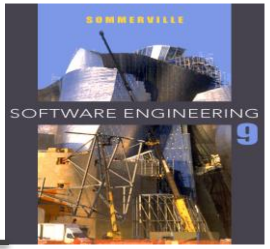
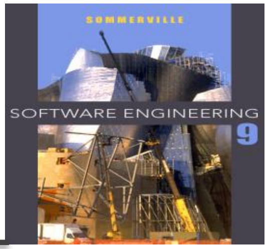


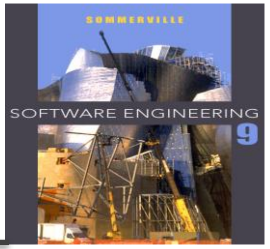
Chapter 4 – Design and Implementation

Topics Covered



- ✧ Design and Implementation
- ✧ An Object-Oriented Design Process
- ✧ Context and Interaction Models
- ✧ Implementation Issues
- ✧ Open Source Development

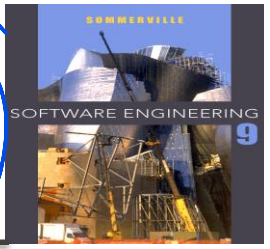
4.1 Design and Implementation



- ✧ Software design and implementation is the stage in the software engineering process at which an executable software system is developed.
- ✧ Software design and implementation activities are invariably inter-leaved.
  - Software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements.
  - Implementation is the process of realizing the design as a program.

4.2 An Object-Oriented Design Process

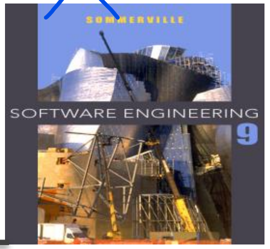
Self $(m < 9)$ x



- ✧ Structured object-oriented design processes involve developing a number of different system models.
- ✧ They require a lot of effort for development and maintenance of these models and, for small systems, this may not be cost-effective.
- ✧ However, for large systems developed by different groups design models are an important communication mechanism.

Process Stages

self
(mcq)



x

✧ There are a variety of different object-oriented design processes that depend on the organization using the process.

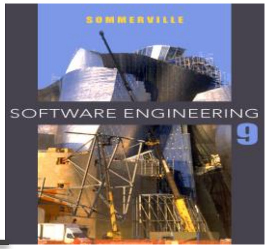
✧ Common activities in these processes include:

- Define the context and modes of use of the system;
- Design the system architecture;
- Identify the principal system objects;
- Develop design models;
- Specify object interfaces.

Self (mcq)

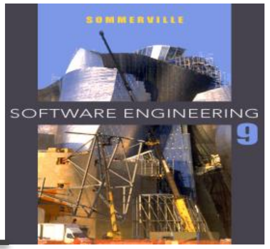
X

4.3 Context and Interaction Models



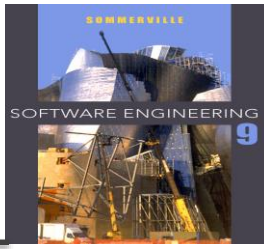
- ✧ A system context model is a structural model that demonstrates the other systems in the environment of the system being developed.
- ✧ An interaction model is a dynamic model that shows how the system interacts with its environment as it is used.

4.4 Implementation Issues



- ✧ Software engineering includes all of the activities involved in software development from the initial requirements of the system through to maintenance and management of the deployed system. A critical stage of this process is, of course, system implementation, where you create an executable version of the software.
- ✧ Implementation may involve developing programs in high- or low-level programming languages or tailoring and adapting generic, off-the-shelf systems to meet the specific requirements of an organization.

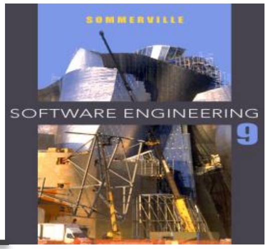
Implementation Issues



✧ Focus here is not on programming, although this is obviously important, but on other implementation issues that are often not covered in programming texts:

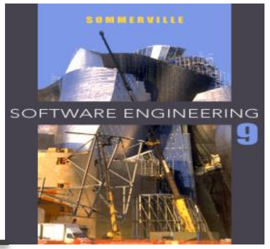
- Reuse Most modern software is constructed by reusing existing components or systems. When you are developing software, you should make as much use as possible of existing code.
- Configuration management During the development process, you have to keep track of the many different versions of each software component in a configuration management system.
- Host-target development Production software does not usually execute on the same computer as the software development environment. Rather, you develop it on one computer (the host system) and execute it on a separate computer (the target system).

4.4.1 Reuse



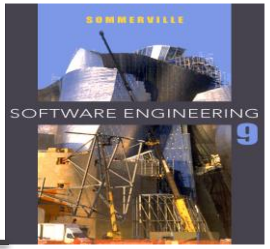
- ✧ From the 1960s to the 1990s, most new software was developed from scratch, by writing all code in a high-level programming language.
  - The only significant reuse or software was the reuse of functions and objects in programming language libraries.
- ✧ Costs and schedule pressure mean that this approach became increasingly unviable, especially for commercial and Internet-based systems.
- ✧ An approach to development based around the reuse of existing software emerged and is now generally used for business and scientific software.

