Module 5
Evolution of Software and Project Management

Vaishnavi
Assistant Professor

Software Evolution

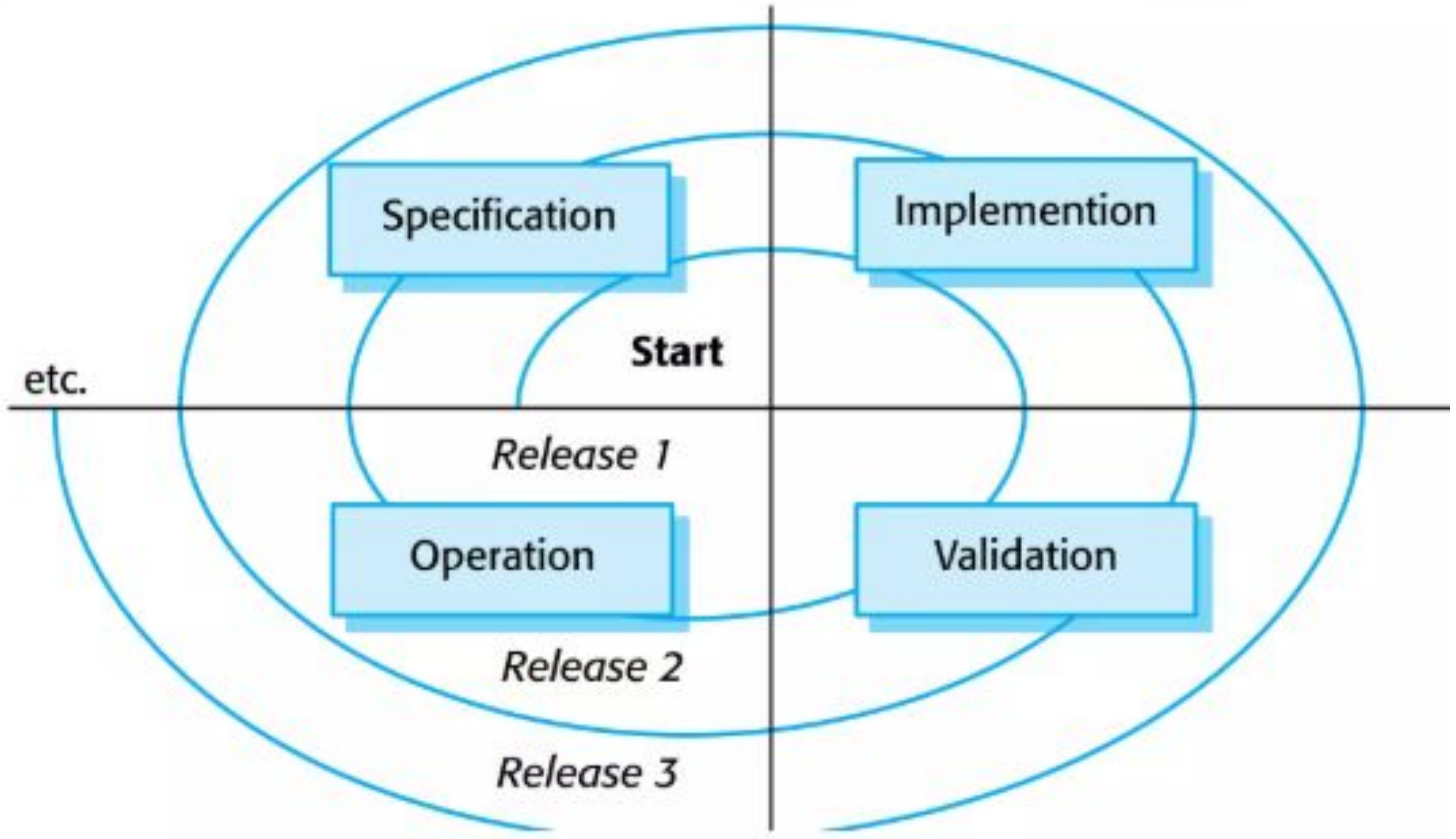
Software change

- ✧ Software change is inevitable
  - New requirements emerge when the software is used;
  - The business environment changes;
  - Errors must be repaired;
  - New computers and equipment is added to the system;
  - The performance or reliability of the system may have to be improved.
- ✧ A key problem for all organizations is implementing and managing change to their existing software systems.

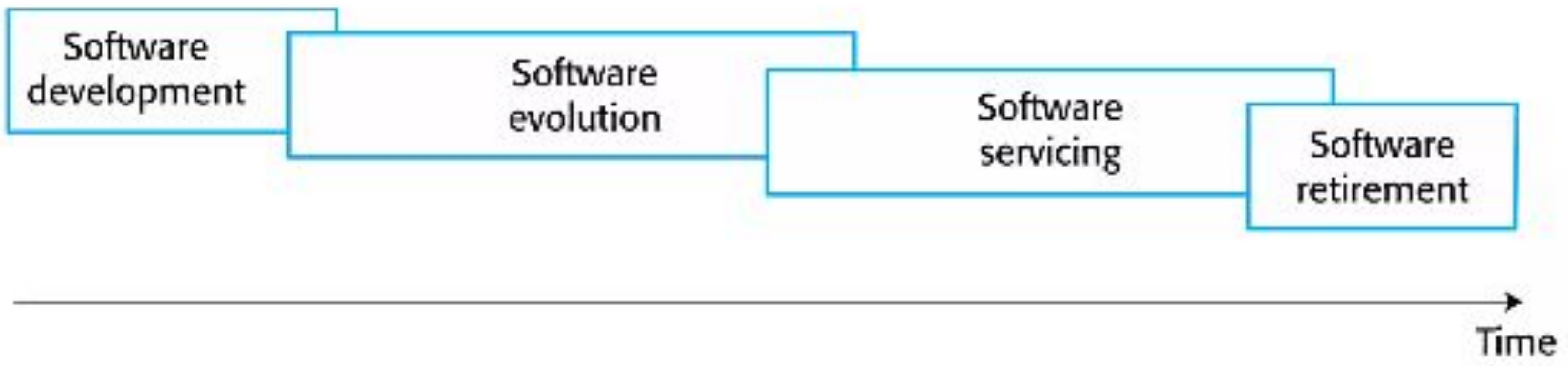
Importance of evolution

- ✧ Organisations have huge investments in their software systems - they are critical business assets.
- ✧ To maintain the value of these assets to the business, they must be changed and updated.
- ✧ The majority of the software budget in large companies is devoted to changing and evolving existing software rather than developing new software.

