Notes

10 Advanced Topics in Software Engineering

Syllabus

Component-based software engineering, Client/Server software engineering, Web engineering, Reenfineering, Computer-aided software engineering, Software process improvement, Emerging trends in software engineering.

Contents

10.1 Component Based Software Engineering (CBSE)

Definition : The CBSE is an approach of defining, implementing, integrating loosely coupled independent components into system.
CBSE is basically a reuse based approach.

Benefits

1. By CBSE, it becomes easy to construct understandable and maintainable software.
2. Components are independent entities and they do not interfere in other component's operation.
3. Component implementation is hidden.
4. Communication among the components is through well defined interfaces.
5. Component platform can be shared and ultimately it reduces the cost of development

Drawbacks

1. It is difficult to verify the trustworthiness of the component without source code.
2. The quality of component can not be verified.
3. It is not possible to predict the emergent properties of component compositions.
4. It is difficult to make the trade-offs between the features of various components.

10.1.1 Component and Component Models

According to Council and Heinmann

"Software component is a software element conforms to a component model and can be independently deployed and composed without modification according to a composition standard."

Various characteristics of component are -

1. **Standardized :** The components confirms to some standardized component model. These standards are for defining component interfaces, metadata, composition, deployment and documentation.
2. **Independent :** The component must not depend upon other component while deployment is made.
3. **Deployable :** The component must have an ability to operate as a standalone entity and it should be self-contained.
4. **Composable :** For a component to be composable, all external interactions must take place through publicly defined interfaces.

5. **Documented :** The component must be fully documented so that any user can decide whether particular component is useful to meet the system requirements.

10.1.2 Component Models

The component model specifies how interfaces should be defined, what are the elements to be included in an interface definition.

Examples of component model are -

1. EJB Model
2. COM Model
3. CORBA Model

10.1.3 CBSE Process

Component composition is the process of 'wiring' components together to create a system. When choosing compositions, you have to consider required functionality, non-functional requirements and system evolution.

The steps for CBSE process are as follows -

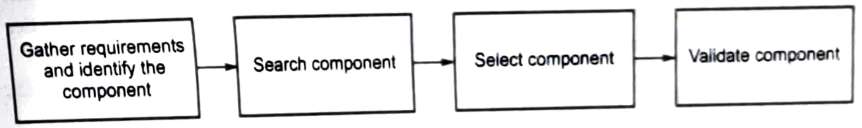


Fig. 10.1.1 CBSE Process

1. Gather the user requirements. Because, using complete set of requirements one can identify as many components as possible for reuse.
2. Refine the requirements depending upon the availability of component. If user requirements can not be satisfied from available components, then sometimes the requirements need to be modified or search for appropriate component is made.
3. During the development process, the discovered component is integrated with software being developed.
4. Then the component validation process is carried out. In this process the compatibility with the interfaces is tested. The selected component must behave as per the requirement. Component validation involves developing a set of test cases for component.

10.2 Client Server Software Engineering

GTU : Summer-2018, Winter-2019,Marks 3

The client-server architecture models the application in such a way that the server consists of a set of services which are demanded by the clients. That means clients demand for the services and servers provide these services to the clients. The clients and servers are the separate processes. The architecture is as shown in the Fig. 10.2.1.

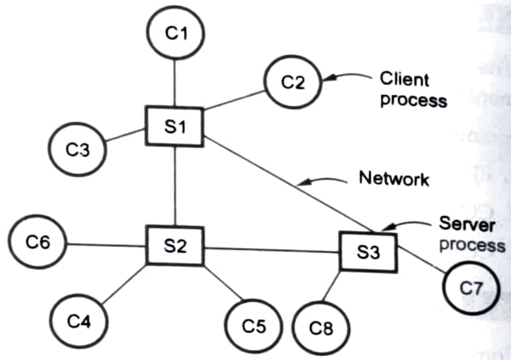


Fig. 10.2.1 Client-server architecture

10.2.1 Two Tier Architecture

In client-server architecture, one server might be connected more than one client. The simplest client-server architecture is called **two-tier** client server architecture in which the application executes on two layers client layer and server layer. The two-tier architecture is of following types -

1. Thin client model

In this model the data management and application logic is implemented on the server and the client is responsible for running the presentation software.

