Developing Distributed System with Service Resource Oriented Architecture

Hermawan[1]*, Riyanarto Sarno[2]
[1]Dep. of Informatics Eng., Fac. of Engineering, Universitas Trunojoyo Madura, Indonesia
[2]Dep. of Informatics Eng., Fac. of Information Tech., Institut Teknologi Sepuluh Nopember, Indonesia
email: hermawan.unijoyo@yahoo.co.id[1]*, riyanarto@if.its.ac.id[2]

Abstrak

Service oriented architecture (SOA) merupakan paradigma desain dari rekayasa perangkat lunak untuk skala enterprise yang dibangun di dalam lingkungan sistem terdistribusi. Paradigma ini ditujukan pada abstraksi fungsionalitas aplikasi sebagai sebuah layanan melalui protokol teknologi layanan web yang disebut dengan simple object access protocol (SOAP). Adapun SOAP bersifat statis dan berorientasikan pada metode layanan sehingga memiliki keterbatasan dalam pembuatan dan pengaksesannya pada jumlah layanan yang besar. Karena itu dalam penelitian ini dilakukan kombinasi SOA dengan resource oriented architecture (ROA) yang berorientasi pada sumberdaya layanan menggunakan representational state transfer (REST) untuk memperluas cakupan layana. Kombinasi arsitektur ini disebut sebagai service resource oriented architecture (SROA). SROA mampu mengoptimalkan distribusi aplikasi dan integrasi layanan yang diimplementasikan untuk membangun perangkat lunak manajemen proyek. Untuk merealisasikan model ini, perangkat lunak dibangun sesuai dengan kerangka kerja Agile model driven development (AMDD) untuk mengurangi kompleksitas pada semua tahapan proses pembangunan perangkat lunak.

Kata Kunci: *AMDD, REST, ROA, SOA, SROA*

Abstract

Service oriented architecture (SOA) is a design paradigm in software engineering for an enterprise scale which built in a distributed system environment. This paradigm aims at abstracting of application functionality as a service through a protocol in web service technology, namely simple object access protocol (SOAP). However, SOAP have static characteristic and oriented by the service methode, so have restrictiveness on creating and accessing for big numbers of service. For this reason, this reasearch aims at combining SOA with resource oriented architecture (ROA) that is oriented by the service resource use representational state transfer (REST) protocol in order to expand scalability of service. This combination is namely service resource oriented architecture (SROA). SROA can optimize distributing of applications and integrating of services where is implemented to develop the project management software. To realize this model, the software is developed according with framework of Agile model driven development (AMDD) to reduce complexities on the whole stage processing of software development.

Keywords: *AMDD, REST, ROA, SOA, SROA*

1. Introduction

SOA is a design paradigm in software engineering [1] which introduces a concept of services that can represent functionalities of an application. In order to help reducing complexities in software development, this paradigm should be flexible in adapting to a heterogenous and decentralized information system [2]. Furthermore, SOA can help to bridge business process (BP) requirements and IT development processes which are represented as services.

One of the modeling frameworks to implement SOA is Service Oriented Modeling Architecture (SOMA) [3]. This framework offers many benefits, e.g. business and Information Technology (IT) alignment, eliminating redundant services while providing high value of services on re-usability, and relatively easy implementation. SOMA was developed by IBM [4] who also defined the relationship between Enterprise Architecture (EA), SOA, and Business Process Modeling (BPM) [5].

Received January 20, 2012; Revised April 18, 2012; Accepted April 27, 2012

Meanwhile, SOA concept enables a hierarchy of services, namely service taxonomy within which services are categorized according to their granularity level. The hierarchy includes business process, business services, domain services, utility services external services and fundamental service [6]. Among these services, the utility services are those that are implemented by SOAP as a SOA protocol.

In SOAP standard, transaction between a software application with others distributed systems are facilitated by services which are abstracted as webservices which encapsulate methods and receive response outputs. Therefore, services can be independent while can perform business function. However, a large enterprise that has a large number of services may have high complexities in developing and managing all of services which are needed to support business process [7]. For this reason, this research considers *entity service* that holds information of each class entity, i.e. application resources. The architecture model to support this new service is *Resource Oriented Architecture* (ROA) [8].

ROA enables creation of services by extracting resources from all of class entities in service directory, instead of creating a service for each application resources [9]. This creation is mediated with a mediator class which is mapped into a Uniform Resource Locator (URL) address and published as an XML, namely *Representational State Transfer* (REST) [10]. The REST is a dynamic language which is able to cover information for an application at a minimum effort. This standard has an HTTP 1.1 header which is referred as RESTful that has four methods: GET, POST, PUT, and DELETE with a simple parameter, e.g. *id*. Those methods represent a standard function of CRUD.

The combination of SOAP and RESTful [11] can improve system performance by addressing all services requirements which are incorporated within service taxonomy. This results in an adaptive architecture, namely Service Resource Oriented Architecture (SROA) which aims at improving flexibility, effectiveness and efficiency of services. For this reason, the design principles of SOA are applied, including *standarized service contracts, service loose coupling, service abstraction, service reusability, service autonomy, service statelessness, service discoverability,* and *service composability* [1].

By applying these principles, the quality of services can be evaluated through a case study in software development. This research uses project management as a case study which is comprised of nine domains according to a standard of project management, namely PMBOK 4.0. These domains cover a highly complex business process, including *integration, scope, time, cost, quality, HRM, communication, procurement, and risk* [12]. For this reason, the design and implementation of the software for project management requires an agile framework in order to meet all business requirements over services. This framework is referred as Agile Model Driven Design (AMDD) [13]. By using this framework, top level managements can determine their business goals and governance by modeling business process based on Business Process Driven (BPD) [14]. This model is implemented by developing components which are incorporated within software artifacts according to Domain Driven Design (DDD) [15].

