MANAGEMENT OF SOFTWARE SYSTEMS

Module 1

Rijin IK

Assistant Professor
Department of Computer Science and Engineering
Vimal Jyothi Engineering College
Chemperi

February 11, 2023

Outline

Introduction to Software Engineering

Software Engineering

- Software engineering is the branch of computer science that deals with the design, development, testing, and maintenance of software applications.
- Software engineers apply engineering principles and knowledge of programming languages to build software solutions for end users.

Frequently asked questions about software engineering

What is Software?

- Computer programs and associated documentation.
- Software products may be developed for a particular customer or may be developed for a general market.

What are the Attributes of good software?

- Good software should deliver the required functionality and performance to the user
- And should be maintainable, dependable and usable

What is Software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- from initial conception to operation and maintenance.

Frequently asked questions about software engineering

What are the Fundamental software engineering activities?

- Software specification
- Software development
- Software validation
- software evolution.

What is the Difference between software engineering and computer science

- Computer science focuses on theory and fundamentals;
- software engineering is concerned with the practicalities of developing and delivering useful software.

Frequently asked questions about software engineering

Difference between software engineering and system engineering

- System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

What are the Key challenges facing software engineering

- Coping with increasing diversity
- Demands for reduced delivery times
- Developing trustworthy software

What are the Costs of software engineering

- Roughly 60% of software costs are development costs, 40% are testing costs.
- For custom software, evolution costs often exceed development costs.

Frequently asked questions about software engineering

What are the Best software engineering techniques and methods

- While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed.
- You can't, therefore, say that one method is better than another.

What differences has the Internet made to software engineering?

- Not only has the Internet led to the development of massive, highly distributed, service-based systems, it has also supported the creation of an "app" industry for mobile devices which has changed the economics of software.

Professional software development

Professional software development

- Software development is a professional activity in which software is developed
  - for business purposes.
  - for inclusion in other devices.
  - as software products.
- Professional software is intended for use by someone apart from its developer.
- It is maintained and changed throughout its life.
- It includes techniques that support program **specification, design, and evolution**, none of which are normally relevant for personal software development.

Professional software development

- Software is not just the programs themselves but also all associated
  - documentation,
  - libraries,
  - support websites,
  - configuration data that are needed to make these programs useful.
- A system may consist of several separate programs and configuration files that are used to set up these programs. It may include
  - System documentation, which describes the structure of the system.
  - User documentation, which explains how to use the system.
  - Websites for users to download recent product information.
- Software engineers are concerned with developing software products, that is, software that can be sold to a customer.

Professional software development

Essential attributes of good software

- Maintainability
  - Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
- Dependability and security
  - Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
- Efficiency
  - Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
- Acceptability
  - Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Professional software development

Software products:-There are two kinds of software product

1. Generic products.
    - These are stand-alone systems that are produced by a development organization and sold on the open market to any customer who is able to buy them.
    - mobile apps
2. Customized software.
    - These are systems that are commissioned by and developed for a particular customer.
    - A software contractor designs and implements the software especially for that customer.

Professional software development

Software engineering

- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
- Two key phrases:
  1. **Engineering discipline.**
     - Using appropriate theories and methods to solve problems within the organizational and financial constraints.
  2. **All aspects of software production**
     - Software engineering is not just concerned with the technical processes of software development.
     - It also includes activities such as software project management and the development of tools, methods, and theories to support software development.

Professional software development

Why software engineering is important?

Software engineering is important for two reasons:

1. More and more, individuals and society rely on advanced software systems.
    - We need to be able to produce reliable and trustworthy systems economically and quickly.
2. It is usually cheaper, in the long run, to use software engineering methods and techniques for professional software systems rather than just write programs as a personal programming project.

Failure to use software engineering method leads to higher costs for testing, quality assurance, and long-term maintenance.

Professional software development

SOFTWARE PROCESS

The systematic approach that is used in software engineering is sometimes called a software process. A software process is a sequence of activities that leads to the production of a software product.

Four fundamental activities are common to all software processes.

1. **Software specification**, where customers and engineers define the software that is to be produced and the constraints on its operation.
2. **Software development**, where the software is designed and programmed.
3. **Software validation**, where the software is checked to ensure that it is what the customer requires.
4. **Software evolution**, where the software is modified to reflect changing customer and market requirements.

