A Model-Driven Approach for Service Oriented Web 2.0 Mashup Development

José Luis Herrero, Pablo Carmona, Fabiola Lucio
Department of Computer and Telematics Systems Engineering
University of Extremadura
Avda. de Elvas s/n, 06006, Badajoz, Spain
{jherrero,pablo,flucio}@unex.es

***Abstract*— Mashup applications are composed by data or functionality extracted from different sources. With the evolution of web 2.0 and the appearance of AJAX technology and the service-oriented architecture, a new breed of mashup applications for the web has emerged. However, software engineers have to deal with the heterogeneous composition of mashup sources, which increases software development cost and complexity. Therefore, it becomes essential to boost a software development approach that can attenuate these problems. This is the reason why we propose in this paper a model-driven and service-oriented architecture for developing mashup applications. Towards this end, the following tasks have been developed: first a new mashup profile extends UML and introduces mashup concepts at design level, and second, a set of transformation rules has been defined with the aim of generating code semi-automatically. These rules have been classified according to the type of the element (web application, mashup or web service). Finally, a transformation tool parses a UML model, identifies mashup elements, and according to the specified set of rules, generates code.**

Keyword: Mashup, Model-driven architecture, web services, web applications.

I. INTRODUCTION

With the appearance Web 2.0 paradigm and the introduction of new technologies such as Asynchronous JavaScript and XML (AJAX) [1] and web services, new types of applications for the web have emerged. Under the umbrella of this new trend, Rich Internet Applications (RIA) [2] has evolved, and a high degree of interactivity and complexity is achieved by this type of applications. One of the most interesting type of applications that have gained much attention in the Web 2.0 community, is mashups. During the last few years, different types of mashups have been defined: on the one hand, data mashups have the ability to produce new information combining data from different sources, and on the other hand, functional mashups are composed by mashup components that can be assembled and combined in order to build the final application.

The service-oriented architecture (SOA) is defined as ‘‘a set of components, which can be invoked, and whose interface descriptions can be published and discovered’’ [3]. One implementation of SOA applications is made possible through the realization of Web Services, which are implemented in eXtended Markup Language (XML), and described through the Web Services Description Language (WSDL), while the simple object access protocol (SOAP) is the main communication protocol adopted.

A mashup is a program that manipulates and composes existing data sources or functionality to create a new piece of data or service that can be plugged into a web application [4]. SOA provides a solid foundation for mashup development. However we argue that the underneath technology that supports mashup applications requires software engineers to locate and combine mashup sources, which implies an increase in the complexity degree, and in particular in the development costs of this type of applications.

In order to solve these problems, we propose in this paper a Model Driven Architecture (MDA) [5]. It simplifies modeling, design, implementation, and integration of applications by defining software mainly at the model level. The primary goals of MDA are portability, interoperability, and reusability through architectural separation of concerns [6], making product development more cost efficient by increasing automation in software development [7].

The objective of this paper is to propose a model-driven architecture to develop mashup applications, and this task has been achieved at the following levels: first, a new profile that includes specialized concepts from the mashup domain has been defined, and then, a transformation model generates mashup applications from the UML profile. This paper is organized as follows: first, Section II explains the motivation and the background of this work. Then, a UML mashup profile is proposed in Section III, and also an example is presented. Next, Section IV describes the transformation model to generate mashup applications from a UML design, and finally, conclusions are clarified in Section V.

II. MOTIVATION AND RELATED WORKS

Two different taxonomies can be mentioned when classifying mashups applications; the first one is focused on the place the composition mechanism takes place [8], and the other one studies the type of the combined elements [9].

This classification brings us the motivation to propose a novel architecture that captures the essence of both alternatives. On the one hand, mashups can be composed at server or client side, while on the other hand, the composition can be focused on data or functional components. As far as we know, this is a novel approach to deal with mashup applications development.

 ISBN: 978-1-61208-280-6

A. Mashup background

Different tools have been proposed to build mashup applications for the web. Dapper [10] is a drag and drop tool that allows users to select contents in several web pages that will be composed in order to generate a new representation. Yahoo Pipes [11] is a composition tool to aggregate and manipulate mashup content from the web. This tool provides a visual editor to create pipes from different sources and provides rules to compose the content. DERI [12] is inspired by Yahoo's Pipes, and proposes an engine and graphical environment for general web data transformations and mashup. Serena Business Manager [13] contains a visual workflow editor to define mashups and presents an online marketplace where mashups can be exchanged. And finally, IBM Mashups Center [14] is a mashup platform that supports rapid assembly of dynamic web applications, enabling the creation, sharing, and discovery of reusable application building blocks that can be easily assembled into new applications.