A spiral model of development and evolution



Evolution and servicing



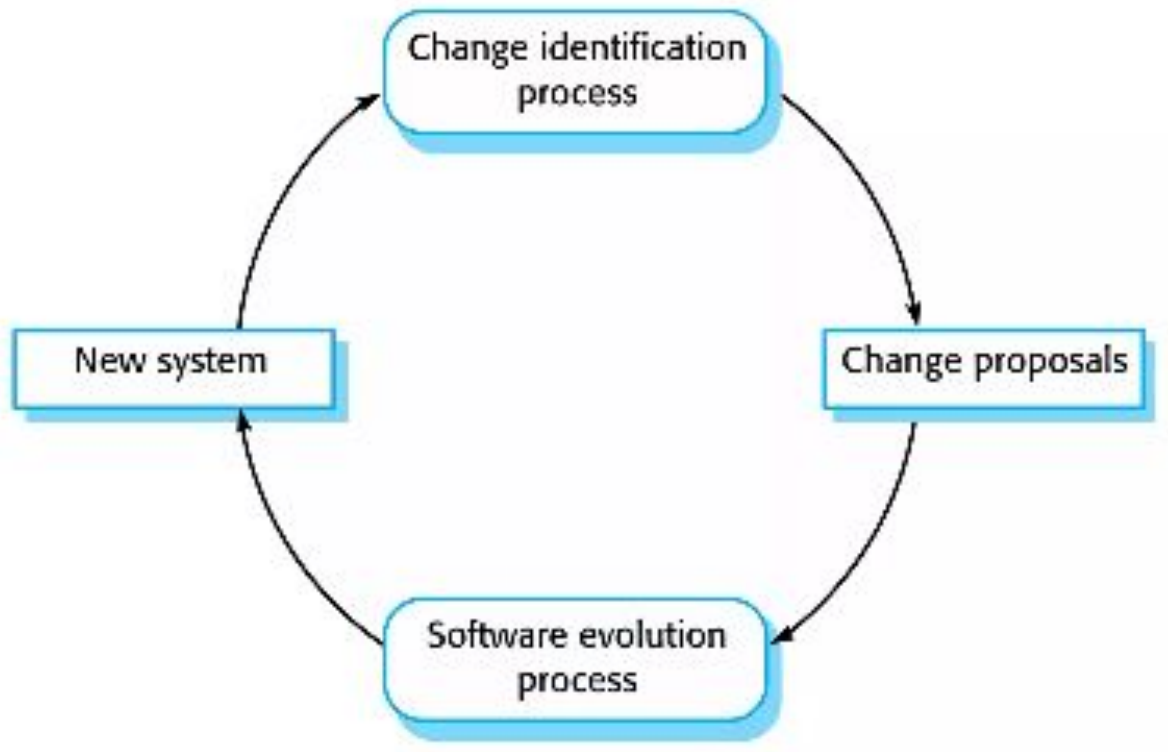
Evolution and servicing

- ✧ Evolution
  - The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system.
- ✧ Servicing
  - At this stage, the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software's environment. No new functionality is added.
- ✧ Phase-out
  - The software may still be used but no further changes are made to it.

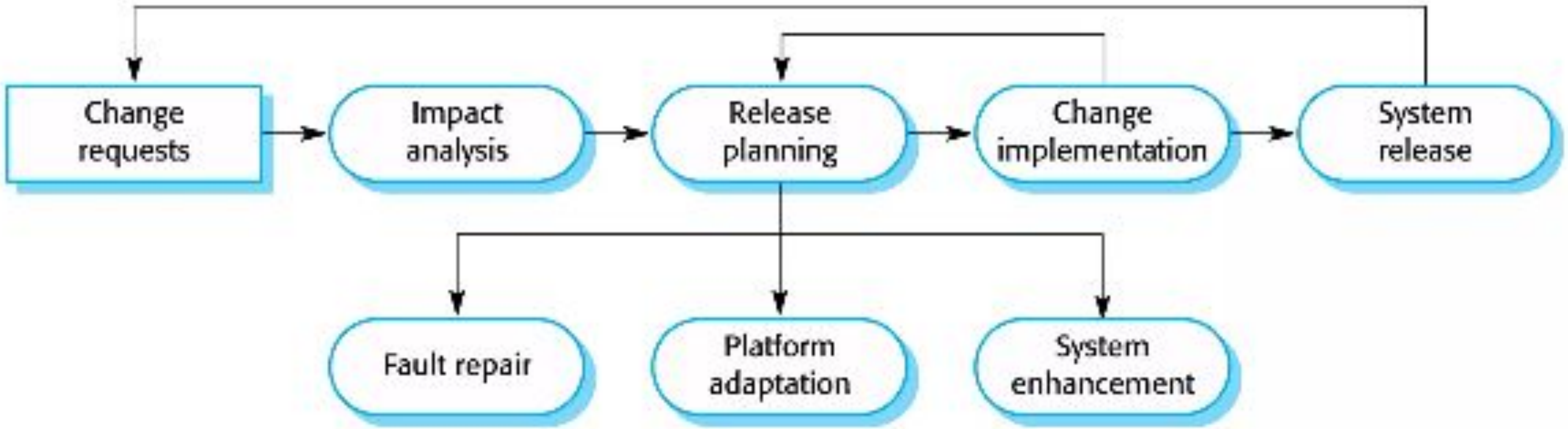
Evolution processes

- ✧ Software evolution processes depend on
  - The type of software being maintained;
  - The development processes used;
  - The skills and experience of the people involved.
- ✧ Proposals for change are the driver for system evolution.
  - Should be linked with components that are affected by the change, thus allowing the cost and impact of the change to be estimated.
- ✧ Change identification and evolution continues throughout the system lifetime.

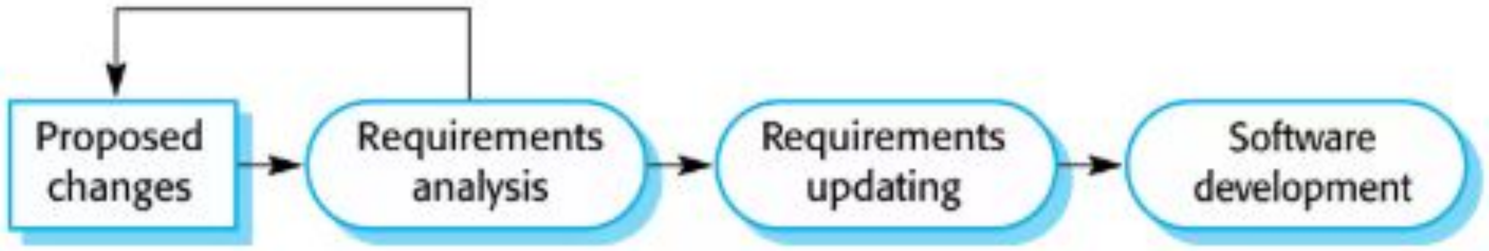
Change identification and evolution processes



The software evolution process



Change implementation



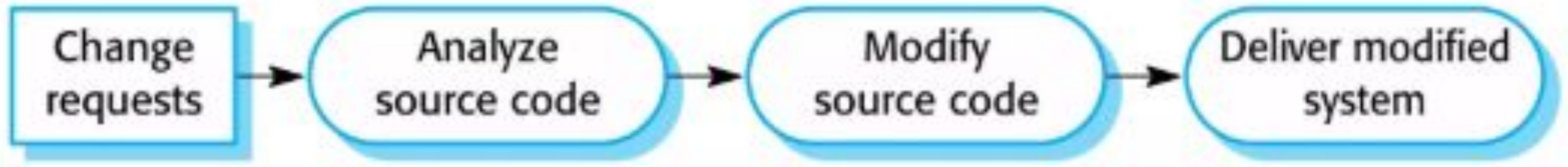
Change implementation

- ✧ Iteration of the development process where the revisions to the system are designed, implemented and tested.
- ✧ A critical difference is that the first stage of change implementation may involve program understanding, especially if the original system developers are not responsible for the change implementation.
- ✧ During the program understanding phase, you have to understand how the program is structured, how it delivers functionality and how the proposed change might affect the program.

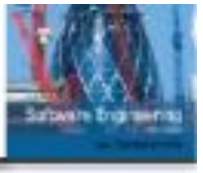
Urgent change requests

- ✧ Urgent changes may have to be implemented without going through all stages of the software engineering process
  - If a serious system fault has to be repaired to allow normal operation to continue;
  - If changes to the system's environment (e.g. an OS upgrade) have unexpected effects;
  - If there are business changes that require a very rapid response (e.g. the release of a competing product).

The emergency repair process



Agile methods and evolution



- ✧ Agile methods are based on incremental development so the transition from development to evolution is a seamless one.
  - Evolution is simply a continuation of the development process based on frequent system releases.
- ✧ Automated regression testing is particularly valuable when changes are made to a system.
- ✧ Changes may be expressed as additional user stories.