Reuse Levels



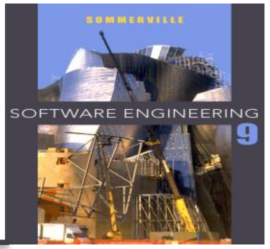
- ✧ The abstraction level
  - At this level, you don’t reuse software directly but use knowledge of successful abstractions in the design of your software.
- ✧ The object level
  - At this level, you directly reuse objects from a library rather than writing the code yourself.
- ✧ The component level
  - Components are collections of objects and object classes that you reuse in application systems.
- ✧ The system level
  - At this level, you reuse entire application systems.

4.4.2 Configuration Management



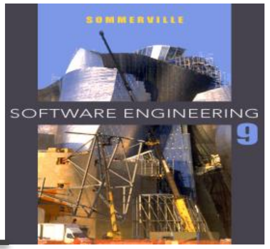
- ✧ Configuration management is the name given to the general process of managing a changing software system.
- ✧ The aim of configuration management is to support the system integration process so that all developers can access the project code and documents in a controlled way, find out what changes have been made, and compile and link components to create a system.

Configuration Management Activities



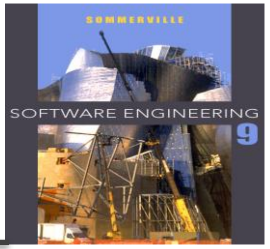
- **Version management**, where support is provided to keep track of the different versions of software components. Version management systems include facilities to coordinate development by several programmers.
- **System integration**, where support is provided to help developers define what versions of components are used to create each version of a system. This description is then used to build a system automatically by compiling and linking the required components.
- **Problem tracking**, where support is provided to allow users to report bugs and other problems, and to allow all developers to see who is working on these problems and when they are fixed.

4.4.3 Host-Target Development



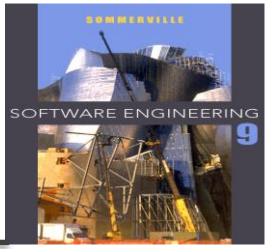
- ✧ Most software is developed on one computer (the host), but runs on a separate machine (the target).
- ✧ More generally, we can talk about a development platform and an execution platform.
  - A platform is more than just hardware.
  - It includes the installed operating system plus other supporting software such as a database management system or, for development platforms, an interactive development environment.
- ✧ Development platform usually has different installed software than execution platform; these platforms may have different architectures.