Example

The compiler application makes use of thin client model.

Advantage

- This model is used in **simple** applications.

Disadvantages

- There is a **heavy load** on both the server and the network. The server is responsible for all the computations. This ultimately results in **heavy network traffic** between client and server.
- There is a lot of processing available in modern computing devices. Executing simply the presentation software on clients mean not utilizing the power of these computing devices.

2. Fat client model

In this model, the server is responsible for only data management. The application logic and presentation software is executed on the client itself.

Example

The automatic teller machine (ATM) is an example of fat client model.

The ATM is connected to the server. User operates the ATMs and the information is processed at the client side. The data management part is handled by the server.

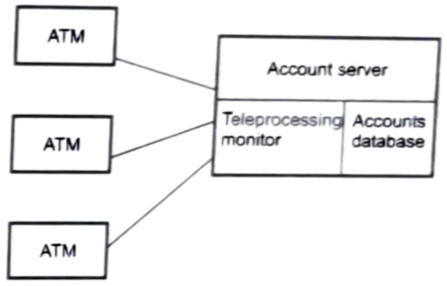


Fig. 10.2.2 ATM system (Fat client model)

Advantages

- This model makes a proper utilization of power computing devices by executing application logic on the clients. Thus processing is distributed effectively in fat client model.
- The application functionality is spread across many computers and thereby efficiency of the overall system gets increased.

Disadvantages

- In this model, the system management is more complex.
- If the application software has to be changed then this involves reinstallation of the software in every client. This process becomes costly.

10.2.2 Three Tier Architecture

In this architecture, the presentation, application processing and data management are logically separate processes and execute on different processors. The architecture is as shown in Fig. 10.2.3.

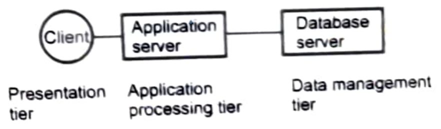


Fig. 10.2.3 Three tier architecture

Example

Internet banking is an example of three tier architecture. In this application the client browses the web page and requests for banking transaction (This is presentation tier). Then the application processes the request and communicates the database server tier). The database to verify the request for transaction. (This is application processing tier).

server stores the bank database and executes the queries. The results of these queries are returned to the application server.

Review Question

1. *Explain client / server software engineering.* **GTU : Summer-2018, Winter-2019, Marks 3**

10.3 Web Engineering

GTU : Summer-2018, Marks 7

Murugesan has defined web engineering as -

Definition : *Web engineering is a sound scientific, engineering and management principles and disciplines. It is also systematic and disciplined approaches to the successful development, deployment and maintenance of high quality web based systems and applications.*

10.3.1 Attributes of Web Based Applications

Web applications are evolving continuously. Hence attributes of web applications are based on the nature of it. Following attributes are encountered in majority of applications -

1. Concurrency

Large number of users can access the same web application concurrently. Sometimes pattern of usage of these web applications among the users may vary greatly.

2. Performance

If a user has to wait for a long time to access the desired web page then he may loose the interest in accessing it.

3. Network intensiveness

The particular web application may be residing on the internet or it may be present on intranet or may be on the extranet. That means the environment in which the web application is residing may vary.

4. Unpredictable load

The load for accessing particular web application varies greatly. That means, some day 10 users might be accessing the web application and immediately on the next day 10000 users may access the same.

5. Availability

100 % availability of particular web application is just impossible. But the application which is used very frequently should be available to its user almost all the time.

6. Data driven

The primary contents on any web application are text, graphics, sound and video. Some web applications may present information to its user using the databases and such databases might be on remote machines.

7. Content sensitivity

It is an important aspect of any web application. The information present of the web application must be authentic. Similarly the contents must be neatly arranged on the web page. The quality of web application is dependant upon this attribute.

8. Continuous evolution

Web applications evolve continuously. There are some web applications that evolve continuously minute to minute. For example web systems showing the share market position.

9. Security

The strong security is the topmost demand for any web application. Sometimes the sensitive information must be conveyed to limited number of users. In such case strong security measure must be applied.

10. Immediacy

Building a web application quickly is today's need. Hence web engineers must us methods for planning, analysis, design, implementation and testing for preparing the target web application within given schedule.