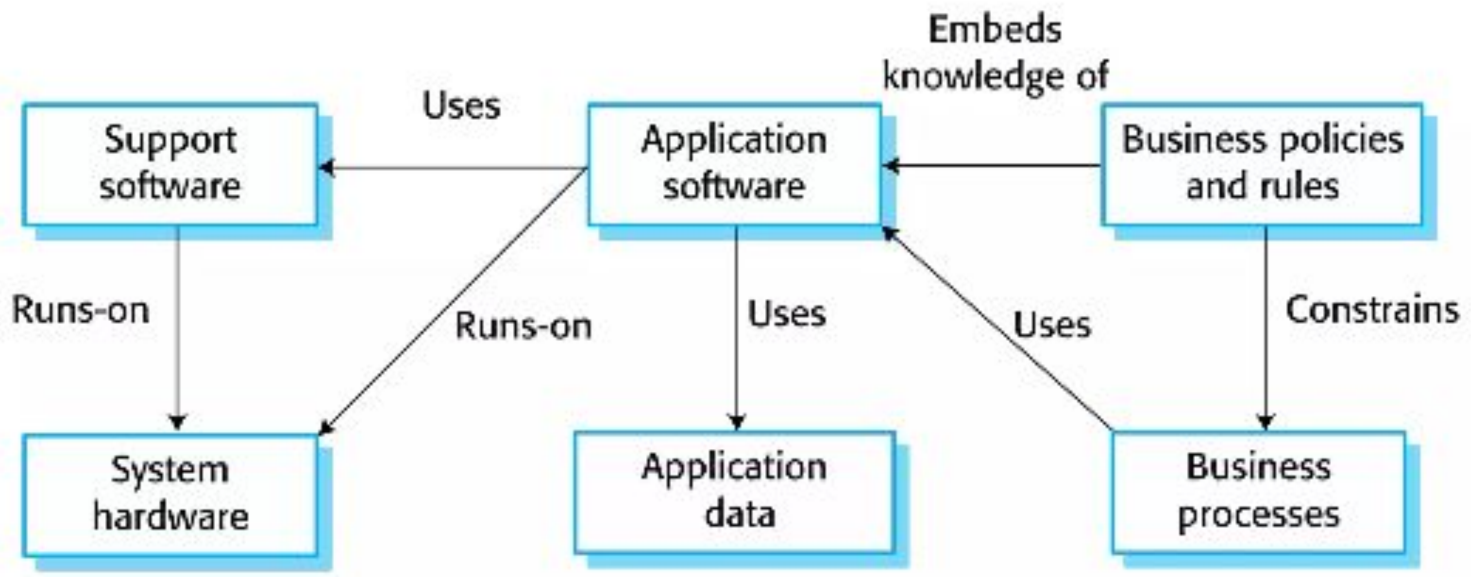
Handover problems

- ✧ Where the development team have used an agile approach but the evolution team is unfamiliar with agile methods and prefer a plan-based approach.
  - The evolution team may expect detailed documentation to support evolution and this is not produced in agile processes.
- ✧ Where a plan-based approach has been used for development but the evolution team prefer to use agile methods.
  - The evolution team may have to start from scratch developing automated tests and the code in the system may not have been refactored and simplified as is expected in agile development.

Legacy systems

- Legacy systems are older systems that rely on languages and technology that are no longer used for new systems development.
- Legacy software may be dependent on older hardware, such as mainframe computers and may have associated legacy processes and procedures.
- Legacy systems are not just software systems but are broader socio-technical systems that include hardware, software, libraries and other supporting software and business processes.

The elements of a legacy system



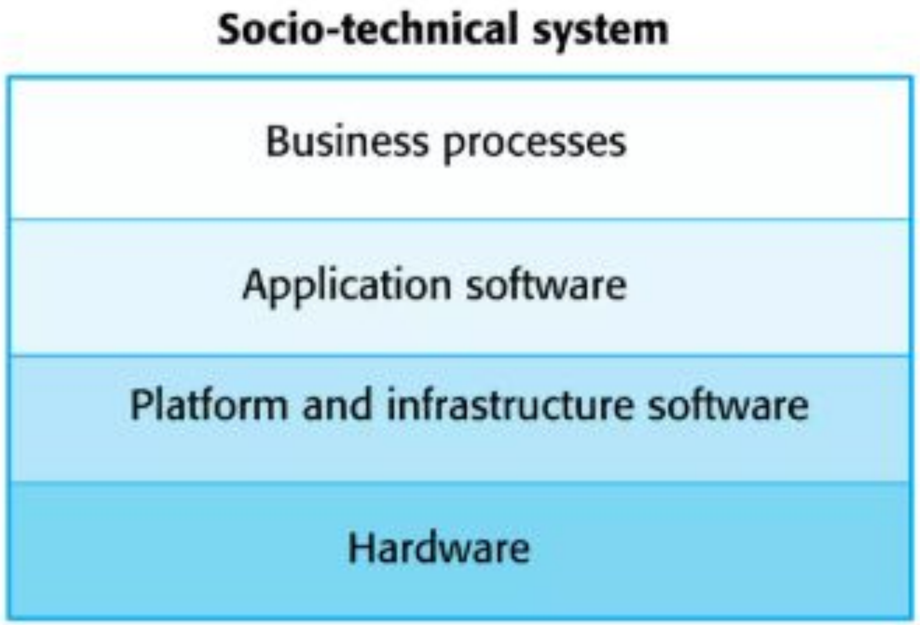
Legacy system components

- *System hardware* Legacy systems may have been written for hardware that is no longer available.
- *Support software* The legacy system may rely on a range of support software, which may be obsolete or unsupported.
- *Application software* The application system that provides the business services is usually made up of a number of application programs.
- *Application data* These are data that are processed by the application system. They may be inconsistent, duplicated or held in different databases.

Legacy system components

- ✧ *Business processes* These are processes that are used in the business to achieve some business objective.
- ✧ Business processes may be designed around a legacy system and constrained by the functionality that it provides.
- ✧ *Business policies and rules* These are definitions of how the business should be carried out and constraints on the business. Use of the legacy application system may be embedded in these policies and rules.

Legacy system layers



Legacy system replacement

- ✧ Legacy system replacement is risky and expensive so businesses continue to use these systems
- ✧ System replacement is risky for a number of reasons
  - Lack of complete system specification
  - Tight integration of system and business processes
  - Undocumented business rules embedded in the legacy system
  - New software development may be late and/or over budget

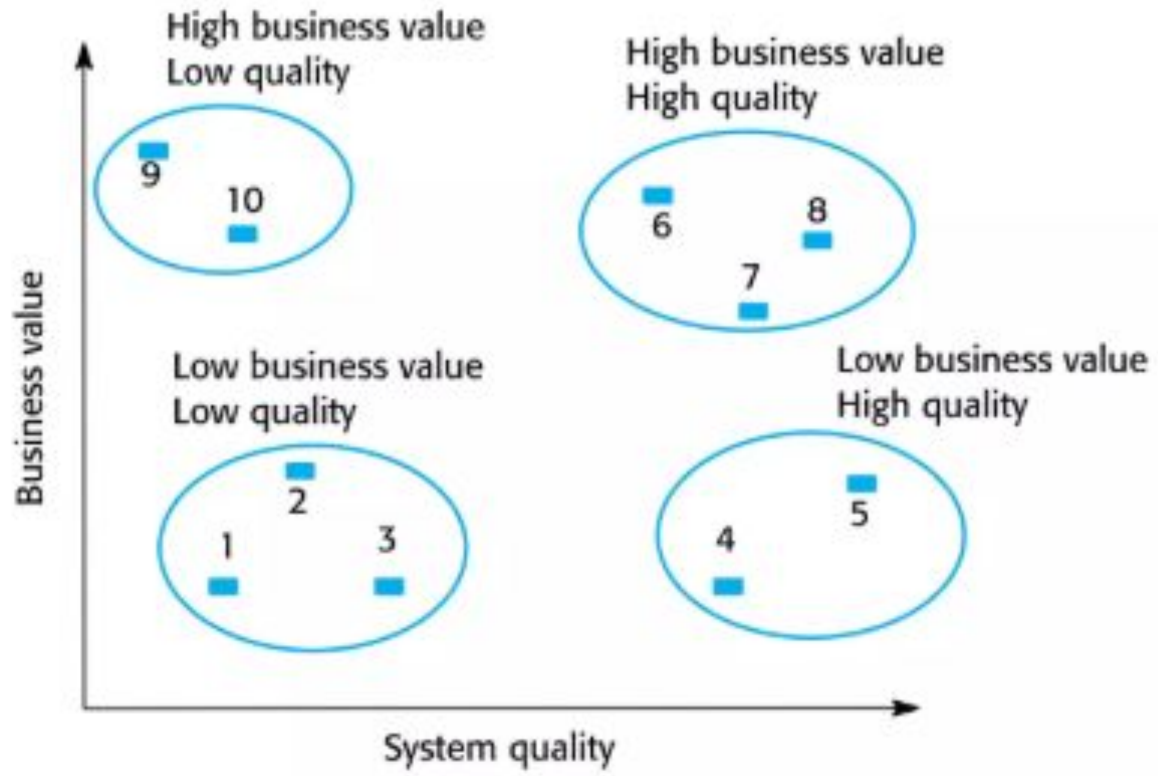
Legacy system change

- ✧ Legacy systems are expensive to change for a number of reasons:
  - No consistent programming style
  - Use of obsolete programming languages with few people available with these language skills
  - Inadequate system documentation
  - System structure degradation
  - Program optimizations may make them hard to understand
  - Data errors, duplication and inconsistency

Legacy system management

- ✧ Organisations that rely on legacy systems must choose a strategy for evolving these systems
  - Scrap the system completely and modify business processes so that it is no longer required;
  - Continue maintaining the system;
  - Transform the system by re-engineering to improve its maintainability;
  - Replace the system with a new system.
- ✧ The strategy chosen should depend on the system quality and its business value.

