2012 International Conference on Software and Computer Applications (ICSCA 2012)
IPCSIT vol. 41 (2012) © (2012) IACSIT Press, Singapore

New Modular Software Development Principles, a Decentralized Approach

Mirali Sharifi Bonab [1], GholamAli Nejad HajAli Irani [2 +]

[1] Islamic Azad University, Bonab Branch, Bonab, Iran
[2] University of Bonab, Bonab, Iran

Abstract. In this paper some principles have been provided to increase modularity of software systems and help to turn an existing system to a modular system. These principles distribute functionalities of each module to them and decrease modules dependency to each other.
To obtain this aim, object oriented principles and heuristics has been analyzed then by considering a module as an object and new modular principles have been provided. In the reminder to evaluate new principles, a new modular architecture has been provided. The strength of new principles has been shown with a complete case study.

Keywords: Modular Software Architecture, Quality Attributes, Object Oriented Analysis and Design.

1. Introduction

In Centralized approach, the most common tasks of system have been extracted and assigned to the common part of system as named Core. Architectures consist of several components [3]. The main purpose of centralized approaches is to reduce the complexity of developing components. So components are developed quickly and many quality attributes of system is improved. In centralized approaches other components are highly depend on Core and if we change the Core, other components has changed as well. So extensibility, modifiability and flexibility of software are increased.

Modularity as an object oriented principle helps to have extensible, modifiable and flexible software. To reach perfect modular software we must have maximum amount of cohesion and minimum amount of coupling [14]. Therefore independent parts of software are excellent.

Based on [14], the biggest problem in modular development is decomposability problem, means that we can't always decompose each system to parts that are independent from each other [14]. So the big trade-off has been made between being modular and modular decomposition.

Most software has used centralized approaches. As increasing the scale and complexity of system, scale and complexity of Core is increasing as well. Then management of Core is turned to a big problem. On the other hand, while extending and modifying the Core, all modules might change. Therefore as increasing scale of Core, the Core can turn a GOD module [1], so we can't reach to modularity and written modules for a system can't be used in other systems.

For example, more than 1200 web portal and CMS is presented in web applications [2]. The CMS is composed of several modules [4]. For example, Drupal is composed of more than 8700 modules [5]. So, each module has been written thousands times in general. These problems are due to lack of modularity.

[+] Corresponding author. Tel.: + 98-914-322-5187.
E-mail address: Irani.GholamAli@Gmail.com.

In this paper, to reach a highest modularity between modules, a new modular principles based on decentralized approach have been provided and use object-oriented principles and heuristics, and distributes the Core complexity between other modules.

2. Problems of centralized approach

Being centralized in comparison with decentralized and modularized can cause numerous problems in development software which categorized as fallowing:

Req1: Modules have to use Core Functionalities (CF) – the common functionalities that have been centralized and implemented in Core. So, modules cohesion is increasing and dependency on Core is increasing as well, therefore modularity of system will be decreased.

Req2: Developing small-scale modules need to follow the CF. Consequently; complexity of developing small-scale modules will be increased.

Req3: The implemented CF is not complete in general. Probably, developing large-scale modules is needed to use a new CF which is not supported by the Core.

Req4: Due to centralized approach and dependency of modules on Core, performing a Unit Test on modules is difficult and quality of testability is decreasing.

Req5: Due to variety of CF types and patterns, considering all of them in the Core cause to complexity of Core.

Req6: Because of centralized approach, integration of implemented modules into different systems with different Cores takes some efforts due to lack of standard interface. Therefore system integrity and modules portability decrease.

Req7: In centralized approach, the overall CP (so-called Big Picture) is apparent to all modules. So, encapsulation of CP is violated.

Table 1 presents the relationship obtained from the categorizing of above-mentioned requirements with software architecture quality attributes. In previous studies which are described in the reminder, none of above-mentioned problems are considered.

Table 1. Quality attributes affected by requirement list.

