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

MEDA-Principles: Towards New Principles on Software Modularity

GholamAli Nejad HajAli Irani [1 +]

[1] University of Bonab, Bonab, Iran

Abstract. Modularity is a critical issue in large-scale software systems. To reach maximum quality of modularity, some principles and guidelines have been provided such as object oriented principles and heuristics.
Everything can be an object. So we can consider a module as an object and apply object oriented heuristics on modules. In this paper, five modular principles as MEDA-Principles have been provided by analysis of object oriented principles and different aspect of modularity.
To evaluate MEDA-Principles, some disadvantages of existing approaches have been investigated and categorized. Finally, a new architecture based on MEDA-Principles has suggested and show that MEDA-Principles can capture disadvantages of existing approaches.

Keywords: Software Architecture, Software modularity, Object Oriented Principles

1. Introduction

Software systems are being larger. As the increasing the scale and complexity of software systems, some variety of approaches have been provided to overcome their complexity.

For example, web portals and Content Management Systems (CMS) as web based information systems are complex systems. More than 1200 web portal and CMS are developed as until [2]. In [2], full comparisons of web portals and CMS can be found. Considering that each CMS is composed of many modules [3], so thousands of similar modules have been developed in the field of Web Application and various existing CMS cannot use the modules of together and a module for a CMS only used in the CMS.

Unlike information systems, most other fields of computer systems are in high degree of portability modularity. For example when you want to buy a pc computer you have some choices in their pieces such as RAM or Mother Board and etc. Even assembling the pc system, you can change most pieces of pc computers. But information systems have not reached to high level of portability and modularity like pc computers.

Another example from the field of information systems, hundreds of Accounting Systems have written [4] that none of them have used other modules. A customer with purchasing and using an Accounting Systems can't change each module of that with another Accounting Systems. In the field of CMS a written module for a CMS can't be use in other CMS. Therefore a public module like News Module in CMS field is written more than 1200 times. So, this is because of lack of standard interfaces between developers of CMS. Towards to reach a standard framework between developers of information systems and increasing the modularity of systems, MEDA-Process has been provided as following:

1. To investigate and categorize problems of existing approaches and architectures
2. MEDA-Principles: To analyze and investigate different aspects of modularity and trying to provide new and more detailed software development principles.
3. To provide base architecture that we want to turn it to architectural style.
4. To provide some rules and guidelines to develop different part of architecture and their modules

[+] Corresponding author. Tel.: + 989143225187;
E-mail address: irani@bonabu.ac.ir

2. Problems of existing approaches

All existing architectures used a centralized approach in there architectures. In centralized approach, common parts of code have grouped and placed in a part with named Core. Considering that there are lots of solutions and patterns for each known problem in computer science, so with gathering common codes in Core, Core turn into a GOD-Class [1]. This means that, with increasing the scale of system, control of Core turn to a big problem as well. In the reminder the problems of existing methods have been investigated.

Req1.1: Simple and small scale modules have to obey existing patterns of Core. So time to market of each small scale modules may increase. Therefore dependency of modules to Core increase and the quality of Portability, Integrability and Reusability of modules will decrease.

Req1.2: Certainly patterns are placed in the Core is not complete in general and may not be suitable for the development of large scale modules. So we may be forced to develop new pattern that is not into Core or we may change our module policy and structures. So Portability, Integrability and Reusability of modules will increase similar Req1.1.

Req1.3: Modules are not autonomous in the choice of their patterns and forced to use the Core's patterns. So to use a written module in the other system, we must change all connecting protocols. Therefore Portability and Integrability of modules are decrease. In order to Req1.1, Req1.2 and Req1.3, written modules for a Core are deeply depending on Core. So each change in Core may change all modules.

Req2: Approaches, patterns or management mechanisms of Core may vary with the modules. So modules and Core no need to aware mechanisms of them (so called Big Picture of them). Therefore if Core and modules aware structures and patterns of each other, security of system may affect and decrease.

Req3: Due to collecting common codes in Core and dependency of modules on Core, performing a Unit Test on modules is difficult and quality of testability is decreasing.

Req4: Due to collecting common codes in Core and dependency of modules on Core, changing the Core my affect all modules of system.

Disadvantages of existing architectures are shown in table 1.

Table 1. Quality attributes affected by requirement list.

	M	T	I	P	R	S	LEGEND
Req1.1			x	x	x		M: Modifiability;
Req1.2			x	x	x		T: Testability;
Req1.3			x	x			I: Integrity;
Req2						x	P: Portability;
Req3		x					R: Reusability;
Req4	x						S: Security.

3. MEDA-principles

As regarding the problems of table 1, we need some modular principles that help us to develop new architectures that can solve existing problems. In [5] five rules and five principles have developed for modular software. But the principles are more general. We need some detailed principles.