4.5 Open Source Development

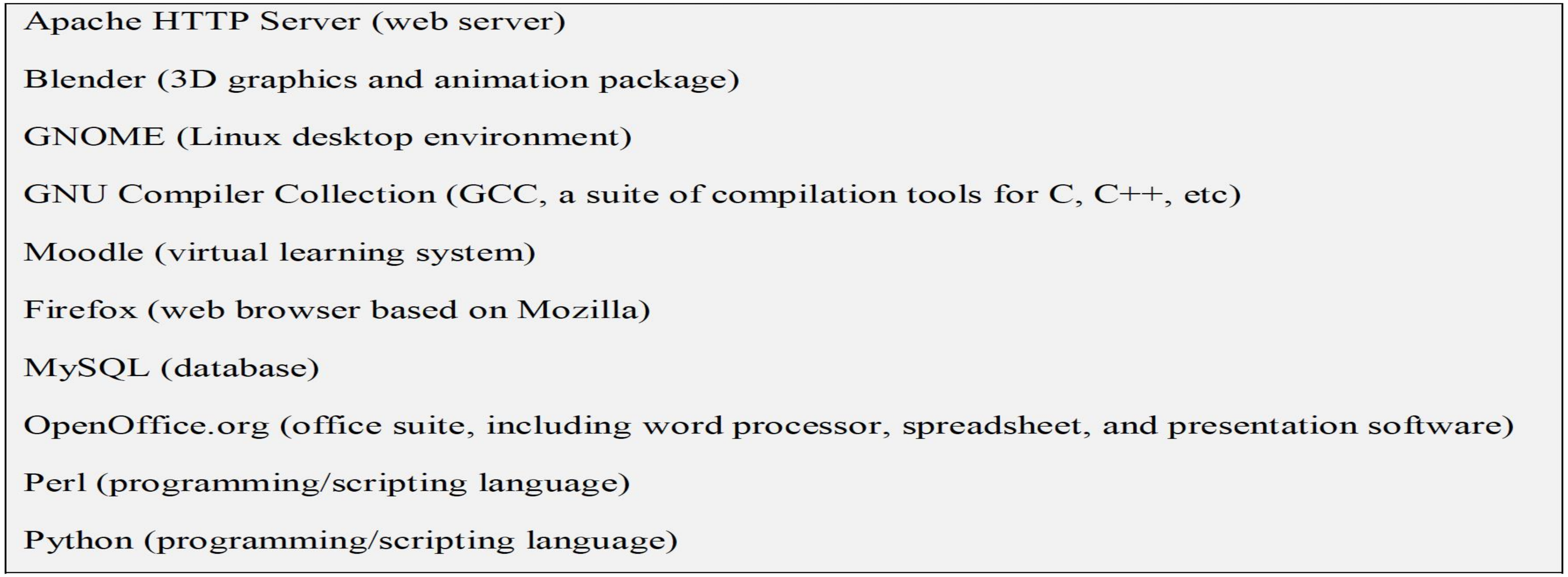


- ✧ Open source development is an approach to software development in which the source code of a software system is published and volunteers are invited to participate in the development process
- ✧ Its roots are in the Free Software Foundation (www.fsf.org), which advocates that source code should not be proprietary but rather should always be available for users to examine and modify as they wish.
- ✧ Open source software extended this idea by using the Internet to recruit a much larger population of volunteer developers. Many of them are also users of the code.

4.5.1 Open Source Systems



- The best-known open source product is, of course, the Linux operating system which is widely used as a server system and, increasingly, as a desktop environment.
- Other important open source products are Java, the Apache web server and the mySQL database management system.



Apache HTTP Server (web server)

Blender (3D graphics and animation package)

GNOME (Linux desktop environment)

GNU Compiler Collection (GCC, a suite of compilation tools for C, C++, etc)

Moodle (virtual learning system)

Firefox (web browser based on Mozilla)

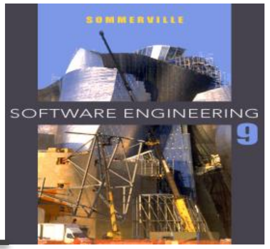
MySQL (database)

OpenOffice.org (office suite, including word processor, spreadsheet, and presentation software)

Perl (programming/scripting language)

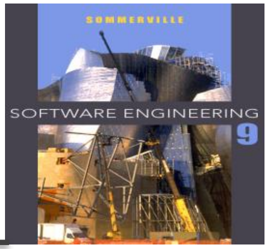
Python (programming/scripting language)

4.5.2 Open Source Business



- ✧ More and more product companies are using an open source approach to development.
- ✧ Their business model is not reliant on selling a software product but on selling support for that product.
- ✧ They believe that involving the open source community will allow software to be developed more cheaply, more quickly and will create a community of users for the software.

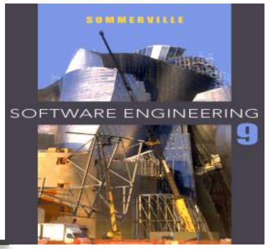
4.5.3 Open Source Licensing



✧ A fundamental principle of open-source development is that source code should be freely available, this does not mean that anyone can do as they wish with that code.

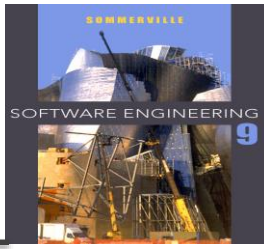
- Legally, the developer of the code (either a company or an individual) still owns the code. They can place restrictions on how it is used by including legally binding conditions in an open source software license.
- Some open source developers believe that if an open source component is used to develop a new system, then that system should also be open source.
- Others are willing to allow their code to be used without this restriction. The developed systems may be proprietary and sold as closed source systems.

Key Points



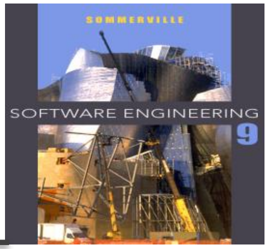
- ✧ Software design and implementation are inter-leaved activities. The level of detail in the design depends on the type of system and whether you are using a plan-driven or agile approach.
- ✧ The process of object-oriented design includes activities to design the system architecture, identify objects in the system, describe the design using different object models and document the component interfaces.
- ✧ A range of different models may be produced during an object-oriented design process. These include static models (class models, generalization models, association models) and dynamic models (sequence models, state machine models).
- ✧ Component interfaces must be defined precisely so that other objects can use them. A UML interface stereotype may be used to define interfaces.

Key Points



- ✧ When developing software, you should always consider the possibility of reusing existing software, either as components, services or complete systems.
- ✧ Configuration management is the process of managing changes to an evolving software system. It is essential when a team of people are cooperating to develop software.
- ✧ Most software development is host-target development. You use an IDE on a host machine to develop the software, which is transferred to a target machine for execution.
- ✧ Open source development involves making the source code of a system publicly available. This means that many people can propose changes and improvements to the software.

Review questions



1. Write short notes on Object-Oriented Design Process. List its common activities.
2. Define: system context model and interaction
3. List and explain various implementation issues.
4. What are the different levels of software reuse
5. What is meant by configuration management? List down its activities.
6. Write short notes on Host-target development.
7. Write briefly about open source development system.