Trends in the mashup community are currently working in different areas, Meditskos et al. [15] proposes an approach for developing mashups with semantic mashup discovery capabilities has been proposed, and an extension of OWL-S advertisements has been defined. A novel service oriented architecture is presented in [16] which addresses reusability and integration needs for building mashup applications, identifying the essential architecture patterns for designing mashups. A different work [17] approaches to represent domain concepts at the mashup composition, and defines an architecture to assist experts in the process of introducing domain concepts in the composition mashup level. Another interesting area [18] studies the privacy problem, which deals with the dynamic data integration from different mashup sources in the presence of privacy concerns, and proposes a service-oriented architecture for privacy-preserving data mashup.

B. MDA background

In the field of MDA, there is a consensus about the benefits that this technology offers for software development: a reduction of sensitivity to the inevitable changes that affect a software system [19], a reduction of cost and complexity [20], and an increase of abstraction [21]. An interesting analysis about the existing problems in the field of web engineering and how they can be solved by model-driven development approaches is presented in [22], which identifies the problems encountered in the development process of web applications such as their dependence on the HTTP protocol, compatibility issues due to the heterogeneity of web browsers, and the lack of performance because of the increase in the latency degree

Different proposals extend web engineering methods for developing web applications. Fraternali et al. [23] present a survey of existing web engineering methods to develop this type of applications. The Object-Oriented Hypermedia Design Model (OOHDM) [24] uses abstraction and composition mechanisms in an object-oriented framework to, on one hand, allow a concise description of complex information items, and on the other hand, allow the specification of complex navigation patterns and interface transformations. The RUX-Model [25] is a representational model that offers a method for engineering the adaptation of legacy model-based Web 1.0 applications to Web 2.0 UI expectations. An extension of this model is proposed in [26] where a model-driven approach to web application development by combining the UML based Web Engineering (UWE) with the RUX-Method is defined.

The combination of these two trends has been studied in [27]. The article proposes a model-driven mashup development (MDMD) as an approach to develop mashups according to flexible framework (PLEF-Ext) for end-user. A Service-Oriented Model Driven Architecture (ODSOMDA) approach has also been proposed in [28], which involves adding service oriented architecture (SOA) elements into a model-driven architecture to facilitate the construction of mashup applications.

III. BUILDING MASHUPS WITH A MDA APPROACH

MDA comprises of three main layers: a) the Computation Independent Model (CIM) is the top layer and represents an abstract model of the system, abstracting from technical details, b) the Platform Independent Model (PIM) defines the conceptual model based on visual diagrams, use-case diagrams and metadata, and c) the Platform Specific Model (PSM) that represents the system from a specific implementation platform viewpoint. In order to achieve an implementation of the system, the PIM must be transformed into PSM, and with this aim, an automatic code generator must guide this process. The most important advantage of this approach is that PSMs can be automatic generated from a PIM model according to a set of transformation rules.

This paper proposes a MDA approach to develop mashup applications for the web. With this aim, a new UML profile is proposed (Figure 2), and a set of a transformation rules has been developed in order to generate mashup applications according to the WCF framework [29].

A. Mashup profile

The main element of this profile is the <<Mashup Application>> stereotype that represents a mashup application, which includes the name and composition type (client or server). A mashup application can be attached to a web application, represented by the <<Web Application>> stereotype. A mashup resource (<<Mashup resource>>) specifies a data or software element that can be combined in order to create new information or add new functionality. Mashup resources are classified as an enterprise mashup (<<Enterprise mashup>>) or data mashup (<<Data Mashup>>). <<Mashup component>>, <<web API>> and <<Widget>> stereotypes represent the different types of enterprise mashups.

Data mashups can be categorized according to data integration patterns: the <<Mashup Pipe>> stereotype defines a set of pipes or filters that must be applied to the information in order to obtain the final output.