Professional software development

General issues that affect most Software

1. Heterogeneity:
    - Operating in a distributed systems across networks that include different types of computer and mobile devices, it is often have to integrate new software with older systems written in different programming languages.
2. Business and social change:
    - Many traditional software engineering techniques are time consuming, and delivery of new systems often takes longer than planned. But it is needed to change their existing software and to rapidly develop new software.
3. Security and trust:
    - We have to make sure that malicious users cannot successfully attack our software and that information security is maintained.
4. Scale:
    - Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

Professional software development

Software Engineering Diversity

- There are many different types of software system and there is no universal set of software techniques that is applicable to all of these.
- The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.

Professional software development

Application types

- Stand-alone applications
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- Interactive transaction-based applications
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- Embedded control systems
  - These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.

Professional software development

Application types cont..

- Batch processing systems
    - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
- Entertainment systems
    - These are systems that are primarily for personal use and which are intended to entertain the user.
- Systems for modeling and simulation
    - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

Professional software development

Application types cont..

- Data collection systems
    - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- Systems of systems
    - These are systems that are composed of a number of other software systems.
    - Software that has already been developed rather than write new software.

Professional software development

Software Engineering and the Web

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
- Web services allow application functionality to be accessed over the web.
- Cloud computing is an approach to the provision of computer services where applications run remotely on the 'cloud'.
    - Users do not buy software buy pay according to use

Professional software development

Web software Engineering

- Software reuse is the dominant approach for constructing web-based systems.
  - When building these systems, you think about how you can assemble them from pre-existing software components and systems.
- Web-based systems should be developed and delivered incrementally.
  - It is now generally recognized that it is impractical to specify all the requirements for such systems in advance.
- User interfaces are constrained by the capabilities of web browsers.
  - Technologies such as AJAX allow rich interfaces to be created within a web browser but are still difficult to use. Web forms with local scripting are more commonly used.

Professional software development

Web based Software Engineering

- Web-based systems are complex distributed systems but the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system
- The fundamental ideas of software engineering, discussed in the previous section, apply to web-based software in the same way that they apply to other types of software system.

Software engineering ethics

Software engineering ethics

- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.

Software engineering ethics

Issues of Professional Responsibility

- Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- Competence
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.
- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

Software engineering ethics

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join
- The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

ACM/IEEE Code of Ethics

Software Engineering Code of Ethics and Professional Practice

- ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

- The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.
- Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

ACM/IEEE Code of Ethics

Ethical principles

1. **PUBLIC** — Software engineers shall act consistently with the public interest.
2. **CLIENT AND EMPLOYER** — Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. **PRODUCT** — Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. **JUDGMENT** — Software engineers shall maintain integrity and independence in their professional judgment.
5. **MANAGEMENT** — Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

ACM/IEEE Code of Ethics

Ethical principles cont...

6 **PROFESSION** — Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7 **COLLEAGUES** — Software engineers shall be fair to and supportive of their colleagues.

8 **SELF** — Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession

ACM/IEEE Code of Ethics

Rationale for the code of ethics

- Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems.
- Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession.

Case Studies

- A personal insulin pump
  - An embedded system in an insulin pump used by diabetics to maintain blood glucose control.
- A mental health case patient management system
  - A system used to maintain records of people receiving care for mental health problems.
- A wilderness weather station
  - A data collection system that collects data about weather conditions in remote areas

Insulin pump control system

An insulin pump is a medical system that simulates the operation of the pancreas (an internal organ). The software controlling this system is an embedded system that collects information from a sensor and controls a pump that delivers a controlled dose of insulin to a user.

- Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- Calculation based on the rate of change of blood sugar levels.
- Sends signals to a micro-pump to deliver the correct dose of insulin.
- Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.

