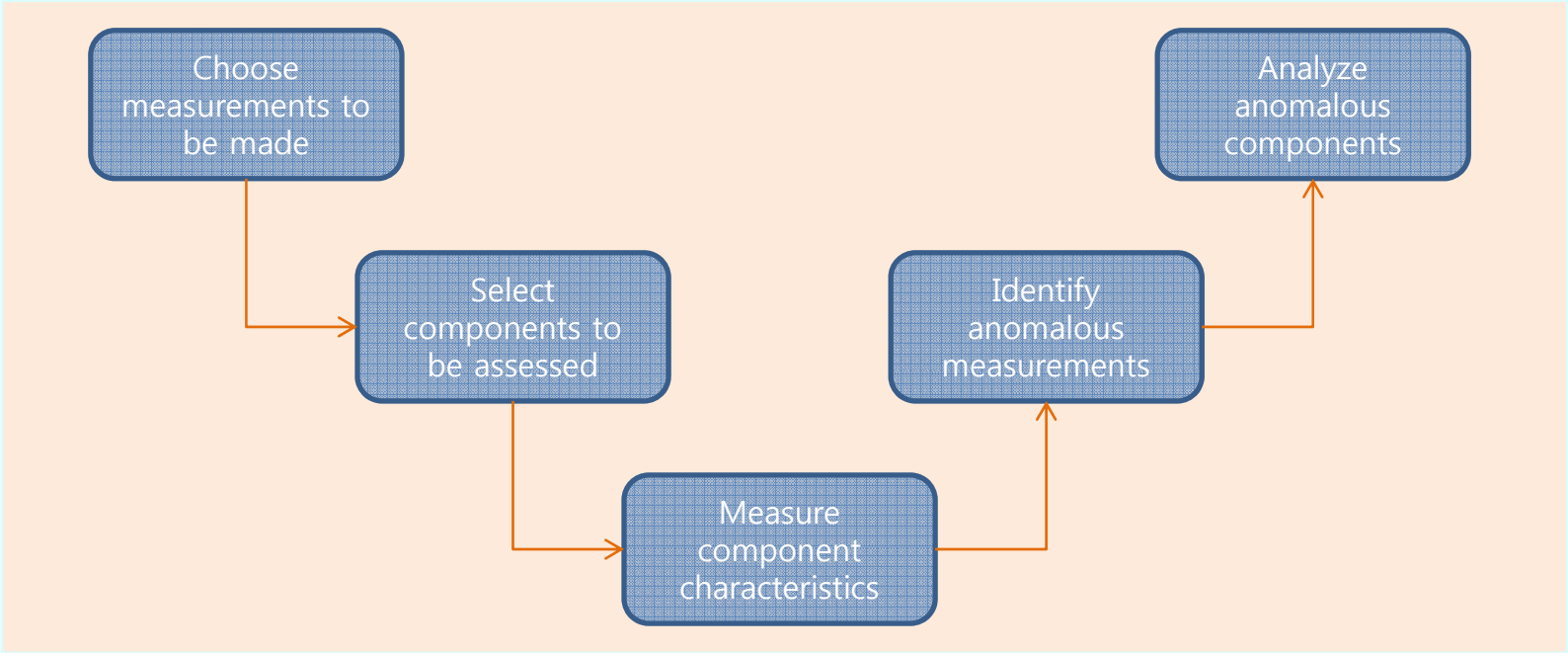
- A software property can be measured.
- The relationship exists between what we can measure and what we want to know. We can only measure internal attributes but are often more interested in external software attributes.
- This relationship has been formalized and validated.
- It may be difficult to relate what can be measured to desirable external quality attributes.

Internal and External Attributes



Measurement Process

- A software measurement process may be a part of a quality control process.
  - Data collected during this process should be maintained as an organizational resource.
  - Once a measurement database has been established, comparisons across projects become possible.



Data Collection

- A metrics programme should be based on a set of product and process data.
- Data should be collected immediately (not in retrospect) and, if possible, automatically.

- Three types of automatic data collection
  - Static product analysis
  - Dynamic product analysis
  - Process data collation

Data Accuracy

- Don't collect unnecessary data
  - The questions to be answered should be decided in advance and the required data identified.
- Tell people why the data is being collected.
  - It should not be part of personnel evaluation.
- Don't rely on memory
  - Collect data when it is generated not after a project has finished.

Product Metrics

- A quality metric should be a predictor of product quality.

- Classes of product metric
  - Dynamic metrics : Collected by measurements made of a program in execution.
  - Static metrics : Collected by measurements made of the system representations

  - Dynamic metrics help assess efficiency and reliability.
  - Static metrics help assess complexity, understandability and maintainability.

Dynamic and Static Metrics

- Dynamic metrics are closely related to software quality attributes
  - Relatively easy to measure the response time of a system (performance attribute) or the number of failures (reliability attribute).

- Static metrics have an indirect relationship with quality attributes
  - Need to try and derive a relationship between these metrics and properties such as complexity, understandability and maintainability.

Software Product Metrics

Software Metric	Description
Fan-in / Fan-out	Fan-in is a measure of the number of functions or methods that call some other 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 how to compute cyclomatic complexity in Chapter 22.
Length of identifiers	This is a measure of the average length of distinct identifiers 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 are potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value for the Fog index, the more difficult the document is to understand.

Object-Oriented Metrics

Object-Oriented Metric	Description
Depth of inheritance tree	This represents the number of discrete levels in the inheritance tree where sub-classes inherit attributes and operations (methods) from super-classes. The deeper the inheritance tree, the more complex the design. Many different object classes may have to be understood to understand the object classes at the leaves of the tree.
Method fan-in/fan-out	This is directly related to fan-in and fan-out as described above and means essentially the same thing. However, it may be appropriate to make a distinction between calls from other methods within the object and calls from external methods.
Weighted methods per class	This is the number of methods that are included in a 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 more difficult to understand. They may not be logically cohesive so cannot be reused effectively as super-classes in an inheritance tree.
Number of overriding operations	This is the number of operations in a super-class that are over-ridden in a sub-class. A high value for this metric indicates that the super-class used may not be an appropriate parent for the sub-class.

Measurement Analysis

- It is not always obvious what data means
  - Analysing collected data is very difficult.

- Professional statisticians should be consulted if available.
- Data analysis must take local circumstances into account.

Summary

- Software quality management is concerned with ensuring that software meets its required standards.
- Quality assurance procedures should be documented in an organizational quality manual.

- Software standards are an encapsulation of best practice.
- Reviews are the most widely used approach for assessing software quality.
- Software measurement gathers information about both the software process and the software product.
- Product quality metrics should be used to identify potentially problematical components.
- There are no standardized and universally applicable software metrics.

Chapter 28.

Process Improvement

Objectives

- To explain the principles of software process improvement
- To explain how software process factors influence software quality and productivity
- To explain how to develop simple models of software processes
- To explain the notion of process capability and the CMMI process improvement model

Process Improvement

- Understanding existing processes and introducing process changes to improve product quality, reduce costs or accelerate schedules.

