

Process Component Plug-in Approach

Jin Myung Choi

School of Computing
University of Soongsil
Seoul City, Republic of KOREA

Sung Yul Rhew

School of Computing
University of Soongsil
Seoul City, Republic of KOREA

Abstract - *We use development methodology and software development processes to develop software effectively. But several processes such as CBD96, Catalysis, and RUP do not offer the detailed descriptions for effective execution of some activities or tasks. In this paper, we propose an approach to substitute plug-in for existing process of insufficient description and definition using reusable minimal external process components. As a result, we apply this process component plug-in approach to MaRMI(Magic and Robust Methodology Integrated)-III – Korean standard information system development methodology. Firstly, this approach helps customizing established processes effectively and substituting (or extending) existing process components for insufficient parts. Secondly, we can develop new processes easily by constructing various process components in several ways.*

Keywords: Process Component Reuse, Software Process Customization, Software Process Plug-in

1 Introduction

Presently, the most of software development, object oriented development languages, component model, and frameworks are used and all these methods are focused on software reusability. And architectures such as MDA (Model Driven Architecture) [1] or PLE (product line engineering) [2] are useful to produce other software using specific software itself. These requirements of software reusability need for reusability on the business domains as well as software functions. For the reusability of the software functions and business domains, components such as COM+, EJB are useful. These components are developed using object oriented languages and various design pattern are used more popular for the software functions and business domains [3] [4]. For this reason, many object oriented development methodology, component development methodology, and component based requirement analysis, design, and development on the processes recommends using the design pattern actively. But, neither methodology such as CBD96, Catalysis, RUP, MaRMI-III which is often used in component based software development does not present specific methods for practical identification of pattern, abstraction, application. The appropriate solutions for the problems are described in [5], but they do not propose the method to use the effective plug-in to the existing process. In this paper, we propose a method to extend existing process and methodology and to add specific domains easily. This

method plugs into the specific domains which does not include existing process or methodology using specific process component. Chapter 2 presents existing process customizing technique and explains Korean standard information system development methodology, MaRMI-III, which is used to apply our plug-in approach. Chapter 3 defines the model of process components that are used as plug-in. Chapter 4 describes the details of plug-in for process components. Chapter 5 shows the example which plugs in MaRMI-III with process component which includes the effective design pattern selection approach [5] and chapter 6 concludes.

2 Process Customizing and MaRMI-III Methodology

2.1 Process Customizing for Component Selection

Abdallah uses selection process options and project domain characteristics and constraint sets [6]. Selection process options are used to select COTS components and project domain characteristics and constraint sets describe the characteristics of the specific project and domains. SPOs consist of process level and activity level [7] [8]. Process level analyses the evaluation strategy and included steps/activities. Activity level analyses techniques in activities, evaluation factors, stage depth analyzing present activity size levels, iteration, knowledge source evaluating different elements, roles assigned to each activity. PDCs define present specific project or domain as a set P: (Eff, Cpx, Crtc, PG, Crtn, Exp). Project domain P consists of Eff, Cpx, Crtc, PG, Crtn, and Exp. Eff means available effort for project, Crtc means Criticality of project domain – for example, is it related to army or national security? PG means project genre, Crtn means the certainty of information and Exp means the available experience. To choose a component, after modeling existing process, it performs customizing of process level by applying SPOs to the PDCs that are the subjects of customizing. In the modeled process of previous result, we eliminate unnecessary activity and task, etc. After we perform activity level customizing of SPOs, we can finally extract recommended items of SPOs which a customized process must have. And we refine the process by repeating this stage several times.

2.2 MaRMI-III Methodology

MaRMI-III is information system development methodology developed by ETRI and several research

centers. MaRMI-III is based on ISO 12207 [14] as a reference mode. MaRMI-III methodology consists of phase, activity, task, procedure, and provides procedure description, results, technique description, example collection, electronic manual, etc. So, we can develop system easily using these materials. Figure 1 shows the whole structure of this methodology.

