Software Engineering

Software Engineering is the science and art of building significant software systems that are:

1) on time
2) on budget
3) with acceptable performance
4) with correct operation.

Software Engineering

- The economies of all developed nations are dependent on software.
- More and more systems are software controlled.
- Software engineering is concerned with theories, methods and tools for professional software development.
- Software engineering expenditure represents a significant fraction of the GNP of developed countries.

Software Costs

- Software costs often dominate system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop.
- Software engineering is concerned with cost-effective software development.

Software Products

- Generic products:
  - Stand-alone systems which are produced by a development organization and sold on the open market to any customer.
- Customized products:
  - Systems which are commissioned by a specific customer and developed specially by some contractor.

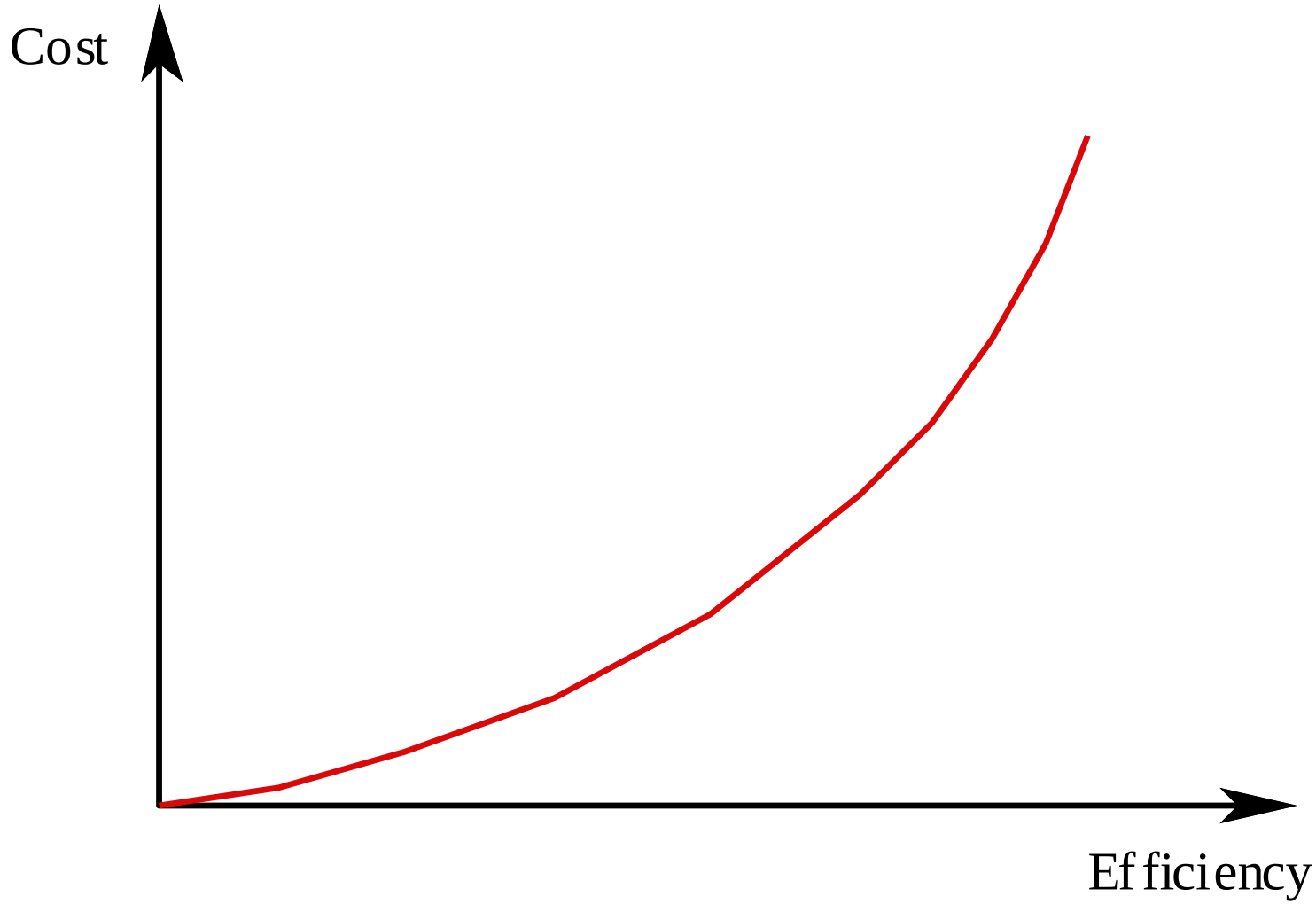
Software Product Attributes

- Maintainability
- Dependability
- Efficiency
- Usability

Importance of Product Characteristics

- The relative importance of these characteristics depends on the product and the environment in which it is to be used.
- In some cases, some attributes may dominate
  - In safety-critical real-time systems, key attributes may be dependability and efficiency.
- Costs tend to rise exponentially if very high levels of any one attribute are required.

Efficiency Costs



The Software Process

- Structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- Activities vary depending on the organization and the type of system being developed.
- Must be explicitly modeled if it is to be managed.

Engineering Process Model

- **Specification**: Set out the requirements and constraints on the system.
- **Design**: Produce a model of the system.
- **Manufacture**: Build the system.
- **Test**: Check the system meets the required specifications.
- **Install**: Deliver the system to the customer and ensure it is operational.
- **Maintain**: Repair faults in the system as they are discovered.