2. Research Method

2.1. Define the Taxonomy Service Hierarchy of SROA

Service Resource Oriented Architecture (SROA) is having capability to transform business process (BP) into a hierarchy of services, namely service taxonomy. This model is developed according to granularity of services. Based on this service taxonomy, functional and non-functional requirements for project management software can be structured. We present Table 1 to represent the hierarchy of service taxonomy for the software. We also present a metamodel based on the hierarchy as depicted in Figure 1. This metamodel is developed by elaborating Rosen's model [6] through an addition of *entity service* level by considering Resource Oriented Architecture (ROA). This new architecture model can be can be compared with other models which were developed by Kulkarni [16], Xiaofeng [17], or Xiwei [18]. In addition, different case studies can also be considred [19] to enrich the implementation of SROA in business practice.

2.2. AMDD Modeling

The Agile Model Driven Design (AMDD) framework that is implemented in this research based on the modeling architecture of "*4+1 view architecture"* that was described by Philippe

Kruchten [20]. This modeling architecture consists of *usecase view, development view, logical view, process view* and *physical view*. These views can be associated with the hierarchy of service taxonomy, Unified Modeling Language (UML) standard as depicted in Table 2. This structure can be compared with other conceptual models [21].

Table 1. Hierarchy of service taxonomy

Taxonomy of Service	**Definition**	**Implementation of service in project management**
Business Process (BP)	BP represents a very coarse-grained service that is focused on inputs, outputs and processes of the whole application functionalities which are developed in order to achieve business goals. Therefore, it is not necessarly focusing on what and how it would be run in a system. The granularity level of this service depends on the size of BP.	*Initiating, planning, executing, controlling,* and *closing.*
Business Service (BS)	BS represents an activity which is inherent in a specific business process. BS is comprised of service components which are aggegrated from one or more services in order to meet the objectives of BP.	PMBOK 4.0 classifies 42 business services, including: *project charter, stakeholder identification, project management plan, Work Breakdown Structure (WBS) creation, activity definition;* and sub business services, including: *project initiating, contract appointing,* and *payment staging for project charter process*.
Domain Service (DS)	DS represents a medium-grained service into which each component of BS is grouped and refined. This classification is based on information of BS and the relationship with other BS in DS.	PMBOK 4.0 classifies nine business services, including: *integration, scope, time, cost, quality, HRM, communication, procurement, and risk.*
Entity Service (ES)	ES represents a fine-grained service which is contained within each entity in a DS. This service consists of all classes and attributes which are exposed as a service resource by using RESTful.	*Project, contract, payment, location,* and *closing.*
Utility Service (US)	US represents a very fine-grained service with a specific-independent function. A business logical layer is thus needed within BS which can be exposed as a method by using SOAP.	Examples of methods: *get Payment Complete, get Workbase Complete, get Total Cost, get Income,* and *get Out Come*.
Integration Service (IS)	IS represents a service component which is to invoke and expose services from and into an application.	*RESTful, SOAP, XML, JSON, WClient, REST-API client,* etc.
External Service (XS)	XS is supplied by vendors or external partners in order to meet the demand of services over a web application.	*Googlemap, Gmail,* and *OpenId* are amongst the most popular provider for an open service on the internet.
Foundation Service (FS)	FS represents infrastructure technology of services which independently support a system. However, FS is not bassociated with overall business service.	*Java, Grails, Hybernate, Apache tomcat, Jquery JS Framework*

Table 2. The AMDD modeling

Architecture view	**Hirarki service**	**View Model**	**Audience**	**AMDD**
Usecase view	Business Service	Usecase, swimlane, BPMN	° Product Manager ° Business Modeller ° Requirement Analyst	BPD
Development view	Domain Service	Package Diagram	° System Architect ° OO Analyst	
Logical view	Entity Service	Class Diagram	° Webservice programmer ° DB desainer ° Soft. Programmer	DDD
	Utility Service		° Web designer	
Process view	Integration Service	Sequence Diagram	° System Architect	
Physical view	Foundation Service	Deployment Diagram	° Network administrator ° Security Analyst	