11. Aesthetics

Aesthetic means something related to **the look and feel**. Aesthetic is one of the important attribute of web application. That means the look and feel of any web application should be sophisticated and appealing.

10.3.2 Design Model for Web Based Applications

We have already discussed the Web Engineering design model in section 5.8.

Review Questions

1. *Define the term web engineering. What are the attributes of web based applications ?* **GTU : Summer-2018, Marks 7**
2. *Explain web engineering.*

10.4 Computer Aided Software Engineering (CASE)

GTU : Winter-2012, 2013, 2014, 2017, 2019, Summer-2011, 2014, 2015, 2016, Marks 7

Importance of CASE Tools

- The Computer Aided Software Engineering (CASE) tools automate the project management activities, manage all the work products. The CASE tools assist to perform various activities such as analysis, design, coding and testing.
- Software engineers and project managers make use of CASE tools.
- The use of CASE tools in the software development process reduces the significant amount of efforts.
- CASE is used in conjunction with the process model that is chosen.
- CASE tools help software engineer to produce the high quality software efficiently.
- Use of CASE tool automates particular task of software development activity. But it is always useful to use a set of CASE tools instead of using a single CASE tool. If different CASE tools are not integrated then the data generated by one tool will be an input to other tools. This may also require a format conversions, as tools developed by different vendors may use different formatting. There are chances, that many tools do not allow exporting data and maintain data in proprietary formats.

10.4.1 Building Blocks of CASE

- The CASE tools may exist as a single tools or a collection of multiple tools. These tools must communicate with various elements such as hardware, database, people, network, operating system and so on. This communication creates an **integrated environment**.
- Fig. 10.4.1 represents the building block for CASE. The bottom most layer consists of **environment architecture** and **hardware platform**. The environment architecture consists of collection of system software and human work pattern that is applied during the software engineering process.
- A set of **probability services** connects the CASE tools with the integration framework.
- The **integration framework** is a collection of specialized programs which allows the CASE tools to communicate with the database and to create same look and feel for the end-user. Using probability services CASE tools can communicate with the cross platform elements.
- At the top of this building block a collection of CASE tools exist. CASE tools basically assist the software engineer in developing a complex component.

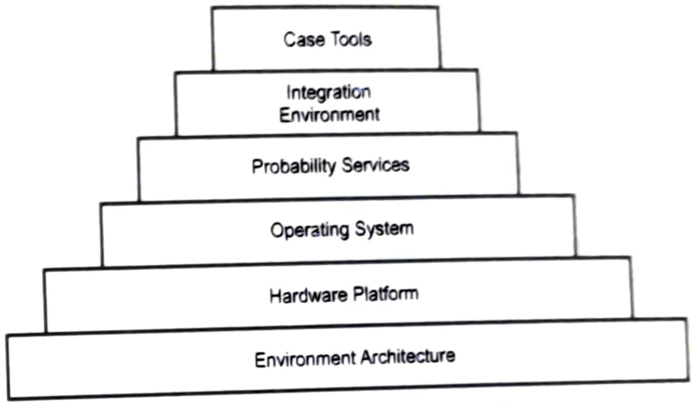


Fig. 10.4.1 Building block for CASE Tools

- CASE tools can exist in variety of manner. A single CASE tool can be used, or a collection of CASE tools may exists which acts as some package. The CASE tools may serve as a bridge between other tools.

10.4.1.1 Taxonomy of CASE Tools

- To create an effective CASE environment, various categories of tools can be developed.
- CASE tools can be classified by
  1. by function or use
  2. by user type (e.g. manager, tester), or
  3. by stage in software engineering process (e.g. requirements, test)
- The taxonomy of CASE tools is as given below.

1) Business process engineering tools

This tool is used to model the business information flow. It represents business data objects, their relationships and how data objects flow between different business areas within a company.

2) Process modeling and management tools

It models software processes. First the processes need to be understood then only it could be modelled. This tool represents the key elements of the processes. Hence it is possible to carry out work tasks efficiently.

3) Project planning tools

These tools help to plan and schedule projects. Examples are PERT and CPM. The objective of this tool is finding parallelism and eliminating bottlenecks in the projects.