	E	**M**	**Mo**	**I**	**Io**	**P**	**T**	**LEGEND**
Req1			x					E: Extendibility;
Req2	x		x					M: Modifiability;
Req3	x	x						Mo: Modularity;
Req4							x	I: Integrity;
Req5	x				x			Io: Interoperability;
Req6			x	x		x		P: Portability;
Req7			x		x			T: Testability.

3. New Modular Development Principles

Based on table 1, theses problems are related to Modularity and in detail related to interoperability, portability, testability, extendibility and modifiability. In this section, to solving these problems, we used Object Oriented principles and heuristics in [1].

Based on [1], we can consider a module as an object then we can apply object oriented principles to obtain new modular development principles. List of used Object Oriented Heuristics from [1] are: Heuristic 2.1, 2.2, 2.4, 2.5, 2.6, 2.9, 2.10, 3.1, 3.2, 3.7, 3.8, 4.1, 4.2, 4.3, 4.4 and 5.3. By analyzing concepts of above-mentioned principles and heuristics and their relation with modularity concepts, we can provide new principles for developing modular systems which is shown in table 2.

Table 2. Mapping Object Oriented heuristics to Modular principles.

Heuristics	Code	Provided Principle
H2.1, H5.3	M1	Each module should hold and manage its own data by itself and Modules can't access each other data.
H2.9, H2.10	M2	Each module should perform all its functionalities by itself and Modules can't access each other functionalities.
H2.2, H2.3, H2.4, H2.5, H2.6, H4.1, H4.2, H4.3	M3	Minimize module relationships and interface. Independent modules are excellent.
H3.1, H3.2, H3.7, H3.8	M4	Beware of creation of God module [1]. (Minimize the Core).

4. Evaluating provided principles

In this section, steps of developing modules based on provided principles are presented as following:

Step 1: Considering that the modular principles presented in table 2 is very general, we must derive detailed modular principles for desired module based on modular principles in table 2.

Step 2: To examine previous methods and approaches in modular systems for the module.

Step 3: To analyze and design of the module based on modular principles and decentralized approach. Use Case Model is necessary, because event list in following steps obtained from Use Case list.

Step 4: Considering M3, we must provide optimized event list of the module based on Use Case list. Event list must cover all use cases of module.

Step 5: To standardize DTD and XML Schema for events of the module.

5. Case Study: Modular Authentication and Authorization

In this section steps of developing modular authentication and authorization (AA) has been provided. In step 1, detailed module development principles for AA has provided in table 3.

Table 3. Derived principles for developing AA.

Base Principle	Obtained Principle
M4, M2	Each module has to perform its authorization by itself.
M4, M1	Each module has to hold and manage its authorization data.
M3	Standardize an AA interface between Core and Modules.

In step 2, we must collect and examine previous methods for AA.

In step 3, we must perform analysis and design step for AA based on derived principles in table 3. Distilled use case model is shown in figure 1.

In step 4, we must provide optimized event list of AA based on its Use Case list. Optimized event list for AA is shown in table 4.

Table 4. Optimized event list for AA.

Use cases	Events	Description
New User, Sign Up	NewUser(String Username):void;	As soon as a new user registers in system, Core should inform all the modules, so the modules can grant default permissions to he/she.
New ModuleAdmin	NewModuleAdmin(String Username):void;	As soon as a new ModuleAdmin registers in system, Core should inform the target module.
Change User Overall Status	ChangeUserStatus(String Username, UserStatus status):void;	When Core changes overall status of a user, should inform all the modules.
Login	getAccessList(String Username):String;	For creating Control Panel for a user.
Other Use Cases	---	For other cases modules act independently.