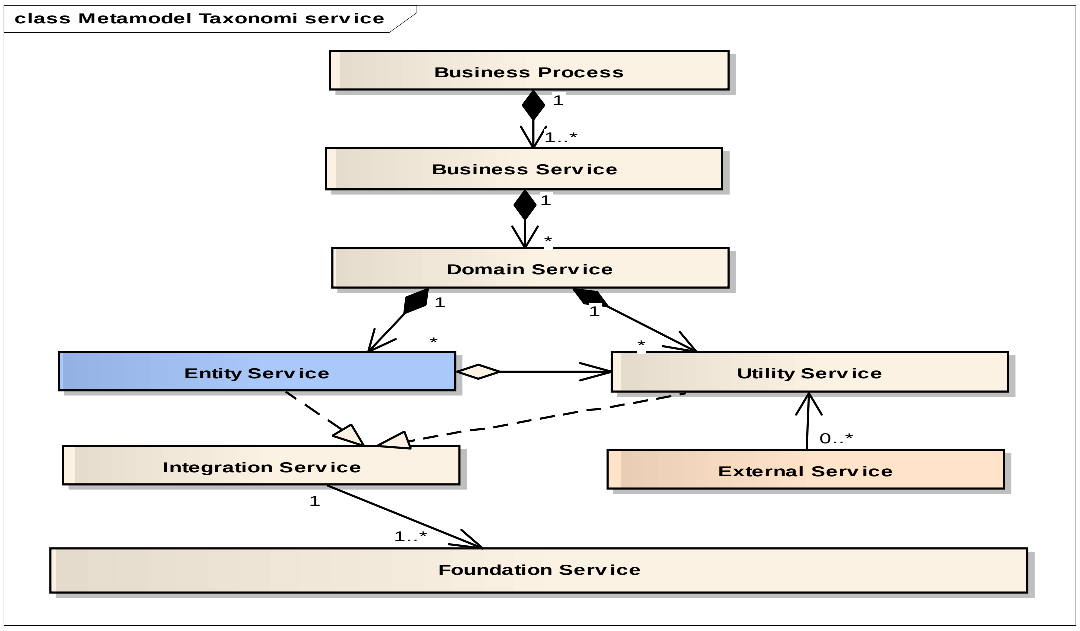


Figure 1. Metamodel hierarcy of the SROA taxonomy

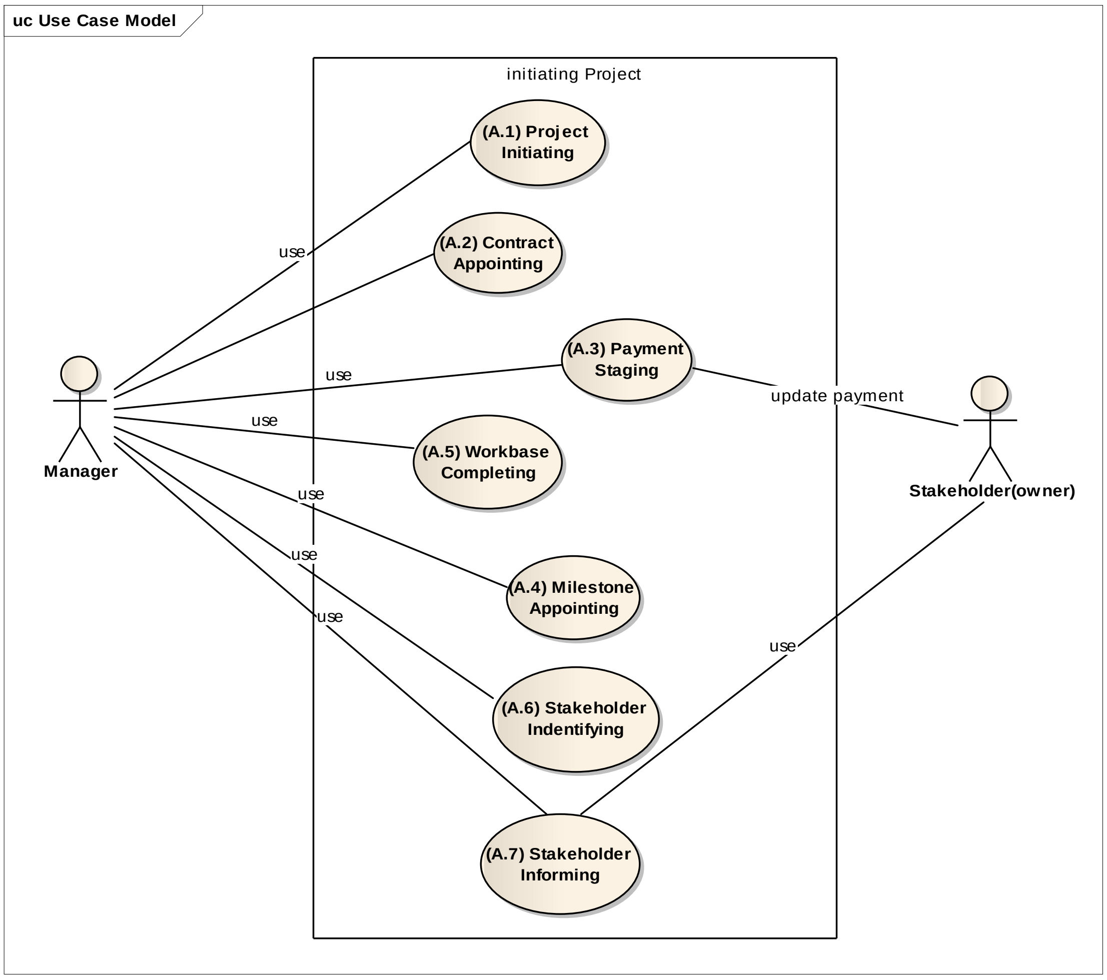

uc Use Case Model
initiating Project
(A.1) Project Initiating
(A.2) Contract Appointing
(A.3) Payment Staging
(A.5) Workbase Completing
(A.4) Milestone Appointing
(A.6) Stakeholder Indentifying
(A.7) Stakeholder Informing
Manager
Stakeholder(owner)
use
update payment

