# Flexibility Point of Customer Relationship Management System based on User Perspective

Hema Subramaniam
PhD Student[1], Lecturer[2]
[1]Universiti Putra Malaysia (UPM)
[2]Universiti Selangor (UNISEL)

Hazura Zulzalil
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia (UPM)

Marzanah A. Jabar
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia (UPM)

Saadah Hassan
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia (UPM)

## ABSTRACT
Software is an essential asset of an organization in organizing business process. Thus, demand for software gradually increase from day-to-day. However, due to the complex structure of a software and tedious software design has cause delays in making it to be available in market. Consequently it increases the needs for customizing software from existing system. Under those circumstances, software customization is influenced by the flexibility of software that indicates how easy the software is modified to suit the current environment. Thus, the easiness is determined by the flexibility point that resides in software. Since flexibility is a desirable software quality characteristic, determination of flexibility point is essential. In this case, user role transformation from one domain to another domain has been viewed as flexibility point that triggers changes into the software. To demonstrate flexibility point exist at the user role, this study emphasize on the change of class design from Customer Relationship Management System (CRMS) to Tuition Centre Management System (TCMS). As a result, some slight modification on the user oriented class attributes and methods effect the changes on other associated classes. Certainly, slight modification only can be made if the user-oriented class definition is written as general purpose rather than dedicated to one system. Therefore, proper definition of user-oriented class assists in promoting flexibility and at the same time it has been view as software flexibility point.

## General Terms
Software Quality, Flexibility Measurement

## Keywords
software flexibility; flexibility point; software quality.

## 1. INTRODUCTION
General purpose functions is the essential part of any software that lead to easier customization[1]. Minimal changes or modification is required by these general purpose program in converting functionality from one environment to another[2]. However, software customization cannot be realized unless software internal structure is created in such a way to be flexible to adapt any new changes. Initially, flexibility is known as one of the software quality characteristics and it is categorized under product revision factor. Under this category, flexibility is viewed as the ease of making changes required by changes in the operating environment [3]. Comparatively, IEEE has defined flexibility as the ease of software to change its component to be worked in the environment other than it was specifically designed [4]. Thus, flexibility can be viewed in two different perspectives, changes within the domain and changes between the domains. Despite flexibility role in customizing software, questions have been raised about how to make software to be flexible since application or system is not a tangible product[5]. Furthermore, as far as we have concern there is no clear explanation on the flexibility point that resides in software. Most studies in software flexibility concentrate on the assessment of it but fail to identify the flexibility point of software or application. The aim of this study is to examine flexibility point of software by demonstrating how the component of Customer Relationship Management System (CRMS) has been customized to develop Tuition Centre Management System (TCMS). Based on the study, user perspective is viewed as flexibility point in promoting easier customization.

The paper begins by outlining the importance of flexibility in software quality models and coverage of software flexibility assessment models. Then the discussion continues by explaining about CRMS and TCMS. Discussion on TCMS and CRMS based on the class design changes in terms of user view. Depth of user perspective in flexibility point determination is covered at result and discussion section.

## 2. RESEARCH BACKGROUND

### 2.1 Software Flexibility point based on software quality models
A flexibility term has been introduced by McCall Quality Model [3] as one of the factors influencing software quality. The model categorized flexibility under product revision factor together with testability and maintainability. It also stressed that the ability to change software internal structure indicates the flexibility of that product. In that case, the point that triggers the change is considered as flexible point of software. Significant to that, end user view trigger the changes

to the existing software since different end user need different function into the system. Hence, end user view can be considered as flexibility point that facilitates changes into software[6].Identically, International Organization of Standard (ISO/IEC 9126) had discuss flexibility in different perspective by determining how easy to transfer software to another environment. Moreover, the standard refers user and maintainer behavior towards system as a point that triggers the modification. In other words ISO considers maintainer's and user's effort as flexibility point [7]. In contrast, a considerable amount of literature [8], [9], [10] had discussed flexibility in different context. For instance, FURPS model stressed product functional and non-functional requirements as support tools that trigger changes[8]. Although FURPS model stressed on functional requirement while measuring flexibility, yet requirement are gathered from user. In that case, user perspective towards system is again considered as flexibility point. Different from that, software product properties such as correctness, internal, contextual and descriptive used in evaluating the software quality characteristics especially on flexibility and maintainability assessment. In general software product properties mostly related to program source code like source code correctness, source code internal standard compatibility, source code context and descriptive nature of software[9]. In other words, code level mechanism used as flexibility point of software. However, varieties in coding style lead to inconsistence and difficulties in identifying flexibility point.