 ISBN: 978-1-61208-280-6

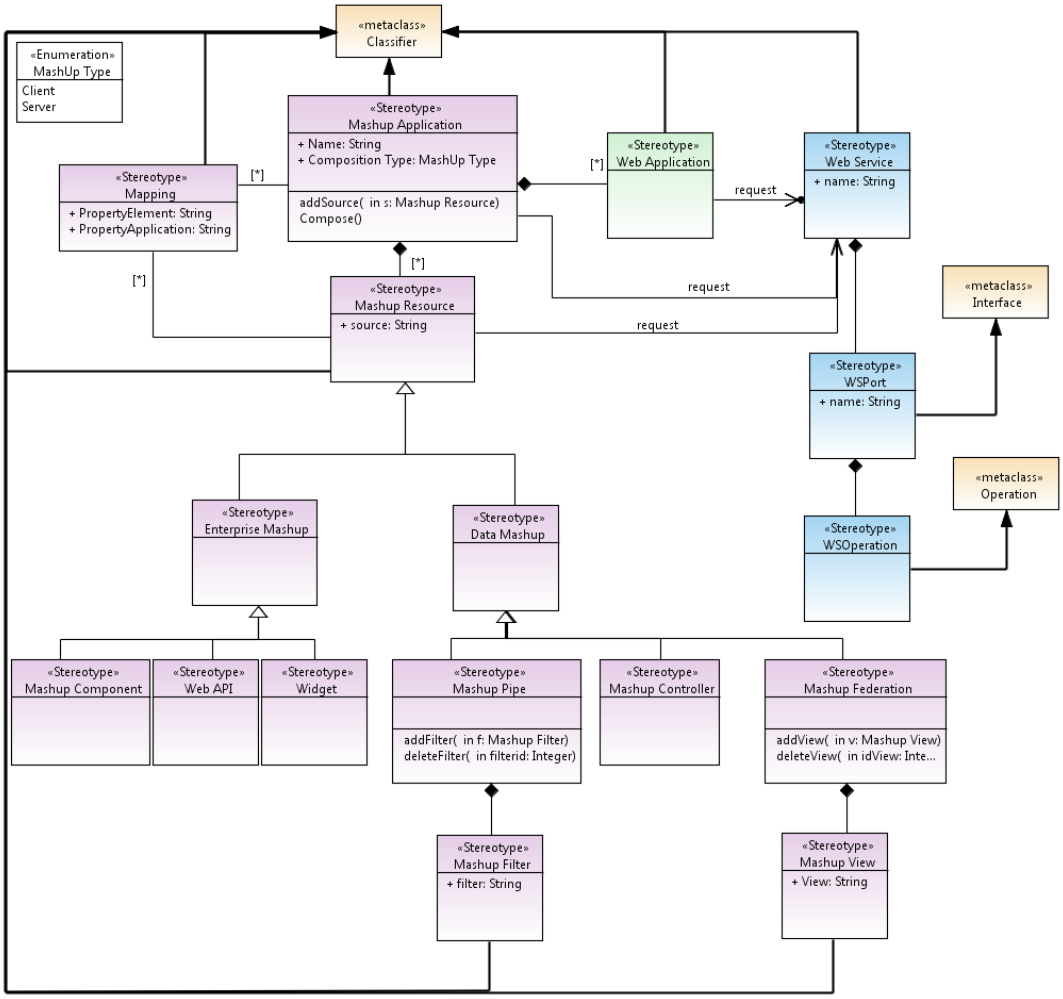

Figure 1. UML mashup profile

The <<Mashup DataFederation>> stereotype specifies different views of the same data, while the <<Mashup Controller>> stereotype describes how the data is rendered. The mapping between a mashup application and its sources is specified by the <<Mapping>> stereotype.

Finally, services are referenced by the <<Web Service>>, <<WSPort>> and <<WSOperation>> stereotypes according to a previous proposal [30], but avoiding unnecessary stereotypes in order to provide a more compact profile.

B. An illustrative scenario

In order to test the proposed profile, the following scenario is suggested: an ecommerce web portal (Figure 3) allows clients to buy products that can be offered by different providers. Additionally, a set of tools are supported in order to provide useful utilities (currency conversions, a calendar tool and a shopping cart). A payment system is defined and connections with bank payment services are also supplied. Clients, providers and banks are connected with the ecommerce portal using web services technology.

