Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 12 Issue 13 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

System Design Principles – Reuse: Online Attendance System

By Ch V M K Hari, K S V Krishna Srikanth & N S S S Girish Kumar
Institute of Technology GITAM University

Abstract - Software engineering is an engineering approach for software development. In order to develop large software several phases has to be followed by the developer to achieve good quality software; cost effectively. System Design is the most important activity in software development which reflects reusability. System Design specifies what a new or modified system is going to do. To achieve good quality software, the primary characteristics of neat module decomposition are low coupling {data coupling}, high cohesion {functional cohesion} and top-down approach has to be followed. We applied these principles on developing Online Attendance System and observed reusability of code. The system has been successfully tested in our institute. Effective design principles always lead to an effective reusability which in turn benefited with Return on Investment (ROI).

Keyterms : Design, Coupling, Cohesion, Reusability, Online Attendance System.

GJCST-C Classification: D.2.0

Strictly as per the compliance and regulations of:

System Design Principles – Reuse: Online Attendance System

Ch V M K Hari [α], K S V Krishna Srikanth [σ] & N S S S Girish Kumar [ρ]

Abstract - Software engineering is an engineering approach for software development. In order to develop large software several phases has to be followed by the developer to achieve good quality software; cost effectively. System Design is the most important activity in software development which reflects reusability. System Design specifies what a new or modified system is going to do. To achieve good quality software, the primary characteristics of neat module decomposition are low coupling {data coupling}, high cohesion {functional cohesion} and top-down approach has to be followed. We applied these principles on developing Online Attendance System and observed reusability of code. The system has been successfully tested in our institute. Effective design principles always lead to an effective reusability which in turn benefited with Return on Investment (ROI).

Keyterms : Design, Coupling, Cohesion, Reusability, Online Attendance System.

I. Introduction

Software engineering is the application of a *systematic, disciplined, quantifiable approach* to the development, operation, maintenance and retirement of software [1]. The use of the term *systematic approach* implies that methodologies are used for developing software. Software engineering includes process, managing techniques, technical methods, and use of tools. Software engineering deals with the problem of developing 'large' software. Software engineering helps to reduce the programming complexity. Software engineering principles or methodologies use two important techniques (1) abstraction and (2) decomposition to reduce problem complexity. The principle of *abstraction* implies that a problem can be simplified by omitting irrelevant details. The principle of *decomposition* states that a complex problem is divided into several smaller problems and then smaller problems are solved one by one.

The ***goal*** of software engineering is to develop high quality software with low cost i.e., within time and budget constraints. New software systems are built from the old ones and all must interoperate and cooperate with each other.

Software is meant to solve some problem of the *client* (the people whose needs are to be satisfied by the software). The problem is to develop software systematically to satisfy the needs of clients [2]. There are some factors for basic problem which affect the approaches selected to solve the problem and these factors are the primary forces that drive the progress and development in the field of software engineering.

Software Development Life Cycle activities will have several stages where in one identifies the problem to be solved, develop a design, writes the code and so on. Software life cycle defines entry and exit criteria for every phase. A phase can start only if its phase-entry criteria have been satisfied. Without software life cycle it becomes difficult for software project managers to monitor the progress of the project. The software life-cycle [5, 6] consists of: feasibility study, requirements analysis, design, construction, testing (validation), deployment and maintenance. The development process tends to run iteratively through these phases rather than linearly.

Upon successfully demonstrating the feasibility of a project, the requirements analysis begins. The design starts after the requirements analysis is complete, and coding begins after the design is complete. Once the programming is completed, the code is integrated and testing is done. Upon successful completion of testing, the system is installed. After this, the regular operation and maintenance of the system takes place.