## 2.2 Software Flexibility point based on software flexibility assessment models

The models or framework that covers on software flexibility assessment have been widely investigated since 2002. Zeng and Zhao [11] reported that software flexibility can be achieved with the existence of data and process independence. They incorporated workflow and agent technique during the process of the redesigned [11]. Such incorporation promotes software flexibility at the process level instead of data level. For this reason, existing and new business processes are considered as flexibility point that triggers adaptability of software. Additionally, Martinho and Domingos [12] mentioned that people who are involved in the software development such as system engineer and development team (also known as process participant) can also trigger the flexibility process[12]. Again this would affect process flexibility instead of data flexibility. Even though development teams are involved in the software process, users are the real person who can inject the changes instead of process participant. Although this may be true, but software element such as business rules, controlled functions and parameter settings in the user interface still can be viewed as flexibility point that can change the functions of the components. In that case, neural network method known as Back-Propagation (BP) used to measure flexibility point [13]. Apart from that, Peng et al. [14] considered user action towards system as software flexibility point [14]. Normally user action triggers the changes into the system. In this case, interface which cause changes to the system such as menu, button, checkbox, component and templates is likely to be flexible point. As a result, user action changes the input interface, output interface, process logic, and information structure of software However, the point in the user action that causes the changes is unseen and hard to measure. Significantly, 3-tier architecture which consists of presentation, business and data layer are viewed as point where the flexibility resides with it. In fact, different layer had promoted different flexibility point. In this situation, data tier considers design as flexibility point, presentation tier considers interaction as flexible point and business tier considers process as flexible point. Each of the layers is coupled with the appropriate index to evaluate the software flexibility rate[15]. Similarly, series of user operation such as information and data setting in the user interface can causes the function of the system changed. Hence, user operation are again viewed as flexibility point that increases the extensibility of software regardless of user role towards system[16]. Based on the above discussion, considerable amount of literature have been highlighted user perspective as desirable flexibility point as compared to other viewpoint. Other perspectives such as business process, functional requirements, source code are also related with user perspective. Most of the flexibility point have been interrelated with user action towards the application.

Table 1 shows the discussion summary on the preferred software flexibility point.

# 3. RESEARCH METHODOLOGY

The aim of study is to show that changes in user role leads to the changes of software functionality. In other words, user role has been viewed as flexibility point that can trigger the changes of application. To demonstrate this concept, Customer Relationship Management System (CRMS) and Tuition Centre Management System (TCMS) were used as subject of study. Customer relationship management is a concept used in establishing great relationship between organization and customer[17]. Meanwhile, CRMS help in systematically keeps the record of customers and their transactions. Report generation process becomes easy since all the records being kept electronically[18]. Meanwhile, TCMS aids in managing academic related activities. Since both systems are totally from different domain, customization becomes a challenge for the software developers. However, clearly defined flexibility point would ease up the customization process. In the first place, CRMS architecture was retrieved from the system analyst. Then functionality which is related to education domain (especially on registration process) was extracted from existing CRMS design. Experience developer's view was taken into consideration in this extraction process. Once done with that, class components which related to shortlisted functionality were derived from the overall system design. Among those class components, existing name of user oriented classes were modified into a new user role. For instance user oriented classes in CRMS known as Customer been changed into Student which is closely related to education domain. Additionally, a considerable amount of changes were applied into the classes which relate with user oriented classes. These changes were based on user action towards system. Accordingly, attribute and behavior of the classes were also modified slightly to meet the education domain (TCMS) instead of business domain (CRMS). Fig 1 shows the flow of process involved while conducting the case study to determine the flexibility point. Detail explanation on CRMS and TCMS design architecture are provided in the next section.

.

