MAGNT Research Report (ISSN. 1444-8939) Vol. 2(6): PP. 1272-1284

Analysis and Design of New Secure Dynamic Structure for Increasing Modularity of ERP Systems

Esmail Amini[1], Farhad Soleimanian Gharehchopogh[2]

[1]Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran

[2]Department of Computer Engineering, Hacettepe University, Beytepe, Ankara, Turkey

Abstract: *With the development of more complex interactions between an organizations and business processes, control and management of processes in organizations is becoming more difficult in a growing fashion and the use of computerized systems such as Enterprise Resource Planning (ERP) systems becomes more evident. This paper attempts to increase the dynamism of ERP systems and systems based on modular architecture and function by introducing a structure by the use of which all sectors and components on these systems can be accessed while the independent nature of each sector be preserved. This might be achieved only by the collection and preservation of information that exist in these sectors and making use of them in order to manage various parts and components. Therefore, the design of a modular registry system not only does the synchronization of modular systems with changes and modifications in business environments, but also enhances the dynamism of ERP systems. In addition, using manual access control in order to secure the privacy of modules will lead to increase in the security rate of systems based on modules and ERP.*

Keywords: *Software Design, ERP Systems, Modularity, Usability, Dynamic Structure, Information Security*

1. INTRODUCTION

Large organizations quickly understood the importance to adapt themselves to changing conditions and technologies and also the importance of building an integrated software to meet all their needs from the management of customer relationship to internal management within organizations. This led to companies to the use of ERP software in order to provide for internal and external needs. Anyway, these attempts will not be sufficient for providing improvement level needed for the competitions of companies in the developing and competitive world of future. In the future, the significant increase in the revenues of companies will not be achieved through the increase in efficiency of processes but by improving the transfer of knowledge from any place to the organization or the staff responsible with organizational processes. Companies have focused their attention and investment on systems that are comprehensive and include the whole organization instead of focusing on small and separate ones in order to meet their growing needs and parameters [1, 2]. In most software systems (especially ERPs), the dynamism capability of systems and the synchronization of various parts in it with business processes in organizations is of great importance. Since change in the level of activities in organizations and the complexity in organizational processes have faced a growing trend in recent decades, if software systems related with these activities do not have a consistent architecture and structure capable of synchronous development with organizational advances, in the short term the performance of these systems and their ability to meet the needs of an organization will be lost and probably will lead to the slowing of operations within the organizations. In this

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

paper, it has been tried to make use of techniques used in the design of ERP systems to introduce a structure by the use of which the dynamism and development capabilities in these systems can be increased to a significant degree [1, 2, and 3]. To achieve this task, you must first investigate the overall architecture of modular systems and the features of ERP systems. In order to design an optimal structure for Module Registry based on object-oriented principles to increase dynamism and security in ERP systems, important and key information of modules and their subsystems are needed. After that, the class and data structure required to meet the needs of an ERP system and maintain its security and dynamism in the long are studies. Finally, a solution and framework is presented to increase dynamism and security within these systems by analysis and optimization. In this paper, it has been tried to present a new structure for Module Registry in systems based on modules. If necessary, the modeling environment of Visual Studio will be used to design this structure.

2. Enterprise Resource Planning

The use of information systems in an organization that cover all activities and tasks and provide timely and necessary information to its users is essential in modern organizations. Without being equipped with these systems, the increase of the capabilities of organizations, improved performance, better decisions, and the achievement of competitive advantages is impossible. ERP is one of the latest management tools that is capable of collecting information in the organization by the use of information technology in all areas of the organization in an integrated way and provide the results and information to users in different levels of an organization. ERP is a thought and theory that companies can and should operationalize in their system by experts in order to integrate different subsystems with each other [2].

Corporate systems are commercial software packages that enable the integration of business processes and business information exchange across organizations. In fact, the data can be integrated by the software packages. Furthermore, business processes in companies can be controlled online and in a complete way. Enterprise systems include ERP software and the related software packages that are capable of advanced planning and scheduling, the automation of sales system, customer communication management, product formation, etc. one of the major characteristics of enterprise systems is their integrity (like the limitless integrity in the information related to accounting and financial systems, human resources systems, chain supply information, and customer information). The enterprise accounting systems are similar functional systems that perform various operations within different levels of an organization. These systems can be considered among the latest information technology in the past decade that, which is rapidly changing and evolving [3].