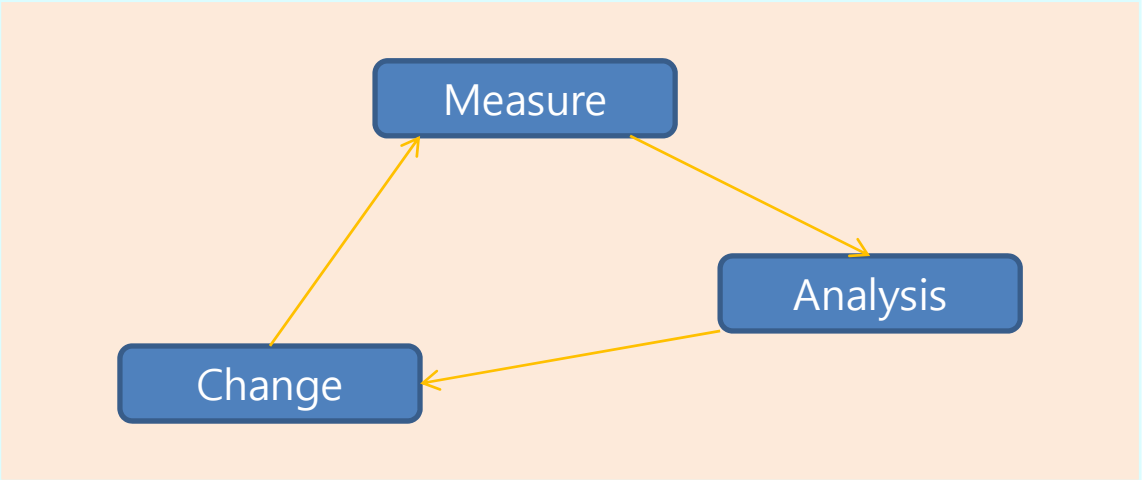
- Most process improvement work so far has focused on defect reduction. This reflects the increasing attention paid by industry to quality.
- However, other process attributes can also be the focus of improvement.

Process Attributes

Process Attributes	Description
Understandability	To what extent is the process explicitly defined and how easy is it to understand the process definition?
Visibility	Do the process activities culminate in clear results so that the progress of the process is externally visible?
Supportability	To what extent can CASE tools be used to support the process activities?
Acceptability	Is the defined process acceptable to and usable by the engineers responsible for producing the software product?
Reliability	Is the process designed in such a way that process errors are avoided or trapped before they result in product errors?
Robustness	Can the process continue in spite of unexpected problems?
Maintainability	Can the process evolve to reflect changing organisational requirements or identified process improvements?
Rapidity	How fast can the process of delivering a system from a given specification be completed?

Process Improvement Cycle

- Process measurement
  - Attributes of the current process are measured. These are a baseline for assessing improvements.
- Process analysis
  - The current process is assessed and bottlenecks and weaknesses are identified.
- Process change
  - Changes to the process that have been identified during the analysis are introduced.

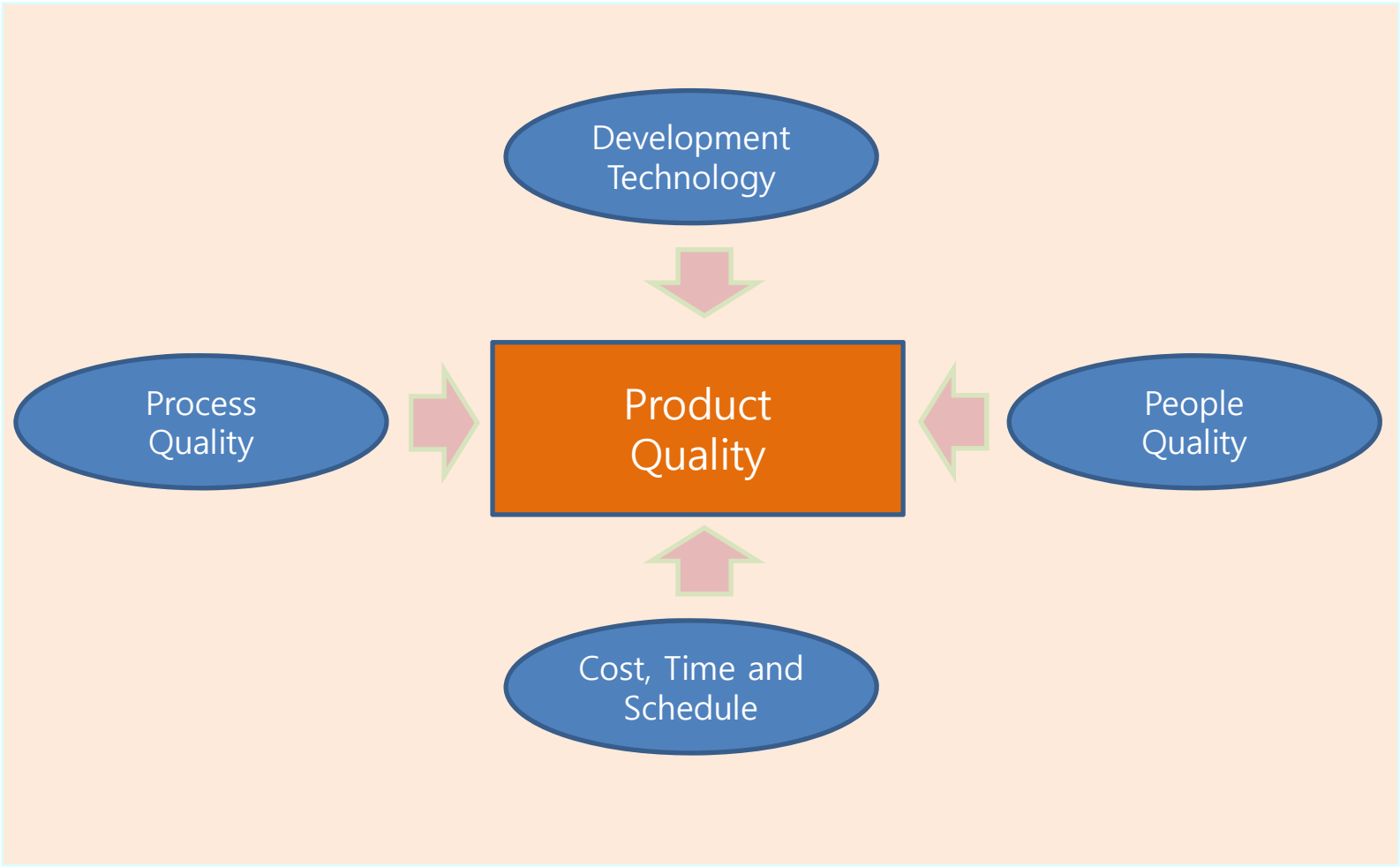


Process and Product Quality

- Process quality and product quality are closely related.
  - The quality of the product depends on its development process.

- A good process is usually required to produce a good product.
  - For manufactured goods, process is the principal quality determinant.
  - For design-based activity, other factors are also involved especially the capabilities of the designers.

Principal Product Quality Factors



Quality Factors

- For large projects with 'average' capabilities, the development process determines product quality.

- For small projects, the capabilities of the developers is the main determinant.
- The development technology is particularly significant for small projects.

- In all cases, if an unrealistic schedule is imposed then product quality will suffer.