IV. MODEL TRANSFORMATIONS

One of the keys of MDA is the capacity of defining transformations from higher-level models to platform specific models guided by a set of transformation rules. With this aim, a generation tool is proposed in this section. This tool is based on Eclipse platform (Eclipse Juno version), and different plugins have been used in order to support UML development (Papyrus Project) and code generation (Acceleo Project). A library has also been developed with the aim of requesting elements from a UML design, and extract information about the model.

The transformation process parses a UML model, next identifies all the elements tagged with the new stereotypes defined in the mashup profile, then extracts all the information about them, and finally generates the specific code according to the Web Component Framework (WCF).

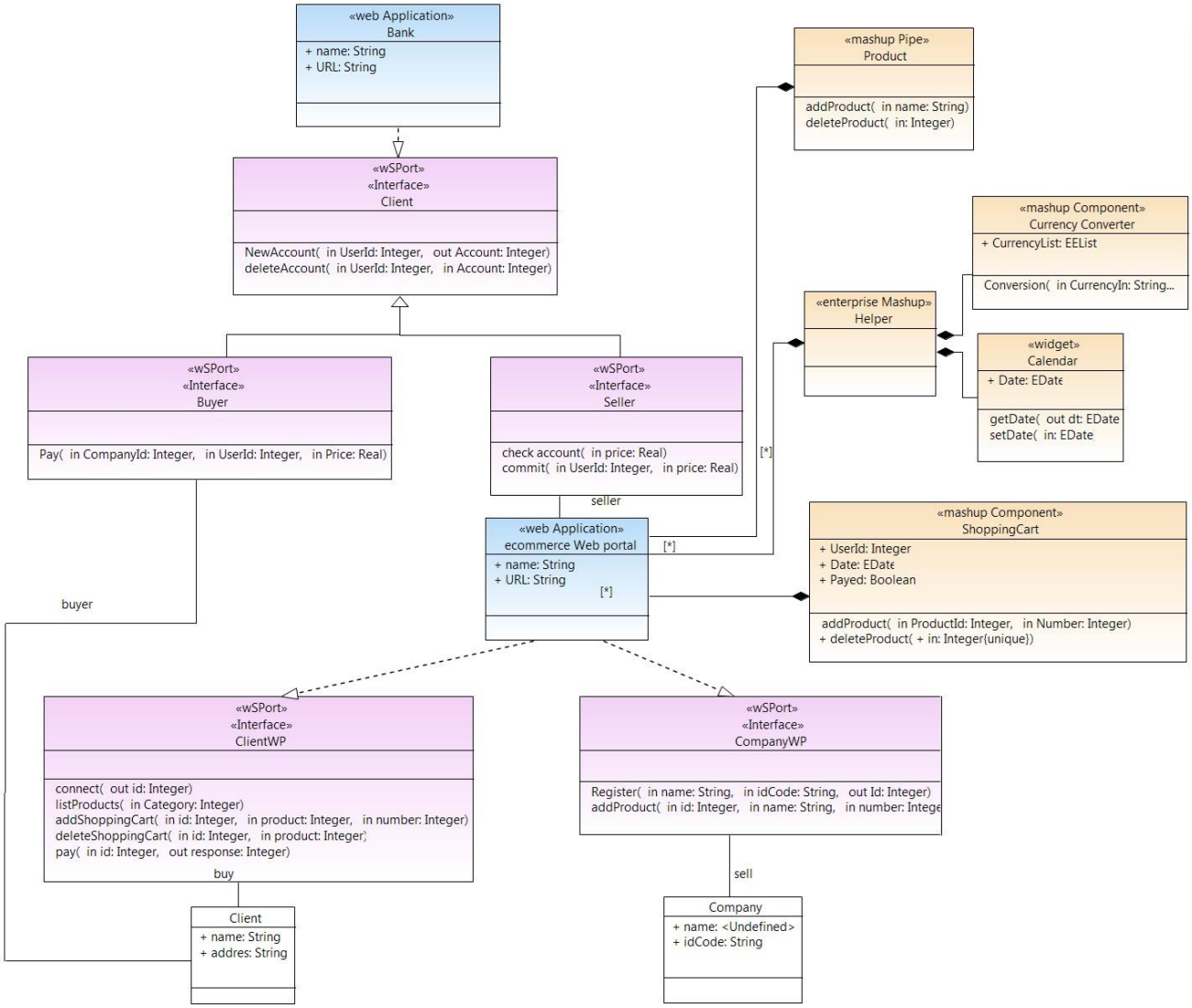

Figure 2. Ecommerce web portal.