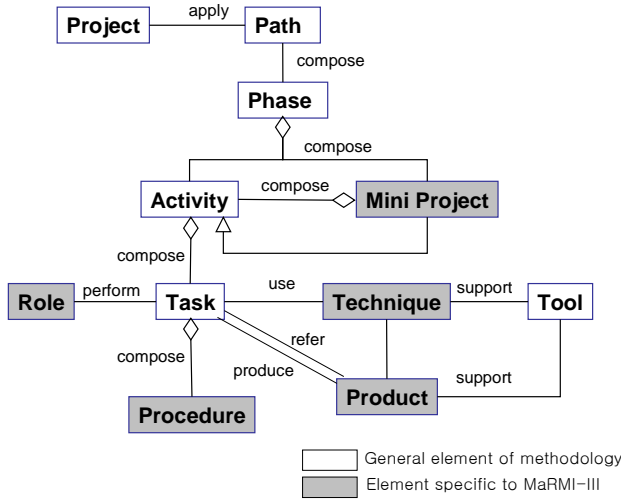


Figure 1: MaRMI-III Meta Model

MaRMI-III methodology consists of 5 phases as follows; planning phase includes understanding requirements, requirement definition, develop strategy, and project management plan. Architecture phase includes analysis of requirements, architecture design, architecture prototyping, architecture development. Project phase defines the life cycle from the beginning to the end of project. Incremental development phase and delivery phase develops and delivers software, respectively. Especially, it provides guidelines for developing component based software using design pattern in gradually development phase using J2EE and .NET.

3 Process Component Model

EJB, COM+ components provide reusability of binary level in application software. Like these components, process component itself is cooperating component [9] and suitable for making new process to the constructible form as one component or adding new process to existing processes. Process component is defined as a part of process integrated with other processes to construct organization or special task-related process like a component. This process component model follows the component diagram notation defined in UML 2.0[12] [13] as figure 2. In Figure 2, Process component name means the name of process component which will be reused. Required interface and provided interface plays a role of link communicating with existing processes when process components are reused to add existing process and new process respectively. Required interface cannot communicate with required interface of other process components and provided interface also cannot communicate with provided interface of other process component. Only the form of required interface and provided interface or form of provided interface and

required interface can communicate with other component process. And this interface contains information on previous task and the result of each process component. Therefore, when we cannot decide the exact link position of reusing process component by the process component name which is process component stereo type, we can decide the exact link position through the required interface and provided interface.

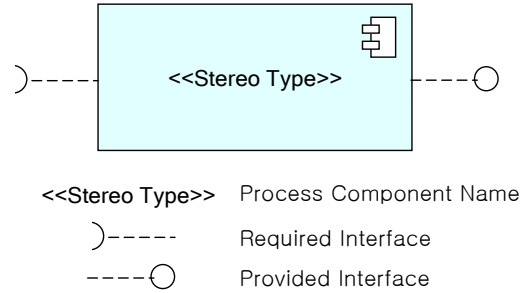


Figure 2: Process Component Model

4 Process Component Plug-in Approach

Process components have the workflow of specific business work by itself or especially multiple workflows for analysis and design of software development. Therefore, this process component can be reused by plug-in as one of component in the process of the existing business domain or the multi-purpose process of software development. In order to plug-in a process component to the existing process, we have to consider in depth the relationship in each phase and activity, and related documents from plug-in object process component and the existing process. For example, in the case of specific activities of a process, the relationships are in orderly fashion; the previous activity's results needs to be delivered to the current activity's input. And the currently performed activity's result needs to be the next activity's input. Therefore, it is the most important that the relationship between such phase and phase, the relationship between activity and activity should not be corrupted to plug in a new process component. Therefore, this paper provides the plug-in method that performs activities of process component analysis, the existing process analysis, plug-in position discernment, process component plug-in, plug-in result validity verification in order to plug in process component effectively. Among these, there is no before-after order in the two activities of process component analysis to plug in and the existing process analysis. Therefore, after the process component analysis activity to plug in is preceded, the existing process analysis activity can be performed or vice versa.