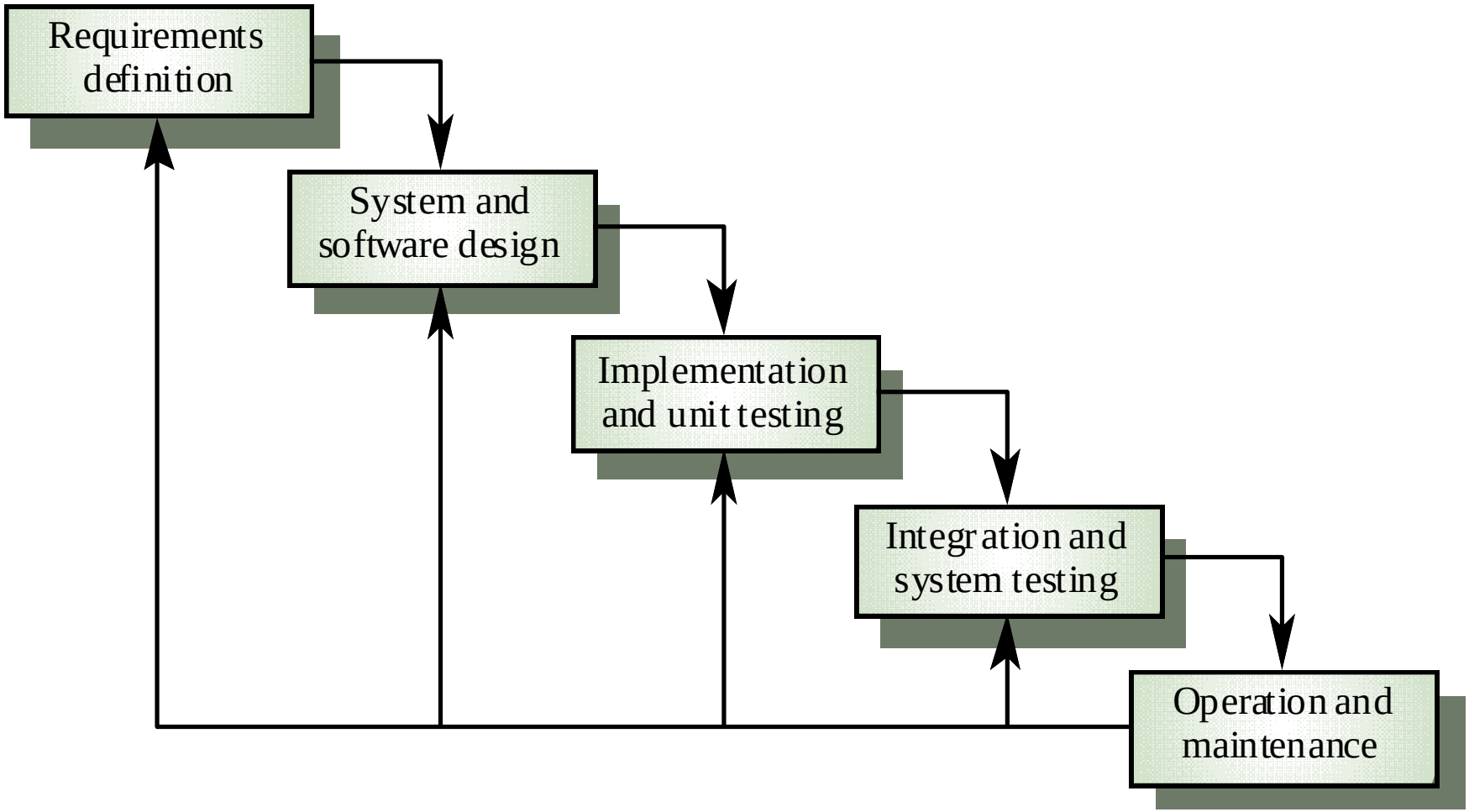
Software Engineering is Different

- Normally, specifications are incomplete.
- Very blurred distinction between specification, design and manufacture.
- No physical realization of the system for testing.
- Software does not wear out - maintenance does not mean component replacement.