The discrete and insular automation of business processes not only would not lead to a uniformity, consistency, and compatibility in a serial data stream in a value-added chain, but also will destroy the capability of the organization to grow. Since the business environment in the current information age has created intense competition in obtaining information and knowledge organization, systems within organizations have to work together and cooperate, rather than being separate. The creation of such a mechanism is not possible unless with the help of an integrated system for all units, which controls everything from finance and administrative sections to production and warehouse. ERP is one of the latest management tools that is capable, of collecting the existing data in organizations by the use of IT techniques in an integrated way in all areas of the organization and providing the

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

results to users at various levels of the organization [2, 3].

Agent-oriented architecture to provide flexibility of ERP systems is a research topic and the results of that can help us achieve the goals in paper. Software architecture is an important part of an ERP software package that can help us in ensuring the flexibility of the system and perform the process of reverse engineering. This part is the development environment in which the possibility of ERP software package to specialize or develop in order to meet the specific needs of the users are provided. On the other hand, one of the main distinctions of ERP system compared to other integrated systems is their being equipped with business processes based on the best experiences, which of course can be specialized based on the needs of customer organizations. One of the major restrictions in current development environments is the lack of flexibility and intelligence to support the creation and modification of processes at the time of performing them. These restrictions can be overcome in an agent-oriented architectural design [4].

The development processes of ERP systems and administrative barriers at the macro level are another study that could help to the process of this paper. The objectives of this study include the investigation of development processes of ERP systems in organizations and institutions and the removal of obstacles that lead to the failure of implementation in these systems. The methodology used in this study is correlation and analytical in nature. In addition, it is a field study type and the data has been collected by the use of a questionnaire and interview. This study tries to answer questions such as, "How ERP can improve business performance in a company?", "What are the factors cause the failures often in ERP projects?", and "To what extent the development of software packages related to these systems been improved in software companies?" The results can be used in designing a more appropriate architecture [5]. The implementation principles of Enterprise Resource Planning are among the most important issues that need to be considered in the design of ERPs. The majority of ministries and large organizations or companies have frequently spent large amounts of money to implement a proper integrated information system in their operational environment and the results are just limited to reports on study phases and knowing the system or just an improper implementation.

Many problems and reasons exist in organizational processes and characteristics of employers and contractors that cause this failure and should be evaluated in a proper place and time. One of the appropriate solutions to overcome these problems and causes can be found in the use of integrated ERP software products that are based on standardized processes and can largely overcome many problems existing in organizational systems after being implemented. Their objective is to scan documents related to companies and experts in the development and implementation of ERP and the manner of proper selection and implementation for ERP products. The results show that however the use of ERP products is costly and expensive, if experienced people help in the proper selection and implementation of ERP products, the change management be able to establish the culture for the use of that in its organization, and the organization contains a secure and stable network, the facilities provided in ERP software can be applied in the proper planning of organizational resources. [5, 6].

In most software systems, especially ERP systems, dynamism of systems and their keeping pace with business processes in organizations is of particular importance. Since the rate of

3. Proposed Structure for Module Registry System

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

change in organizations and their activities has become faster and more complex, if software systems related to these activities do not have an integrated structure and architecture capable of keeping pace with organizational developments, they will lose their capability and efficiency to meet organizational needs in the short term and might lead to the slowing of organizational processes implemented in organizations. [12].

In this section, we will try to use existing techniques in the design of ERP system to implement a system that can be used to increase development and dynamism capabilities in these systems to a great extent so that ERP systems can keep pace with changes and modifications in organizations and update their information. In order to introduce and design an efficient structure based on object-oriented principles to increase dynamism and security of ERP systems, important and key information related to modules and subsystems is needed. Then, the class and data structures required to meet the needs of ERP systems and to guarantee the security and dynamism of these systems in the long-run have been investigated. Finally, by doing analysis and optimization we will towards finding a solution and framework that can be used to increase dynamism and security in these systems [12, 13].

3.1 Metadata Management of Modules in Module Registry system

The expansion in the levels of inter-organizational interactions and the complexity of business process have made control and management difficult in these organizations and the necessity to use computerized systems such as ERPs has become more evident. Nevertheless, what seems remarkable in ERP systems is their informational and structural dynamism that has become a vital issue in the survival of such systems due to continuous changes in the quantity and quality of business processes in organizations over time. In this section, we will try to increase the dynamism of ERP systems and systems based on modules by the introduction of an architecture and structure by the use of which, all parts and components in these systems can be accessible. Of course, the independent nature of each components will be preserved the and this would only be possible through the collection and preservation of information from these sections and making use of them in managing different parts of the system [11, 13].