4.1 Process Component Analysis to Plug-in

Object – It is to analyze the process component functionalities (or business workflow) that is to be reused by plug-in and the construction information (input, output results, etc) that will work as interface dynamics.

Input Criteria – It is when the requirements are collected and the requirements need this process

component regarding the specific business domain or process.

Activities – We grasp the business workflow of process components to reuse by plug-in and the main functionalities. We analyze the process component definition document to grasp the business workflow and then core functionalities then analyze the provided interface and the required interface of process component that work as link to the plug-in object process. In the process of analyzing the required interface, we find out the related input documents that describe the limitations and pre-requisite activities or preparation to perform the process. In the process of analyzing provided interface, we find out the related documents - The functional limitation of process component is stated - given after the process component performs the unique business workflow.

Result – We extract the process definition document that states the business role of process component to plug in and limitations, etc. and the required interface information (document or input information to the process component) by this process component, the provided interface information (output results).

Output criteria – Information such as the business workflow of process component to plug in, functional definition, limitation contents, input, and output result is analyzed.

4.2 The Existing Process Analysis

Object – In order to find out whether the process component to plug in can be plugged in the existing process, we analyze the construction information of the existing process. The construction information of this process includes the information such as the phases of process, the activities by each phase, and the core functionalities of each activity, business workflow, the input criteria of each activity, input, output criteria, and output results.

Input criteria – It is when we secure the document set or record media that includes all the contents of the existing process that are the analysis objects.

Activities – We analyze the process construction information such as the existing process phases, activities of each phase, the core function of each activity, business workflow, input/output criteria of each activity, input, and output results. Then, we modelize each phase of existing process to a conceptual process component just as Figure 2's process component. And we make out the summary data as to the analysis result to find out the proper position of the future plug in object process component to the existing process.

Results – the construction information of existing process – phases, activities by each phase, the core functions of each activity, business workflow, input criteria as to each activity, input, output criteria, output results – are identified and each identified phase, activity, and the relationship between input and output, etc. are summarized and documented. Then the conceptual process component model as to the existing process is made out.

Output criteria – Existing process's phase, activities at each phase, and task areas of phase or activity,

constraints of each activity, input/output criteria, input, and output results are identified and documented.

4.3 The Identification of Plug-in Position

Objective – We find out the information of the proper position - object phases, activities, and task areas - to plug in after we grasp whether the process component to plug in is related to the phase, activity, task area, or functionality of the existing process, which is completed in analysis.

Input criteria – The analysis of the object process component to plug in and the analysis of existing process is completed.

Activities – We compare the definition of process component, business areas and the task areas of existing process's activities at each phase. In order to do that, we utilize the information of the process component's function definition, limitations, workflow, input, output results, which is the output result of 4.1 activity, and the summary document of existing process derived from 4.2 activity and conceptual process component. In order for the process component to plug in not to damage the continuation by activity of existing process and relationship among various results, we grasp the reusing process component based on the existing process. Then we derive the phase or activity from existing process where the new process component is to be plugged in.

Results – The phase or activity from existing process where the process component to be plugged in is identified. Then, the identified phase or the preceding activity, the next phase and activity information, the task area of each phase and activity, interface information(input, output), etc.

Output criteria – The position information to plug in a new component to existing process is identified. Based on the position information, existing process's preceding phase and the next process or activity's relation information is found out.

4.4 Process Component Plug-in

Objective – We plug plug-in object process component in identified position for existing process.

Table 1: Process Component Plug-in Order

Order	Process component	Existing process
1	Required interface	Plug-in (Substitution) phase Previous phase of activity Provided interface information in activity
2	Business area and function of process component	Plug-in (Substitution) phase Business area and function of activity
3	Provided interface	Plug-in (Substitution) phase Next phase of activity Required interface information in activity

Input Criteria – We identify the position information where new process component based on task area of existing process plugged in.