Figure 9.13 An example of a legacy system assessment



Legacy system categories

- ✧ Low quality, low business value
  - These systems should be scrapped.
- ✧ Low-quality, high-business value
  - These make an important business contribution but are expensive to maintain. Should be re-engineered or replaced if a suitable system is available.
- ✧ High-quality, low-business value
  - Replace with COTS, scrap completely or maintain.
- ✧ High-quality, high business value
  - Continue in operation using normal system maintenance.

Business value assessment

- ✧ Assessment should take different viewpoints into account
  - System end-users;
  - Business customers;
  - Line managers;
  - IT managers;
  - Senior managers.
- ✧ Interview different stakeholders and collate results.

Issues in business value assessment

- ✧ The use of the system
  - If systems are only used occasionally or by a small number of people, they may have a low business value.
- ✧ The business processes that are supported
  - A system may have a low business value if it forces the use of inefficient business processes.
- ✧ System dependability
  - If a system is not dependable and the problems directly affect business customers, the system has a low business value.
- ✧ The system outputs
  - If the business depends on system outputs, then the system has a high business value.

System quality assessment

- ✧ Business process assessment
  - How well does the business process support the current goals of the business?
- ✧ Environment assessment
  - How effective is the system's environment and how expensive is it to maintain?
- ✧ Application assessment
  - What is the quality of the application software system?

Business process assessment

- ✧ Use a viewpoint-oriented approach and seek answers from system stakeholders
  - Is there a defined process model and is it followed?
  - Do different parts of the organisation use different processes for the same function?
  - How has the process been adapted?
  - What are the relationships with other business processes and are these necessary?
  - Is the process effectively supported by the legacy application software?
- ✧ Example - a travel ordering system may have a low business value because of the widespread use of web-based ordering.

Factors used in environment assessment

Factor	Questions
Supplier stability	Is the supplier still in existence? Is the supplier financially stable and likely to continue in existence? If the supplier is no longer in business, does someone else maintain the systems?
Failure rate	Does the hardware have a high rate of reported failures? Does the support software crash and force system restarts?
Age	How old is the hardware and software? The older the hardware and support software, the more obsolete it will be. It may still function correctly but there could be significant economic and business benefits to moving to a more modern system.
Performance	Is the performance of the system adequate? Do performance problems have a significant effect on system users?

Factors used in environment assessment

Factor	Questions
Support requirements	What local support is required by the hardware and software? If there are high costs associated with this support, it may be worth considering system replacement.
Maintenance costs	What are the costs of hardware maintenance and support software licences? Older hardware may have higher maintenance costs than modern systems. Support software may have high annual licensing costs.
Interoperability	Are there problems interfacing the system to other systems? Can compilers, for example, be used with current versions of the operating system? Is hardware emulation required?

Factors used in application assessment

Factor	Questions
Understandability	How difficult is it to understand the source code of the current system? How complex are the control structures that are used? Do variables have meaningful names that reflect their function?
Documentation	What system documentation is available? Is the documentation complete, consistent, and current?
Data	Is there an explicit data model for the system? To what extent is data duplicated across files? Is the data used by the system up to date and consistent?
Performance	Is the performance of the application adequate? Do performance problems have a significant effect on system users?

Factors used in application assessment

Factor	Questions
Programming language	Are modern compilers available for the programming language used to develop the system? Is the programming language still used for new system development?
Configuration management	Are all versions of all parts of the system managed by a configuration management system? Is there an explicit description of the versions of components that are used in the current system?
Test data	Does test data for the system exist? Is there a record of regression tests carried out when new features have been added to the system?
Personnel skills	Are there people available who have the skills to maintain the application? Are there people available who have experience with the system?

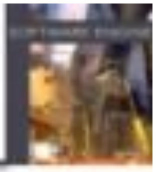
System measurement

✧ You may collect quantitative data to make an assessment of the quality of the application system

- The number of system change requests; The higher this accumulated value, the lower the quality of the system.
- The number of different user interfaces used by the system; The more interfaces, the more likely it is that there will be inconsistencies and redundancies in these interfaces.
- The volume of data used by the system. As the volume of data (number of files, size of database, etc.) processed by the system increases, so too do the inconsistencies and errors in that data.
- Cleaning up old data is a very expensive and time-consuming process

Project Management

Software project management



- Concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organisations developing and procuring the software.
- Project management is needed because software development is always subject to budget and schedule constraints that are set by the organisation developing the software.

Success criteria

- Deliver the software to the customer at the agreed time.
- Keep overall costs within budget.
- Deliver software that meets the customer’s expectations.
- Maintain a happy and well-functioning development team.

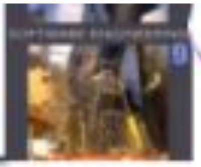
Software management distinctions

- The product is intangible.
  - Software cannot be seen or touched. Software project managers cannot see progress by simply looking at the artefact that is being constructed.
- Many software projects are 'one-off' projects.
  - Large software projects are usually different in some ways from previous projects. Even managers who have lots of previous experience may find it difficult to anticipate problems.
- Software processes are variable and organization specific.
  - We still cannot reliably predict when a particular software process is likely to lead to development problems.

Management activities

- ✧ *Project planning*
  - Project managers are responsible for planning. estimating and scheduling project development and assigning people to tasks.
- ✧ *Reporting*
  - Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software.
- ✧ *Risk management*
  - Project managers assess the risks that may affect a project, monitor these risks and take action when problems arise.

Management activities



- ✧ *People management*
  - Project managers have to choose people for their team and establish ways of working that leads to effective team performance
- ✧ *Proposal writing*
  - The first stage in a software project may involve writing a proposal to win a contract to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out.

Risk management

- ✧ Risk management is concerned with identifying risks and drawing up plans to minimise their effect on a project.
- ✧ A risk is a probability that some adverse circumstance will occur
  - Project risks affect schedule or resources;
  - Product risks affect the quality or performance of the software being developed;
  - Business risks affect the organisation developing or procuring the software.

