NIBM WORLD WIDE

ebook

SOFTWARE ENGINEERING DSE

APPLICATION DOMAINS

Lesson 01 – Software Application Domains

What is a Software?

Software is a collection of computer programs, procedures, rules and associated documentation and data pertaining to the operation of a computer system.

Software is traditionally divided into two categories.

- **System Software**
- **Application Software**

Figure 1.0.1 Software vs Hardware

System Software: Designed to operate and control the computer hardware and provide a platform for running the application software. Operating Systems and computer drivers.

Figure 1.0.2 Examples for System Software

Application Software: Stand-alone programs that solve a specific business need. Office package, Skype etc.

Figure 1.0.3 Examples for Application Software

Engineering/Scientific Software: Characterized by number crunching algorithms such as automotive stress analysis, space engineering.

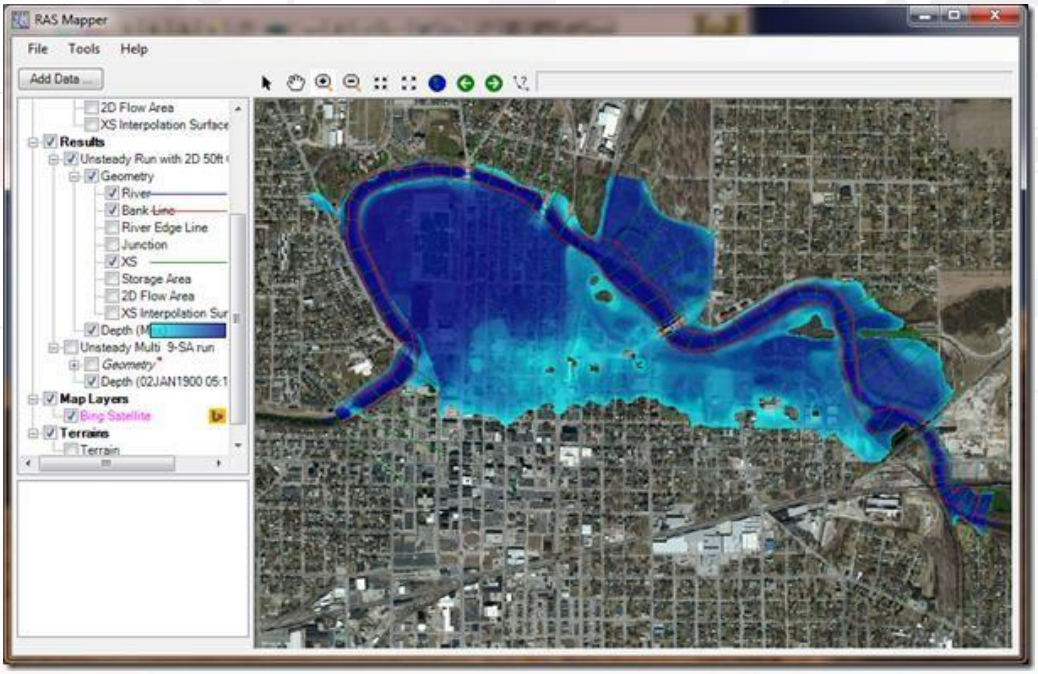

Figure 1.0.4 Engineering/Scientific software

Embedded Software: Resides within a product or system and is used to implement and control features and functions for the end user and for the system itself.

Ex: Key pad control for microwave oven, Anti-lock brakes, Auto-focus camera, Teller machines.