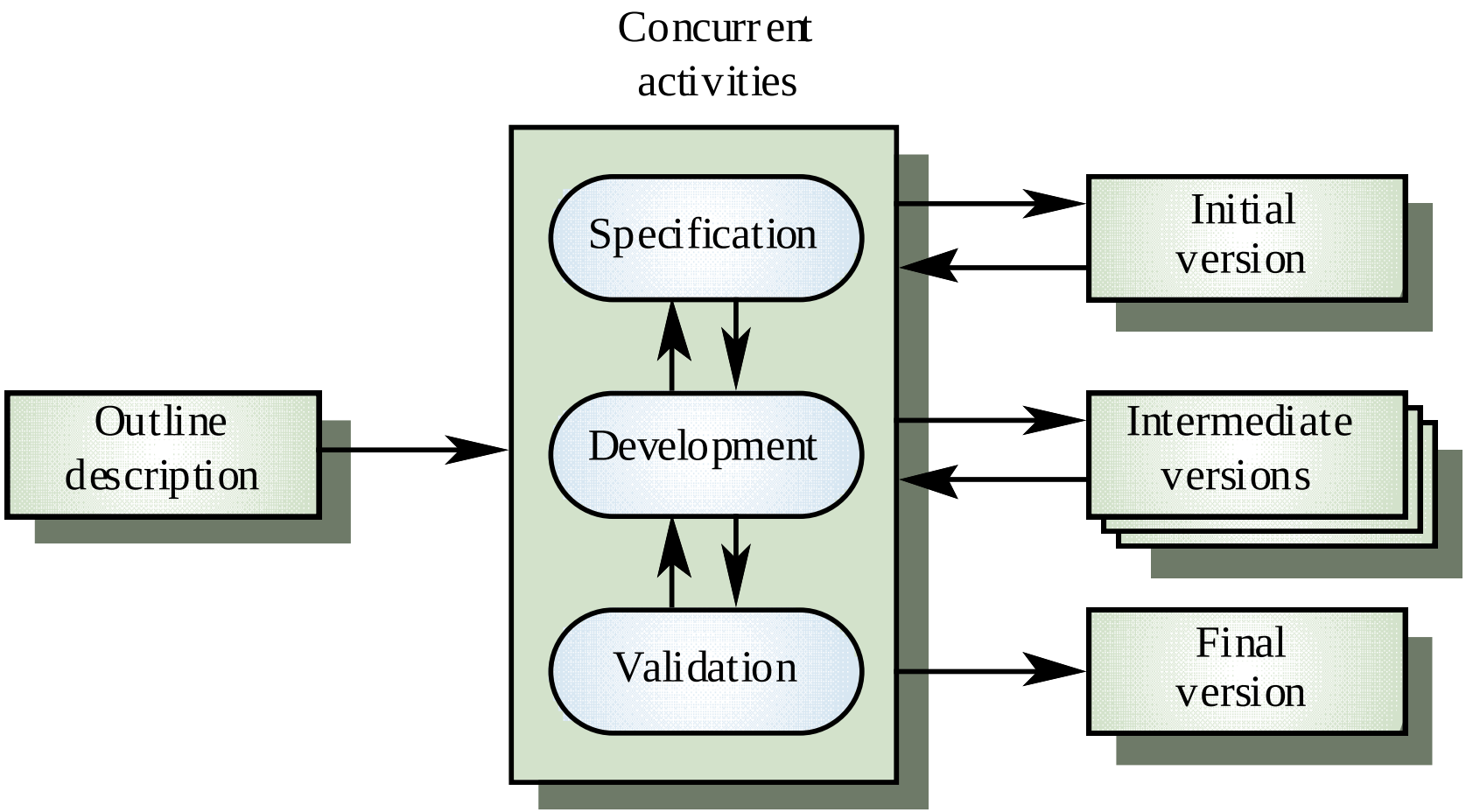
Generic Software Process Models

- **Waterfall**
  - Separate and distinct phases of specification and development
- **Evolutionary**
  - Specification and development are interleaved
- **Formal Transformation**
  - A mathematical system model is formally transformed to an implementation
- **Reuse-based**
  - The system is assembled from existing components