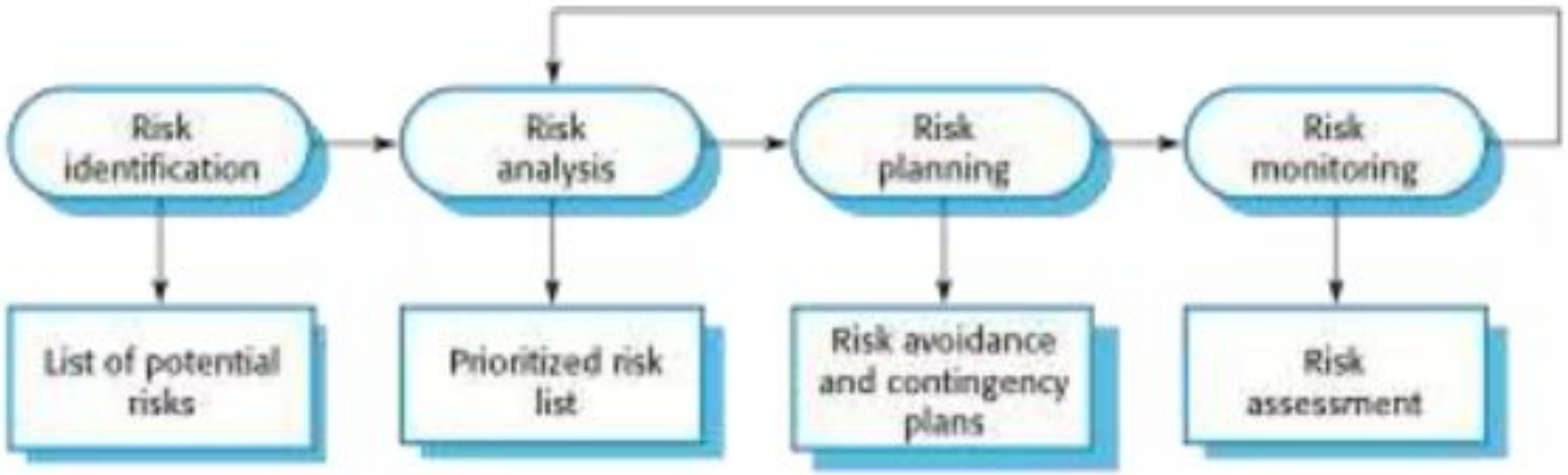
Examples of common project, product, and business risks

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

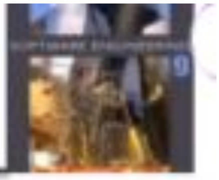
The risk management process

- Risk identification
  - Identify project, product and business risks;
- Risk analysis
  - Assess the likelihood and consequences of these risks;
- Risk planning
  - Draw up plans to avoid or minimise the effects of the risk;
- Risk monitoring
  - Monitor the risks throughout the project;

The risk management process



Risk identification

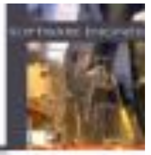


- ✧ May be a team activities or based on the individual project manager’s experience.
- ✧ A checklist of common risks may be used to identify risks in a project
  - Technology risks.
  - People risks.
  - Organisational risks.
  - Requirements risks.
  - Estimation risks.

Examples of different risk types

Risk type	Possible risks
Technology	The database used in the system cannot process as many transactions per second as expected. (1) Reusable software components contain defects that mean they cannot be reused as planned. (2)
People	It is impossible to recruit staff with the skills required. (3) Key staff are ill and unavailable at critical times. (4) Required training for staff is not available. (5)
Organizational	The organization is restructured so that different management are responsible for the project. (6) Organizational financial problems force reductions in the project budget. (7)
Tools	The code generated by software code generation tools is inefficient. (8) Software tools cannot work together in an integrated way. (9)
Requirements	Changes to requirements that require major design rework are proposed. (10) Customers fail to understand the impact of requirements changes. (11)
Estimation	The time required to develop the software is underestimated. (12) The rate of defect repair is underestimated. (13) The size of the software is underestimated. (14)

Risk analysis

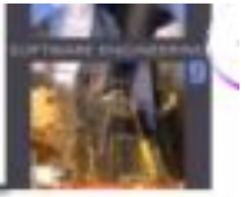


- ✧ Assess probability and seriousness of each risk.
- ✧ Probability may be very low, low, moderate, high or very high.
- ✧ Risk consequences might be catastrophic, serious, tolerable or insignificant.

Risk types and examples

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget (7).	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project (3).	High	Catastrophic
Key staff are ill at critical times in the project (4).	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused. (2).	Moderate	Serious
Changes to requirements that require major design rework are proposed (10).	Moderate	Serious
The organization is restructured so that different management are responsible for the project (6).	High	Serious
The database used in the system cannot process as many transactions per second as expected (1).	Moderate	Serious

Risk types and examples



Risk	Probability	Effects
The time required to develop the software is underestimated (12).	High	Serious
Software tools cannot be integrated (9).	High	Tolerable
Customers fail to understand the impact of requirements changes (11).	Moderate	Tolerable
Required training for staff is not available (5).	Moderate	Tolerable
The rate of defect repair is underestimated (13).	Moderate	Tolerable
The size of the software is underestimated (14).	High	Tolerable
Code generated by code generation tools is inefficient (8).	Moderate	Insignificant

Risk planning

- Consider each risk and develop a strategy to manage that risk.
- Avoidance strategies
  - The probability that the risk will arise is reduced;
- Minimisation strategies
  - The impact of the risk on the project or product will be reduced;
- Contingency plans
  - If the risk arises, contingency plans are plans to deal with that risk;

Strategies to help manage risk

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.

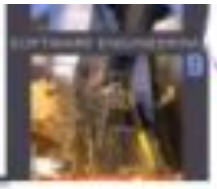
Strategies to help manage risk

Risk	Strategy
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program generator.

Risk monitoring

- ✧ Assess each identified risks regularly to decide whether or not it is becoming less or more probable.
- ✧ Also assess whether the effects of the risk have changed.
- ✧ Each key risk should be discussed at management progress meetings.

Risk indicators



Risk type	Potential indicators
Technology	Late delivery of hardware or support software; many reported technology problems.
People	Poor staff morale; poor relationships amongst team members; high staff turnover.
Organizational	Organizational gossip; lack of action by senior management.
Tools	Reluctance by team members to use tools; complaints about CASE tools; demands for higher-powered workstations.
Requirements	Many requirements change requests; customer complaints.
Estimation	Failure to meet agreed schedule; failure to clear reported defects.

Project Planning

Project planning

- ✧ Project planning involves breaking down the work into parts and assign these to project team members, anticipate problems that might arise and prepare tentative solutions to those problems.
- ✧ The project plan, which is created at the start of a project, is used to communicate how the work will be done to the project team and customers, and to help assess progress on the project.

Planning stages

- At the proposal stage, when you are bidding for a contract to develop or provide a software system.
- During the project startup phase, when you have to plan who will work on the project, how the project will be broken down into increments, how resources will be allocated across your company, etc.
- Periodically throughout the project, when you modify your plan in the light of experience gained and information from monitoring the progress of the work.

Proposal planning

- Planning may be necessary with only outline software requirements.
- The aim of planning at this stage is to provide information that will be used in setting a price for the system to customers.
- Project pricing involves estimating how much the software will cost to develop, taking factors such as staff costs, hardware costs, software costs, etc. into account

Project startup planning

- ✧ At this stage, you know more about the system requirements but do not have design or implementation information
- ✧ Create a plan with enough detail to make decisions about the project budget and staffing.
  - This plan is the basis for project resource allocation
- ✧ The startup plan should also define project monitoring mechanisms
- ✧ A startup plan is still needed for agile development to allow resources to be allocated to the project

Development planning

- ✧ The project plan should be regularly amended as the project progresses and you know more about the software and its development
- ✧ The project schedule, cost-estimate and risks have to be regularly revised

Software pricing

- ✧ Estimates are made to discover the cost, to the developer, of producing a software system.
  - You take into account, hardware, software, travel, training and effort costs.
- ✧ There is not a simple relationship between the development cost and the price charged to the customer.
- ✧ Broader organisational, economic, political and business considerations influence the price charged.

Factors affecting software pricing

Factor	Description
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Cost estimate uncertainty	If an organization is unsure of its cost estimate, it may increase its price by a contingency over and above its normal profit.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a smaller than normal profit or break even than to go out of business. Cash flow is more important than profit in difficult economic times.

Factors affecting software pricing

Factor	Description
Market opportunity	A development organization may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the organization the opportunity to make a greater profit later. The experience gained may also help it develop new products.
Requirements volatility	If the requirements are likely to change, an organization may lower its price to win a contract. After the contract is awarded, high prices can be charged for changes to the requirements.