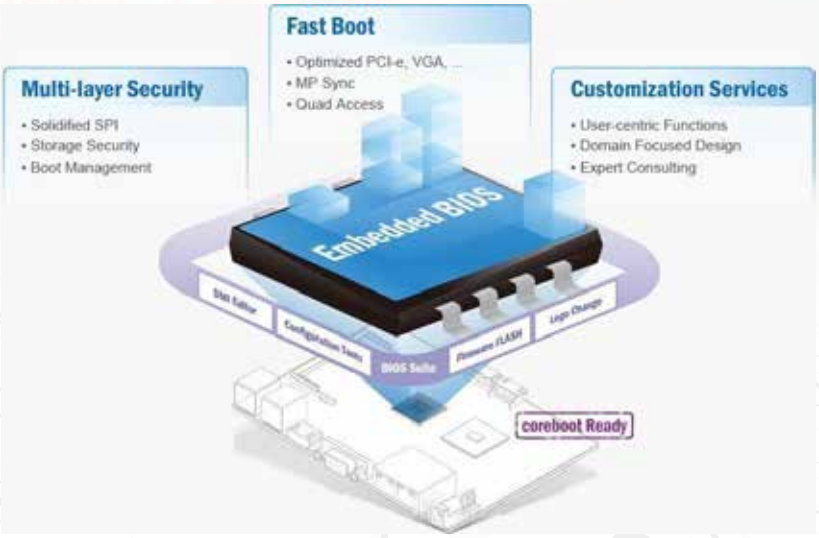

Figure 1.0.5 Embedded Software

Product Line Software: Focus on a limited marketplace to address mass consumer market.

Ex: Database management systems, Adobe Photoshop

Figure 1.0.6 Product Line Software

Web Applications: The network-centric software category spans a wide array of applications.

Figure 1.0.7 Web applications

Open Source: Free source and free tools open to the computing community.

Figure 1.0.7 Open Source Software

Software Engineering & Software Engineer

What is Software Engineering?

The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

Software Engineering is a Layered Technology.

Figure 1.0.8 Software Engineering Layers

- **Quality Focus**: The bedrock that support SE is quality focus.

- **Process**: The foundation of the SE. Process defines a framework that must be established for effective delivery of software engineering technology.

- **Methods**: Provide the technical for building software. Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, testing and support.

- **Tools**: Provide automated or semi-automated support for the process and the methods.

Importance of Software Engineering

The economies of all developed nations are dependent on software.

Individuals, business and governments increasingly rely on software for strategic and tactical decision making as well as day to day operations and control.

If the software fails it will be a huge impact to those parties. It follows that software should exhibit high quality.

Attributes of a Good Software

- **Maintainability**

  The software should be written in a way that it can be evolve to meet changing needs of the customer.

- **Dependability**

  A software must be Trustworthy, Reliable, Safe and Secured.

- **Efficiency**

  A software should be efficient in every way. The software should not make wasteful of system resources (ex. memory, processing cycles).

- **Acceptability (Usability)**

  The software must be acceptable to the group of users for which it's designed for.

Key Challenges of a Software Engineer

- **The Legacy Challenge**

  The legacy challenge is the challenge of maintaining and updating old Software.

- **The Heterogeneity Challenge**

  The heterogeneity challenge increasingly, systems are required to operate as distributed systems across networks that include different types of computers and with different kinds of support systems.

- **The Delivery Challenge**

  The delivery challenge is the challenge of shortening delivery times for large and complex systems without compromising system quality.

- **Update the knowledge & qualifications to latest technologies**

  It is a challenge to update the knowledge and qualifications while working in the industry.

- **Technical Challenges**

It is a challenge to solve day to day technical issues of systems without any support.

- **Dealing with Management & Customer**

  Dealing with your Manager and Customer is the most difficult challenge the Software Engineer ever face.

- **Operational Challenges**

  It is a challenge to run the day to day operations in the company.

Ethical Responsibilities of Software Engineer

Software engineers must behave in an ethical and morally responsible way if they are to be respected as professionals.

- **Confidentiality**

  Engineers should normally respect the confidentiality of their employers or clients.

- **Competence**

  Engineers should not misrepresent their level of competence.

- **Intellectual Property Rights**

  Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc.

- **Computer Misuse**

  Software engineers should not use their technical skills to misuse other people's computers.

Seven Core Principles of Software Engineering

1) **The First Principle: The Reason It All Exists:**

   A software system exists for one reason: to provide value to its users. Ask yourself questions such as: "Does this add real value to the system?"

2) **The Second Principle: KISS (Keep It Simple, Stupid!):**

   All design should be as simple as possible, but no simpler.

3) **The Third Principle: Maintain the Vision:**

A clear vision is essential to the success of a software project.

4) **The Fourth Principle: What You Produce, Others Will Consume:**

In some way or other, someone else will use, maintain, document, or otherwise depend on being able to understand your system.

5) **The Fifth Principle: Be Open to the Future:**

In today software lifetimes are typically measured in months instead of years because of the business environment change.

6) **The Sixth Principle: Plan Ahead for Reuse:**

Reuse saves time and effort. Achieving a high level of reuse is arguably the hardest goal to accomplish in developing a software system.

7) **The Seventh principle: Think:**

This last principle is probably the most overlooked. Placing clear, complete thought before action almost always produces better results.