Based on object oriented principles and heuristics and five modular principles in [5], Four strategies can be defined to achieve the main goal that is reach to highest modularity. These are as following:

1. To minimize the Core: with decreasing the scale of Core, dependencies of modules to Core are increasing as well. Then changes and extensions in Core may have minimum affects on modules. With this strategy the complexity of Core can distribute between modules.
2. To minimize the relationships between modules: to increasing the independency of modules, structures and architectures must be suggested that minimize the relationships and direct access between modules. So modularity of modules increase.
3. To standardize the externally visible of all modules and Core for specific Information System.
4. Just use dynamic structures to have a highest extensibility and modifiability.

3.1. Decentralize Principle

There are several object oriented heuristics for the management of GOD-Classes and other object oriented problems [1]. But there are about classes and objects. So we can consider a module as an object. Then we can use object oriented heuristics to overcome the complexity of GOD-Modules in modular software. In object oriented thinking everything can be an object [6]. So, we can consider a module as an object. Then by the analysis of concepts of encapsulation principle [6] and object oriented heuristics [1], we can say that each module must do all it functionality by itself and each module must hold and manage all its data by itself. This means that collecting common parts of system in the Core is not appropriate. For example managing all data access of system with one data access layer [7] is not good idea, because all of these approaches are type of GOD-Class [1]. Therefore principle 1 of MEDA is as given below:

P1: All functionalities of each module have to do with and within it.

For example in modular systems each module must return its search result by itself. Core just wants each module to search a title and each module have to gather the result and passed it to Core, finally Core show the results. For another example, based on this principle we can say that Authorization of each module is module's functionality and must be assigned to modules.

3.2. Pure Modular Principles

To have a system with maximize quality of modularity, modules must be independent from each other. The human body architecture is an excellent example. Each modules of the human body are independent and all modules related to the blood vessels and neural vessels. All relations in the human body are based on some standard Hormones. Therefore we can use that architecture to have maximum quality of modularity. Then we can eliminate all direct relationships in software and all modules only depend on a communication channel like vessels and standards like hormones.

In the other hand, by analysis of H2.2, H2.3, H2.4, H2.5, H4.3 and H4.4 from [1], to have a robust object oriented system, all relations and standard interfaces should be increased. At best, there should not be relations between modules. Based on the concepts described, in order to have pure modular systems two following principles have provided:

P2: There should be no direct relation between the modules.

P3: Modules are not aware of any other modules existence, functionalities and data.

3.3. Standard Interface Principle

Based on [5] and object oriented principles [1] it is so clear that the basic condition of modularity is about using standard interfaces in all relationships. All modules must aware from these standards like the genetic map in the all human body cells. So forth principle of MEDA is provided as following:

P4: All externally visible parts of modules have to use standard interfaces.

Based on P4, all relationships in all parts of system have to use standards interfaces and based on P2 there should be no direct relations and awareness between modules. So, all modules must be in relationship with Core.

3.4. Dynamic Principle

Extensibility and modifiability are most important elements in developing of software systems. So, if we want to have extensible and modifiable software, we must have dynamic architecture. In software engineering, aspects of dynamic architecture have not been investigated widely. Based on P1, P2, P3 and P4, to have a dynamic architecture, for example we can use an instance of event-driven architecture [8]. In event-driven architecture, events can be everything [8]. For example an event can be a method or object. Events structure should be dynamic and standard. For example, we can use XML [9] format for each event. Therefore to support extensibility and modifiability we have to use dynamic protocols.

P5: Dynamic structures should be used for whole communications.

All MEDA-Principles show in table 2.

Table 2: MEDA-Principles.

Main Goal	Strategies	MEDA-Principle
To reach maximum modularity	Decentralization (Minimize the Core)	P1: All functionalities of each module have to do with and within it.
	Pure Modularity (Minimize the Relationships)	P2: There should be no direct relation between the modules. P3: Modules are not aware of any other modules existence, functionalities and data.
	Just Standard Interfaces	P4: All externally visible parts of modules have to use standard interfaces.
	Dynamic Structures	P5: Dynamic structures should be used for whole communications.

4. Suggested Architecture

In this section, a new suggested architecture based on MEDA-Principles has been provided which is shown in Fig 1. Based on P1, Core must be minimized and just a structure for other modules. Core just manages modules and standards and base runtime functionalities and other functionalities distribute between modules.

In suggested architecture, there are two categories of modules. First category is Internal-Modules that are inside the Core and include Base-Modules and Collectivity-Modules. Second category is System-Modules that are outside the Core. Base-Modules control basic functionalities of system. This part is constant for all architectures and never changes. Collectivity-Modules perform functionalities that related to all modules. For example Search-Manager as Collectivity-Module, sent constructions to all System-Modules and after receives the results and analysis them, prepares them for display.

Based on P2 and P3, there is not any direct relationship between modules and all messages as events manage by Event-Bus. There is not any received list in any message and Event-List decides about that. In this architecture a combination of event driven architecture [8] and a type of Bus-Architecture [10] are used.

Based on P4 and P5, we have to use standard interfaces that have saved in Interface-Repository and a copy of that should exist in all modules.