**Table 1. Flexibility point summary**

| Model | Software Flexibility point | Reference |
|---|---|---|
| ***Software Quality Model*** | | |
| International Organization of Standard (ISO/IEC 9126) | User / Maintainer behavior | [7] |
| Boehm Model | End user or user perspective | [6] |
| FURPS model | Product requirements | [8] |
| Dromey model | Code level mechanism | [9] |
| ***Software Flexibility Assessment Model*** | | |
| Intelligent Workflow technique | Business process | [11] |
| Two step approach | Process participant | [12] |
| BP neural network | Business rule, controlled functions, parameter settings | [13] |
| User oriented measurement | User action | [14] |
| Grey evaluation model | Software design, interaction between component and business process | [15] |
| Assessment model | Information and data settings in the interface | [16] |

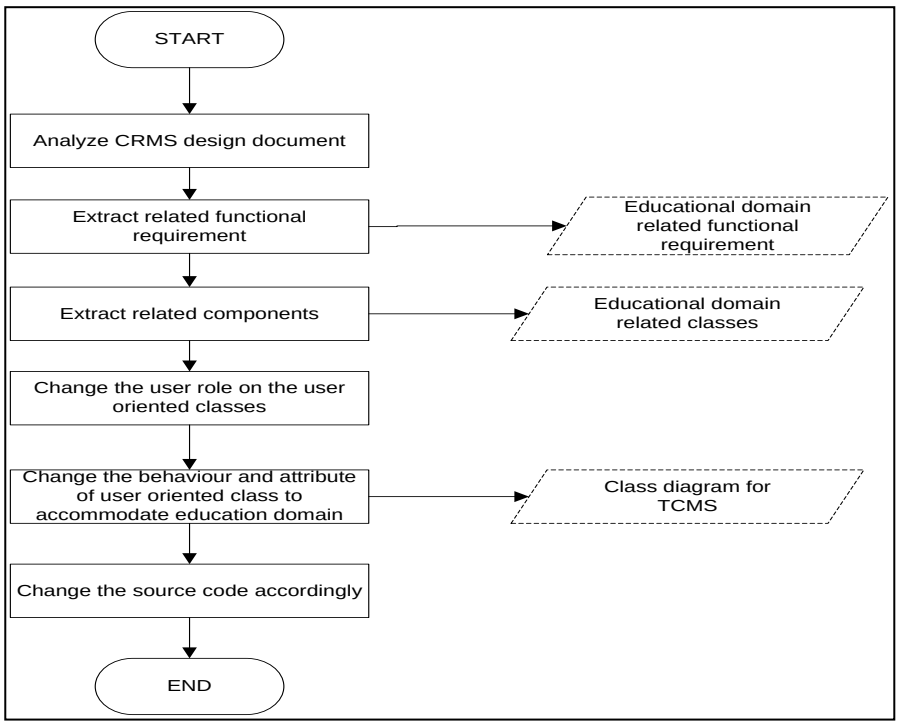


**Fig 1. Flow in customizing CRMS into TCMS**

## 3.1 CRMS Design

System design is an important mechanism that can illustrate the overall flow of the system. It can be based on any modeling language that can increase the understanding towards the system. In that case, Unified Modeling Language (UML) used to design CRMS by creating sequence diagram and class diagram. Since the company's intention was to promote flexibility in the software, most of the classes were design as general usage. Moreover, there was no dedicated classes had been used in CRMS. CRMS contain four core classes that derive the system, namely Customer, Account, Report and Product. Each class was build up based on the assumption that it would serve the customer and their business relationship matters. Among the classes, Customer class is a user-oriented class that triggers the action on the other associated classes. Fig 2 shows the CRMS class design that emphasized on user-oriented class. However, only a portion of classes, attributes and operations were extracted from the overall class scheme. Based on the class diagram, Customer class becomes the user oriented class which triggers the behavior of other classes. Since Customer class is the central focus of the CRMS, it was created to be general. Moreover, the operation specified in the Customer class is used in other system without much modification. Similarly, other classes such as Account, Product and Report which closely related to Customer class also can be modified easily. For instance, register_user() method is aim to adding customer record into database. Definition for this method can be used for any other system which is needed to insert record to a database. Fig 3 shows the flexibility point segment of register_user() method. In most of the cases, the modification involves the table and field name.In this case, is not advisable to change class function name since it involved in other related classes as well. At the same time this ensures minimal changes into the system.