Figure 2. Usecase diagram of BP initiating project

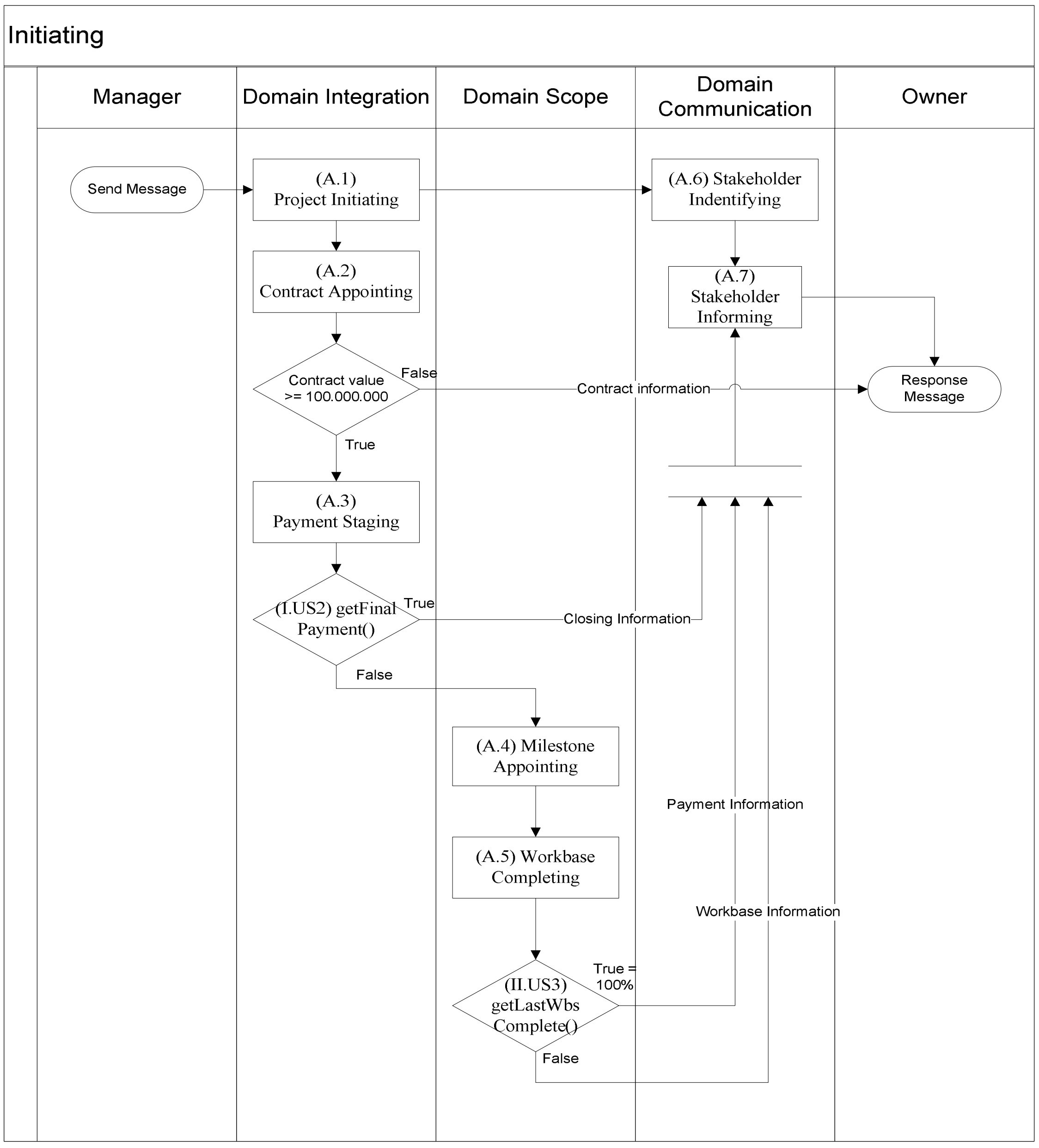


Figure 3. Swimlane diagram of BP initiating project

Rational Unified Process (RUP) in AMDD follows the Software development Life Cycle (SDLC) by applying the iteratively phase role [13], within which each design process runs in parallel with the implementation, the deployment and the testing in order to create a software artifact and services of an application. Meanwhile, in the Business Process Driven (BPD) modeling, the usecase views can be divided into three models, which are: *usecase, swimlane* and *Business Process Modeling Notation (BPMN) diagram*. A usecase diagram represents a basic of business processes which are then detailed into a swimlane diagram which incorporates business proceses and logical business process. This behavioral diagram represents conceptualization of business practice. The example of usecase diagram and swimlane diagram for the business process of initiating project is depicted in Figure 2 and Figure 3, respectively.

According to both diagrams of usecase and swimlane, an executable model of BPMN can be developed. This model covers processes and logics which can be generated into webservices of Business Process Execution Langunge (BPEL) that is deployed in Enterprise

Service Bus (ESB). We develop the BPMN diagram and execute the BPEL within *Intalio BPMS* software as depicted in Figure 4. This figure shows an orchestration process of services which are built from a composed of services based on REST and SOAP, including *Project charter, Contract, Workbase, Milestone, and Payment* within REST, and *Workbasecomplete* that is a service method of SOAP.