Insulin pump control system

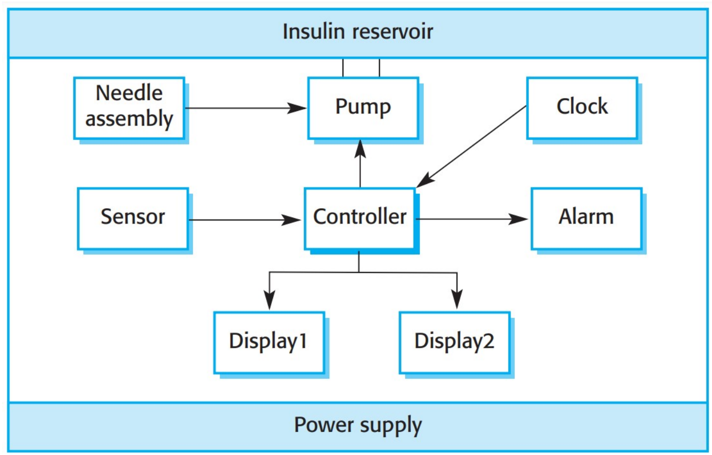

Figure 1: Insulin pump hardware architecture

Insulin pump control system

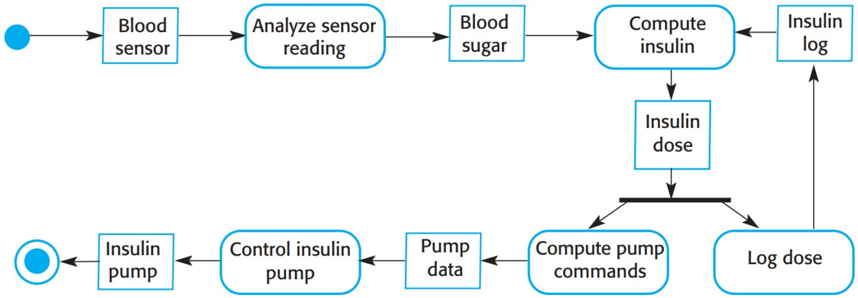

Figure 2: Activity model of the insulin pump

Insulin pump control system

Essential high-level requirements

- The system shall be available to deliver insulin when required.
- The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.
- The system must therefore be designed and implemented to ensure that the system always meets these requirements.

A Patient Information System for Mental Health Care

- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
- Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
- To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

A Patient Information System for Mental Health Care

- The **MHC-PMS (Mental Health Care-Patient Management System)** is an information system that is intended for use in clinics.
- It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.

A Patient Information System for Mental Health Care

MHC-PMS goals

- To generate management information that allows health service managers to assess performance against local and government targets.
- To provide medical staff with timely information to support the treatment of patients. The organization of the MHC-PMS

A Patient Information System for Mental Health Care

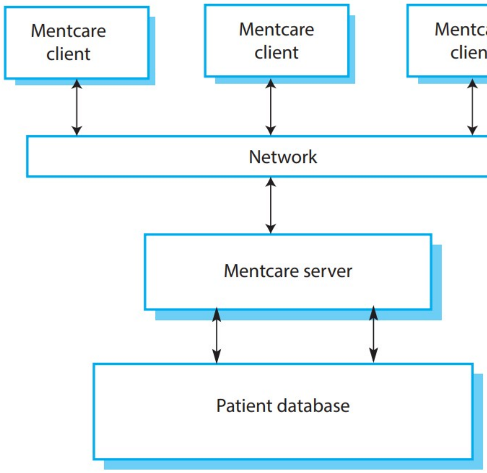

Figure 3: The organization of the Mentcare system

A Patient Information System for Mental Health Care

MHC-PMS key features

- **Individual care management**
  - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.
- **Patient monitoring**
  - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.
- **Administrative reporting**
  - The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

A Patient Information System for Mental Health Care

MHC-PMS concerns

- Privacy
  - It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.
- Safety
  - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
  - The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

Wilderness weather station

Wilderness weather station

- The government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.
- Weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed and wind direction.
    - The weather station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure and the rainfall over a 24-hour period.
    - Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.

Wilderness weather station

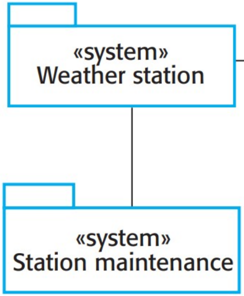

Figure 4: The weather station's environment

Wilderness weather station

The weather station system

- This is responsible for collecting weather data, carrying out some initial data processing and transmitting it to the data management system.