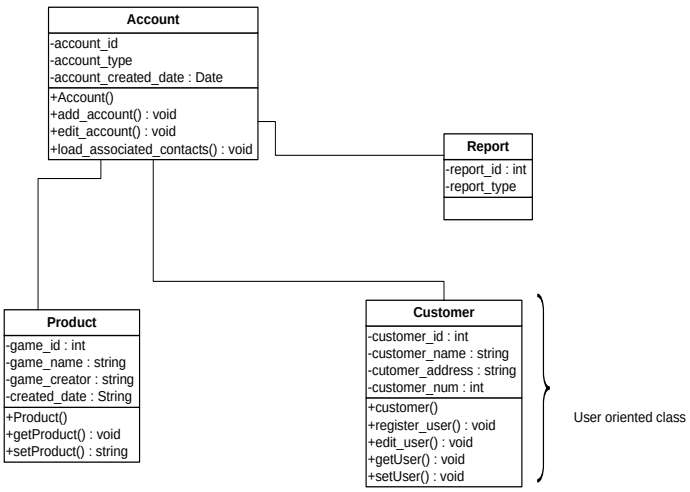


Fig 2: CRMS class design

```
<cfscript>
<cffunction name="register_user" access="public"
returmtype="void" hint="adding customer record to
database">
        <cfquery name="myquery" datasource="myCRM">
        INSERT INTO customer
        VALUES("abc", "rawang", "0123312433")
        </cfquery>
</cffunction>

</cfscript>
```

*Element that need modification*

Fig 3: register_user definition

## 3.2 TCMS class design

To demonstrate this, TCMS design mirror the CRMS design with some minimal changes on the user oriented classes and its associated methods and attributes. Since TCMS categorized under education domain, student is the main user of the system. Changes in user-oriented classes consequently affect the other related classes that inherit the function of CRMS. Student registration, subject registration, payment process and administrative section are among the functions related to TCMS. To emphasize the use of flexibility point, only few classes from the overall TCMS are explained in this section. Upon changes of the user role from Customer to Student, classes such as Account and Product that are closely related to Customer class need to be changed as well. The changes are only involved at attributes name, database query properties (table and field name) and associated link between classes. Fig 4 shows the TCMS class diagrams that emphasizes on student class as user-oriented class and at the same time as flexible point of the system.

Based on the class diagram (Fig 4), Customer class from CRMS is changed to Student class. This modification is happened due to the nature of TCMS system which involve student as user of the system. However, the behavior of Student class towards the system still remain the same as in the customer whereby Student class still can register_user, edit_user, getUser and setUser. In detail, any associated classes with Student class are derived similar function from CRMS even though the user role had been changed. Indeed certain modification is applied to existing Account and Product classes, due to the user perspective towards the system. For instance Student class only can be associated with Registration class whereby it helps in registering the subject instead of creating account. For this purpose, Account class name is changed to Registration while the function definition still remains the same. Fig 5 shows the transformation of CRMS into TCMS in terms of interface.

.