Therefore, by designing a registry module we will attempt to synchronize modular systems with changes and modifications in business environments. Finally, it will be attempted to increase dynamism in ERP systems and preserve the privacy of modules by controlling access to the, and increasing security rates in systems based on module and ERPs. In order to design modular systems and ERPs, which are capable of being developed, the presence of a system capable of maintaining metadata needed for the proper performance of modules is necessary. Module registry is part of the architecture of ERP systems used to maintain metadata related to the modules present in ERP systems. Information will be registered in module registry in times of configuring a new subsystem and in later recalls and accesses to subsystems, these information will be used [10, 11, 13]. This module is made up of three basic classes, the structure of which is shown in Figure (1).

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

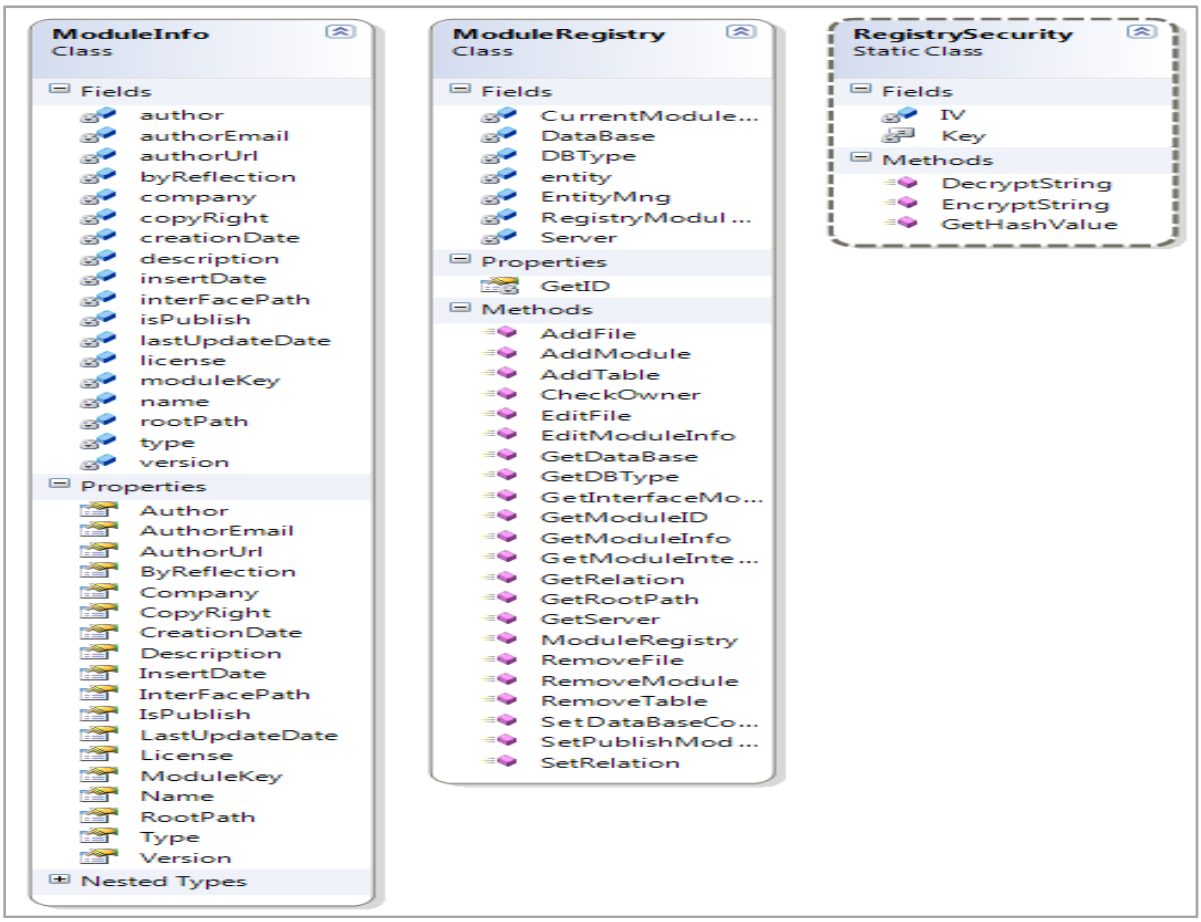

Figure 1. The Overall Structure for the Module Registry System

Module Registry Class has the main task in this module and is responsible for all work required when installing a new subsystem to record relevant metadata. In addition, it monitors all activities when running a specific module in order to prevent the unauthorized acts. Module Info Class Entity is an Entity class type used for storing metadata associated with a module. Registry Security Class will be used for security purposes such as Encryption, Decryption, and Hashing. Security of applications, especially web applications has always been a major concern for software designers. This concern is more evident in modular systems because if in these systems the required security is not guaranteed, it will be possible for modules and subsystems to sabotage information in other modules and the system itself and lead to disruption in the system. [14, 15]. In this regard, security has been ensured largely in the proposed architecture so that modules and system core can exchange their information with confidence. In addition, the way to access internal information of each subsystem has been blocked. The manner of controlling access to modules and subsystems in the proposed architecture is as follows: in the table related to modules and subsystems as well as all the tables and parts that need to be controlled, there is an attribute called Access. There, the codes related to those users who are qualified to access this part are kept. The interfaces for Module Registry Class are shown in Table (1).

Table. 1. Interfaces of Module Registry Class

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

Task description	Task title
The documentation of information related to modules	Add File (File Info)
Documenting the general information of modules	Add Module(Module Info)
Documents tables of modules	Add Table(Table Info)
Surveying the possession of a file or table related to modules	Check Owner(Source Type, string Name, string Module ID)
The revision of information related to a certain module	Edit File(File Info fi)
Revision of information related to a module	Edit Module Info(Module Info)
Access to information in the database	Get Data Base (string Module ID)
Determining the type of database	Get DB Type(string Module ID)
Access to the main interface file related to a file	Get Interface Modules()
Access to module codes. These codes are used for the authentication of a module within the system	Get Module ID(string Module Name)
Access to information in a module by the use of module codes	Get Module Info(string Module ID)
Studying the interactional capabilities of modules	Get Relation(string Source Module ID, string Destination Module ID)
Access to modules' main path	Get Root Path(string Module ID)
Access to information of the server in a database	Get Server(string Module ID)
Removal of the information of a file related to a module	Remove File(File Info)
The removal of a module	Remove Module(string Module ID)
The removal of a certain table related to a module	Remove Table(Table Info ti)
The documentation of information related to the server in a database	Set Data Base Config (string Server Name, string Data Base Name, DBMS Type DB Type, string Module ID)
Publication and non-publication for a certain module	Set Publish Module(string Module ID, bool State)
Confirming the interaction of a module with other modules	Set Relation(string Source Module ID, string Destination Module ID, Relation Type)

3.2. Data Structure for Module Registry System

If we want to implement a data structure for Module Registry System, we should first look at components modules in the system. Given that the majority of modules contain files for their input and output operation, Module Registry System must be able to identify and distinguish files related to each module and prevent unauthorized access. This is also the case for the base environment of each module and each module has several tables to preserve its data [16]. Considering these points, the data structure of Module Registry System can be seen in Figure (2).