4) Risk analysis tools

It helps in identifying potential risks. These tools are useful for building the risk table and thereby providing detailed guidance in identification and analysis of risks. Using this tool one can categorize the risks as catastrophic, critical, marginal, or negligible. A cost is associated with each risk which can be caluculated at each stage of development.

5) Project management tools

These track the progress of the project. These tools are extension to the project planning tools and the use of these tools is to update plans and schedules.

6) Requirements tracing tools

The objective of these tools is to provide a systematic approach to isolate customer requirements and then to trace these requirements in each stage of development.

7) Metrics and management tools

These tools assist to capture specific metrics that provide an overall measure of quality. These tools are indetended to focus on process and product characteristics. For example "defects per function point", "LOC/person-month".

8) Documentation tools

Most of the software development organizations spend lot of time in developing the documents. For this reason the documentation tools provide a way to develop documents efficiently. For example - word processors that give templates for the organization process documents.

9) System software tools

These tools provide services to network system software, object management and distributed component support. For example - e-mail, bulletin boards, and www access.

10) Quality assurance tools

These are actually metrics tools that audit source code to insure compliance with language standards.

11) Database management tool

It provides consistent interfaces for the project for all data, in particular the configuration objects are primary repository elements.

12) Software configuration management tools

It assists with identification, version control, change control, auditing, and status accounting.

13) Analysis and design tools

It creates models of the system. Some create formal models. Others construct data flow models. These models contain representation of data, function and behavior. Such tools helps in architectural, component level and interface design.

14) PRO/SIM tools

These are prototyping and simulation tools. They can help predict real time system response and allow mockups of such systems to be fashioned.

15) Interface design and development tools

These tools are used in developing user interface. It includes various components such as menu, icons, buttons, scrolling mechanisms etc. For example - JAVA, Visual Studio.

16) Prototyping tools

These tools support to define screen layout rapidly for interactive applications.

17) Programming tools

The programming tool category include the programs that support most of the conventional programming languages. For example - compilers, debuggers, editors, database query languages.

18) Web development tools

These tools help in developing the web based applications. The various components of these tools are text, graphics, form, scripts, and applets.

19) Integration and testing tools

These tools include various category of tools such as data acquisition tools, static measurement, dynamic measurement, simulation, cross functional tools.

20) Static analysis tools

The static testing tools are used for deriving the test cases. There are three types of static testing tools.

i. Code based testing tools – These tools take source code as input and generate test cases.

ii. Specialized testing language – Using this language the detailed test specification can be written for each test case.

iii. Requirement-based testing tools – These tools help in designing the test cases as per user requirements.

21) Dynamic analysis tools

These interact with an executing program, checking path coverage, and testing assertions. Intrusive tools insert code in the tested program. Non intrusive tools use a separate hardware processor.

22) Test management tools

These tools manage and co-ordinate regression testing, perform comparisons of output, and act as test drivers.

23) Client/server testing tools

The client server tools are used in client server environment to exercise the GUI and network communication requirements for client and server.

24) Reengineering tools

These tools performs a set of maintenance activities. These tools perform various functions such as

- Reverse engineering to specification tools.
- Code restructuring.
- On-line system re-engineering.

10.4.2 Integrated CASE Environment

- CASE tools create a pool of software engineering information. The integrated CASE environment allows a transfer of information into and out of this pool. For such transfer there is a need for some architectural components. These components are –
  - Database for storing the information
  - Object management system. Because using the objects information can be transferred in the information pool.
  - Control mechanism.
  - User interface
- The Fig. 10.4.2 shows the simple model for integrated CASE environment. This environment consists of various levels.
- The **user interface layer** consists of **interface tool kit** and **presentation protocols.** The **interface tool kit** consists of collection of software required **for** interface management and display objects. The **presentation protocol** decides **a** common look and feel of the presentation interface.
- Then comes **a tools layer.** IT consists of set of tools management **services** (TMS).