Process Classification

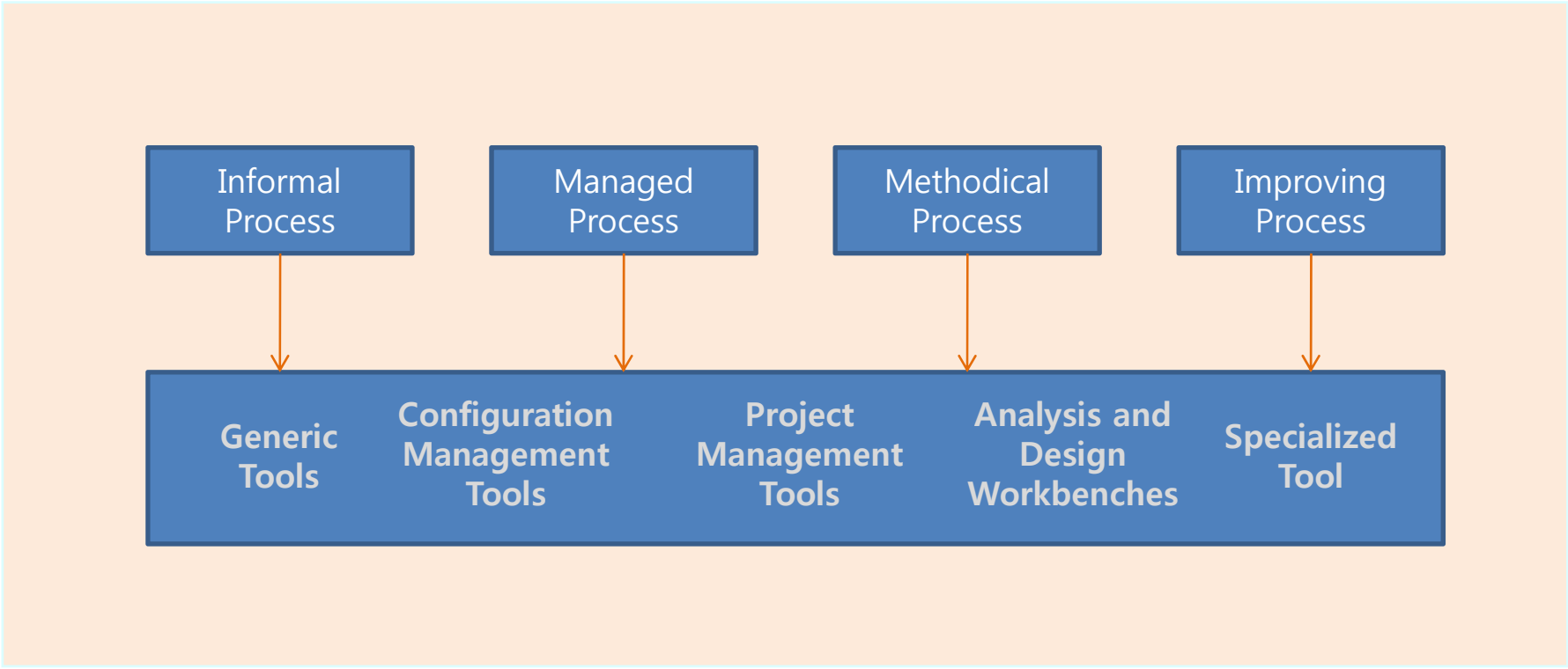
- Informal
  - No detailed process model.
  - Development team chose their own way of working.
- Managed
  - Defined process model which drives the development process.
- Methodical
  - Processes supported by some development method such as the RUP.
- Supported
  - Processes supported by automated CASE tools.

Process Choice

- Process used should depend on type of product being developed
  - For large systems, management is usually the principal problem so we need a strictly managed process.
  - For smaller systems, more informality is possible.

- No uniformly applicable process which should be standardized within an organisation
  - High costs may be incurred if you force an inappropriate process on a development team.
  - Inappropriate methods can also increase costs and lead to reduced quality.

Process Tool Support



Process Measurement

- Wherever possible, quantitative process data should be collected
  - However, where organizations do not have clearly defined process standards, this is very difficult as you don't know what to measure.
  - A process may have to be defined before any measurement is possible.

- Process measurements should be used to assess process improvements
  - But, this does not mean that measurements should drive the improvements.
  - The improvement driver should be the organizational objectives.

Classes of Process Measurement

- Time taken for process activities to be completed
  - E.g. Calendar time or effort to complete an activity or process
- Resources required for processes or activities
  - E.g. Total effort in person-days
- Number of occurrences of a particular event
  - E.g. Number of defects discovered

Goal-Question-Metric Paradigm

- Goals
  - What is the organisation trying to achieve?
  - The objective of process improvement is to satisfy these goals.
- Questions
  - Questions about areas of uncertainty related to the goals.
  - You need process knowledge to derive these.
- Metrics
  - Measurements to be collected to answer the questions

Process Analysis and Modelling

- Process analysis
  - Study existing processes to understand the relationships between parts of the process and to compare them with other processes.

- Process modelling
  - Documentation of a process which records the tasks, the roles and the entities used
  - May be presented from different perspectives.

- Study an existing process to understand its activities.
- Produce an abstract model of the process.
  - Normally represent the model graphically.
  - Several different views (e.g. activities, deliverables, etc.) may be required.
- Analyse the model to discover process problems.
  - Involves discussing process activities with stakeholders and discovering problems and possible process changes.

Process Analysis Techniques

- Published process models and process standards
  - It is always best to start process analysis with an existing model.
  - People then may extend and change it.
- Questionnaires and interviews
  - Must be carefully designed.
  - Participants may tell you what they think you want to hear.
- Ethnographic analysis
  - Involves assimilating process knowledge by observation.
  - Best for in-depth analysis of process fragments rather than for whole-process understanding.

Process Model Elements 1

Process Model Elements	Graphical Notation	Description
Activity	A round-edged rectangle with no drop shadow	An activity has a clearly defined objective, entry and exit conditions. Examples of activities are preparing a set of test data to test a module, coding a function or a module, proof-reading a document, etc. Generally, an activity is atomic i.e. it is the responsibility of one person or group. It is not decomposed into sub-activities.
Process	A round-edged rectangle with drop shadow	A process is a set of activities which have some coherence and whose objective is generally agreed within an organisation. Examples of processes are requirements analysis, architectural design, test planning, etc.
Deliverable	A rectangle with drop shadow	A deliverable is a tangible output of an activity that is predicted in a project plan.
Condition	A parallelogram	A condition is either a pre-condition that must hold before a process or activity can start or a post-condition that holds after a process or activity has finished.
Role	A circle with drop	A role is a bounded area of responsibility. Examples of roles might be configuration manager, test engineer, software designer, etc. One person may have several different roles and a single role may be associated with several different people.
Exception	May be represented as a double edged box	An exception is a description of how to modify the process if some anticipated or unanticipated event occurs. Exceptions are often undefined and it is left to the ingenuity of the project managers and engineers to handle the exception.
Communication	An arrow	An interchange of information between people or between people and supporting computer systems. Communications may be informal or formal. Formal communications might be the approval of a deliverable by a project manager; informal communications might be the interchange of electronic mail to resolve ambiguities in a document.

Process Exceptions

- Software processes are complex and process models cannot effectively represent how to handle exceptions
  - Several key people becoming ill just before a critical review.
  - A breach of security that means all external communications are out of action for several days.
  - Organizational reorganization
  - A need to respond to an unanticipated request for new proposals.

- Under these circumstances, the model is suspended and managers use their initiative to deal with the exception.
- We have to avoid the exceptions or change the process itself.