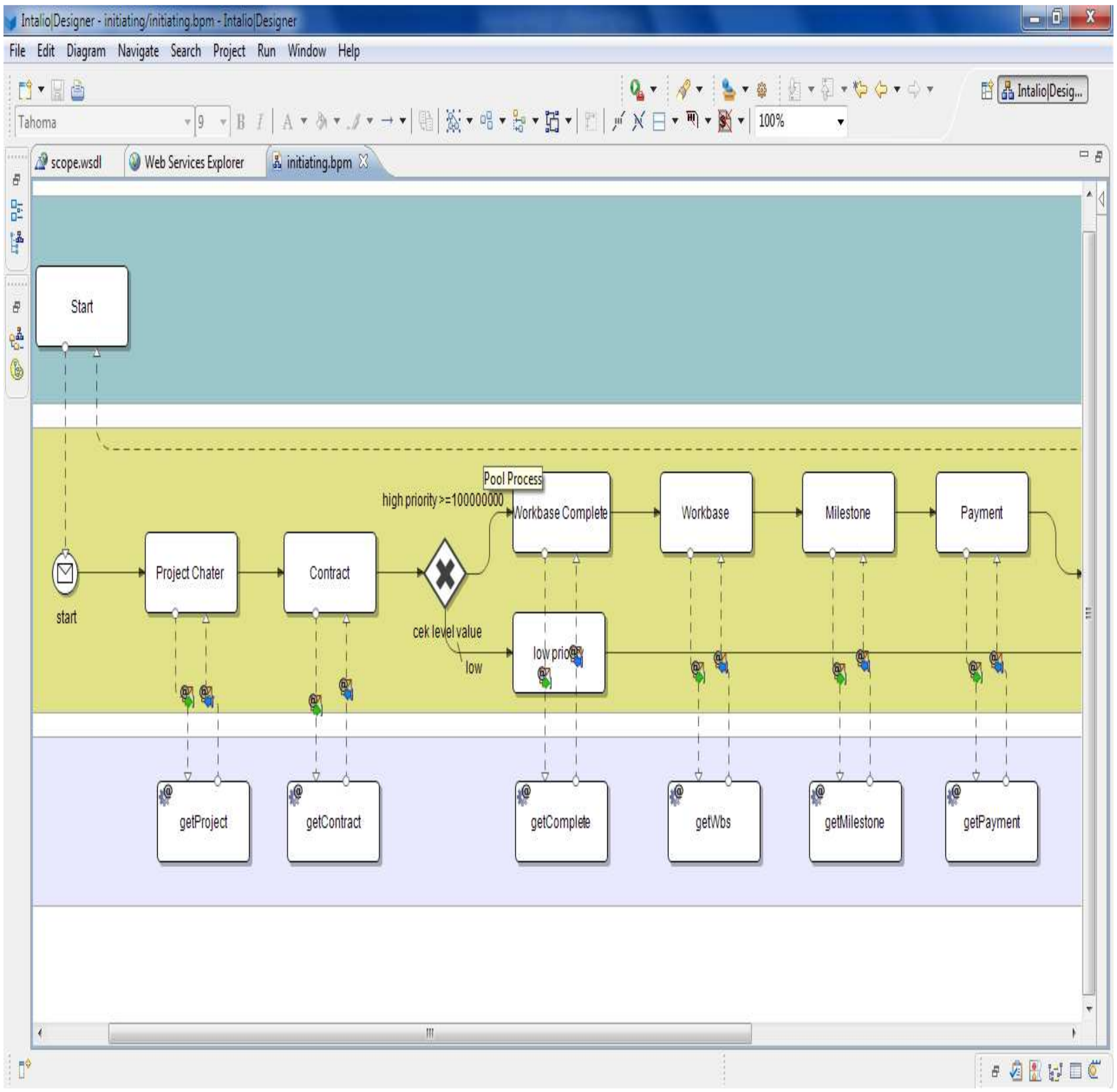


Figure 4. BPMN diagram of BP Initiating project in Intalio BPMS

Next step in the DDD process group is a *bottom up processing* which is intended to build applications and services according to business process requirements. The DDD has four models, which are*: package, class, sequence and deployment diagram*. In this research, we focus on the two models, which are *design package diagram* and *class diagram.* All of DDD models are developed based on Object Oriented Analyze and Design (OOAD) [22].

Package diagram represents the composition and decomposition of applications and services inside the domain services. Each of domain services is deployed independently and therefore it employs loose-coupling principle which enables services to be distributed in different infrastructure environments. Entity and utility services are created by applying domain analysis [23] which considers that tight-coupled services are located in the same domain package which is separated with other domains.

The DDD framework organizes the design of a class in order to generate software artifacts by applying a specific modern platform, e.g. Ruby on Rails (Ruby), Symphony (PHP), Django (Python), Naked Object (Java), Grails (Groovy) or C# (.Net) framework. This research implements the Grails framework of technology platform by applying Java-based and dynamic language of Groovy. Therefore, there are four elements should be considered, including: *domain model, controller, view,* and *service.* The REST is developed by applying controller as a mediator and the SOAP service methods are developed by using Xfire service provider. We developed an infrastructure model to implement the overal processes of BPD and DDD within SROA as depicted in Figure 5.