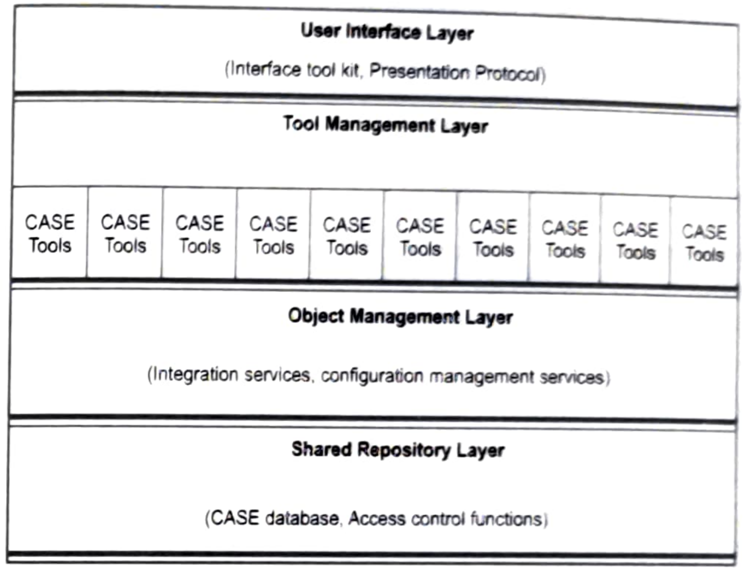


Fig. 10.4.2 Model for integrated CASE environment

- The next layer is **Object management Layer (OML).** It performs the configuration management. The services of this layer allows the identification of all the configuration objects. So that the case tool can be plugged into the integrated CASE environment.
- The bottom most layers is shared **repository layer**. It consists of **CASE database** and **access control functions.** These access control functions help the object management layer to access the CASE Database.

Review Questions

1. *Describe integrated CASE environment.* **GTU : Summer-2011, Winter-2013, Marks 7**
2. *What does CASE stand for ? Explain all CASE components.* **GTU : Winter-2014, Marks 7**
3. *Explain CASE and building blocks of CASE.* **GTU : Summer-2014, Marks 7**
4. *Explain CASE tools and its use/importance in software engineering.* **GTU : Winter-2012, Summer-2015, Marks 7**
5. *Write a short note on : CASE.* **GTU : Summer-2016, Winter-2019, Marks 7**
6. *Explain CASE tools and its use in software engineering.* **GTU : Winter-2017, Marks 3**

10.5 Software Process Improvement

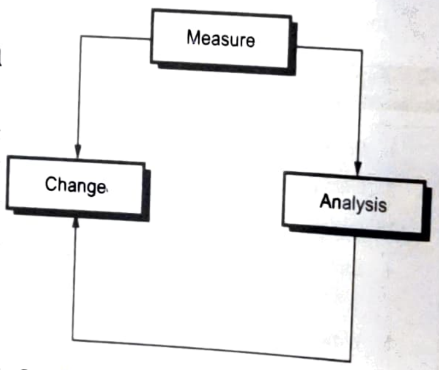
GTU : Summer-2013, 2016, Winter-2018, Marks 7

Process improvement means understanding the existing process and changing these processes to increase the product quality, to reduce the cost and/or to reduce the development time in order to accelerate the project.

Following are process attributes focusing on the concept of process improvement -

1. **Understandability :** "Is the process definition easy to understand ?" - This aspect is focused for understandability.
2. **Visibility :** "Do the process activities happen in such a manner that the progress of process is visible ?" - This aspect is focused for visibility.
3. **Reliability :** "Is the process design in such a manner that the process errors are avoided before getting introduced in the product as the product error ?" - This aspect is focused for reliability.
4. **Supportability :** "To what extent the CASE tools support the process activities ?" - This aspect is focused.
5. **Robustness :** "Will the system continue to work even if some unexpected errors occur ?" - This aspect is focused.
6. **Acceptability :** "Is the desired process acceptable for producing the software product ?" - This aspect is focused.
7. **Rapidity :** "How fast the processes can be delivered into the system ?" - This aspect is focused.
8. **Maintainability :** "Can the process evolve if any changes or modifications occur in the system ?" - This aspect is focused.

10.5.1 SPI Model



Process improvement is the crucial activity and involves following stages -

1) **Process Improvement :** Current project attributes are measured. Then identify the process that needs to be improved.

2) **Process Analysis :** The identified process is analysed to obtain the bottlenecks and weaknesses.

3) **Process Change :** After analysis phase some processed need to get changed. In this phase the required changes are incorporated in the process.