In order to perform the transformation, the following definitions have been assumed.

TABLE I. DEFINITIONS

Definition 1	Let *M* be the model designed according to the mashup profile
Definition 2	Let *Cm* be an element extracted from the model.
Definition 3	Let *As* be an Association between two *Cm* elements,.
Definition 4	Let *End* be a target element of an Association *As*.
Definition 5	Let *Ma* be a model element tagged as a <<Mashup Application>>.
Definition 6	Let *WSP* be a model element tagged as a <<WSPort>>.

The set of rules proposed are classified according to the type of element generated: web application, mashup and web service.

1) Web Application rule

This rule searches each *Cm* in the model *M*, tagged as a "Web Application", looks for its associations and checks if a Mashup application is attached. In this case a mashup application is created.

Input: a UML model (M)
Output: generates the definition of a Web application as a collection of mashup applications

```
1  for each Cm ϵ Elements(M) do
2    if (getStereotype(Cm)="Web Application")
3      for each As ϵ Associations(Cm) do
4          if (typeof(As)="Aggregation")
5          for each End ϵ AssociationEnds(As) do
6            if (getStereotype(End)="Mashup Application")
```

```
7                out (“createMashupApplication”)
8                out (“ (“+End.getAttribute(“name”)+”,”)
9                out (End.getAttribute(“type”)+”)”)
10               generateMashupResources(End)
11             end if
12           end for
13       end for
14     end if
15  end for
```

2) Mashup rule

First, the rule checks every mashup resource attached to a mashup application *Ma.* Next, according to the type a specific mashup resource is generated (data or enterprise).

Name: generateMashupResources
Input: mashup Application (Ma)
Output: generates the definition of mashups resources.

```
1     for each As є Associations(Ma) do
2       if (typeof(As)=”Aggregation”)
3         for each End є AssociationEnds(As) do
4           if (End .isSubtypeof(”Mashup Resource”)
5             if (End .isSubtypeof(”Enterprise Mashup”)
6               out(CreateEnterpriseMashup+”(”)
7             else
8               if (End .isSubtypeof(”Data Mashup”)
9                   out(CreateDataMashup+”(”)
10            end if
11          end if
12          out(End.getAttribute(“name”)+”,”)
13          out(End.getAttribute(“Composition Type”)+”,”)
14          out(End.getAttribute(“source”)+”)”)
15          out (“addsource(‘“+Ma.getAttribute(“name”)+”’,’”+
            End.getAttribute(“name”)+”’)”)
16        end if
17      end for
18    end if
19  end for
```

3) Web Services

The last elements considered in this transformation are web services, which are represented using different stereotypes (Web services, WSPort and WSOperation). The transformation from these elements to the Web Services Description Languag is guided by the following rules:

a) Header and definitions rule

The header and the definition part of the web service are generated by this rule. With this aim, each element tagged with the *Web Service* stereotype is located and every *WSPort* element attached is extracted.

Input: UML model (M)
Output: generates the web service header and definition

```
1   for each Cm є Elements(M) do
2     if (getStereotype(Cm)=”Web Service”)
3       out(“<?xml version=’1.0’ encoding=’UTF-8’?>”)
4       out(“ <definitions name=’”+Cm.getAttribute(“name”)
         +”Service’>”)
5       for each As є Associations(Cm) do
6         if (typeof(As)=”Aggregation”)
7           for each End є AssociationEnds(As) do
8             if (getStereotype(End)=”WSPort”)
9               generateMessages(End)
10              out(“</definitions>”)
11            end if
12          end for
13        end if
14      end for
15    end if
16  end for
```

b) Messages rule

The set of request and response messages are extracted from each *WSOperation* attached to a *WSPort* and the WDSL message part is generated.

Name: generateMessages
Input: WSP is a “WsPort” element
Output: generates web service messages

```
1   for each As є Associations(WSP) do
3     if (typeof(As)=”Aggregation”)
4       for each End є AssociationEnds(As) do
5         if (getStereotype(End)= ”WSOperation”)
6           out(“<message name=”+End.getAttribute(“name”)+
            ”Request>”
7           for each Op є Operation do
8             out(Op.getAttribute(“name”)+”Request”)
9             for each P є Params(Op) do
10              if (P.direction=”in”)
11                out(“<part name=”+P.getAttibute(“name”)+
                  ”type=xs:”+P.getAttribute.type+”’/>”
12              end if
13            end for
14            out(“</message>”)
12            out(Op.getAttribute(“name”)+”Response”)
13            for each P є Params(Op) do
14              if (P.direction=”out”)
15                out(“<part name=”+P.getAttibute(“name”)+
                  ”type=xs:”+P.getAttribute.type+”’/>”
16              end if
17            end for
18        end for
19      end if
20    end for
21  end if
22 end for
```

c) Binding rule

This rule generates the binding part of the web service and describes how the service is bound to a SOAP message protocol.