The data management and archiving system

- This system collects the data from all of the wilderness weather stations, carries out data processing and analysis and archives the data.

The station maintenance system

- This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems.

Wilderness weather station

Additional software functionality

- Monitor the instruments, power and communication hardware and report faults to the management system.
- Manage the system power, ensuring that batteries are charged whenever the environmental conditions permit but also that generators are shut down in potentially damaging weather conditions, such as high wind.
- Support dynamic reconfiguration where parts of the software are replaced with new versions and where backup instruments are switched into the system in the event of system failure.

The software process

The software process

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
    - **Specification** – defining what the system should do;
    - **Design and implementation** – defining the organization of the system and implementing the system;
    - **Validation** – checking that it does what the customer wants;
    - **Evolution** – changing the system in response to changing customer needs.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective

The software process

Software process descriptions: When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.

- Process descriptions may also include
  - **Products**, which are the outcomes of a process activity;
  - **Roles**, which reflect the responsibilities of the people involved in the process;
  - **Pre- and post-conditions**, which are statements that are true before and after a process activity has been enacted or a product produced.

The software process

Plan-driven and agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software process models

Software process models

- Sometimes called a Software Development Life Cycle (SDLC model)
- a simplified representation of a software process.
- Each process model represents a process from a particular perspective and thus only provides partial information about that process.
- For example, a process activity model shows the activities and their sequence but may not show the roles of the people involved in these activities.
- The general process models (sometimes called process paradigms)
    1. The waterfall model
    2. Incremental development
    3. Integration and configuration

THE WATERFALL MODEL

The Waterfall model

This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as

- Requirements specification
- software design
- Implementation
- testing.

- The first published model of the software development process
- The waterfall model is an example of a plan-driven process
- Plan and schedule all of the process activities before starting software development.

THE WATERFALL MODEL

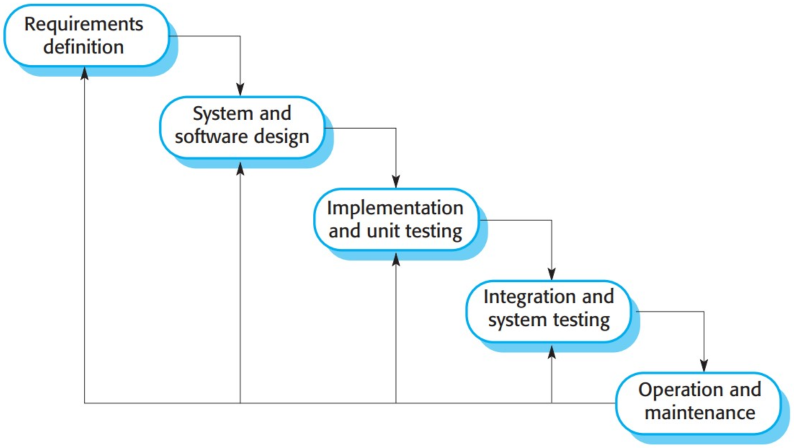
Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 5: The Waterfall model

THE WATERFALL MODEL

Activities are,

1. **Requirements analysis and definition** The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.
2. **System and software design** The systems design process allocates the requirements to either hardware or software systems. It establishes an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.
3. **Implementation and unit testing** During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

THE WATERFALL MODEL

4. **Integration and system testing** The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
5. **Operation and maintenance** Normally, this is the longest life-cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors that were not discovered in earlier stages of the life cycle, improving the implementation of system units, and enhancing the system's services as new requirements are discovered.

THE WATERFALL MODEL

Waterfall Model Problems:

- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.
- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
    - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
    - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
    - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

INCREMENTAL DEVELOPMENT

Incremental Development

- This approach interleaves the activities of specification, development, and validation.
- The system is developed as a series of versions (increments), with each version adding functionality to the previous version.