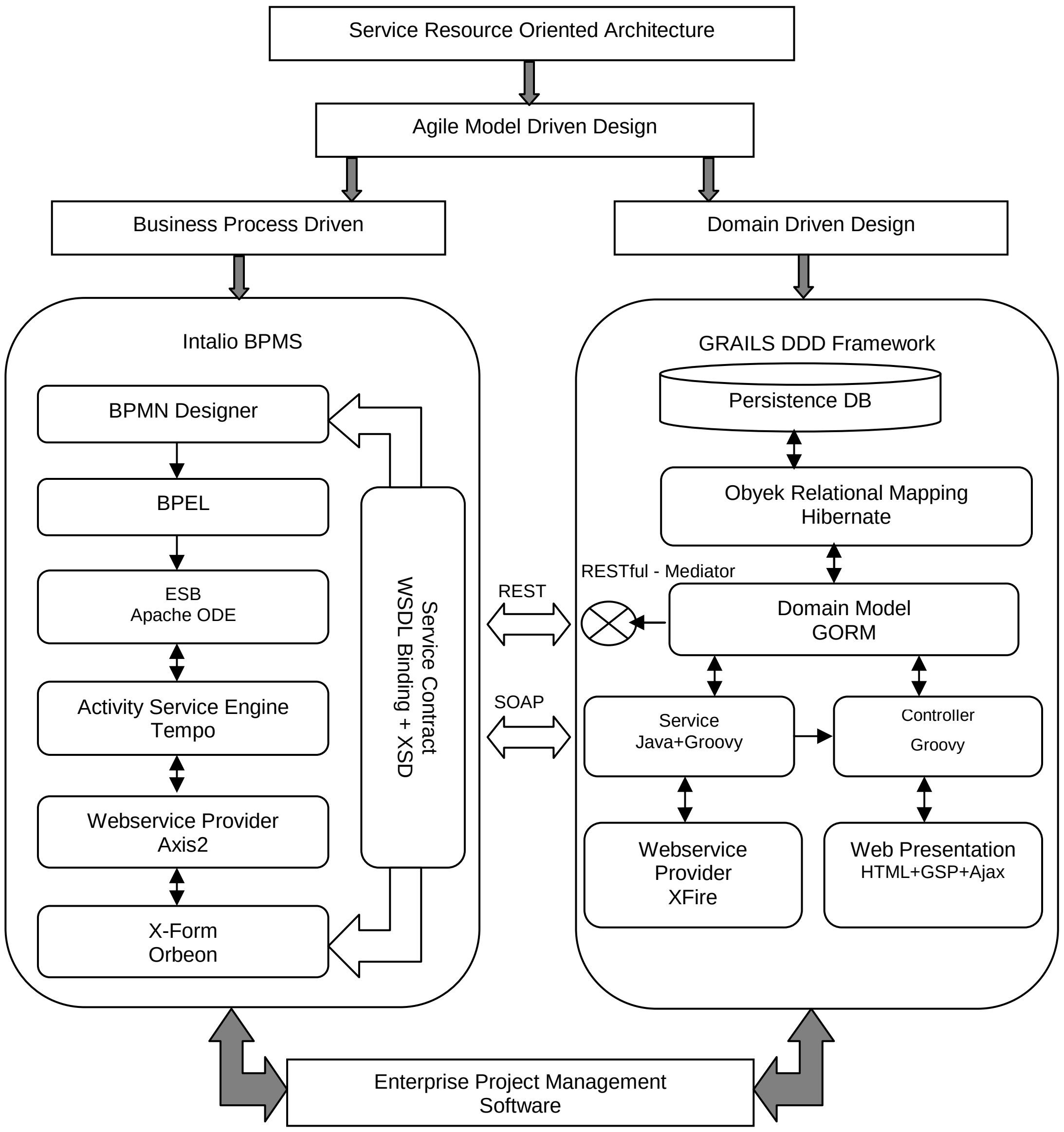


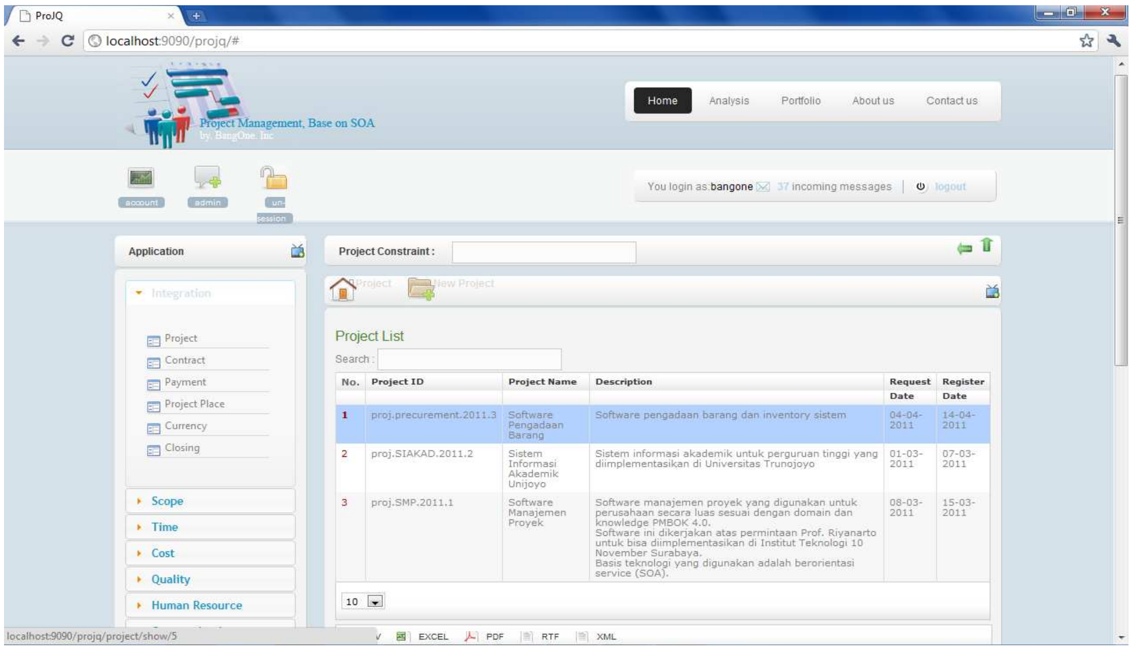
Figure 5. Infrastructure and overall processes of BPD and DDD within SROA

3. Results and Analysis

This research aims at presenting two significant results. The first result is to contribute to the design and implementation of an integrated and distributed system for application and services; while the second result is related to improve system quality by ensuring that both design and implementation follow the design principles of Service Oriented Architecture (SOA).

3.1. Integrated System

Integrated system includes two levels of integration, which are at the top-tier and at the middle-tier. Integration at the top-tier represents application level within which the web presentation can be accessed directly from other applications that are included within intra or inter server-infrastructure. Web-2.0 technology is also applied based on Asynchronous JavaScript and XMLHTTP (AJAX) in order to be able to send a message and get a response. The web applications which are resulted from integration at the top tier are presented on Figure 6. This figure shows web applications which are accessed from a web browser from an integrated system.

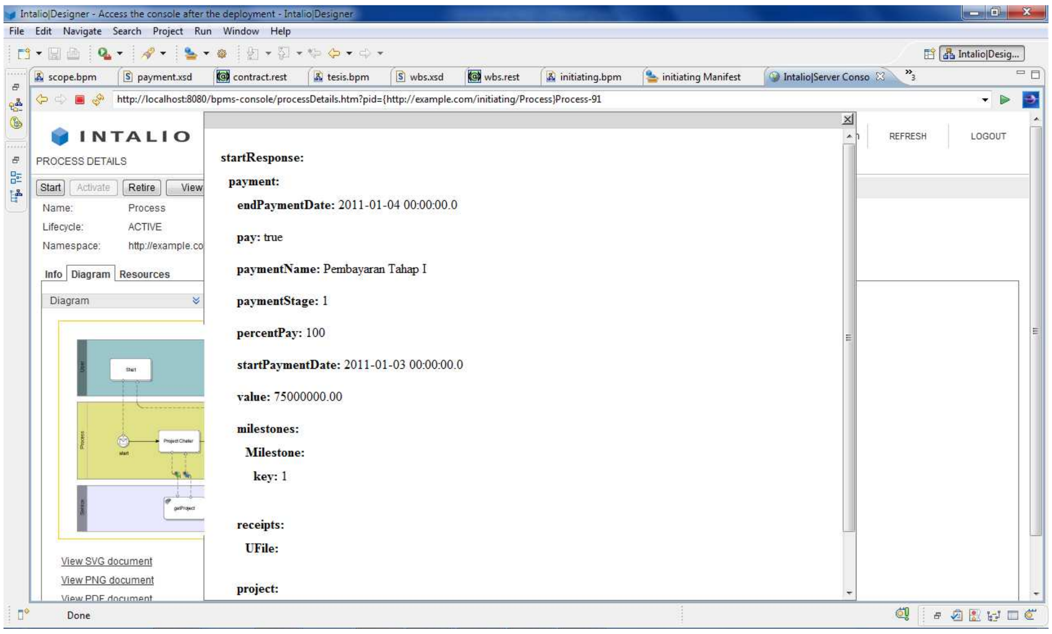


Project List

Search :