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

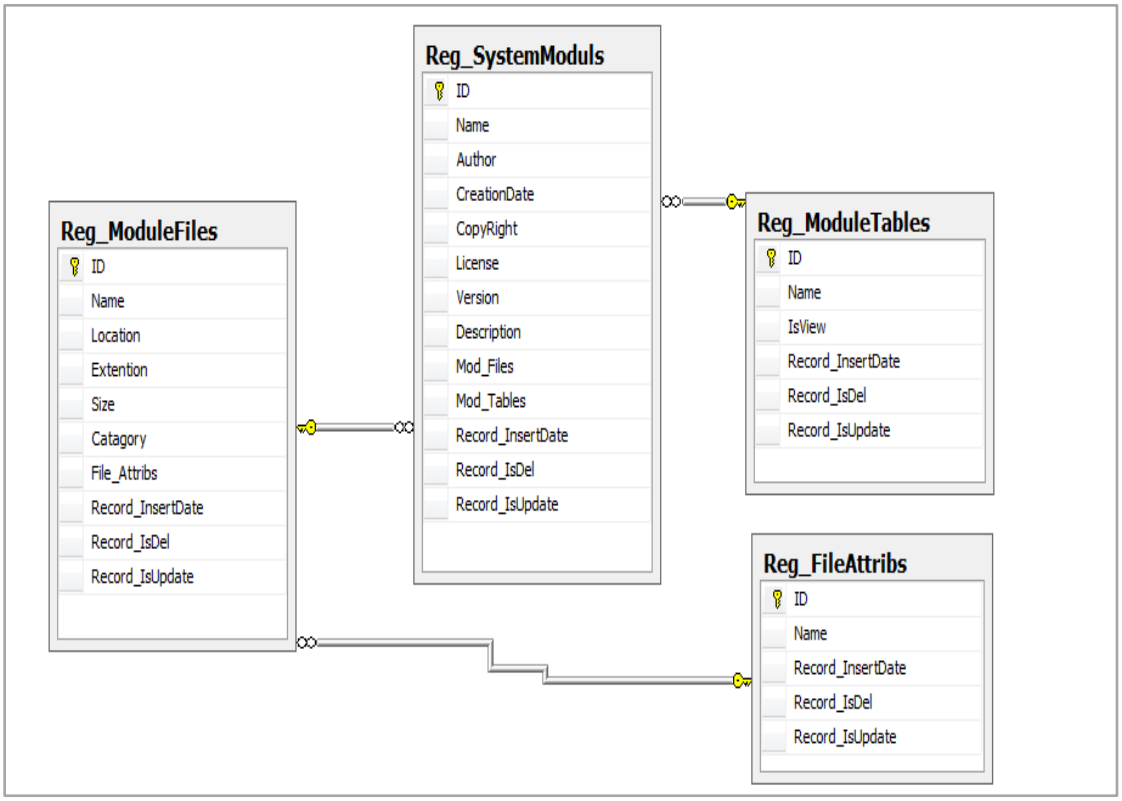

Figure 2: The Structure of Module Registry System

4. The Structure of New Modules

In the previous section, we described the general concept of modular systems. Whether these concepts can assist us in the world of programming and software architecture in order to create a modular system or not? It is clear that these concepts could reveal new ways to develop software systems. By making use of facilities that exist in programming languages and concepts discussed in the architecture, software systems can be designed that have characteristics such as Extensibility, Dynamism, Compatibility , and many others. In the design of the proposed architecture, it has been tried to get closer to the implementation of these concepts and principles as much as possible and make their principles the principles that govern the project. In order for modules to operate based on a system architecture, they have to implement the interfaces that systems have set up for them.

In this section, the system default interface of the system for subsystems will be reviewed. Based on the proposed structure, we have assigned eight overall interfaces for Module Registry that subsystems must follow [14, 15, and 16]. For example, consider an interface named “ISearch”. For modules and subsystems to be able to respond to the request or event Search performed by the core, this Interface must be implemented. The list of default interfaces and their initial structure have been shown in Figure (3).