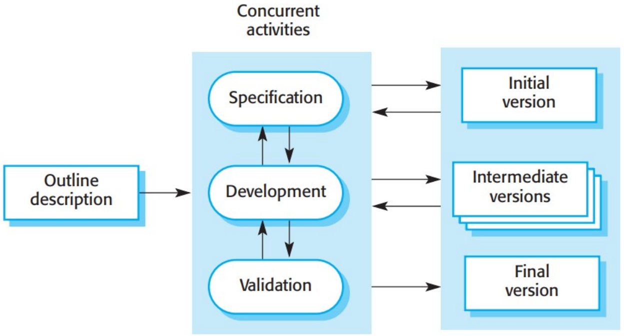

Figure 6: Incremental development

INCREMENTAL DEVELOPMENT

Incremental development has three major advantages over the waterfall model:

1. The cost of accommodating changing customer requirements is reduced.
   - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
2. It is easier to get customer feedback on the development work that has been done.
   - Customers can comment on demonstrations of the software and see how much has been implemented.
3. Early delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included.
   - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

INCREMENTAL DEVELOPMENT

From a management perspective, the incremental approach has two problems:

1. The process is not visible.
   - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost effective to produce documents that reflect every version of the system.
2. System structure tends to degrade as new increments are added.
   - Regular change leads to messy code as new functionality is added in whatever way is possible. It becomes increasingly difficult and costly to add new features to a system.

INTEGRATION AND CONFIGURATION(Reuse-oriented software engineering)

- This approach relies on the availability of reusable components or systems.
- The system development process focuses on configuring these components for use in a new setting and integrating them into a system.

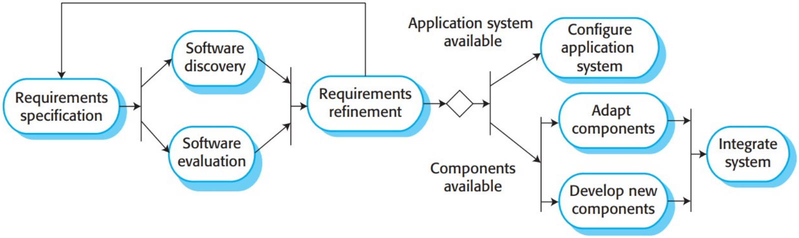

Figure 7: Reuse oriented software engineering

INTEGRATION AND CONFIGURATION

- This often happens informally when people working on the project know of or search for code that is similar to what is required.
- They look for these, modify them as needed, and integrate them with the new code that they have developed.
- Three types of software components are frequently reused:
    1. Stand-alone application systems that are configured for use in a particular environment. These systems are general-purpose systems that have many features, but they have to be adapted for use in a specific application.
    2. Collections of objects that are developed as a component or as a package to be integrated with a component framework.
    3. Web services that are developed according to service standards and that are available for remote invocation over the Internet.

INTEGRATION AND CONFIGURATION

The stages in this process are:

1. **Requirements specification** The initial requirements for the system are proposed. These do not have to be elaborated in detail but should include brief descriptions of essential requirements and desirable system features.
2. **Software discovery and evaluation** Given an outline of the software requirements, a search is made for components and systems that provide the functionality required. Candidate components and systems are evaluated to see if they meet the essential requirements and if they are generally suitable for use in the system.

INTEGRATION AND CONFIGURATION

The stages in this process are cont...

3. **Requirements refinement** During this stage, the requirements are refined using information about the reusable components and applications that have been discovered. The requirements are modified to reflect the available components, and the system specification is re-defined. Where modifications are impossible, the component analysis activity may be reentered to search for alternative solutions.
4. **Application system configuration** If an off-the-shelf application system that meets the requirements is available, it may then be configured for use to create the new system.
5. **Component adaptation and integration** If there is no off-the-shelf system, individual reusable components may be modified and new components developed. These are then integrated to create the system.

Process activities

Process activities

- Real software processes are interleaved sequences of
  - Technical
  - Collaborative
  - Managerial activities
- Generally, processes are now tool-supported.
- This means that software developers may use a range of software tools to help them, like
  - Requirements management systems
  - Design model editors
  - Program editors
  - Automated testing tools
  - Debuggers.

Process activities

- The four basic process activities of
  1. Specification
  2. Development
  3. Validation
  4. Evolution

  are organized differently in different development processes.
- In the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.
- How these activities are carried out depends on the type of
  - Software being developed.
  - The experience and competence of the developers.
  - The type of organization developing the software.