Name: generateBinding
Input: Ws is a “WsPort” element
Output: generates web service binding

```
1   out(“<binding name=’+Ws.getAttribute(“name”)+”Bind’
    type =’tns:”+Ws.getAttribute(“name”)+”Port’>)
2   out “(<soap:binding style='document' transport='http://schemas.
    xmlsoap.org/soap/http'/>”)
2   for each As є Associations(End) do
3     if (typeof(As)=”Aggregation”)
4       for each End є AssociationEnds(As) do
5         if (getStereotype(End)= ”WSOperation”)
6           for each Op є Operation do
8             out <operation name=’”+Op.name/]+”’>”
9             out(“<input><soap:body use=’literal’/></input>”)
10            out(“<output><soap:body use=’literal’/></input>”)
12          end for
```

```
13          end if
14        end for
15      end if
16  end for
17  out("</binding")
18  out("<service name="+Ws.getAttribute("name") +"Service'>")
19  out("<port binding='tns:"+ Ws.getAttribute("name") +"Bind")
20  out("name="+Ws.getAttribute("name") +"Port")
21  out(</port></service>")
```

V. CONCLUSIONS AND FUTURE WORKS

Mashup applications provide the capacity of creating new elements by composing existing resources, and this is the reason why they are being widely adopted in the web 2.0 community.

The presented work tries to enrich mashups with the definition of a model-driven approach that provides new advantages in the development process of this type of applications. The main purpose of this proposal is making mashup development more cost efficient by increasing automation in software development. With this aim, the following task has been performed: a new profile extends UML in order to specify new concepts involved in a mashup application, and a set of transformation rules have been propose with the aim of guiding the transformation process.

Future works will try to study how to increase the performance degree of our approach. With this aim, we will try to incorporate prefetching techniques to download web contents in advance.

REFERENCES

[1] J. Garrett. "Ajax: A new approach to web applications", http://www.adaptivepath.com/publications/essays/archives/000385.php. [retrieved:December, 2012.

[2] L.D.Paulson, "Building rich web applications with Ajax", Computer, vol.38, no.10, 2005, pp. 14-17], doi: 10.1109/MC.2005.330.

[3] W3C, World Wide Web Consortium. http://www.w3c.org. [retrieved:November, 2012]

[4] K. Stolee and S. Elbaum, "Refactoring pipe-like mashups for end-user programmers", 33rd International Conference on Software Engineering (ICSE '11), 2011, pp. 81-90, doi:10.1145/1985793.1985805.

[5] OMG. Model Driven Architecture, http://www.omg.org/mda [retrieved:November, 2012]

[6] J Estefan, "Survey of Model-Based Systems Engineering (MBSE) methodologies",http://www.incose.org/productspubs/pdf/techdata/mttc/mbsemethodogy_survey_2008-0610_revb-jae2.pdf, [retrieved: January, 2013]

[7] S. Teppola, P. Parviainen, and J.Takal, "Challenges in Deployment of Model Driven Development", Fourth International Conference on Software Engineering Advances (ICSEA '09), 2009, pp.15-20, doi: 10.1109/ICSEA.2009.11.

[8] S. Aghaee and C Pautasso, "Mashup development with HTML5", 3rd and 4th International Workshop on Web APIs and Services Mashups. article No. 10, 2010, doi:10.1145/1944999.1945009

[9] P Vrieze, L. Xu, A. Bouguettaya, J. Yang, and J. Chen, "Building enterprise mashups", Future Generation Computer Systems, vol. 27, issue 5,2011, pp.637-642.