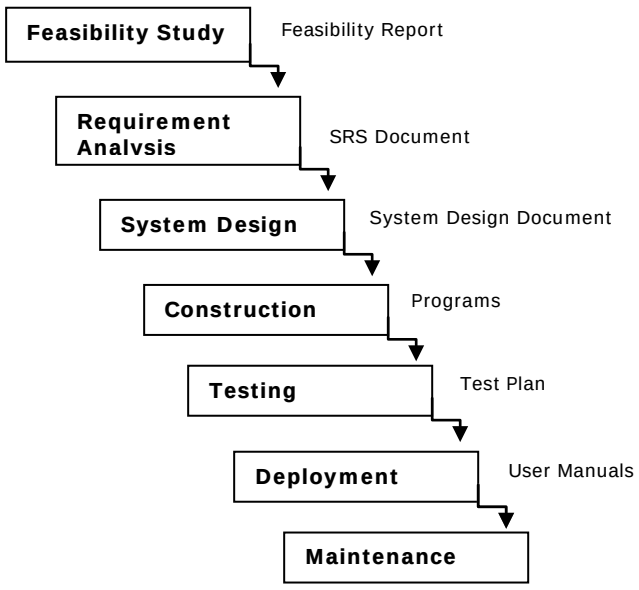

Fig. 1 : Software Life Cycle

Author α : Department of IT GITAM Institute of Technology GITAM University. E-mail : kurmahari@gmail.com
Author σ : Department of IT GITAM Institute of Technology GITAM University. E-mail : ksvksrikanth@gmail.com
Author ρ : Department of IT GITAM Institute of Technology GITAM University. E-mail : girishnsss@hotmail.com

System Design is the process of defining architecture, modules, interfaces and data for a system to specify the requirements of proposed system. The design of a system is essentially a blueprint or a plan for a solution for the system. It specifies what components are needed for the system, their behavior and how they should be interconnected [2, 3]. The design activity begins when the SRS document is available. During design we further refine the architecture. The *goal* is to transform the requirements specified in the document into a structure that is suitable for implementation in some programming language.

Design focuses on the *module view*. A module of a system can be considered a system, with its own modules. A system as set of modules with defined behavior interacts with each other in a defined manner may produce some behavior or services for its environment. A good software design can be arrived infrequently by using single step procedure but rather through several iterations through a series of steps.

The design characteristics include the following:

1. *Top-down approach*: A top-down design approach starts by identifying the major components of the system, decomposing them into their lower-level components and iterating until the desired level of detail is achieved.
2. *Coupling*: Coupling between modules is the strength of interconnections between modules or a measure of the degree of interdependence between two modules. Classification of different types of coupling will help to estimate the degree between modules. The classification starts from *low to high*:
   a. *Data Coupling*: Two modules are said to be data coupled, if communication of modules is through a parameter.
   b. *Stamp Coupling*: Two modules are said to be stamp coupled provided, if communication of modules is through composite parameters.
   c. *Control Coupling*: Two modules are said to be control coupled when one module controls the execution behavior of another module.
   d. *Common Coupling*: Two modules are said to be common coupled, if they share data through some global data items.
   e. *Content Coupling*: Two modules are said to be content coupled provided one module refers to a piece of information defined in other module.
3. *Cohesion*: Cohesion of a module represents how the internal elements of the module are tightly bound to one another. Cohesion of a module gives the designer an idea about the different functions in it and how they belong together in the same module.

The different classes of cohesion that a module may possess from *high to low* are:

   a. *Functional Cohesion*: It is the highest. In this, all the elements of the module contribute to achieve a single function.
   b. *Sequential Cohesion*: When the elements are together in a module, the output of one element forms the input to another.
   c. *Communication Cohesion*: In this, the elements are together and they operate on the same input or output data.
   d. *Procedural Cohesion*: In this, a module contains number of functions in which certain sequences have to be carried out for achieving an objective.
   e. *Temporal Cohesion*: In this, elements of the module are executed in the same time span.
   f. *Logical Cohesion*: It occurs if all the elements of a module have some logical relationship between them and perform similar operations.
   g. *Coincidental Cohesion*: It is the lowest. A module is said to be coincidental cohesive, if it performs set of tasks that relate to each other very loosely and functions put in are out of pure coincidence without any design.
4. *Span of Control*: Number of subordinate modules under given modules.
5. *Size*: Indicates the overall code size.
6. *Sharability of modules:* Identify the commonalities with the program.

A module with high cohesion and low coupling is said to be functionally independent of other modules i.e., cohesive module performs a single task or function and has minimal interaction with other modules. Functional independence is a sign to a good design as it reduces error propagation; reuse of a module becomes possible; and the complexity of the design is reduced.