Activity – To plug a process component in the existing process, we follow the procedures as Table 1. Firstly, as order 1 of Table 1, we apply as input of process component, which will be plugged in provided interface in previous phase or activity based on the identified position of existing process plugging in existing process. At this time, compared to the required interface information of process component plugging in, we maintain the unique interface information. Secondly, after eliminating the repetition using existing process identified as plug-in object or the core business area of activity, the core business area of process component plugging in function and provided function, we substitute it, as Order 2 shows. At this procedure, not to make identified phase or core function of activity in existing process spoiled, we include unique business area which is identified in the phase of existing process or activity that is not contained in process component which is plugged in. At last, we apply the provided interface of process component which will be plugged in as an input for the next phase of existing process or the required interface of activity. We eliminate or integrate common interface element by comparing required interface in next phase or activity and provided interface of process which will be plugged in used as input of it. By performing this, we ensure additional business domain and function through process component plugged in while we maintain the order of flow among each phase or activity which existing processes have.

Result Output – By process component being plugged in, we can find out the definition description of phase or activity of newly defined existing process, interface information (input, output).

Output criteria – New process component is plugged in identified position (phase or activity) of existing process.

4.5 Plug-in Validation Test

Objective – It is shown that process component was correctly plugged into the position identified as plug-in position of existing process and process component does not interfere with task order and the flow of existing process.

Input criteria – Process component was plugged in the position of identified existing process.

Activity – To find out the full task order and flow of existing process, we analyze the relation of previous and next phase or activity in existing process about identified position information by 4.3 activity. We analyze task definition, input criteria, and require input, generated output, output criteria of phase or activity of existing process identified as plug-in position. Then we compare the result with the result output of 4.4 activity -definition description, interface information (input, output) of phase or activity of newly defined existing process after plugging in. By comparing this comparison result with full order and flow about phase and the activity of existing process and comparing this with interface of process component which is plugged in and fulfill change

of core business or function and input/output, we verify the correctness of input and output result.

Result output – We make a checklist comparing task definition about phase or activity of existing process identified as plug-in position, input/output information, input/output information of previous/next activity related to corresponding phase or activity with task definition of phase or activity after process component is plugged in, input/output.

Output criteria – Process component which is plugged in does not interfere with the order and relation of existing process.

5 Application Example

5.1 Analysis of Design Pattern choice process Component

When we develop pattern based software, the most important thing is how the most suitable pattern for resulting problem domain can be chosen and applied in analysis and design phase. The main object of design pattern selection process [5] is applying base technology selection from requirement analysis, software/hardware architecture selection, classifying design pattern suitable for hierarchical structure of analyzed object system. The process to perform this consists of 4 components. Table 2 shows the function definition of design pattern selection process component. Design pattern classification and pattern mapping are subcomponent used importantly among this process component. Table 3 shows interface of process component.

Table 2: Function of Design pattern selection process component

In requirement analysis phase, we select core technology needed to develop pattern based software by analyzing present system or object system with user's requirement analysis result for a basis. In architecture selection phase, after verifying technology selected to design analyzed system in requirement analysis phase, we select hardware/software architecture. And we verify hardware/software architecture selected for object system in this phase. In design pattern classification phase, we classify system modeled on the basis of selected architecture to 6 hierarchies proposed in [4]. Next, we collect design patterns defined in [3] [4] and design pattern which is newly published and used. In pattern mapping phase, we map designed system architecture and related pattern to classified 6 hierarchies properly. At this time, we have to consider the characteristic of each hierarchy and problem, solution, and the consequence of pattern to be mapped.
--

Table 3: Design Pattern Selection Process Interface Information

Required interface	Requirement description, present system definition, system definition, network infrastructure definition, technology description
Provided interface	Component design description, design pattern description, design pattern relation description

Table 4: Comparison of Activities in each stage of Process component to be plugged in and MaRMI-III