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

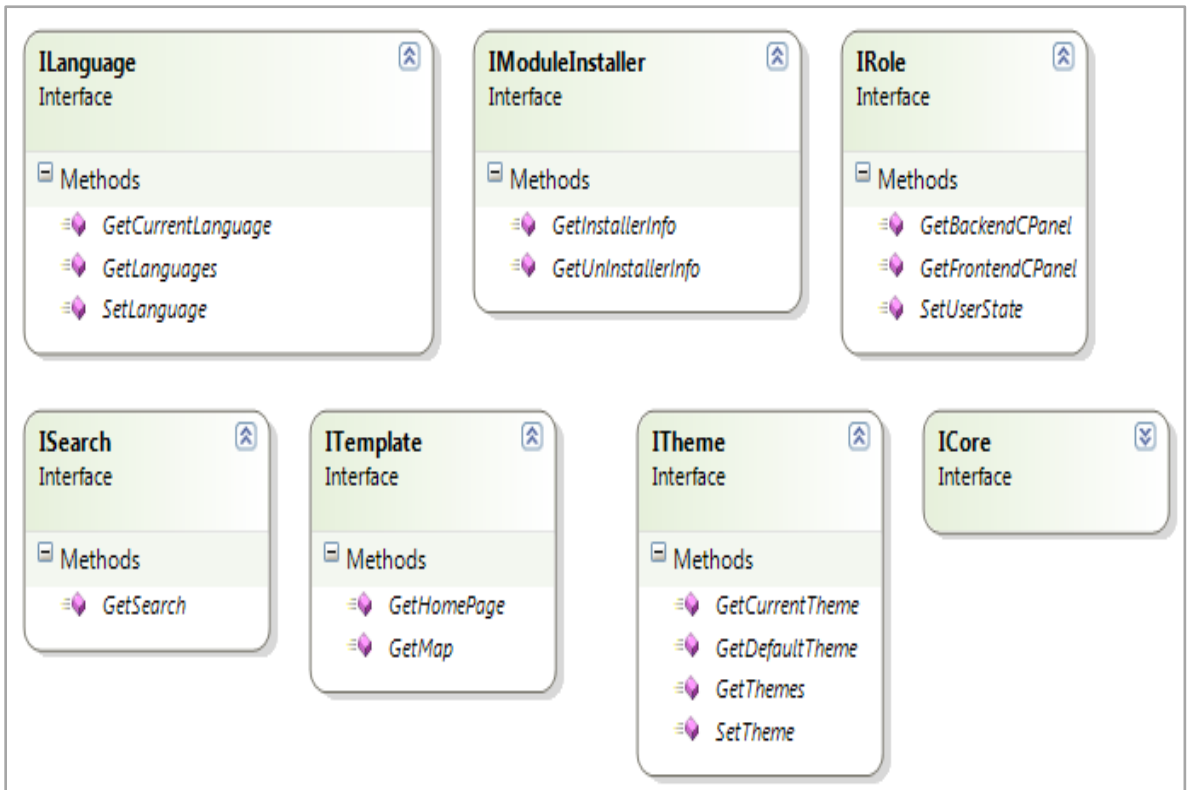

Figure 3: The List of Defaults Interface for Subsystems Based on the Proposed Structure

As seen in Figure (3), a module has to re-write all events in the core in the form of the presented interfaces in order to be able to respond to all events in the core. In order to help to facilitate the easy development of modules and system expandability, a basic module called Base Module has been designed in a way that this basic interface inherits all default interfaces in the core. Now, subsystems can easily inherit this basic module and implement interfaces [15, 16]. The structure of the Base Module Interface is shown in Figure (4). Considering Figure (4), when the core wants to communicate with a module, it recalls the desired event by the use of module interface that is present standardly in all modules.

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

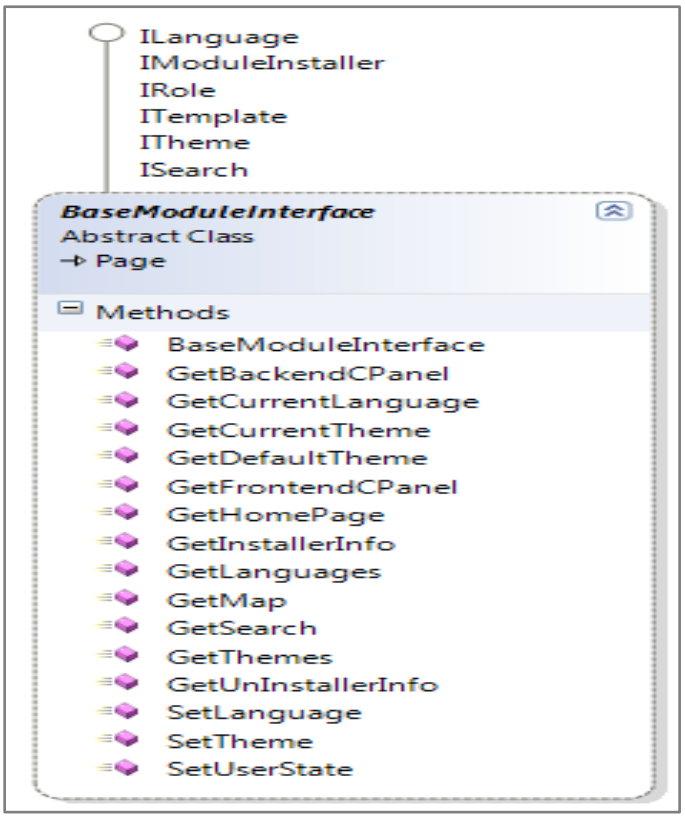

Figure 4: The Internal Structure of the Base Module Interface

4.1. Configuration of the New Module

In the architecture of ERP systems, the core establishes all necessary coordination between systems and modules based on interface that have been implemented by modules. A number of modules in the system are internal and the Core and other modules can use them within themselves. The manner of communication between system and modules is that if any module in the system intends to operate within the system has to introduce a module interface to the system in times of configuration.

By making use of this interface, the core communicates with a subsystem. In this structure, each module should have a file with a particular name and format to be conveyed by the Installer to their specific location. Later, the core generates the address specific to each module and refers to the file based on the Metadata existing in Module Registry. When the module has been developed, it will be introduced to module installer by the use of a descriptor in the form of a zip file. Then, the installer will decompress the files and will transfer them based on type and task to the pre-designed places. If the process of module configuration is successful, the new module can be implemented in systems and provide services to the final users [16, 17].