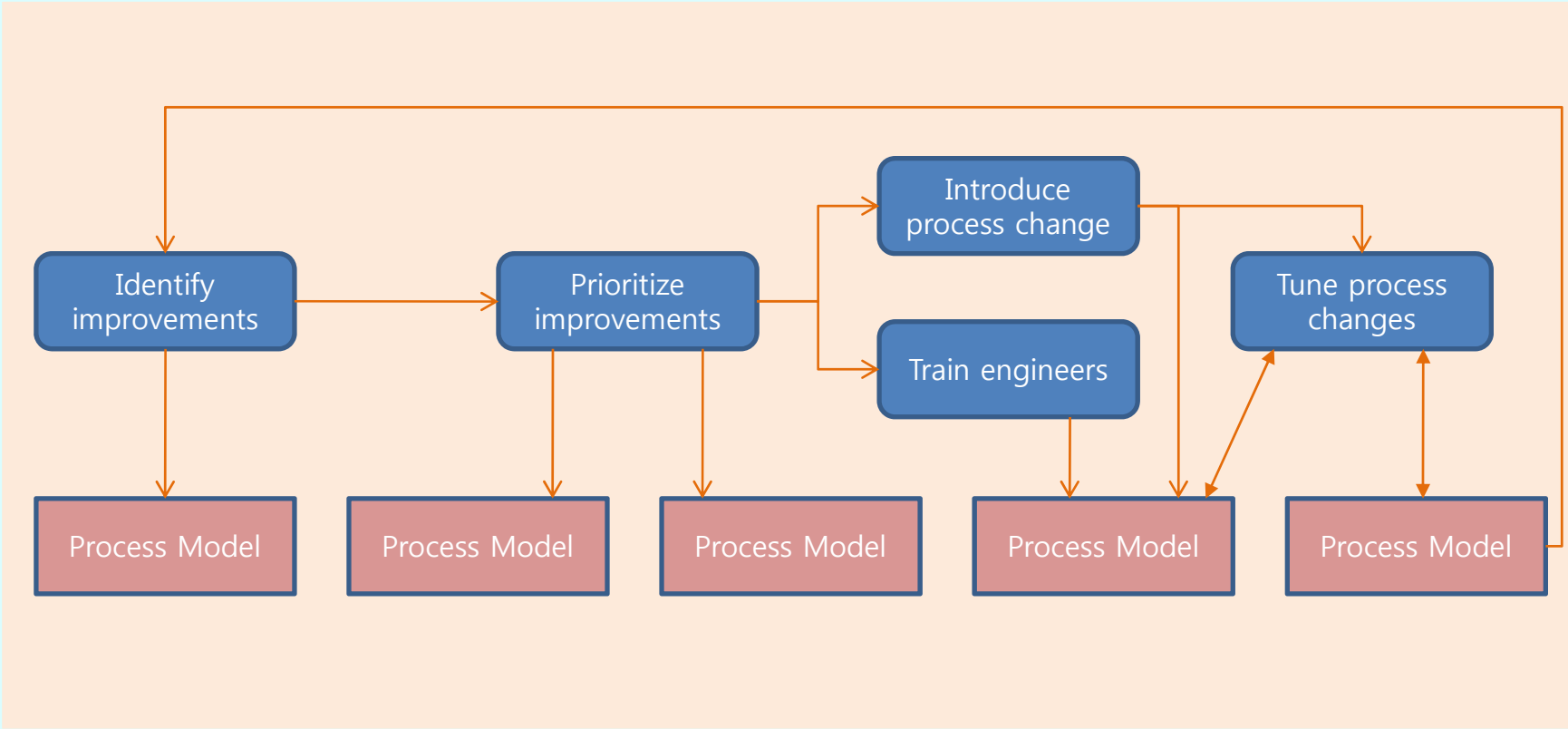
Process Change

- Process changes involve making modifications to existing processes.
  - Introduce new practices, methods or processes.
  - Change the ordering of process activities.
  - Introduce or remove deliverables.
  - Introduce new roles or responsibilities.

- Change should be driven by measurable goals.

- Process change stages
  - Improvement identification
  - Improvement prioritization
  - Process change introduction
  - Process change training
  - Change tuning

Process Change Process



The CMMI Framework

- The CMMI framework is the current stage of work on process assessment and improvement.
  - Started at the SEI(Software Engineering Institute) in the 1980s.
  - The SEI's mission is to promote software technology transfer particularly to US defence contractors.

- It has had a profound influence on process improvement
  - Capability Maturity Model introduced in the early 1990s.
  - Revised maturity framework (CMMI) introduced in 2001.

Process Capability Assessment

- Intended as a means to assess the extent to which an organization's processes follow best practice.
  - It is possible to identify areas of weakness for process improvement.
  - There have been various process assessment and improvement models but the SEI work has been most influential.

The SEI Capability Maturity Model

- Initial
  - Essentially uncontrolled.
- Repeatable
  - Product management procedures defined and used.
- Defined
  - Process management procedures and strategies defined and used.
- Managed
  - Quality management strategies defined and used.
- Optimizing
  - Process improvement strategies defined and used.

Problems with the CMM

- Practices associated with model levels
  - Companies could be using practices from different levels at the same time, but if all practices from a lower level were not used, it was not possible to move beyond that level.
- Discrete rather than continuous
  - Did not recognize distinctions between the top and the bottom of levels.
- Practice-oriented
  - Concerned with how things were done (the practices) rather than the goals to be achieved.

The CMMI Model

- An integrated capability model that includes software and systems engineering capability assessment.

- Two instantiations
  - Staged where the model is expressed in terms of capability levels.
  - Continuous where a capability rating is computed.

CMMI model components

- Process areas
  - 24 process areas that are relevant to process capability and improvement are identified.
  - Organized into 4 groups.
- Goals
  - Goals are descriptions of desirable organizational states.
  - Each process area has associated goals.
- Practices
  - Practices are ways of achieving a goal.
  - They are just advisory and other approaches to achieve the goal may be used.

CMMI Process Areas

CMMI Process Area	Description
Process Management	Organisational process definition Organisational process focus Organisational training Organisational process performance Organisational innovation and deployment
Project Management	Project planning Project monitoring and control Supplier agreement management Integrated project management
	Risk management Integrated teaming Quantitative project management
Engineering	Requirements management Requirements development Technical solution Product integration Verification Validation
Support	Configuration management Process and product quality management Measurement and analysis Decision analysis and resolution Organisational environment for integration Causal analysis and resolution

CMMI Goals

CMMI Goals	Process Area
Corrective actions are managed to closure when the project's performance or results deviate significantly from the plan.	**Project Monitoring and control**
Actual performance and progress of the project is monitored against the project plan.	**Project monitoring and control**
The requirements are analysed and validated and a definition of the required functionality is developed.	**Requirements development**
Root causes of defects and other problems are systematically determined.	**Causal analysis and resolution**
The process is institutionalised as a defined process.	**Generic goal**

CMMI Practices

Practice	Associated Goal
Analyse derived requirements to ensure that they are necessary and sufficient	The requirements are analysed and validated and a definition of the required functionality is developed.
Validate requirements to ensure that the resulting product will perform as intended in the user's environment using multiple techniques as appropriate.	
Select the defects and other problems for analysis.	Root causes of defects and other problems are systematically determined.
Perform causal analysis of selected defects and other problems and propose actions to address them.	
Establish and maintain an organisational policy for planning and performing the requirements development process.	The process is institutionalised as a defined process.
Assign responsibility and authority for performing the process, developing the work products and providing the services of the requirements development process.	