Waterfall Process Model



Evolutionary Process Model



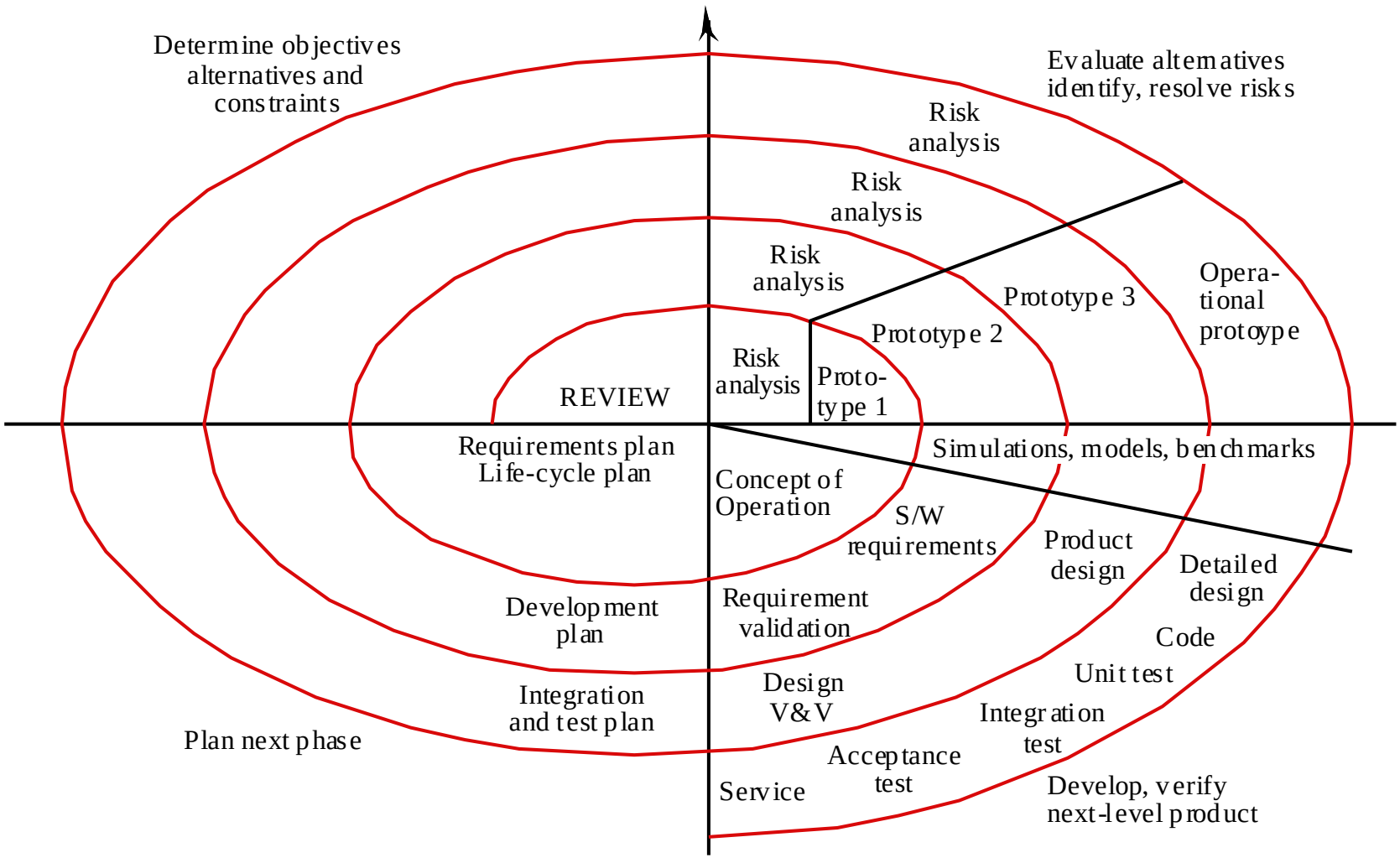
Process Model Problems

- **Waterfall**
  - High risk for new systems because of specification and design problems.
  - Low risk for well-understood developments using familiar technology.
- **Prototyping**
  - Low risk for new applications because specification and program stay in step.
  - High risk because of lack of process visibility.
- **Transformational**
  - High risk because of need for advanced technology and staff skills.

Hybrid Process Models

- Large systems are usually made up of several sub-systems.
- The same process model need not be used for all subsystems.
- Prototyping for high-risk specifications.
- Waterfall model for well-understood developments.

Spiral Process Model



Spiral Model Advantages

- Focuses attention on reuse options.
- Focuses attention on early error elimination.
- Puts quality objectives up front.
- Integrates development and maintenance.
- Provides a framework for hardware/software development.

Spiral Model Problems

- Contractual development often specifies process model and deliverables in advance.
- Requires risk assessment expertise.

Process Visibility

- Software systems are intangible so managers need documents to assess progress.
- Waterfall model is still the most widely used model.

Waterfall Model Documents

Activity	Output documents
Requirements analysis	Feasibility study, Outline requirements
Requirements definition	Requirements document
System specification	Functional specification, Acceptance test plan Draft user manual
Architectural design	Architectural specification, System test plan
Interface design	Interface specification, Integration test plan
Detailed design	Design specification, Unit test plan
Coding	Program code
Unit testing	Unit test report
Module testing	Module test report
Integration testing	Integration test report, Final user manual
System testing	System test report
Acceptance testing	Final system plus documentation

Process Model Visibility

Process model	Process visibility
Waterfall model	Good visibility, each activity produces some deliverable
Evolutionary development	Poor visibility, uneconomic to produce documents during rapid iteration
Formal transformations	Good visibility, documents must be produced from each phase for the process to continue
Reuse-oriented development	Moderate visibility, it may be artificial to produce documents describing reuse and reusable components.
Spiral model	Good visibility, each segment and each ring of the spiral should produce some document.

Professional Responsibility

- Software engineers should not just be concerned with technical considerations. They have wider ethical, social and professional responsibilities.
- No clear rights and wrongs about many of these issues:
  - Development of military systems
  - Whistle blowing

Ethical Issues

- Confidentiality
- Competence
- Intellectual property rights
- Computer misuse