Here is a question that must be answered: How and by the use of what structure a descriptor has to be designed that contains information and metadata related to the newly entered module, can cover all aspects of a module, and be standard? What is for sure is that the designer can enhance the system's performance in any manner and define the structure of module descriptor. In describing the structure of descriptor, the point that must be observed is that

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

it must provide the installer module with the basic information of modules in the best manner possible and quite legible. The Describing structure must be in a way that the component of modules including tables and files are completely determined [17]. Therefore, in this paper, we have used the XML well-defined structure to design the describing structure. Figure (5) shows a view of the structure proposed for the new module descriptor.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Install Type="Module">
    <ModuleInfo>
        <Name>Module Name</Name>
        .....
    </ModuleInfo>
    <EntityInfo SetupFilePath="Path">
        <Entity Name="EntityName" Type="Table | View | Procedure"
            Description="">Entity Information</Entity>
        .....
    </EntityInfo>
    <FileInfo>
        <File Name="FileName" AbsolutePath="FilePath" Description=""/>
        .....
    </FileInfo>
    <Data>
        <Record FieldInfo> </Record>
    </Data>
</Install>
```

Figure 5: Structure of the New Module Descriptor based on XML

Considering Figure (5), the objective of designing a module descriptor structure based on XML to maximize readability by specifying tags that can be defined for different parts therefore, enhance the processing speed of module descriptor by the use of module installer. In addition, the development of descriptor structure in order to achieve specific goals in the system is easy to implement [10, 14, and 18]. Figure (6) offers example of a module descriptor for content management module that has been designed according to the structure shown in Figure (5) and is available to module installer in times of configuration for content management module.

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