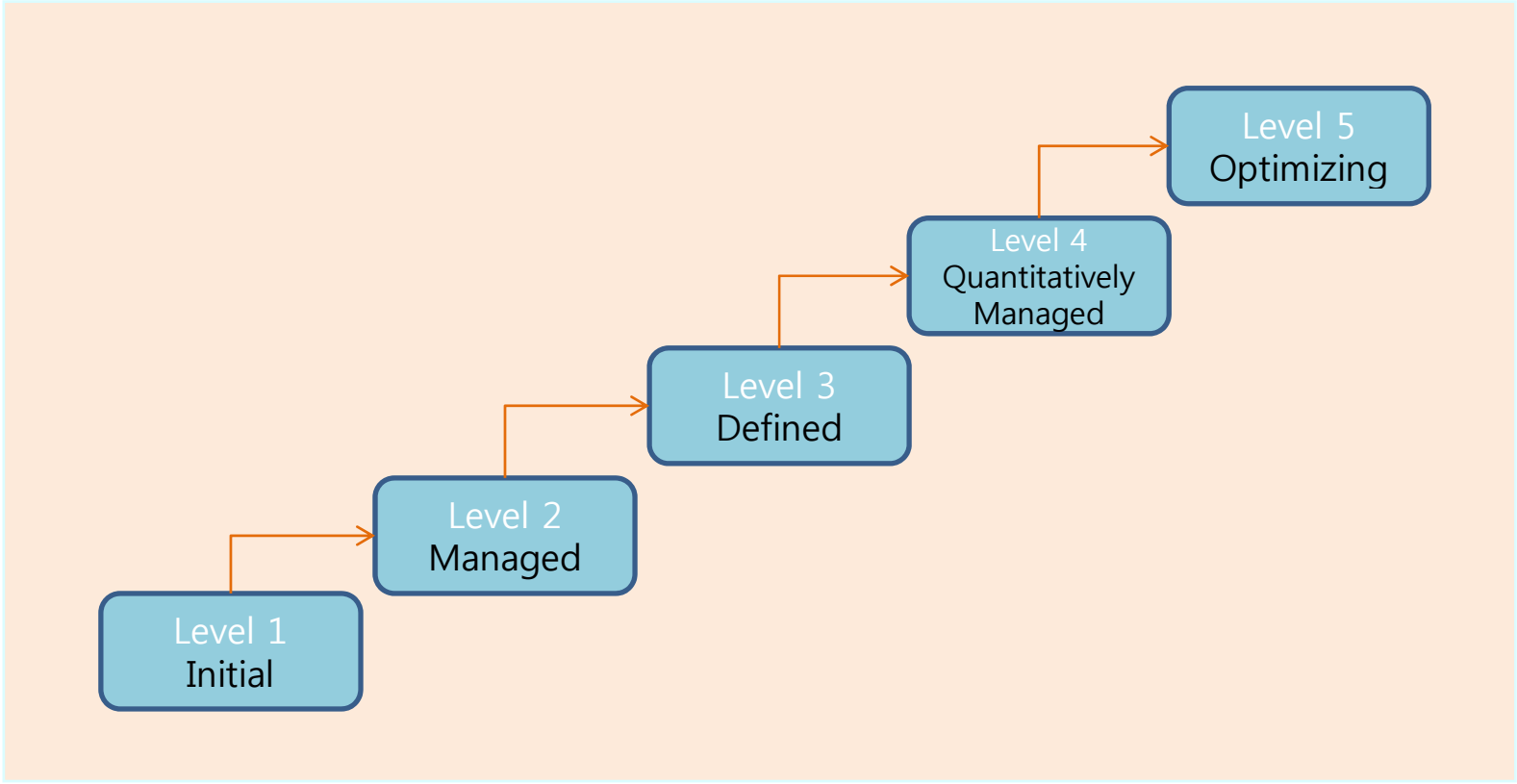
CMMI Assessment

- Examines the processes used in an organization and assesses their maturity in each process area.

- Based on a 6-point scale (6 levels)
  - Not performed
  - Performed
  - Managed
  - Defined
  - Quantitatively managed
  - Optimizing

The Staged CMMI Model

- Comparable with the software CMM.
- Each maturity level has process areas and goals.



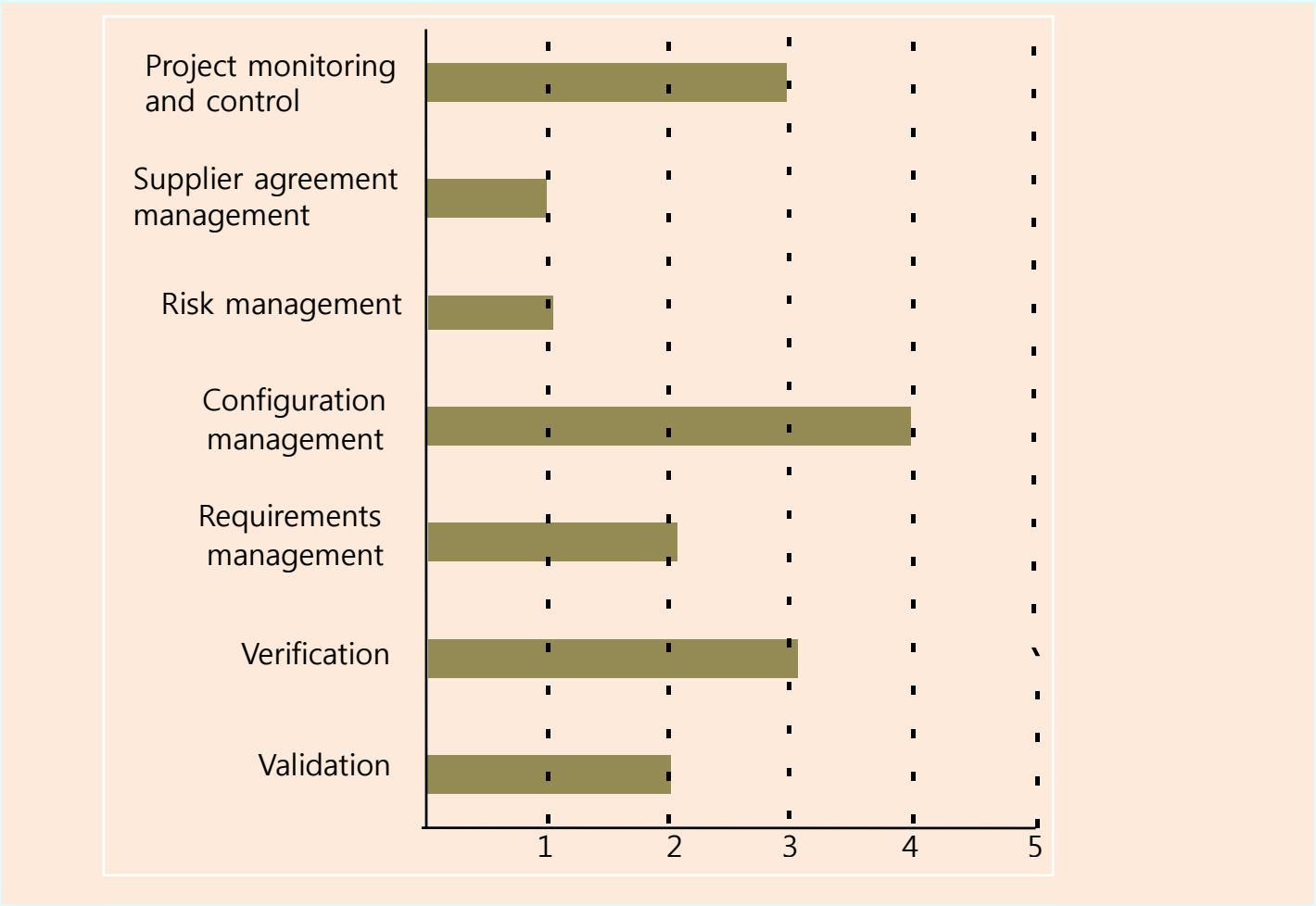
Institutional Practices

- Institutions operating at the managed level should have institutionalized practices that are geared to standardization. (Level 2 → Level 3)
  - Establish and maintain policy for performing the project management process.
  - Provide adequate resources for performing the project management process.
  - Monitor and control the project planning process.
  - Review the activities, status and results of the project planning process.