No.	Project ID	Project Name	Description	Request Date	Register Date
1	proj.precurement.2011.3	Software Pengadaan Barang	Software pengadaan barang dan inventory sistem	04-04-2011	14-04-2011
2	proj.SIAKAD.2011.2	Sistem Informasi Akademik Unijoyo	Sistem informasi akademik untuk perguruan tinggi yang diimplementasikan di Universitas Trunojoyo	01-03-2011	07-03-2011
3	proj.SMP.2011.1	Software Manajemen Proyek	Software manajemen proyek yang digunakan untuk perusahaan secara luas sesuai dengan domain dan knowledge PMBOK 4.0. Software ini dikerjakan atas permintaan Prof. Riyanarto untuk bisa diimplementasikan di Institut Teknologi 10 November Surabaya. Basis teknologi yang digunakan adalah berorientasi service (SOA).	08-03-2011	15-03-2011

Figure 6. The top-tier integrated system



```
startResponse:
  payment:
    endPaymentDate: 2011-01-04 00:00:00.0

    pay: true

    paymentName: Pembayaran Tahap I

    paymentStage: 1

    percentPay: 100

    startPaymentDate: 2011-01-03 00:00:00.0

    value: 75000000.00

    milestones:
      Milestone:
        key: 1


    receipts:
      UFile:

    project:
```

Figure 7. The middle-tier integrated system

Meanwhile, integration at the middle layer represents accessing data information from the RESTful and logic from the SOAP service between inter server infrastructure. This integrated system is of the main topic discussion in the resuarch. An integrated system in a complex data information and logic in the middle tier environment is depicted in Figure 7. This figure shows the development of the system in Intalio BPMS.

3.2. Distributed System

By applying the design of domain analysis, each domain service can be easily deployed in different (infrastructure) servers, and can be distributed to other locations. Within this distributed system, high utility domain services with a large number of entities can be run effectively and efficiently during implementation phase.

Meanwhile, in the enterprise side, a distributed system is usually performed within an organizational domain. For instance, Human Resource Management (HRM) domain service in project management can be implemented in HRD office which is separately from an integrated system of an enterprise that is implemented in central office. Although a domain service is implemented in different infrastructure and location, it can communicate with other domain services which are loosely-coupled toward one to another.

3.3. Analysis and Discuss

In order to improve system quality in the research, we evaluate to what extent both integrated system and distributed system which are developed in the research, conform to the design principles of Service Oriented Architecture (SOA) that was proposed by Erl [1]. The design principles of SOA are *standardized service contract, service loose coupling, service abstraction, service reusability, service autonomy, service statelessness, service discovery,* and *service composability*.

The SROA incorporates webservices in order to encapsulate data and methods of every services related to project management. Webservices employ a standard of XML-based document which is published by W3C standardization. The XML-based documents include WSDL, XSD, and composite services which are developed by using BPEL. Since webservices within SROA consider a certain standardized contract, we can conclude that s*tandardized service contract* principle is applied.

Within the distributed system, a class in a domain service interacts tightly toward other classes while being separated from other domain services. For this reason, we can conclude that *service loose coupling* principle is applied.

Meanwhile, SOAP conforms to *service abstraction* principle by considering that each method from this protocol is wrapped inside WSDL document wherein security and encryption of WS-* can be added in order to prevent unauthorized accesses. However, service providers on REST do not conform to *service abstraction* principle. This is due to security authorization is performed on URL-based by delivering a certain parameters, e.g. user and password, and encryption is also performed on HTTPS-based protocol.

Services can be composed and decomposed according to BP goals through BPMN and therefore *service reusability* principle is applied. In order to increase service reusability within a service inventory, WSDL which incorporates service contracts, can be created for each REST and SOAP within BPMS server. Following this, reusability of services can be determined by calculating frequency of services dependency as shown in Table 3.

According to Table 3, frequency of services can be calculated. If an entity service has a higher instance in domain services, it has a higher functionality in terms of highly consumed services. It also has higher reusability that utility service. For this reason, we can prove that REST protocol has a higher reusability than SOAP protocol.

In terms of service granularity, consumption of services produces two types of composite services, which are *coarse grained services* that are orchestrated in business service, and *fine grained services* that are orchestrated across entity services or utility services. Table 4 shows characteristic of services based on service granularity.

The distributed system can be built in different infrastructures, e.g. different webserver and database server. For this reason, we can conclude that *service autonomy* principle is applied. Furthermore, application and services that have been deployed in a package ".War" can be installed in various types of webservers, e.g. Tomcat, GlassFish, JBos and other java-based servers. With regard to ORM, databases are supported with Java JDBC library, including

MySQL, PostgresSQL, Oracle, and other JavaDBs. Similarly, application and services can also be consumed by other programming languages.

Table 3. Service dependency analysis for project management software

Service	Numbers of Instance	Numbers of service consumption
(I.ES1) Project	4	8
(III.ES2) Activity	6	6
(VI.ES3) Employee	5	3
(VII.ES3) Stakeholder	5	2
(I.ES3) Payment	2	2
(II.ES6) Wbs	3	1
(VIII.ES2) Material Resource	3	1
(VI.ES4Salary	2	2
(III.ES3) getActiveManDay()		1
(III.ES4) getOverManDay()		1
(III.US3) getActiveDay ByDuration()		1
(III.US4)getOverDay ByDuration()		1

Table 4. Characteristic of services according to service granularity

Granularity of service	Characteristic
Coarse grained services	• Complex business logic • Large data volume • Less used services • Weak relations between services • Less flexibility of services
Fine grained services	° Simple business logic ° Small data volume ° More used services ° Strong relations between services ° Better flexibility of services

The use of HTTP and HTTPS protocol shows that *service statelessness* principle is applied. Every web connection is established by considering fully statelessnes of fine grained services. Meanwhile, standardization of documentation in WSDL incorporates descriptions and process of services in service inventory. This shows that *service discovery* principle is applied by which service registry is created in order identify and locate services which are incorporated within service inventory during BPMN orchestration.