Pricing strategies

- ✧ Under pricing
  - A company may underprice a system in order to gain a contract that allows them to retain staff for future opportunities
  - A company may underprice a system to gain access to a new market area
- ✧ Increased pricing
  - The price may be increased when a buyer wishes a fixed-price contract and so the seller increases the price to allow for unexpected risks

Pricing to win

- ✧ The software is priced according to what the software developer believes the buyer is willing to pay
- ✧ If this is less that the development costs, the software functionality may be reduced accordingly with a view to extra functionality being added in a later release
- ✧ Additional costs may be added as the requirements change and these may be priced at a higher level to make up the shortfall in the original price

Plan-driven development

- ✧ Plan-driven or plan-based development is an approach to software engineering where the development process is planned in detail.
  - Plan-driven development is based on engineering project management techniques and is the 'traditional' way of managing large software development projects.
- ✧ A project plan is created that records the work to be done, who will do it, the development schedule and the work products.
- ✧ Managers use the plan to support project decision making and as a way of measuring progress.

Plan-driven development – pros and cons

- The arguments in favor of a plan-driven approach are that early planning allows organizational issues (availability of staff, other projects, etc.) to be closely taken into account, and that potential problems and dependencies are discovered before the project starts, rather than once the project is underway.
- The principal argument against plan-driven development is that many early decisions have to be revised because of changes to the environment in which the software is to be developed and used.

Project plans

✧ In a plan-driven development project, a project plan sets out the resources available to the project, the work breakdown and a schedule for carrying out the work.

✧ Plan sections

- Introduction
- Project organization
- Risk analysis
- Hardware and software resource requirements
- Work breakdown
- Project schedule
- Monitoring and reporting mechanisms

Project plan supplements

Plan	Description
Configuration management plan	Describes the configuration management procedures and structures to be used.
Deployment plan	Describes how the software and associated hardware (if required) will be deployed in the customer's environment. This should include a plan for migrating data from existing systems.
Maintenance plan	Predicts the maintenance requirements, costs, and effort.
Quality plan	Describes the quality procedures and standards that will be used in a project.
Validation plan	Describes the approach, resources, and schedule used for system validation.

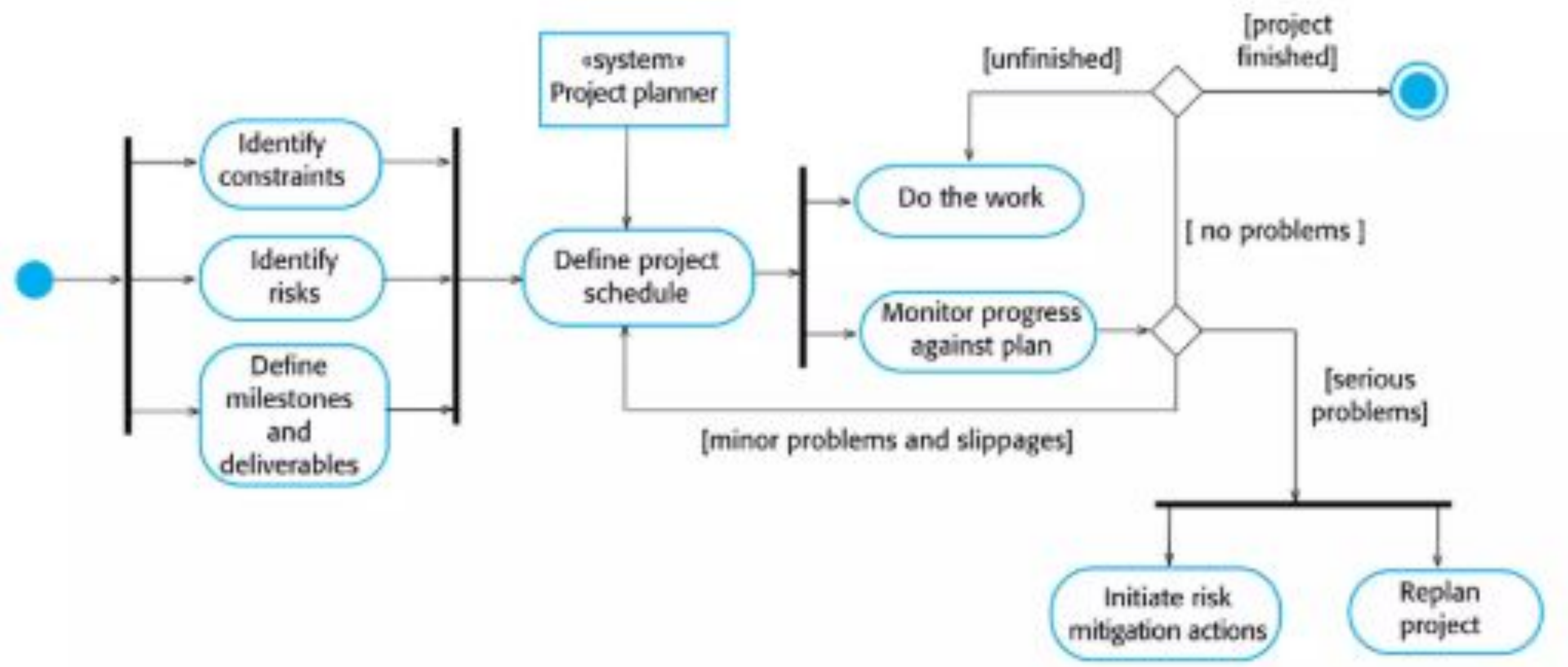
The planning process

✧ Project planning is an iterative process that starts when you create an initial project plan during the project startup phase.

✧ Plan changes are inevitable.

- As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule and risk changes.
- Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be re-planned.

The project planning process



Planning assumptions

- ✧ You should make realistic rather than optimistic assumptions when you are defining a project plan.
- ✧ Problems of some description always arise during a project, and these lead to project delays.
- ✧ Your initial assumptions and scheduling should therefore take unexpected problems into account.
- ✧ You should include contingency in your plan so that if things go wrong, then your delivery schedule is not seriously disrupted.

Risk mitigation

- ✧ If there are serious problems with the development work that are likely to lead to significant delays, you need to initiate risk mitigation actions to reduce the risks of project failure.
- ✧ In conjunction with these actions, you also have to re-plan the project.
- ✧ This may involve renegotiating the project constraints and deliverables with the customer. A new schedule of when work should be completed also has to be established and agreed with the customer.

Quality management

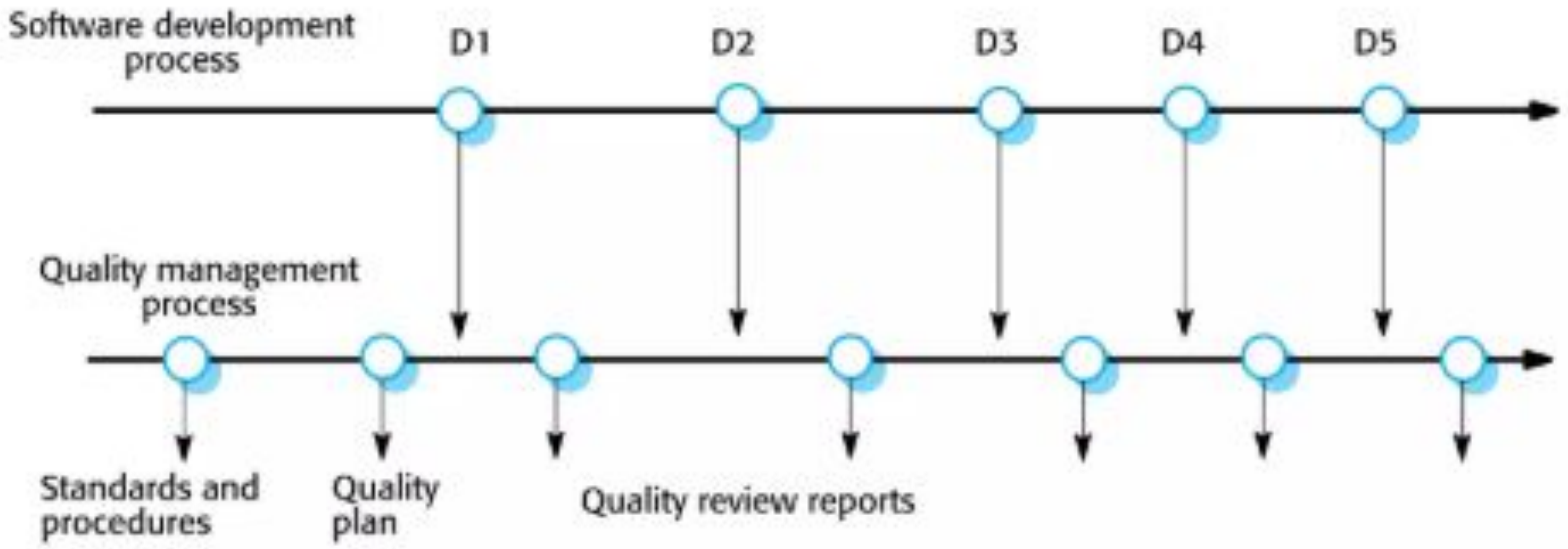
Software quality management

- ✧ Concerned with ensuring that the required level of quality is achieved in a software product.
- ✧ Three principal concerns:
  - At the organizational level, quality management is concerned with establishing a framework of organizational processes and standards that will lead to high-quality software.
  - At the project level, quality management involves the application of specific quality processes and checking that these planned processes have been followed.
  - At the project level, quality management is also concerned with establishing a quality plan for a project. The quality plan should set out the quality goals for the project and define what processes and standards are to be used.