II. Proposed Design Principles

a) Top-down Approach

The top-down approach starts from the higher levels and decompose downwards to lower levels, identifying connections/collaborations at every stage. The top-down approach (also called stepwise design) starts from high level design description and break it down into different sub design or systems to gain observation into its composed sub systems. This gives good understanding of the problem. This starts with system specifications. It specifies/defines a module to implement the specifications. It specifies subordinate modules and then treats each specified module as the problem. Top-down design methods result in some form of elaboration where we reach to a level when no more refinement is needed and the design can be implemented directly. The top-down approach published by many researchers is found to be extremely useful for design. Most design methodologies are based on the top-down approach.

b) Coupling

Coupling is a measure of the relationship (i.e., dependency) between two modules. Coupling measures the degree to which each program module depends on each one of the other modules [3, 7]. Coupling is a measure of interconnection among modules in a program structure. Coupling captures the notion of dependence. Coupling tries to capture how strongly modules are interconnected. Coupling depends on type of information flow.

If two modules interchange large amounts of data, then they are highly independent. The degree of coupling depends on their interface complexity. The interface complexity is determined by number of types of parameters that are interchanged while invoking the functions of the module. Low coupling is often a note of good design as it supports goals of high readability and maintainability.

Data Coupling : Data coupling occurs between two modules when data are passed by parameters using a simple argument list and every item in the list is used. An example is an elementary data item (which should be problem related) passed as parameter between two modules. Example can be an integer, a character, a string etc.

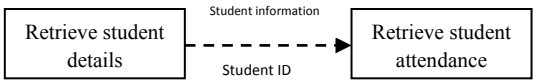

Fig. 2 : illustrates the module that retrieves student information using student id

Strengths of data coupling are: a module sees only the data elements it requires. *Weakness* of data coupling is, a module can be difficult to maintain if many data elements are passed.

c) Cohesion

Cohesion considers maximizing relationship between elements of same module. Cohesion is the measure of functional strength of a module [4, 7]. High cohesion is a mark for associating desirable features of software including robustness, reliability, reusability and understandability.

Functional Cohesion : Functional cohesion is the strongest cohesion. In a functionally bound module, all the elements of the module are related to performing a single function. By function, we mean modules accomplishing a single goal. A functionally cohesive module performs one and only one problem related task. Functionally cohesive modules may be simple and perform one task, such as Read Customer Record.

For example, a module containing all the functions required to manage employees' pay-roll exhibits functional cohesion. When a module exhibits functional cohesion, then we could be able to describe it using a single sentence.

Strengths of functional cohesion are functionally cohesive modules are good candidates for re-use, systems built with functionally cohesive modules are easily understood and, therefore, easier to maintain. *Weakness* of functional cohesion is designers should guard against designing over-simplified modules or methods. If functional cohesion is taken too far in structured design, the system design consists of hundreds of modules comprised of two or three lines of code.

d) Span of Control

Span of Control is a measure of the number of modules directly controlled by a higher-level routine. It is the number of sub-modules under a module. The number of subordinate modules for a project can be in 3 or 5 modules or levels.

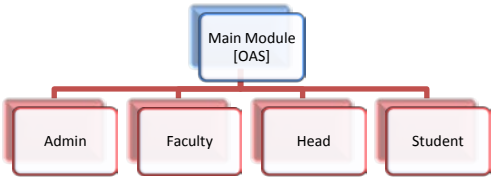

Fig. 3 : illustrates the span of control for OAS main module as 4

e) Size

The size indicates the overall code size. Example, 50 lines of code.

f) Sharability of modules

Identify the commonalities with the program and identify reuse components before development.

III. Case Study: Online Attendance System

Online Attendance System is software developed for daily student attendance in colleges and institutes. It facilitates to access the attendance information of a particular student in a particular class. The attendance information is sorted by the system, which will be provided by the faculty for a particular class. This system will also help in evaluating attendance eligibility criteria of a student [8]. This helps faculty marks student's class attendance easily and quickly.

The **purpose** of developing online attendance system is to produce a computerized solution to manual attendance procedure as manual process is time consuming & mostly not effective and another purpose is to generate the report automatically at the end of the session, and also at end of the academic.

This attendance system lets faculty and administration do the following easily:

- Prints class attendance sheets when needed.
- Faculty checks student attendance instantly.

- The Head of the department or institution can check the attendance monthly, date-wise, at end of the academic and also check the summarized attendance of particular student when required.

The **scope** of the project is this system is intended for engineering institutions which is a web application. In this system there are mainly four entities; admin, faculty, HOD and student. The admin is the main secretary of the system who enters the data into the system.

a) Admin Module

The first entity is Admin who is the powerful in the system. The admin has the possibility to add new students, new faculty, new subjects and new courses; edit and delete the existing ones. The admin can update details of multiple students where students can be promoted to next class or semester easily.