By applying BPD, each service can be composed and decomposed according to business objectives. For this reason, we can conclude that *service composability* principle is applied. Moreover, the DDD supports production and management of application and services via entity classes. Therefore, any changes related to application and services can be performed in a relatively easy manner with fewer risks in order to support software evolution.

4. Conclusion

From this study, we can prove that SROA can employ various types of service protocol, including SOAP and REST. Following this, SOA concept can be regarded as neutral from any interventions which might arise from implementation and technology sides. Therefore, SROA implementation can be established in compliance with the design principles of SOA, particularly with regard to service reusability principle. SROA modeling by employing AMDD can help to reduce the complexity in SOA implementation in a complex business process, e.g. project management system. Furthermore, other systems can also be implemented further by considering the result from this study, i.e. Geographic Information System [24].

Within software enginering field, SROA can contribute to the development of an IT system with minimum efforts in a scalable enterprise, while considering an open source platform in order to save costs. Moreover, this study also contributes to scientific communities by providing a base research for semantic ontology as a light searching engine which is developed at low-costs [25].

References

[1] Erl Thomas. SOA: Principles of Service Design. Boston: Prentice Hall. 2007; 37: 71 – 75.
[2] Daniel Minoli. Enterprise Architecture A to Z. Boca Raton: Auerbach Publications. 2008: 408, 415.
[3] Ali Arsanjani, Abdul Allam**.** *Service-Oriented Modeling and Architecture for Realization of an SOA*. IEEE International Conference on Services Computing. Washington, DC. 2006: 521.
[4] Iyan Supriyana. Model Arsitektur Bisnis. Sistem Informasi dan Teknologi di BAKOSURTANAL berbasis TOGAF. *TELKOMNIKA.* 2010; 8(1): 17-24.
[5] Claus Torp Jensen, eds. Leveraging *SOA. BPM and EA for Strategic Business and IT Alignment.* IBM. December 2008.
[6] Rosen Mike, Boris Lublinsky. Service Oriented Architecture and Design Strategis. Indianapolis: Addison-Wesley. 2008: 93-94
[7] Cesare Pautasso, Olaf Zimmermann. *RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision*. The 17th International Conference on World Wide Web. Beijing. 2008.
[8] R.T. Fielding. Architectural Styles and the Design of Network-Based Software Architectures. Doctoral Dissertation. Dept. of Computer Science. Irvine: Univ. of California. 2000.
[9] Erich Gamma, Richard Helm. Design Patterns Elements of Reusable Object-Oriented Software. Zurich: Addison-Wesley. 1997: 305.
[10] Sujit Kumar Chakrabarti, Prashant Kumar. Test-the-REST: An Approach to Testing RESTful Web-Services. *IEEE Computation World*. 2009.
[11] Cesare Pautasso, Olaf Zimmermann. *RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision*, the 17th international conference on World Wide Web. Beijing. 2008.
[12] R.Duncan Wiliam. A Guide to the Project Management Body of Knowledge PMBOK 4. Atlanta: PMI. 2008.
[13] Ambler Scott, Agile Model Driven Development. Cambridge: University Press. 2004: Chapter 4.
[14] Yinsheng Li a, Jianping Shen. Multi-Model Driven Collaborative Development Platform for Service-Oriented e-Business Systems. *Science Direct: Advanced Engineering Informatics.* 2008; 22: 328–339.
[15] Evans Eric. Domain Driven Design. Boston: Addison-Wesley. 2003: Chapter 4.
[16] Naveen Kulkarni, Vishal Dwivedi. *The Role of Service Granularity in a Successful SOA Realization - A Case Study*. IEEE Congress on Services 2008 - Part I.
[17] Xiaofeng Guo, Jianjing Shen. *On Software Development Based on SOA and ROA*. Chinese Control and Decision Conference. China 2010.
[18] Xiwei Xu, Liming Zhu, *Resource-Oriented Architecture for Business Processes*. 15th Asia-Pacific Software Engineering Conference. 2008.
[19] J. Wolferta, C.N. Verdouwa. Organizing Information Integration in Agri-food, A Method Based on a Service-Oriented Architecture and Living Lab. Approach. *Science Direct: Computers and Electronics in Agriculture*. 2010; 70: 389–405.
[20] Kruchten Philipe. Architectural Blueprints The "4+1" View. *IEEE.* 1995: 42-50.
[21] Jyhjong Lin. A Conceptual Model for Negotiating in Service-Oriented Environments. *Science Direct: Information Processing Letters*. 2008; 108: 192–203.
[22] Grady Booch. Object Oriented Analysis and Design with Applications Third Edition. Boston: Addison-Wesley. 2007.
[23] Dae-Kyoo Kim, Charbel El Khawand. An Approach to Precisely Specifying the Problem Domain of Design Patterns. *Science Direct: Journal of Visual Languages and Computing*. 2007; 18: 560–591.
[24] Nasaruddin, Khairul Munadi, M. Dirhamsyah, and Dedi Yuliansyah. A Web-Based Geographic Information System for Aceh Natural Hazards. *TELKOMNIKA.* April 2011; 9(1): 89 – 98.
[25] Paulus Insap Santosa. Cost and Benefit of Information Search using Two Different Strategies*. TELKOMNIKA.* December 2010; 8(3): 195-206.