```
<?xml version="1.0" encoding="UTF-8" ?>
<Install Type="Module">
    <ModuleInfo>
        <Name>ContentManagment</Name>
        <Author>EsmailAmini</Author>
        <AuthorEmail>esmailamini@gmail.com</AuthorEmail>
        <AuthorUrl>www.UniqueSystem.Com</AuthorUrl>
        <Company>Unique System Boukan</Company>
        <CreationDate>18/6/1390</CreationDate>
        <CopyRight>@2011</CopyRight>
        <License>Unique</License>
        <Version>1.0</Version>
        <Description></Description>
        <Directory>ContentManagment</Directory>
        <InterfacePath>\\bin\\ContentManagment.dll</InterfacePath>
    </ModuleInfo>
    <EntityInfo SqlEntityPath="Entity">
        <Entity Name="Con_Content" Type="Table"  Description="">Con_Content.sql</Entity>
        <Entity Name="Con_Section" Type="Table"  Description="">Con_Section.sql</Entity>
        <Entity Name="Con_VContent" Type="View" Description="">Con_VContent.sql</Entity>
        <Entity Name="Con_SCatagory" Type="Procedure" Description="">Con_SCatagory.sql</Entity>
    </EntityInfo>
    <FileInfo>
        <File Name="t.txt" AbsolutePath="~\\bin\\t.txt" Description=""/>
        <File Name="Xtra.htm" AbsolutePath="~\\bin\\Xtra.htm" Description=""/>
    </FileInfo>
    <Data>
        <Info Type="Table" Name="Con_Section">
            <Record ID="1" Order="0" ParentID="0" Name="Sport" />
            <Record ID="2" Order="0" ParentID="0" Name="Game" />
            <Record ID="3" Order="0" ParentID="1" Name="Football" />
        </Info>
    </Data>
</Install>
```

Figure 6: The Structure of Descriptor in Content Management Module for System Configuration

As shown in Figure (6), in addition to having a high degree of readability, the descriptor in content management module provides module installer with more details related to the new module within XML structure through the major module information such as module name, the path that connects main module with outer environment, and the name of module designer (for copyright purposes). These details include tables, files, and basic data in content management modules that are being introduced using tags such as Entity Info and File Info and Data. Given them, module installer collects full information of module and records them at Module Registry. In addition, module interface is mentioned in module descriptor, which is important and vital in the communication of modules with outer environment and the main core central core [15, 18].

5. Evaluation and Discussion

Nowadays, organizations compete with each other in the way towards globalization and this competition has led to unprecedented levels with the entry of ERP systems. To stay in the field, ERP systems need to better work processes with more flexible and reliable structures to effectively manage organizational processes and lead to benefits such as cost reduction, increased

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

productivity, and improved customer services. The introduces a structure that promotes the modularity and flexibility of ERP systems to a desirable level. Using the proposed structure, the major problem of integration of systems within an organization is overcome and responding the professional needs improves, too.

In addition, the division of projects into smaller components that can be performed independently of each other is achieved in a simpler manner. The integration resulted from the proposed structure is not only with respect to the possibility of adding new modules, but is integration with regard to the coverage of all processes needed in a section within an organization, with respect to lack of the need for the repetition of some process or parts of a process, and integration of information and lack of the need to repeat data in the whole system.

Our proposed architecture introduces a clear and agreed method for the definition and communication between software in a way that modules can be used as software components technology platforms. In fact, the proposed structure amplifies the pre-defined patch of components for software development rather than developing and implementing them.

6. Conclusion and Future Works