The Continuous CMMI Model

- A finer-grain model that considers individual or groups of practices and assesses their use.
  - The maturity assessment is not a single value but is a set of values showing the organisations maturity in each area.
  - The CMMI rates each process area from levels 1 to 5.
  - The advantage of a continuous approach is that organizations can pick and choose process areas to improve according to their local needs.

A Process Capability Profile



Summary

- Process improvement involves process analysis, standardisation, measurement and change.
- Processes can be classified as informal, managed, methodical and improving. This classification can be used to identify process tool support.
- The process improvement cycle involves process measurement, process analysis and process change.
- Process measurement should be used to answer specific process questions, based on organisational improvement goals.
- The three types of process metrics used in the measurement process are time metrics, resource utilisation metrics and event metrics.
- Process models include descriptions of tasks, activities, roles, exceptions, communications, deliverables and other processes.
- The CMMI process maturity model integrates software and systems engineering process improvement.
- Process improvement in the CMMI model is based on reaching a set of goals related to good software engineering practice.

Chapter 29.

Configuration Management

Objectives

- To explain the importance of software configuration management (CM)
- To describe key CM activities namely CM planning, change management, version management and system building
- To discuss the use of CASE tools to support configuration management processes

Configuration Management

- New versions of software systems are created as they change
  - For different machines/OS
  - Offering different functionality
  - Tailored for particular user requirements

- Configuration management(CM) is concerned with managing evolving software systems
  - System change is a team activity.
  - Aims to control the costs and effort involved in making changes.
  - Involves the development and application of procedures and standards to manage an evolving software product.
  - May be seen as part of a more general quality management process.
  - When released to CM, software systems are sometimes called baselines.

CM Standards

- CM should always be based on a set of standards which are applied within an organization.
  - Standards should define how items are identified, how changes are controlled and how new versions are managed.
  - Standards may be based on external CM standards (e.g. IEEE standard for CM).
  - Some existing standards are based on a waterfall process model.
  - New CM standards are needed for evolutionary development.

Frequent System Building

- Frequent system building
  - A new version of a system is built from components by compiling and linking them.
  - This new version is delivered for testing using pre-defined tests.
  - Faults that are discovered during testing are documented and returned to the system developers.

- It is easier to find problems that stem from component interactions early in the process.
  - This encourages thorough unit testing - developers are under pressure not to 'break the build'.
  - A stringent change management process is required to keep track of problems that have been discovered and repaired.

Configuration Management Planning

- All products of the software process may have to be managed
  - Specifications
  - Designs
  - Programs
  - Test data
  - User manuals

- Thousands of separate documents may be generated for a large, complex software system.

The Configuration Management Plan

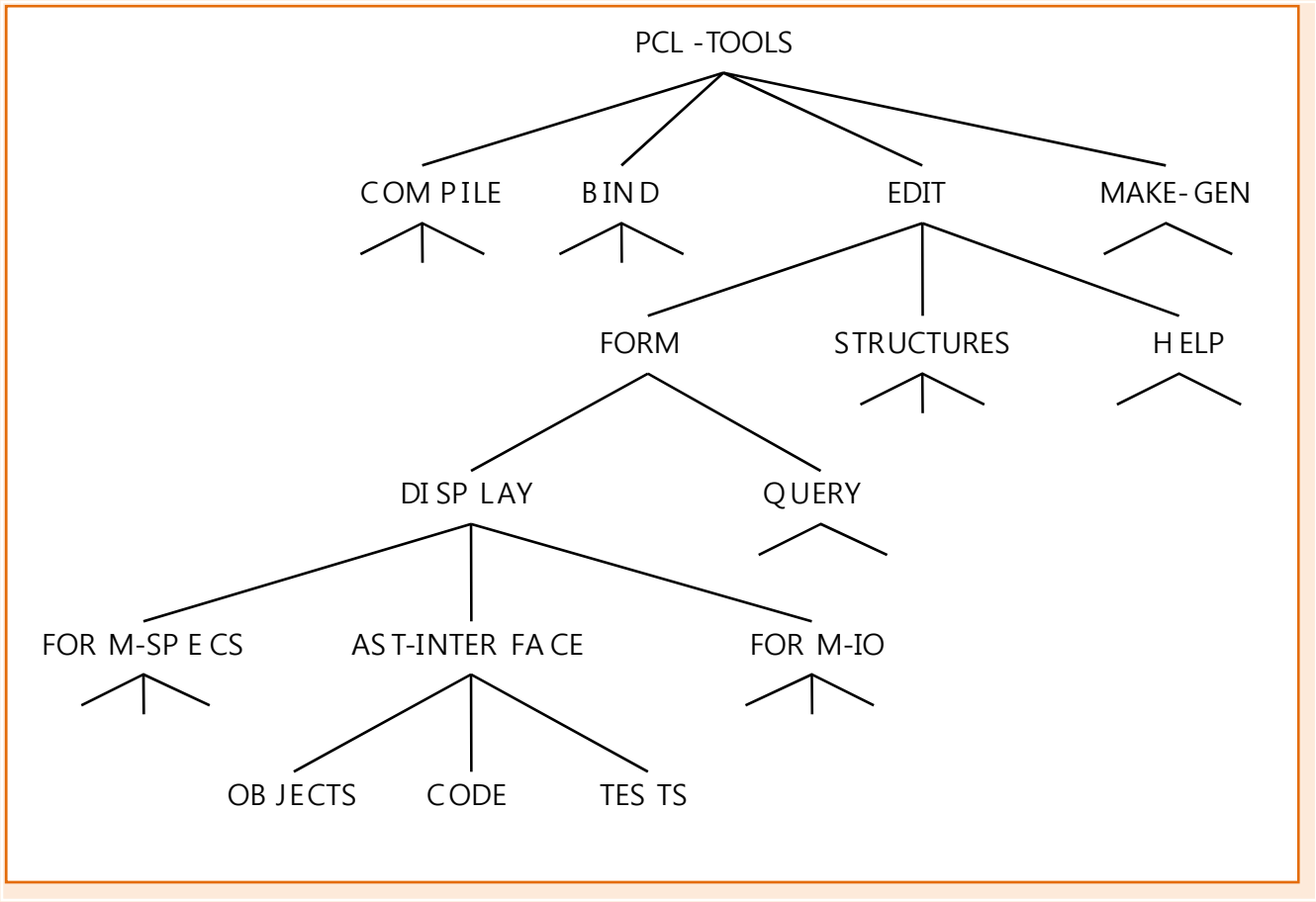
- Defines the types of documents to be managed and a document naming scheme.
- Defines who takes responsibility for the CM procedures and creation of baselines.
- Defines policies for change control and version management.
- Defines the CM records which must be maintained.
- Describes the tools which should be used to assist the CM process and any limitations on their use.
- Defines the process of tool use.
- Defines the CM database used to record configuration information.
- May include information such as the CM of external software, process auditing, etc.

Configuration Item Identification

- Large projects typically produce thousands of documents which must be uniquely identified.
- Some of these documents must be maintained for the lifetime of the software.