Quality management activities

- ✧ Quality management provides an independent check on the software development process.
- ✧ The quality management process checks the project deliverables to ensure that they are consistent with organizational standards and goals
- ✧ The quality team should be independent from the development team so that they can take an objective view of the software. This allows them to report on software quality without being influenced by software development issues.

Quality management and software development



Quality planning

- ✧ A quality plan sets out the desired product qualities and how these are assessed and defines the most significant quality attributes.
- ✧ The quality plan should define the quality assessment process.
- ✧ It should set out which organisational standards should be applied and, where necessary, define new standards to be used.

Quality plans

- 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.

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.
- ✧ Techniques have to evolve when agile development is used.

Software quality

- Quality, simplistically, means that a product should meet its specification.
- This is problematical for software systems
  - There is a tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.);
  - Some quality requirements are difficult to specify in an unambiguous way;
  - Software specifications are usually incomplete and often inconsistent.
- The focus may be 'fitness for purpose' rather than specification conformance.

Software fitness for purpose

- ✧ Has the software been properly tested?
- ✧ Is the software sufficiently dependable to be put into use?
- ✧ Is the performance of the software acceptable for normal use?
- ✧ Is the software usable?
- ✧ Is the software well-structured and understandable?
- ✧ Have programming and documentation standards been followed in the development process?

Non-functional characteristics

- ✧ The subjective quality of a software system is largely based on its non-functional characteristics.
- ✧ This reflects practical user experience – if the software's functionality is not what is expected, then users will often just work around this and find other ways to do what they want to do.
- ✧ However, if the software is unreliable or too slow, then it is practically impossible for them to achieve their goals.

Software quality attributes

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

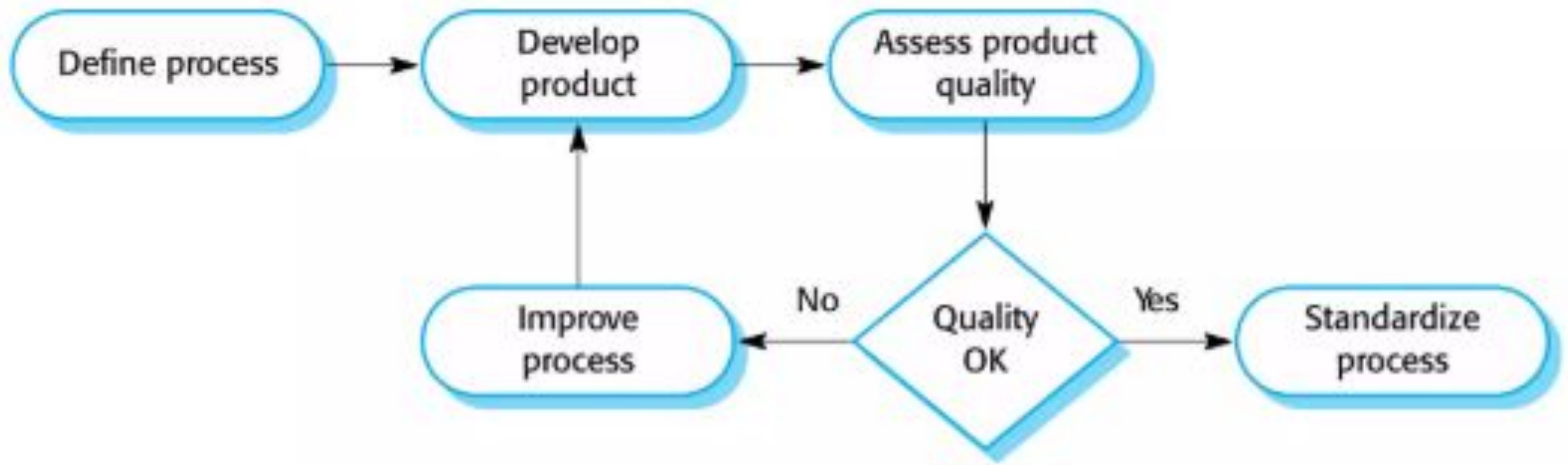
Quality conflicts

- ✧ It is not possible for any system to be optimized for all of these attributes – for example, improving robustness may lead to loss of performance.
- ✧ The quality plan should therefore define the most important quality attributes for the software that is being developed.
- ✧ The plan should also include a definition of the quality assessment process, an agreed way of assessing whether some quality, such as maintainability or robustness, is present in the product.

Process and product quality

- ✧ The quality of a developed product is influenced by the quality of the production process.
- ✧ This is 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.
  - The application of individual skills and experience is particularly important in software development;
  - External factors such as the novelty of an application or the need for an accelerated development schedule may impair product quality.

Process-based quality



Quality culture

✧ Quality managers should aim to develop a ‘quality culture’ where everyone responsible for software development is committed to achieving a high level of product quality.

✧ They should encourage teams to take responsibility for the quality of their work and to develop new approaches to quality improvement.

✧ They should support people who are interested in the intangible aspects of quality and encourage professional behavior in all team members.

Software standards

- ✧ Standards define the required attributes of a product or process. They play an important role in quality management.
- ✧ Standards may be international, national, organizational or project standards.

Importance of standards

✧ Encapsulation of best practice- avoids repetition of past mistakes.

✧ They are a framework for defining what quality means in a particular setting i.e. that organization's view of quality.

✧ They provide continuity - new staff can understand the organisation by understanding the standards that are used.

Product and process standards

- *Product standards*
  - Apply to the software product being developed. They include document standards, such as the structure of requirements documents, documentation standards, such as a standard comment header for an object class definition, and coding standards, which define how a programming language should be used.
- *Process standards*
  - These define the processes that should be followed during software development. Process standards may include definitions of specification, design and validation processes, process support tools and a description of the documents that should be written during these processes.

Product and process standards

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
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 form filling work is often involved to maintain the documentation associated with the standards.