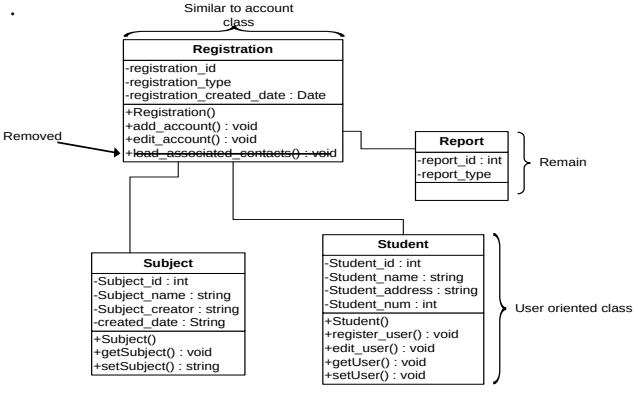


Fig 4. TCMS class diagram

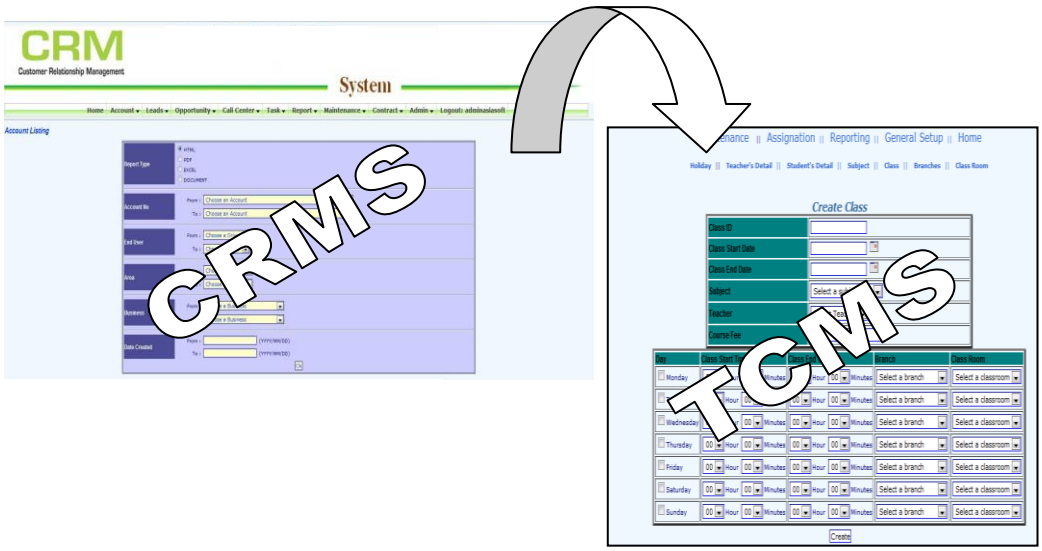


Fig 5: Interface transformation from CRMS into TCMS

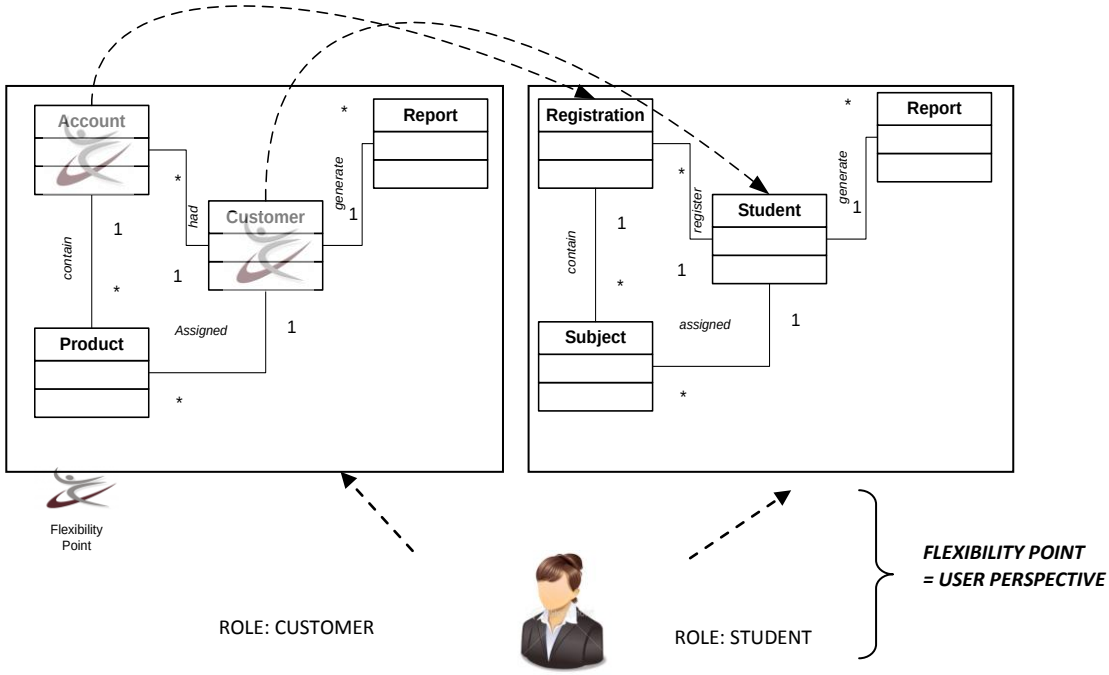


Fig 6. CRMS and TCMS flexibility point

## 4. PROPOSED SOFTWARE FLEXIBILITY POINT

Based on the CRMS and TCMS class design, we can say that transformation from CRMS to TCMS is based on user perspective. Hence, a comprehensive and flexible CRMS class design contributes towards easier customization. At the same time improper function in CRMS design can be easily solved and adjust while implementing it into TCMS. For instance, load_associated_contacts() function is not applicable for TCMS hence is removed from Registration class. Under those circumstances, the user view plays significant role in deciding the modification of the existing system. At the same time user role is determined by the domain where the system resides on. As a result, **Error! Reference source not found.** illustrates the proposed flexibility point for CRMS that trigger the development of TCMS. As can be seen on the illustration, changes in user oriented class (Customer and Student class), are obviously effected by changes on user role into the system. User holds a role as customer or student based on the domain it's belongs to. In other word, application domain determines software flexibility point with regards to user role. For this reason, user-oriented class definition need to be equipped with generalize attributes therefore it can be suited to any domain. Besides that, user class method also needs to be in general purpose to serve any other domain.