The admin can allocate subjects to faculty for a particular class, so that the attendance registers are created dynamically. If allocation for a subject is updated with new faculty, the created registers are transferred to that particular faculty. The admin can view the allocations and registers as and when required.

For faculties to take attendance of an academic, the admin has the possibility to set attendance start date and end date. For new academic, the existing dates are deleted and added again.

To send the attendance report of the student to their parents, the admin gets the absentees' details of all the classes on the given date and details of the students whose attendance is less than the required percentage for the given month.

b) Faculty Module

The second entity is Faculty who plays an important role in the system. The Faculty takes the attendance of the students in this module.

The Faculty when selects the date, period and the register, navigates to the selected register where the attendance is taken to the students and saves the details which cannot be updated. When selecting the date, the faculty can give attendance from the mentioned start date and end date added by the admin.

To view the register of the subjects handled by the Faculty, the faculty selects the register, and views the attendance details of all the students till date. The Faculty can also view the overall number of classes, total attended classes and the percentage.

c) HOD Module

The HOD module also plays a major role in the system. This module is exceptionally used to view the attendance reports of the student. The user of this module can be Head of the Department or Head of the Institution.

The Reports include:

1) *Subject-Wise Report*, where faculty is selected, and obtains the list of registers of the particular faculty. This is similar to the register view of the faculty module.
2) *Student-Wise Report,* where the student ID is given to get the cumulative attendance report of the student for all subjects where report contains total classes, attended classes and the percentage of the attendance.
3) *Date-Wise Report,* where particular class and date are selected to view the attendance of the given date for all periods.
4) *Monthly Report,* where particular class and month are selected to view the attendance report in a cumulative format for all subjects with total classes, attended classes and percentage.
5) *Semester-Wise Report*, where particular class is selected to view the overall attendance report of the semester or academic up to the attendance end date. This report will be generated only after the attendance end date. This report is also similar to the cumulative format of monthly report.

d) Student Module

The final entity is the Student where he/she can get details that include the profile, and attendance report. The attendance details contain the total classes, attended classes. To get the percentage of the attendance to the classes attended, it is generated only after the attendance end date. This is also similar to Student-Wise report in the HOD module.

The four entities or modules mentioned above can access the features given to them in the system, where they have to login with their own username and password.

The Online Attendance System applies the proposed design principles, where the operations performed by each entity satisfy the characteristics of design low coupling and high cohesion which provides an efficient system for ease of usage. The top-down approach when used illustrated the software engineering principle decomposition where the system is broken into different sub-modules so that the module required to start is easily identified and implemented from that level. When applying system design principles to the application, application quality has improved and is operated at high level of efficiency and the requirements of the system specified are satisfied.