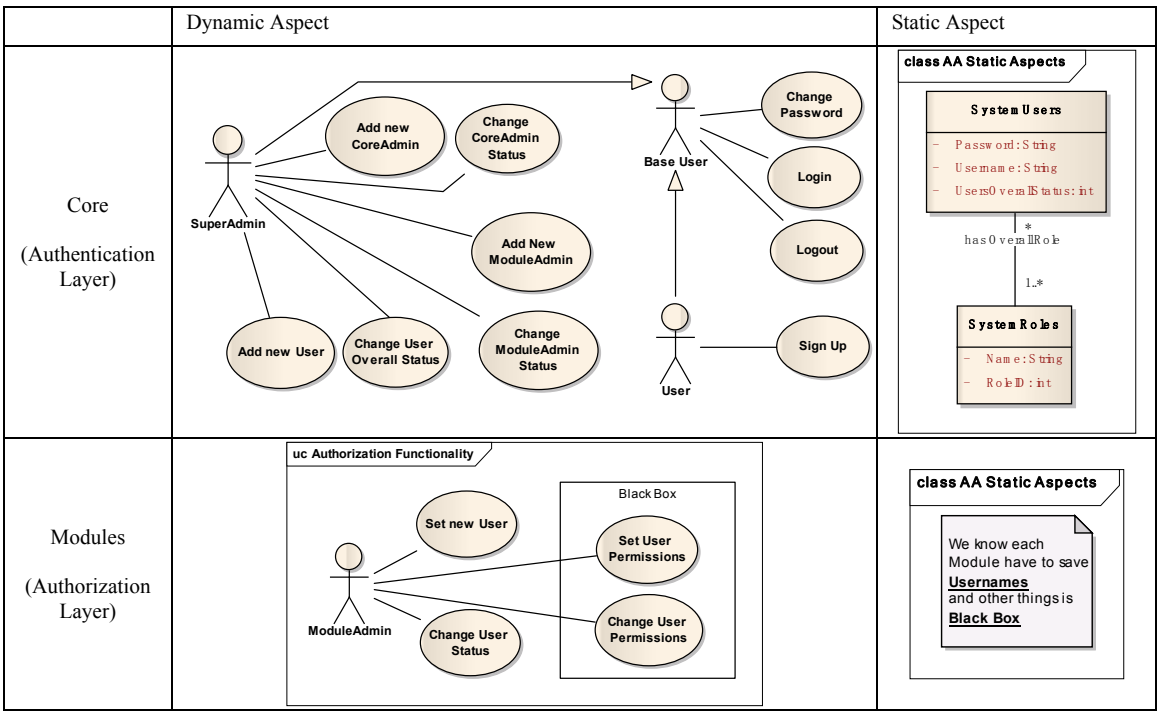

Figure 1. Distilled AA analysis and design.

In step 5, we must standardize DTD and XML Schema for events of the module. Regard as this step outputs is large, this part postpone as future works.

6. Evaluation

In section 2, we categorized a requirement list as problems of centralized approaches. Decentralized principles capture all of these requirements which are shown in table 1. In fact, decentralized principles improve all quality attributes which are mentioned in table 8.

Table 8. Requirement list is captured by new principles.

Description	Captured Requirements
Modules are independent in selecting their own patterns. They just have to consider Core's standard interface.	Req1, Req2, Req3
Since modules are not dependent to Core, unit test of each module can perform easily far from the Core.	Req4
Modules are free to choose their patterns and we don't need to collect all patterns in Core.	Req5
As establishing a new standard interface for Core and modules communication, integrity and portability of modules was increased and modularity of each module was increased to higher degree.	Req6, Req7

7. Conclusion

In this paper, new decentralized principles for modular systems have been provided. These principles distribute Core functionalities between modules based on robust object oriented thinking and dependency between modules decrease saliently and turn existing systems to more modular systems. So module development will take extra effort than before. Although it could be a disadvantage in comparison with centralized systems, this extra effort is worth benefiting of being decentralized.

8. References

[1] A. J. Riel, Object-Oriented Design Heuristics, Addison Wesley, 1996.

[2] The Content Management Comparision Tool, available at http://www.cmsmatrix.org

[3] Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471-2000, IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee, 2000

[4] B. Boiko, Content Management Bible, 2nd Edition, Wiley Publishing, Inc., Indianapolis, Indiana, 2005

[5] Drupal, Open Source CMS, available at http://Drupal.org/Project/Modules

[6] B. Meyer, Object Oriented Software Construction, Second Edition, Prentice Hall, 1997