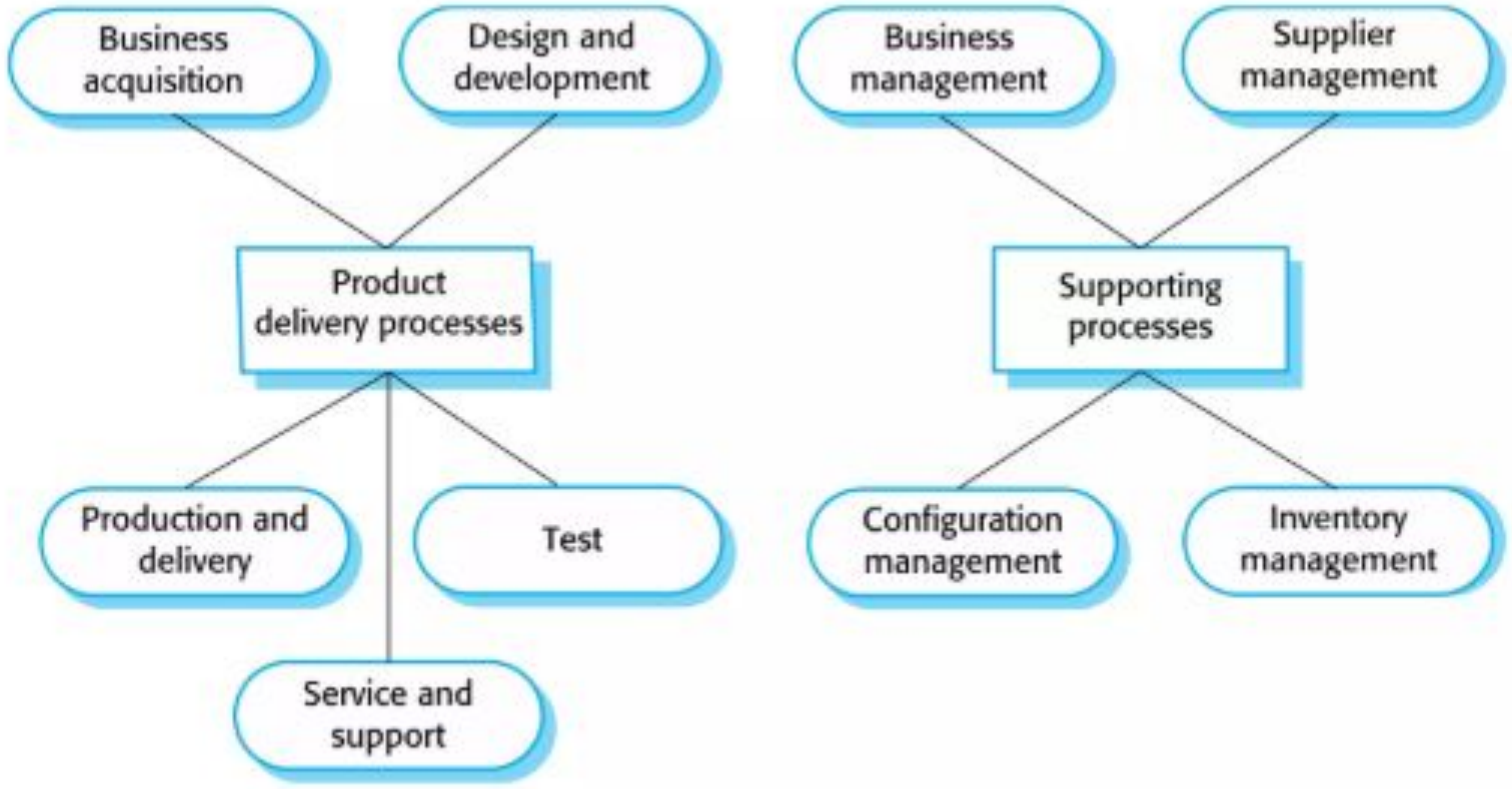
Standards development

- ✧ Involve practitioners in development. Engineers should understand the rationale underlying a standard.
- ✧ Review standards and their usage regularly. Standards can quickly become outdated and this reduces their credibility amongst practitioners.
- ✧ Detailed standards should have specialized tool support. Excessive clerical work is the most significant complaint against standards.
  - Web-based forms are not good enough.

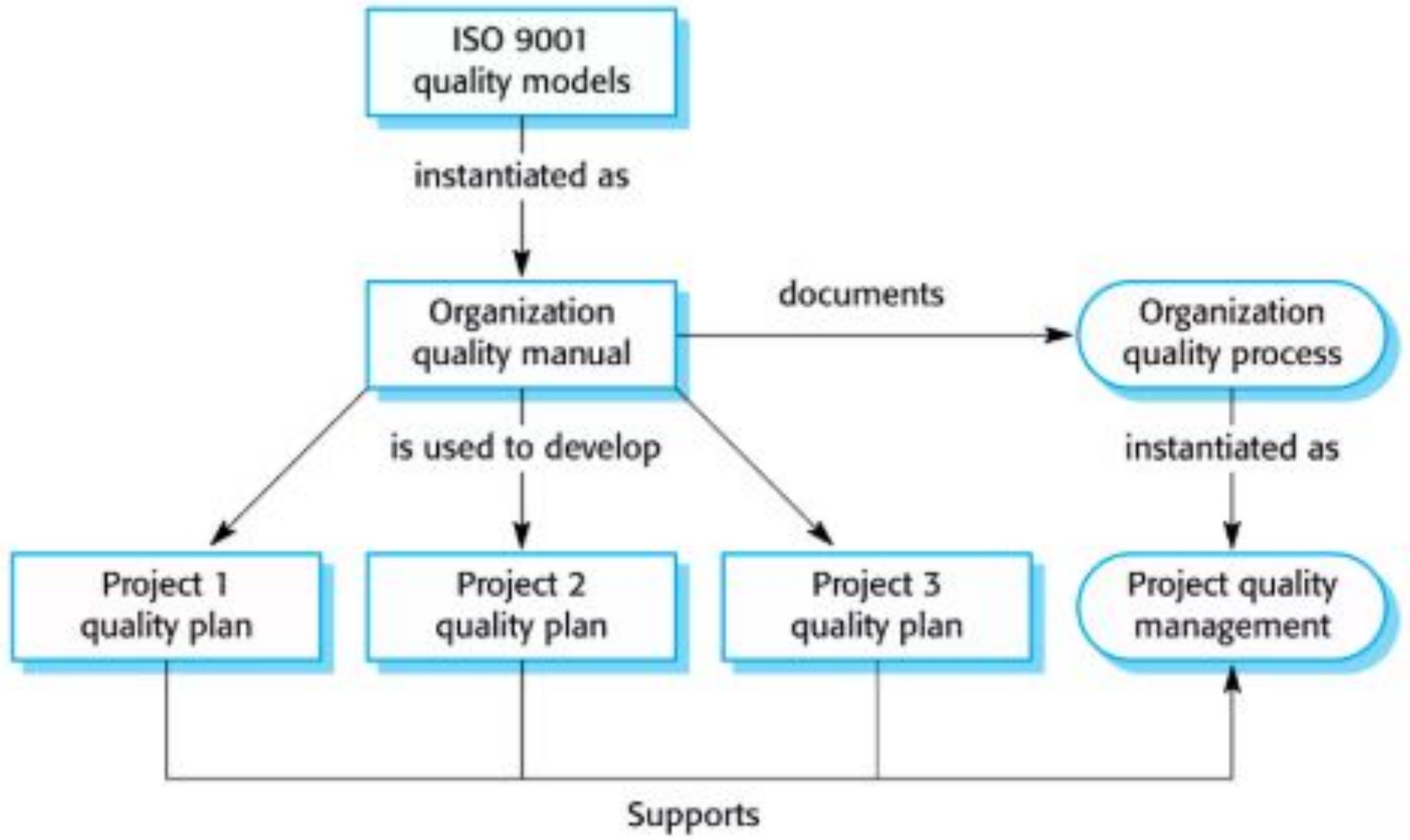
ISO 9001 standards framework

- ✧ An international set of standards that can be used as a basis for developing quality management systems.
- ✧ ISO 9001, the most general of these standards, applies to organizations that design, develop and maintain products, including software.
- ✧ The ISO 9001 standard is a framework for developing software standards.
  - It sets out general quality principles, describes quality processes in general and lays out the organizational standards and procedures that should be defined. These should be documented in an organizational quality manual.

ISO 9001 core processes



ISO 9001 and quality management



ISO 9001 certification

- Quality standards and procedures should be documented in an organisational quality manual.
- An external body may certify that an organisation's quality manual conforms to ISO 9000 standards.
- Some customers require suppliers to be ISO 9000 certified although the need for flexibility here is increasingly recognised.

Software quality and ISO9001

- The ISO 9001 certification is inadequate because it defines quality to be the conformance to standards.
- It takes no account of quality as experienced by users of the software. For example, a company could define test coverage standards specifying that all methods in objects must be called at least once.
- Unfortunately, this standard can be met by incomplete software testing that does not include tests with different method parameters. So long as the defined testing procedures are followed and test records maintained, the company could be ISO 9001 certified.

Thank you