In this paper, we have provided a structure to increase the modularity of ERP systems. We divided the functions of ERP systems into two parts: independent modules and the core. Later, module data were kept within a part called Module Registry and a module called Module Interface was used as a bridge for the communication between modules and with the core. One of the most important tasks that each module responsible for is the establishment of communication with Event Bus (which acts as the communication highway) present in the core. In this paper, a standardized XML-based framework was designed for the module descriptor to increase its processing speed and readability.

This descriptor is delivered to the installer in times of new module configuration and the components and specifications of the new module are introduced to the system by the use of that. Some important features of the proposed structure can be considered as implemented as independent structure of modules, easy implementation, easy expandability and modularity, and its dynamic structure. Considering the above points, the proposed structure can be used as a reference for the communications between modules and the core in a variety of systems based on modules with different architecture in a way that the security of system data is ensured together with the maintenance of privacy for modules.

REFERENCES

1. Al-Mashari, M. , “Enterprise resource planning (ERP) systems: a research agenda”, Industrial Management & Data Systems, Vol. 103 No. 1, pp. 22-7, 2003.
2. Al-Mudimigh, A., Zairi, M. and Al-Mashari, M. , “ERP software implementation: an integrative framework”, European Journal of Information Systems, Vol. 10 No. 4, pp. 216-26, 2001.
3. Boykin, R.F., “Enterprise resource-planning software: a solution to the return material authorization problem”, Computers in Industry, Vol. 45, pp. 99-109, 2001.
4. F. Irmert, M. Daum, K. Meyer-Wegener, “A New Approach to Modular Database Systems”, SETMDM '08 Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)

management ACM New York, NY, USA ,2008

5. G. Hohpe and B. Woolf. "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions". The Addison-Wesley Signature Series. Addison- Wesley, 2004.
6. Bernroider, E. and Koch, S., "ERP selection process in mid-size and large organizations", Business Process Management Journal, Vol. 7 No. 3, pp. 251-7, 2001.
7. F. Soleimanian Gharehchopogh, E. Amini and B. Zebardast , "A Three-Layer Architecture based Approach for Data Access Layer in the Information Systems Production", International Journal of Advanced Research in Computer Engineering & Technology ,Vol. 2, Issue 2, 2013.
8. F. Soleimanian Gharehchopogh, E. Amini and B. Zebardast, " Aspect-Oriented Software Development based Solution for Intervention Concerns Problems : Case Study", International Journal of Computer Applications ,Vol. 63,No.4, 2013
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. P. Clements, F. Bachmann, L. Bass, D. Darlan, J. Ivers, R. Little, R. Nord, and J. Stafford. "Documenting Software Architectures", Views and Beyond. The SEI Series in Software Engineering. Addison-Wesley, 2003.
11. Frankel, David S., "Model Driven Architecture: Applying MDA to Enterprise Computing", OMG Press, Wiley Publishing, 2003.
12. Bass, Len Clements, Paul, Kazman, Rick," Software Architecture in Practice", Second Edition, ISBN: 0-32115495-9, Addison Wesley, April 11, 2003.
13. James McGovern, Scott W. Ambler, Michael E. Stevens, James Linn, Vikas Sharan, Elias K. Jo," A Practical Guide to Enterprise Architecture", Prentice Hall PTR, 2003.
14. Thomas Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall PTR, 2005.
15. F. Soleimanian Gharehchopogh, B. Zebardast and E.Amini, "Analysis and Design by Agent based MaSE Methodology: A Case Study", International Journal of Computer Applications ,Vol. 63,No.4, 2013
16. Steven, H. Spewak, "Enterprise Architecture Planning: Developing a Blueprint for Data, Applications and Technology", John Wiley & Sons Ltd., 1993.
17. F. J. Armour, S. H. Kaisler, and S. Y. Liu, "Building an Enterprise Architecture Step by Step", IT Pro, July/August, pp.31-39, 1999.
18. R. Fielding and R. Taylor. "Principled Design of the Modern Web Architecture". ACM Transactions on Internet Technology (TOIT), 2(2):115–150, 2002.

(DOI: dx.doi.org/14.9831/1444-8939.2014/2-6/MAGNT.151)