Review Questions

1. *Explain software process improvement. Explain various elements of SPI framework and maturity model.* **GTU : Summer-2013, Marks 7**
2. *Explain software process improvement with various elements of SPI framework.* **GTU : Summer-2016, Winter-2018, Marks 7**

10.6 Emerging Trends in Software Engineering

Software engineering is a continuously changing stream. Software Intensive systems have become foundation of virtually every modern technology. The software must be demonstrably safe, secure and reliable. Requirements will emerge as systems evolve. Thus the word is demanding for better, more reliable software. Hence new techniques and trends are used in software engineering. Let us learn few emerging trends in software engineering.

10.6.1 Process Trends

The fundamental unit of business, organizational and cultural trends is **process.**

The **six process trends** suggested by Conradi and Fuggetta are as follows -

1. In a rapid software development the focus will be on short term goals that have product innovation.
2. Software engineers have a good sense that where the process is weak. Therefore the process change will be driven by this knowledge.
3. Automation software process technology is used only within those processes which will get benefited most by such automation.
4. Emphasis of process development will be on return of investment of software development activities.
5. As time passes the software community will understand that the software development has a greater impact on sociology and anthropology.
6. New modes of learning will facilitate the effective software processes.

10.6.2 Collaborative Development

- Today, software engineers collaborate across the international boundaries, and every one of them shares the information.
- The challenges over next decade will be to develop methods and tools that facilitate the collaboration for software development.
- Number of success factors that lead to successful collaborative efforts are -

1. **Shared Goals** : The project goals must be clearly specified and all the stakeholders must understand and agree with them.
2. **Shared Culture** : The cultural difference should be clearly defined and proper communication and educational approach must be adopted.
3. **Shared Process** : Process is the basic unit of collaborative project. All the team members work on it to create a successful working system.
4. **Shared Responsibilities** : Every team member must recognize the needs of the customer and should work to give best possible solution.

10.6.3 Model Driven Development

- During the software design phase, architectural and component level abstractions of the system are represented and assessed.
- In the subsequent phases of software development, these design components must be translated into programming language representations. Thus high level abstraction is transformed into low level abstractions. This low level abstract model should work in a specific computing environment.
- **Model driven software development** is an approach of software development in which Domain Specific Modeling Language (DSML) is combined with transformation engines and generators in such a way that high level abstraction is converted into low level abstraction.
- **Domain specific modeling language** represents the application structure, behavior and requirements within particular application domain.
- The domain specific modeling language describes the relationships among various domains, their semantics and the constraints associated with these domains.

10.6.4 Test Driven Development

Test Driven software Development is software development technique in which there occurs very short development cycle followed by repetitive tests. A Test First Design (TFD) approach is adopted in TDD. That means, the first step is always to add the test. If test is passed then add another test. But if test is failed then make some corrections in the code, run the automated test until all the tests get success.

The test driven development cycle

Following are the basic steps followed in TDD cycle.

- **Add a Test**

In the test driven development the development cycle begins by writing a test. To write these tests, developer must understand the feature and requirements clearly.

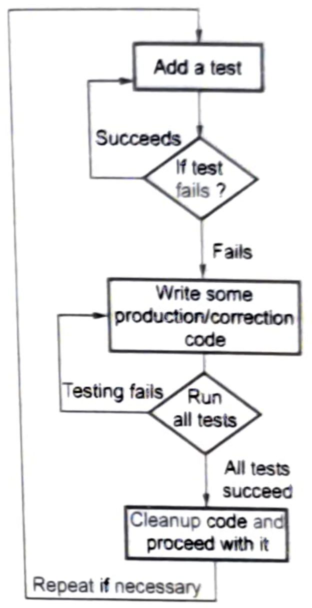


Fig. 10.6.1 Test driven development cycle

- **Run tests**

Execute the test for existing code. If the tests succeed then iteratively add new test.

- **Write correcting code**

If the tests fail then correct the existing code and send it for testing. This will increase the confidence in the code.

- **Run automated tests**

If all the test cases pass in this manner then programmer can be confident that code meets all the tested requirements.

- **Repeat**

Starting with another new test case the cycle will be repeated to push forward the functionality.

Finally the code can be cleaned up. By re-running the test cases the developer is re-factoring the code, testing is without affecting the its functionality.

Review Questions

1. *Explain various emerging trends in software engineering.*
2. *Write a short note on - Test driven development.*

Notes