Phase	Design Pattern Selection Process	MaRMI-III	Phase
Requirement Analysis Stage	R100(user requirement analysis)	R0104 (Requirement Collection)	Requirement Acquisition Stage
	R200(present system analysis)	R0302(Initial System Architecture Definition)	
	R300(Object system analysis/Design)	R0302(Initial System Architecture Definition)	
	R400(Object Technique Selection to apply)	R0304(reusable component check)	
	R500(Verification of Suitability for Selected Technique)	A0304(Architecture Prototype Development)	Architecture Definition Stage
Architecture Selection Stage	A100(Hardware Architecture Selection)	A0208(System Architecture Definition)	Architecture Definition Stage
	A200(Software Architecture Selection)	A0202(Software Architecture Definition)	
	A300(Verification of selected H/W, S/W Architecture)	A0304(Architecture Prototype Development) A0306(Architecture Prototype Evaluation)	
	A400(Writing Description of H/W, S/W Architecture Description)		
Design Pattern Classification	D100(Writing System Hierarchy Structure)	D0104(Architecture Refinement)	Design Stage
	D200(Collecting known Design pattern)		
	D300(Classification of collected design pattern)		
	D400(relation analysis of design pattern)		
	D500(Extraction of Pattern related to written system architecture)		
Pattern mapping	P100(Extracted design pattern mapping to each hierarchies of system)		
	P200(Final Design Pattern Management)		

5.2 MaRMI-III Methodology Analysis

MaRMI-III methodology consists of phase, activity, task, procedure, mini project, technique, instrument, function, and result.

- Plan phase: phase preparation, understanding requirement, defining requirement, setting up development strategy, project plan, and phase verification.
- Architecture phase: phase preparation, requirement analysis, component identification, architecture definition, writing component description, architecture prototyping, incremental development plan, and phase verification.
- Incremental development phase: mini project preparation, requirement and architecture refinement, component design, component implementation, guideline development, component test, component integration test, mini project verification.
- Delivery phase: phase preparation, system installation, user education, management after installation, user acceptance test, phase verification.

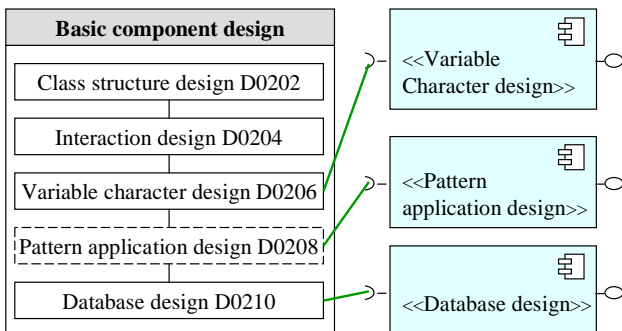


Figure 3: Component design activity

Especially, the component basic design activity of incremental development phase specifies detailed procedures for component design applying pattern as Figure 3. But the pattern application design (D0208) shown in dotted line does not specify detailed guideline, in example, how pattern can be selected, which relations are there with each pattern, in which domain each pattern is used, and so on.

5.3 Position Identification of Plug-in of Design pattern selection Process Component

To find out the position in MaRMI-III methodology where Design pattern selection process component proposed in [5] plugged in, we compared activities in each phase of process component and MaRMI-III. Table 4 shows the comparison result. As shown in Table 4, all activities in design pattern classification phase of process component and pattern mapping phase are missing in the design phase of MaRMI-III. Therefore, the pattern application design (D0208) activity of MaRMI-III is the position where design pattern selection process component plugged in as Figure 4.

Table 5: Pattern Application Design Activity Information

Previous Activity	Variation Design (D0206)
Next Activity	UI Detailed Design (D0302)
Operation Definition	Design Component using pattern
Interface Information	Input : Component Design Description (Design Pattern) Output : Component Design Description (Design Pattern Description)