- Document naming scheme should be defined so that related documents have related names.
- A hierarchical scheme with multi-level names is probably the most flexible approach.
  - PCL-TOOLS/EDIT/FORMS/DISPLAY/AST-INTERFACE/CODE

Configuration Hierarchy



Configuration Database

- All CM information should be maintained in a configuration database.

- This should allow queries about configurations to be answered
  - Who has a particular system version?
  - What platform is required for a particular version?
  - What versions are affected by a change to component X?
  - How many reported faults in version T?

- The CM database should preferably be linked to the software being managed.

CM Database Implementation

- May be part of an integrated environment to support software development.
  - The CM database and the managed documents are all maintained on the same system.
- CASE tools may be integrated.
  - A close relationship between the CASE tools and the CM tools.
- More commonly, the CM database is maintained separately as this is cheaper and more flexible.

Change Management

- Software systems are subject to continual change requests
  - from users
  - from developers
  - from market forces

- Change management is concerned with
  - Keeping track of these changes
  - Ensuring that they are implemented in the most cost-effective way.

Change Management Process

Request change by completing a change request form
Analyze change request
if change is valid**then**
    Assess how change might be implemented
    Assess change cost
    Submit request to change control board
  if change is accepted**then**
     repeat
        make changes to software
        submit changed software for quality approval
    until software quality is adequate
    create new system version
else
    reject change request
else
    reject change request

Change Request Form

- A change request form records
  - The change proposed
  - Requestor of change
  - The reason why change was suggested
  - The urgency of change (from requestor of the change)

- It also records
  - Change evaluation
  - Impact analysis
  - Change cost
  - Recommendations from system maintenance staff

Change Request Form

Change Request Form

Project: Proteus/PCL-Tools **Number:** 23/02
Change requester: I. Sommerville **Date:** 1/12/02
Requested change: When a component is selected from the structure, display the name of the file where it is stored.

Change analyser: G. Dean **Analysis date:** 10/12/02
Components affected: Display-Icon.Select, Display-Icon.Display

Associated components: FileTable

Change assessment: Relatively simple to implement as a file name table is available. Requires the design and implementation of a display field. No changes to associated components are required.

Change priority: Low

Change implementation:
Estimated effort: 0.5 days

Date to CCB: 15/12/02 **CCB decision date:** 1/2/03
CCB decision: Accept change. Change to be implemented in Release 2.1.
Change implementor: **Date of change:**
Date submitted to QA: **QA decision:**
Date submitted to CM:
Comments

Change Tracking Tools

- A major problem in change management is tracking change status.

- Change tracking tools
  - Keep track the status of each change request .
  - Ensure automatically that change requests are sent to the right people at the right time.
  - Integrated with E-mail systems allowing electronic change request distribution.

Change Control Board

- Changes should be reviewed by an external group who decide whether or not they are cost-effective from a strategic and organizational viewpoint rather than a technical viewpoint.

- The group is called a change control board(CCB).
  - May include representatives from client and contractor staff.

Derivation History

- Derivation history is a record of changes applied to a document or code component.
  - Should record, in outline,
    - The change made
    - The rationale for the change
    - Who made the change
    - When it was implemented.
  - May be included as a comment in code.

Component Header Information

```
// BANKSEC project (IST 6087)
//
// BANKSEC-TOOLS/AUTH/RBAC/USER_ROLE
//
// Object: currentRole
// Author: N. Perwaiz
// Creation date: 10th November 2002
//
// © Lancaster University 2002
//
// Modification history
// Version  Modifier     Date        Change       Reason
// 1.0      J. Jones     1/12/2002   Add header   Submitted to CM
// 1.1      N. Perwaiz   9/4/2003    New field    Change req. R07/02
```

Version and Release Management

- Version and release management
  - Invent an identification scheme for system versions.
  - Plan when a new system version is to be produced.
  - Ensure that version management procedures and tools are properly applied.
  - Plan and distribute new system releases.

- Version
  - An instance of a system which is functionally distinct in some way from other system instances.
- Variant
  - An instance of a system which is functionally identical but non-functionally distinct from other instances of a system.
- Release
  - An instance of a system which is distributed to users outside of the development team.

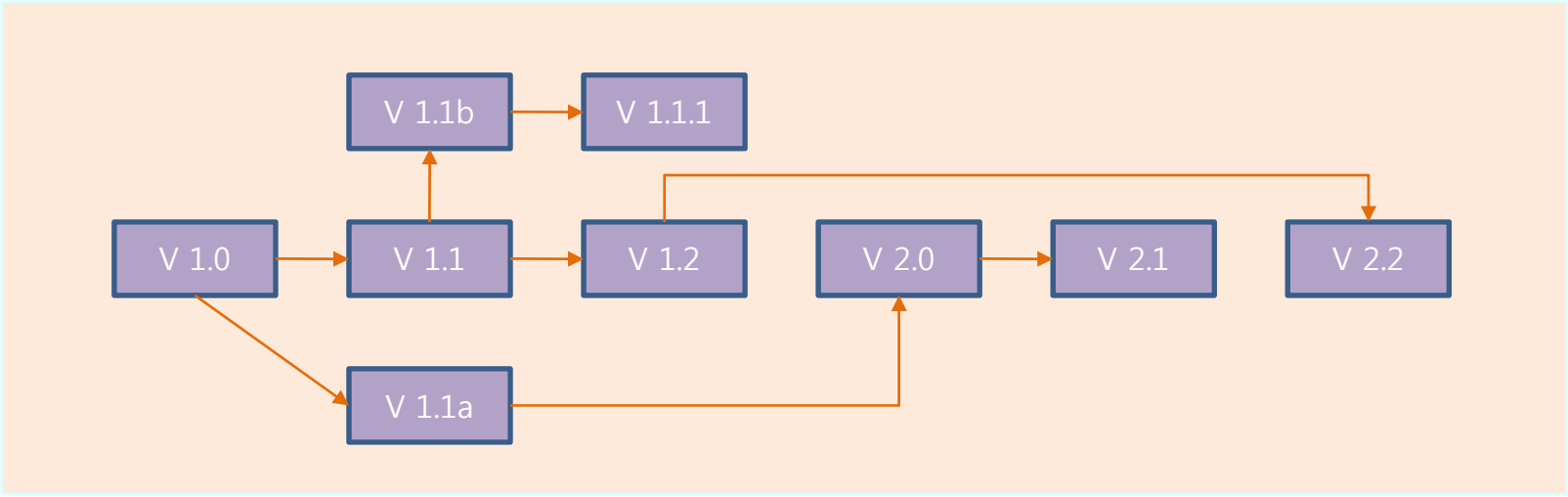
Version Identification

- Version identification should define an unambiguous way of identifying component versions.

- Three basic techniques for component identification
  - Version numbering
  - Attribute-based identification
  - Change-oriented identification

Version Numbering

- Simple naming scheme uses a linear derivation.
  - V1, V1.1, V1.2, V2.1, V2.2 etc.
- The actual derivation structure is a tree or a network rather than a sequence.
  - Version names are not meaningful.
  - A hierarchical naming scheme leads to fewer errors in version identification.



Attribute-Based Identification

- Attributes can be associated with a version with the combination of attributes identifying that version
  - Examples of attributes are Date, Creator, Programming Language, Customer, Status etc.

- More flexible than an explicit naming scheme for version retrieval.
  - May cause problems with uniqueness.
  - The set of attributes have to be chosen so that all versions can be uniquely identified.