IV. Screen Shots

Fig. 4 : Online Attendance System Home Screen

Fig. 5 : Faculty Module - Opening Attendance Register

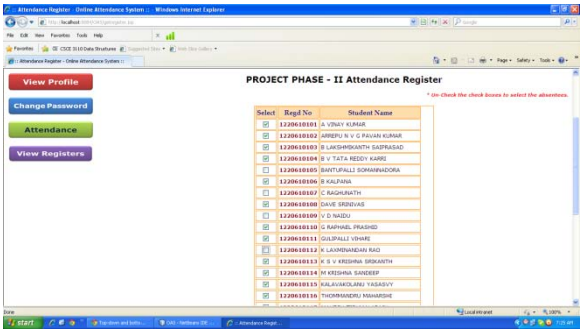

Fig. 6 : Faculty Module - Taking student Attendance for the selected Register

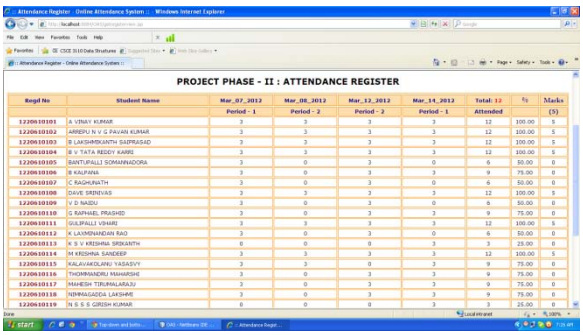

Fig. 7 : Attendance Register View

Fig. 8 : HOD Module – Selection Interface for Monthly Report

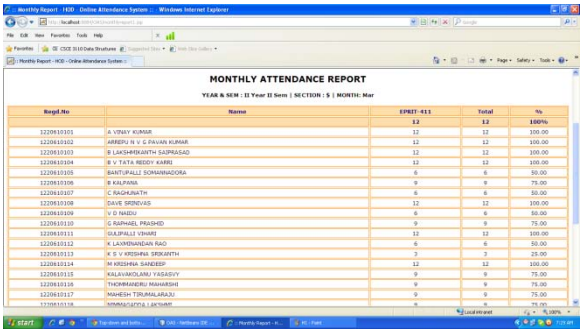

Fig. 9 : Cumulative Report for selected Month with all subjects

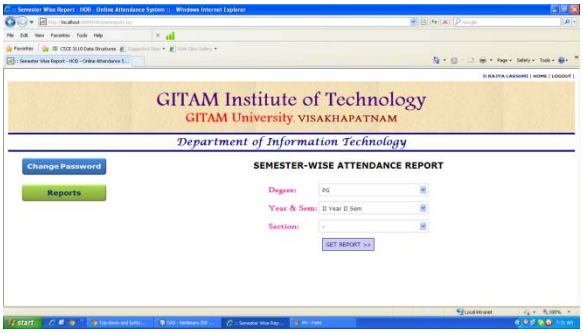

Fig. 10 : HOD Module – Selection Interface for Semester-Wise Attendance Report

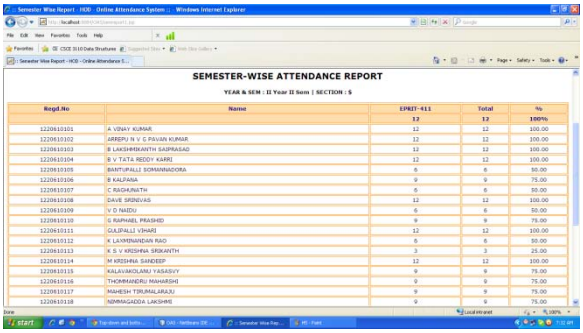

Fig. 11 : Cumulative Semester Wise Report with all subjects

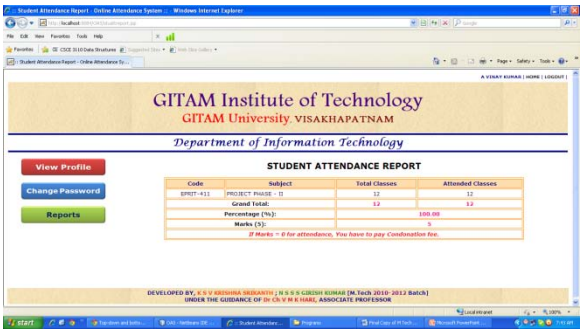

Fig. 12 : Attendance Report Viewed by Student

V. Conclusion

This paper mainly elaborates basic principles of System Design and enumerates reusability is best practice for deliver product facility. The Online Attendance System that is developed meets the design objectives of the system design for which it has been developed. The users associated with the system understand its advantage and easily navigates with the user interface. It was intended to solve as requirement specification. The current system can be a good reference when implementing a similar system in other institutions as the system is proved to be workable and effective.

References Références Referencias

1. Software Engineering: A Practitioner's Approach – Roger S Pressman
2. An Integrated Approach for Software Engineering – Pankaj Jalote
3. Data Coupling Design Principle: http://it. toolbox. com/blogs/enterprise-solutions/ design-principles-coupling-data-and-otherwise-160 61
4. Functional Cohesion Design Principle: http:// it. toolbox.com/blogs/enterprise-solutions/ design-principles-cohesion-16069
5. Nabil.M.A.M and A.Govardhan, A Comparison Between Five Models of Software Engineering, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, pp: 94-102,September 2010.
6. Sanjana Taya and Shaveta Gupta, Comparative Analysis of Software Development Life Cycle Models, IJCST Vol. 2, Issue 4, pp:536-539, Oct . - Dec. 2011.
7. Imran Baig, Measuring Cohesion and Coupling of Object-Oriented Systems, Master Thesis Software Engineering, Thesis no: MSE-2004:29, Month: August Year: 2004
8. H.C. Ting and T.O. Ting, An Online Attendance Record System, ICEED 2009, Month: December Year: 2009