## 5. CONCLUSION AND FUTURE WORK

Flexibility points become central discussion in making software or application to be easily customizable. However, software flexibility becomes an issue due to abstract nature of software. Additionally, flexibility point is hard to be seen and measure since it relies on the functionality of the system. Based on the study, we can conclude that software flexibility point is resided on user perspective towards system. Since, the changes happened in the functionality of the system always caused by the user reaction towards the system. This assumption also is supported by a considerable amount of literature that stressed on user-oriented measurement of software flexibility. Moreover, software flexibility point which is represented by user oriented classes is expected to affect the overall system functionality and at the same time support the modification on the existing applications. Modification takes place with attention to user role in the system. In order to realize that, user-oriented classes should design to be general and customizable. Further experimental researches are needed to estimate the flexibility point measurement that is associated with user-oriented classes. Additionally software flexibility measurement also needs to be evaluated at the design level instead of coding level. Considerably, more work need to be done in measuring software flexibility such as design level measurement and degree of flexibility in making software to be more customizable.

## 6. ACKNOWLEDGMENT

This work was supported by FCSIT, University Putra Malaysia (UPM). So, we would like to give our highest gratitude to UPM for supporting us in determining the flexibility point of software based on user perspective.

## 7. REFERENCES

[1] X. Zhu and S. Wang, "Software Customization Based on Model-Driven Architecture Over SaaS Platforms," *2009 International Conference on Management and Service Science*, pp. 1–4, Sep. 2009.

[2] C. Rohleder and A. Sciences, "Software customization with xml," vol. VI, no. 2, pp. 345–351.

[3] J. A. Mccall and P. K. Richards, "Concept and Definitions of Software Quality," *Quality*, vol. I, no. November, 1977.

[4] T. Committees, "IEEE Standard Glossary of Software Engineering Terminology," 1990.

[5] M. H. Meyer and R. Seliger, "Product Platforms in Software Development," *Sloan Management Review*, vol. 40, no. 1, pp. 61–74, 1998.

[6] Boehm, "Software Quality Models and Philosophies," in *Management*, 1978.

[7] I. JTC 1/SC, "ISO/IEC 9126-2: Software Engineering-Product Quality-Part 2: External Metrics." Canada, 2002.

[8] R. B. Grady, *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, 1992.

[9] R. G. Dromey, "A model for software product quality," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146–162, 1995.

[10] A. H. Eden and T. Mens, "Measuring Software Flexibility," *Computer*, vol. 153, no. 3, pp. 113–126, 2006.

[11] D. D. Zeng and J. L. Zhao, "Achieving Software Flexibility via Intelligent Workflow Techniques," in *Proceeding of the 35th Annual Hawaii International Conference on System Sciences*, 2002, vol. 00, no. c, pp. 7–10,606–615.

[12] R. Martinho and D. Domingos, "A Two-Step Approach for Modelling Flexibility in Software Processes," *23rd IEEE/ACM International Conference on Automated Software Engineering*, pp. 427–430, 2008.

[13] J. Niu, L. Shen, S. Peng, and F. Li, "A Measurement Method of Software Flexibility Based on BP Network," *International Workshop on Intelligent Systems and Applications*, pp. 1–4, 2009.

[14] S. Peng, L. Shen, H. Liu, and F. Li, "User-Oriented Measurement of Software Flexibility," in *2009 WRI World Congress on Computer Science and Information Engineering*, 2009, vol. 7, pp. 629–633.

[15] S. Wang and X. Liu, "A Study on Flexibility of ERP System Based on Grey Evaluation Model," *Technology*, pp. 3–6, 2010.

[16] Y. Wang, M. Jia, J. Guo, and B. Zhang, "Evaluating Model of Software Flexibility of Domestic Foundational Software," in *International Conference on Electrical and Control Engineering*, 2011, pp. 5906–5909.

[17] G. K. Agrawal, "The Development of Services in Customer Relationship Management (CRM) Environment from 'Technology' Perspective," *Journal of Service Science and Management*, vol. 02, no. 04, pp. 432–438, 2009.

[18] C. Shen, B. Han, Q. Zhou, and W. Song, "Design and Implementation of Customer Relationship Management System Based on Structured Object-Oriented Methodology," *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, pp. 390–393, Sep. 2011.