Table 6: Each activity information before and after process component plug-in

Relation before Plug-in		Relation after Plug-in	
Variable Character Design (D0206)	Provided interface: Component Description, Component Design, Variable Character Design	Variable Character Design (D0206)	Provided interface: Component Description, Component Design Description, Variable Character Design Description
	Task Definition: Variable Character Verification, Variable Character Implementation Design, Component Detail Repletion		Task Definition: Variable Character Verification, Variable Character Implementation Design, Component Detail Repletion
Pattern Application Design (D0208)	Requiring Interface: Component Design(Design Pattern)	Design Pattern Selection Process Component	Requiring Interface: Requirement Description, Present System Definition, Network Infra Description, Technique Description, Component Design Description(Design Pattern)
	Task Definition: Finding Out Design Problem, Design Pattern Application, Pattern Application Result Verification		Task Definition: Finding Out Design Problem, Writing System Structure, Known Design Pattern Collection, Collected Design Pattern Classification, Relation among Design Pattern Analysis, Extraction of Written System Architecture and related pattern, Mapping Design Pattern to each System Hierarchy, Final Design Pattern Management
	Provided Interface: Component Design Description (Design Pattern Description)		Provided Interface: Component Design Description, Design Pattern Description, Design Pattern Relation Description
Database Design (D0210)	Requiring Interface: Component Design Description, Variable Character Description, Transaction Definition Description, System Architecture Definition Description, UI Design Description	Database Design (D0210)	Requiring Interface: Component Design Description, Variable Character Definition Description, Transaction Definition Description, System Architecture Definition Description, UI Design Description
	Task Definition: Web Package Structure Definition, Java Beans Definition, Server Page Design, Arrangement Design		Task Definition: Web Package Structure Definition, Java Beans Definition, Server Page Design, Arrangement Design

Related information about pattern application design (D0208) activity identified as target position plugged into is as Table 5.

Phase	Design Pattern Selection Process	MaRMI-III	Phase
Classification of design pattern	D100(Creation layers of system)	D0104(Architecture refinement)	Refinement of requirement and architecture
	D200(Collection of well known design pattern)		
	D300(Classification of collected design pattern)		
	D400(Analysis between relation of design pattern)	D0208 (Pattern Application design)	
	D500(Pattern extraction related to created system architecture)		
Pattern mapping	P100(Extracted design pattern mapping at each layer of system)		
	P200(Management of final design pattern)		

Figure 4: Exact Plug-in Position of Process Component in MaRMI-III

5.4 Design Pattern Selection Process Plug-in

Required interface information of process component includes component description, component design description, variable character design description, which is the provided interface information of variable character design (D0206) activity which is a previous activity of plug-in target activity identified in 5.3. Next, the core performance task of pattern application design (D0208) of MaRMI-III is substituted for the core function of design pattern selection process component.

In other words, design problem understanding, design pattern application, and pattern application result verification procedure specified in pattern application design activity are substituted for design pattern classification, specific activity of pattern mapping component of Figure 4. At last, provided interface – component design description, design pattern description, design pattern relation description - described in Table 3 in section 5.1 are included in required interface information of UI detailed design (D0302) activity which is the next activity of pattern application design (D0208) described in Table 5. Figure 5 show the model after applying plug-in.

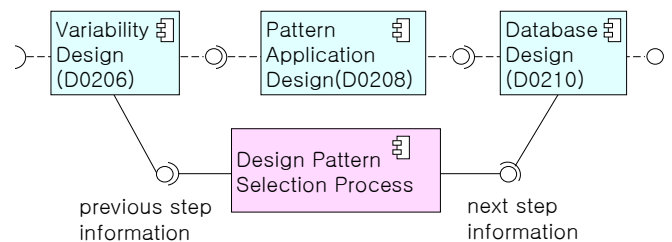


Figure 5: Activity Flow of MaRMI-III after Process Component is plugged in.

5.5 Suitability Verification

The related information comparison table of previous and next activity pattern application design (D0208) which is identified in 5.2 and 5.3 and design pattern selection process component substituted for existing activity newly by plug-in approach in 5.4 are described in Table 6. Table 7 shows the checklist used to decide suitability and suitability verification result.