[10] Dapper: The Data Mapper. http://www.dapper.net/. [retrieved:January, 2013]

[11] Yahoo Pipes. http://pipes.yahoo.com/pipes/. [retrieved:January, 2013]

[12] DERI Pipes. http://pipes.deri.org/. [retrieved:January, 2013]

[13] Serena Business Manager. http://www.serena.com/index.php/en/products/sbm. [retrieved:January, 2013]

[14] IBM Mashup Center. http://www.ibm.com/developerworks/lotus/products/mashups/. [retrieved:January, 2013]

[15] G. Meditskos and N. Bassiliades, "A combinatory framework of Web 2.0 mashup tools, OWL-S and UDDI", Journal Expert Systems with Applications, volume 38 Issue 6, 2011, pp. 6657-6668, doi:10.1016/j.eswa.2010.11.072

[16] Y. Liu, X. Liang, L. Xu, M. Staples, and L.Zhu, "Composing enterprise mashup components and services using architecture integration patterns". Journal of Systems and Software archive, volume 84, issue 9, 2011, pp. 1436-1446, doi:10.1016/j.jss.2011.01.030

[17] S. Soi and M. Baez, "Domain-specific Mashups: From All to All You Need", Proceedings of the 10th international conference on Current trends in web engineering (ICWE'10), 2010, pp. 384-395.

[18] B.C.M. Fung, T. Trojer, P.C.K. Hung, L. Xiong, and K.Al-Hussaeni, "Service-Oriented Architecture for High-Dimensional Private Data Mashup", IEEE Transactions on Services Computing, 2012, vol. 5, no. 3, pp. 373-386, doi: 10.1109/TSC.2011.13.

[19] C. Atkinson and T. Kühne, "The role of metamodeling in MDA". International Workshop in Software Model Engineering, 2002.

[20] J. Mukerji and J. Miller. MDA Guide version 1.0.1, http://www.omg.org/cgi-bin/doc?omg/2003-06-01, [retrieved: December, 2012]

[21] G. Booch, A.W. Brown, S. Iyengar, J. Rumbaugh, and B Selic, "An MDA manifesto", MDA Journal., 2004. http://www.bptrends.com/publicationfiles/05-04 COL IBM Manifest 20 -Frankel- 3.pdf. [retrieved:January, 2013]

[22] R. Gitzel, A. Korthaus, and M. Schader, "Using established Web Engineering knowledge in model-driven approaches", Science of Computer Programming, 2007, vol. 66, issue 2, pp. 105-124, doi:10.1016/j.scico.2006.09.001.

[23] P. Fraternali, S. Comai, A. Bozzon, and G.T. Carughi, "Engineering rich internet applications with a model-driven approach", Journal ACM Transactions on the Web (TWEB), 2010, vol. 4, issue 2, doi:10.1145/1734200.1734204

[24] D. Schwabe, G. Rossi, and S.D.J. Barbosa, "Developing Hypermedia Applications using OOHDM", Seventh ACM conference on Hypertext, 1996, pp. 116-128.

[25] M. Linaje, J.C. Preciado, and F. Sanchez, "Engineering Rich Internet Application User Interfaces over Legacy Web Models", Journal IEEE Internet Computing archive, volume 11, issue 6, 2007, pp. 53-59.

[26] J.C. Preciado, M. Linaje, R. Morales, F.Sanchez, G. Zhang, C. Kroiβ, and N. Koch, "Designing Rich Internet Applications Combining UWE and RUX-Method", Eighth International Conference on Web Engineering, 2008, pp. 148-154, doi:10.1109/ICWE.2008.26.

[27] M.A. Chatti, M. Jarke, M. Specht, U. Schroeder, and D. Dahl, "Model-Driven Mashup Personal Learning Environments", International Journal of Technology Enhanced Learning, volume 3, issue 1, 2011, pp. 21-39, doi:10.1504/IJTEL.2011.039062.

[28] K .He, Wang, J. Wang, J.Liu, C. Wang, and H. Lu, "On-Demand Service-Oriented MDA Approach for SaaS and Enterprise Mashup Application Development", International Conference on Cloud and Service Computing (CSC), 2012, pp. 96–103.

[29] J.L Herrero, P. Carmona, and F. Lucio, "Web services and web components". Seventh International Conference on Next Generation Web Services Practices, 2011, pp. 164-170.

[30] D.Skogan, R. Groenmo, and I. Solheim, "Web Service Composition in UML", Eighth IEEE International of the Enterprise Distributed Object Computing Conference, 2004, pp. 47-57.

 ISBN: 978-1-61208-280-6