- In practice, a version also needs an associated name for easy reference.
  - Example: AC3D (language =Java, platform = XP, date = Jan 2003)

Change-Oriented Identification

- Change-oriented identification integrates versions and the changes made to create these versions.
  - Used for systems rather than components.
  - Each proposed change has a change set that describes changes made to implement that change.
  - Change sets are applied in sequence so that, in principle, a version of the system that incorporates an arbitrary set of changes may be created.

Release Management

- Releases must incorporate changes forced on the system by errors discovered by users and by hardware changes.
  - Must also incorporate new system functionality.

- Release planning is concerned with when to issue a system version as a release.

System Releases

- System release is not just a set of executable programs
- May also include
  - Configuration files defining how the release is configured for a particular installation
  - Data files needed for system operation
  - An installation program or shell script to install the system on target hardware
  - Electronic and paper documentation
  - Packaging and associated publicity

Release Decision Making

- All files required for a release should be re-created when a new release is installed.

- Preparing and distributing a system release is an expensive process.
- Factors such as the technical quality of the system, competition, marketing requirements and customer change requests should all influence the decision of when to issue a new system release.

System Release Strategy

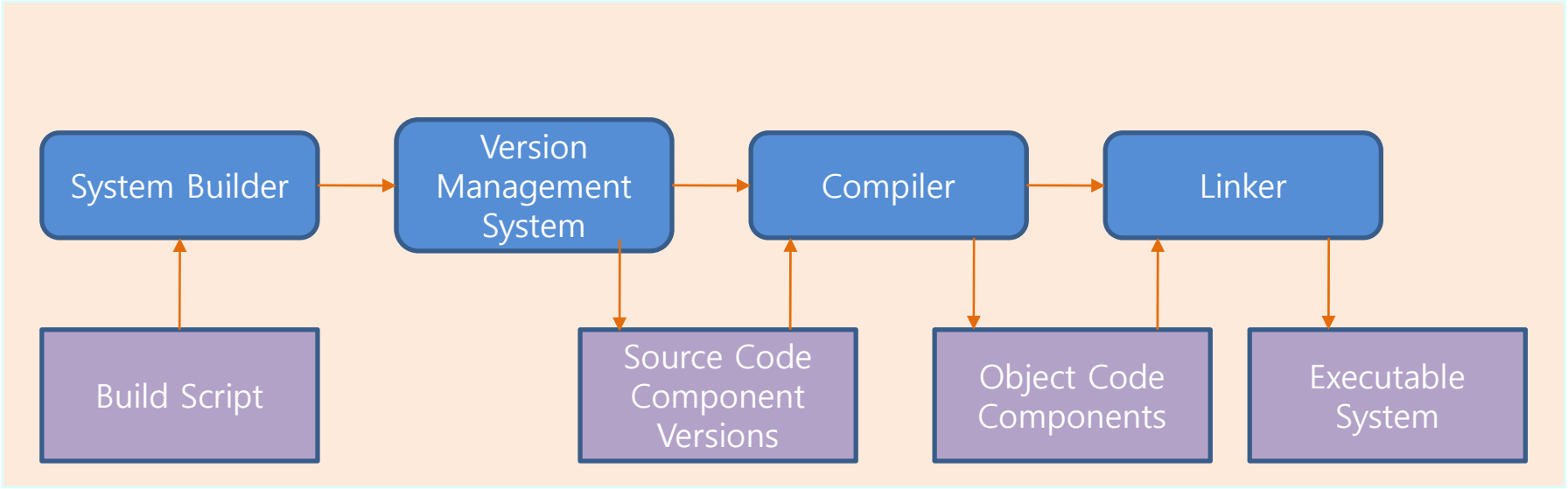
Factor	Description
Technical quality of the system	If serious system faults are reported which affect the way in which many customers use the system, it may be necessary to issue a fault repair release. However, minor system faults may be repaired by issuing patches (often distributed over the Internet) that can be applied to the current release of the system.
Platform changes	You may have to create a new release of a software application when a new version of the operating system platform is released.
Lehman's fifth law (See chapter 21)	This suggests that the increment of functionality that is included in each release is approximately constant. Therefore, if there has been a system release with significant new functionality, then it may have to be followed by a repair release.
Competition	A new system release may be necessary because a competing product is available.
Marketing requirements	The marketing department of an organisation may have made a commitment for releases to be available at a particular date.
Customer change proposals	For customised systems, customers may have made and paid for a specific set of system change proposals and they expect a system release as soon as these have been implemented.

Release Creation

- Release creation involves collecting all files and documentation required to create a system release.
  - Configuration descriptions have to be written for different hardware.
  - Installation scripts have to be written.
  - The specific release must be documented to record exactly what files were used to create it. This allows it to be re-created if necessary.

System Building

- The process of compiling and linking software components into an executable system
  - Different systems are built from different combinations of components.
  - Now always supported by automated tools that are driven by 'build scripts'.



System Building Problems

- Do the build instructions include all required components?
  - When there are many hundreds of components making up a system, it is easy to miss one out. This should normally be detected by the linker.
- Is the appropriate component version specified?
  - A more significant problem. A system built with the wrong version may work initially but fail after delivery.
- Are all data files available?
  - The build should not rely on 'standard' data files. Standards vary from place to place.

System Building Problems

- Are data file references within components correct?
  - Embedding absolute names in code almost always causes problems as naming conventions differ from place to place.
- Is the system being built for the right platform
  - Sometimes you must build for a specific OS version or hardware configuration.
- Is the right version of the compiler and other software tools specified?
  - Different compiler versions may actually generate different code and the compiled component will exhibit different behaviour.

CASE Tools for Configuration Management

- CASE tool support for CM is essential, because
  - CM processes are standardized and involve applying pre-defined procedures.
  - Large amounts of data must be managed.

- Mature CASE tools to support configuration management are available ranging from stand-alone tools to integrated CM workbenches.

CM Workbenches

- Open workbenches
  - Tools for each stage in the CM process are integrated through organizational procedures and scripts.
  - Gives flexibility in tool selection.

- Integrated workbenches
  - Provide whole-process, integrated support for configuration management.
  - More tightly integrated tools so easier to use.
  - However, the cost is less flexibility in the tools used.

Change Management Tools

- Change management is a procedural process so it can be modelled and integrated with a version management system.

- Change management tools
  - Form editor to support processing the change request forms
  - Workflow system to define who does what and to automate information transfer
  - Change database that manages change proposals and is linked to a VM system
  - Change reporting system that generates management reports on the status of change requests

Version Management Tools

- Version and release identification
  - Systems assign identifiers automatically when a new version is submitted to the system.
- Storage management.
  - System stores the differences between versions rather than all the version code.
- Change history recording
  - Record reasons for version creation.
- Independent development
  - Only one version at a time may be checked out for change. Parallel working on different versions.
- Project support
  - Can manage groups of files associated with a project rather than just single files.

System Building

- Building a large system is computationally expensive and may take several hours.
- Hundreds of files may be involved.

- System building tools may provide
  - A dependency specification language and interpreter
  - Tool selection and instantiation support
  - Distributed compilation
  - Derived object management

Summary

- Configuration management is the management of system change to software products.
- A formal document naming scheme should be established and documents should be managed in a database.
- The configuration data base should record information about changes and change requests.
- A consistent scheme of version identification should be established using version numbers, attributes or change sets.
- System releases include executable code, data, configuration files and documentation.
- System building involves assembling components into a system.
- CASE tools are available to support all CM activities.
- CASE tools may be stand-alone tools or may be integrated systems which integrate support for version management, system building and change management.