Table 7: Suitability verification checklists and result

Comparison	Agreement	Process plugged-in changed items in component
Task Definition of Activity	✓	Writing System Hierarchy Structure, Collection Know Design Pattern, Classifying Collected Design Pattern, Analysis of Relation among Collected Design Pattern, Extraction of Written System Architecture and related pattern, Mapping Design Pattern to each System Hierarchy, Final Design Pattern Management
Required Interface	✓	Addition of Requirement Description, Present System Definition Description, Network Infra Description, Technique Description
Provided Interface	✓	Addition of Design Pattern Description, Design Pattern Relation

6 Conclusions

In software development, component development approach is mainly used. As essential element techniques for these effective components development, design pattern is noticeable and applied largely. For these reasons, it is recommended to apply design pattern to the processes (such as CBD96, Catalysis, RUP, and MaRMI-III, etc). MaRMI-III is used as the main development methodology in many software development projects in Korea. But even MaRMI-III does not provide detailed guideline to apply design pattern effectively in real development phase. So we are not convinced of the performance and quality of developed component. For this reason, we propose process customizing techniques to substitute componentized entirely different process component for existing process or methodology using plug-in. It consists of process component analysis, existing process analysis, plug-in position identification, process component plug-in, and plug-in suitability validation. We show the example which compensates the weakness by plugging design pattern selection process component substituting pattern application design (D0208) of MaRMI-III in MaRMI-III and plugging external process component in the domain which has insufficient application guide. By this approach, we can customize several existing processes or development methodology effectively, and we can easily extend or substitute other process component for insufficient domains. In addition, we can easily develop new process or methodology by combining several process components in several forms.

To use methodology or process effectively, Ibarguengoitia proposes customizing procedures for software process like 9 steps of activities suggested in [9]. But, Ibarguengoitia does not present reuse basis and specific method for customized processes. In the future,

we will propose the componentization technique for software process. This technique can make several forms of processes easily by combining process components as one block.

7 References

- [1] OMG, MDA Guide Version 1.0.1, omg/2003-06-01, June, 2003.
- [2] Paul Clements, Linda Northrop, *Software Product Lines*, Addison-Wesley, 2002.
- [3] E. Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.
- [4] Deepak Alur, John Crupi and Dan Malks, *Core J2EE Patterns: Best Practices and Design Strategies*, Prentice Hall, 2001.
- [5] Jin Myung Choi, Sung Yul Rhew, "An Effective Pattern Selection Process for Developing of Pattern - Based Software," *KISS: Software and Application*, Vol 32, No. 5, pp. 346-356, May. 2005.
- [6] Abdallah Mohamed, Guenther Ruhe, and Armin Eberlein, "Decision Support for Customization of the COTS Selection Process," Proc. 2nd International workshop on Models and processes for the evaluation of off-the-shelf components (ICSE'05), ACM, pp. 1-4, May. 2005.
- [7] John C. Dean, Mark R. Vidger, "COTS Software Evaluation Techniques," Proc. The NATO Information Systems Technology Panel Symposium on Commercial Off-the-Shelf Products in Defence Applications, NRC 43625, April. 2000.
- [8] Ncube, C., Dean, J.C., "The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Components," Proc. 1st International Conference on COTS-Based Software System (ICCBSS'02), (LNCS 2255, Springer), NRC 44926, pp. 176-187, Feb. 2002.
- [9] G. Ibarguengoitia G., J.A. Salazar C., M.G. Sánchez M., and A. Y. Ramirez M., "A Procedure for Customizing a Software Process," Proc. 4th Mexican International Conference on Computer Science (ENC'03), IEEE, pp. 68-72, 2005.
- [10] Kevin A. Gray, Timothy E. Lindquist, "Cooperating Process Components," 23rd Annual International Computer Software and Applications Conference (COMSAC'99), IEEE, pp. 218-223, 1999.
- [11] Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)*, Prentice Hall, 2002.
- [12] OMG, UML Superstructure Specification v2.0, formal/05-07-04, August, 2005.
- [13] Rational Unified Process (Rational Unified Process V7.0), Available from: <http://www-128.ibm.com/developerworks/downloads/r/rup/>
- [14] ISO, ISO/IEC 12207 Information technology - Software life cycle processes, August, 1995
- [15] Dong-Han Ham, Jin-Sam Kim, Jin-Hee Cho, and Su-Jung Ha, "MaRMI-III: A Methodology for Component-Based Development," *ETRI Journal*, Vol 26, No. 2, pp. 167-180, April. 2004.