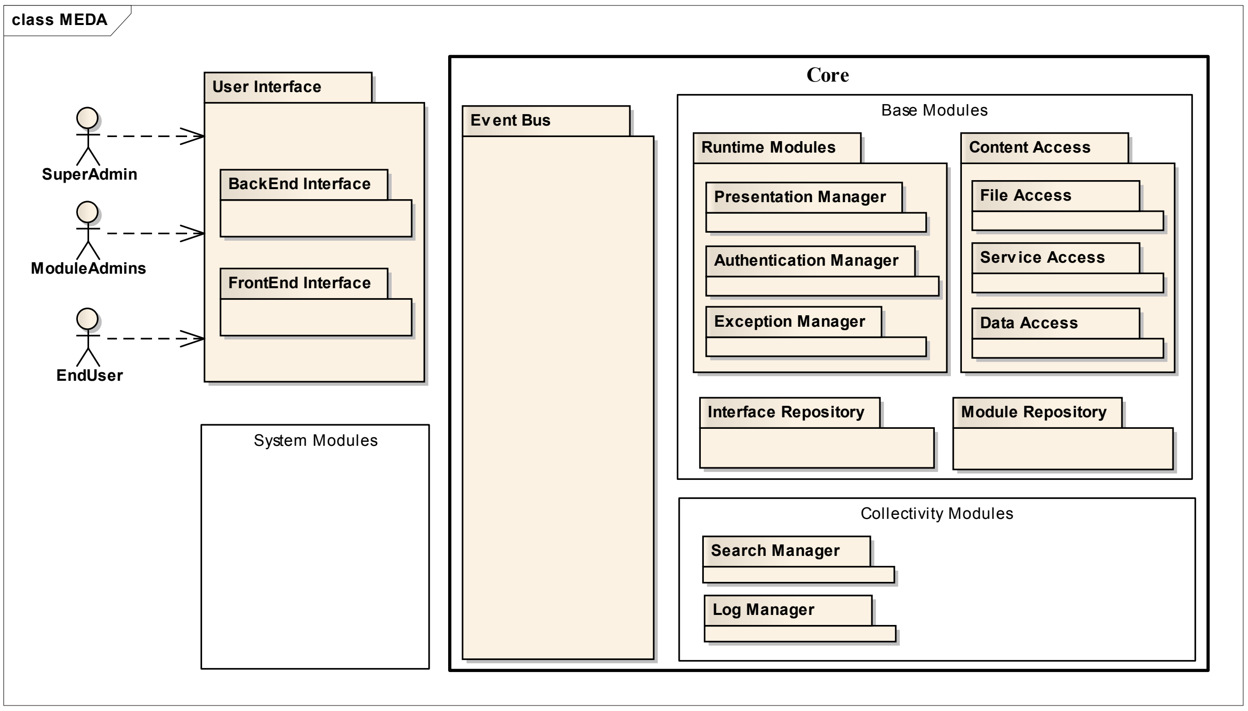

Fig. 1: A suggested Architecture based on MEDA-Principles.

5. Evaluation

In section 2, we categorized a requirement list as problems of existing approaches which are shown in table 1. By the use of MEDA-Principles all of these requirements capture. Totally, 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.

Table 2. Requirement list is captured by MEDA-Principles.

Description	**Captured Requirements**
Modules are independent in selecting their own patterns. They just have to consider Core's standard interface.	Req1.1, Req1.2, Req1.3
Each module can access just its contents and modules encapsulation and security are increased.	Req2
Since modules are not dependent to Core, unit test of each module can perform easily far from the Core.	Req3
Modules are free to choose their patterns and we don't need to collect all patterns in Core.	Req4

6. Conclusion and the future works

In order to MEDA-Principles, Core complexity distribute 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 and MEDA-Principles solved problems of existing architectures.

MEDA-Process can be use in other types of information systems such as Service Oriented Platforms and any large-scale modular software. For example in MEDA-Portal, MEDA-Principles can be applied on large-scale web based application or in MEDA-BPM; MEDA-Principles can be applied on Business Processes.

7. References

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

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

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

[4] Comparisions of Accounting Softwares, available at http://en.wikipedia.org/wiki/Comparison_of_accounting_software.

[5] B. Meyer, Object Oriented Software Construction, Second Edition, Prentice Hall International Series in Computer Science, 1994.

[6] G. Booch, R. A. Maksimchuk, M. W. Engel, B. J. Young, J. Conallen, K. A. Houston, Object-Oriented Analysis and Design with Applications (3rd Edition), Addison-Wesley Professional , 2007.

[7] C. Nock, Data Access Patterns: Database Interactions in Object-Oriented Applications, Addison Wesley, 2003.

[8] O. Etzion, P. Niblett, Event Processing in Action, Manning Publications, USA, 2011.

[9] R. Bourret, A. B. Coates, B. Harvey, G. K. Holman, M. Kay and et al, Advanced XML Applications from the Experts at The XML Guild, Thomson Learning Inc, 2007.

[10] D. Chappell, Enterprise Service Bus, O'Reilly Media, Inc, 2004.