Can Rule-Based Mashups play a role in the Cloud?

A Position Paper

Adrian Giurca

Brandenburgische Technische Universität, Germany
Giurca@tu-cottbus.de

Abstract. In the last time the cloud computing stays more and more in the attention of the major IT players. Large companies start moving their businesses towards such an architectural approach. Our proposal is to investigate the potential use of the rule-based mashups to perform Enterprise 2.0 implementations in the cloud. We argue that some of the issues of modeling and executing mashups on the cloud can be addressed by using intelligent, rule-based, mashups and derive some open research questions. We look towards other researchers' feedback including ones which are interested to join our initiative.

1 Basics on Cloud Computing

In a report from June 2008, Gartner defines cloud computing as "a style of computing where massively scalable IT-related capabilities are provided *as a service* using Internet technologies to multiple external customers." *"During the past 15 years, a continuing trend toward IT industrialization has grown in popularity as IT services delivered via hardware, software and people are becoming repeatable and usable by a wide range of customers and service providers,"* said Daryl Plummer, managing vice president and Gartner Fellow. *"This is due, in part to the commoditization and standardization of technologies, in part to virtualization and the rise of service-oriented software architectures, and most importantly, to the dramatic growth in popularity of the Internet."*

Cloud computing is a computing paradigm in which business processes are assigned to a combination of *data*, *software* and *services* available over a network - all of them known as "the cloud" (see Figure 1). Cloud computing enables users and developers to utilize already existent services without any knowledge about the technology infrastructure that supports them.

The basic idea is that instead of writing your own home page or blog or e-commerce web site, and running that on someone else's servers, you write a software application instead. This idea has been pioneered on the web by Amazon Web Services and Salesforce.com, though similar approaches have been used before.

The cloud computing architecture comprises three levels:

1. *Infrastructure as a Service* (IaaS) - offering storage data, network capacity, computing power, and other resources for which you have to pay only the actual resources used. This is the case for the infrastructure offered by IBM Cloud, Amazon Web Services or Apple mobile.me.
2. *Platform as a Service* (PaaS) - where developers create, test and execute out their own applications on the runtime environment provided by the cloud. The runtime can be sold to cloud customers. Significant examples are Google AppEngine Microsoft Azure Services Platform, CogHead (bought by SAP in February 2009), Bungee Labs or Quickbase.
3. Software as a Service (SaaS) - the provider operates a variety of applications in the cloud. They are used by many customers over the Web and only end-user services have to be offered or sold. Significant providers are Oracle OnDemand, Microsoft Office Live Salesforce.com, Zoho or Animoto.

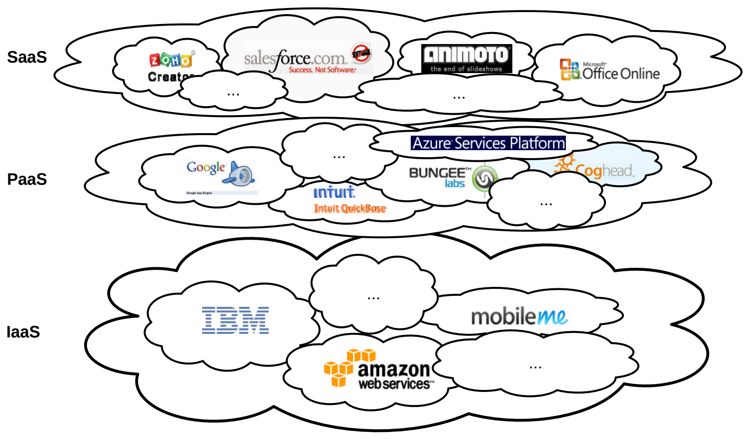

Fig. 1. The Cloud

Cloud computing was started and is developed mainly in enterprises while universities start joining the enterprise research teams just recently. The National Science Foundation announced it had awarded $5 million in grants to fourteen universities as part of its Cluster Exploratory (CLuE) program. The universities receiving money include Carnegie-Mellon, Florida International, MIT, University of Wisconsin, Yale, Purdue, UC-Irvine, UC-San Diego and the San Diego Supercomputer Center, UC-Santa Barbara, University of Maryland-College Park, University of Massachusetts-Amherst, University of Virginia, University of Washington, University of Utah. Corporations include IBM and Google on their "Cloud Computing University Initiative", which serves as a type of spear-head group for the project's goals and focuses results on industry-oriented needs. [1].

[1] See IBM press release, April 23, 2009.

1.1 Enterprise 2.0

Enterprise 2.0 or *Web 2.0 in the Enterprise* was introduced in [11, MacAfee, 2006] to describe how the use of Web 2.0 techniques within an organization can improve productivity and efficiency.

Adopting Web 2.0 techniques allows information workers to control their own experiences with simplified support guidance from IT, and, consequently, create for themselves a more intuitive and efficient work environment.

However, the Enterprise 2.0 scenario remains connected to the main thread of Enterprise Architecture Planning (EAP). In early 90's [19, Spewak and Hill, 1993] defined Enterprise Architecture Planning (EAP) as *"the process of defining architectures for the use of information in support of the business and the plan for implementing those architectures"*.

Spewak's approach to EAP sees four sequential modeling layers:

Q1: *What is my business mission?*
Q2: *What data is required to satisfy the mission?*
Q3: *What application I have to build/use using that data?*
Q4: *What is the technology to implement my applications?*

In an Enterprise 2.0 scenario, while the first layer is much more devoted to the business of the enterprise, the next three are clearly related to the IT landscape.

But how can we use these layers on the cloud?

2 Intelligent Mashups

One can start using the cloud at any of its levels. However, despite the case that most of the service providers in the cloud deliver open APIs, another perspective of services aggregation on the Web is offered by mashups. Basically, a mashup is a Web application that combines data or functionality of one or more services into a single integrated application. In its early stage this concept has been seen just like any other software engineering approach, but in the last time it started to receive attention also from the academia's side (see for example, [12, Morbidoni et al., 2007], [1, Abiteboul et al., 2008], [7, Jarrar and Dikaiakos, 2008]). Artificial intelligence techniques are now applied on a large scale in enterprise applications. Nowadays, in many cases, businesses behavior is expressed naturally through business rules(see [16, Ross, 1997], [17, Ross, 2003]).

2.1 A Basic Mashup Classification

We are not aware of a well established mashup classification, therefore this section will provide a basic one following the aspect of data processing and presentation and the aspect of content processing.

Considering the aspect of data processing and presentation we see the following kinds of mashups:

1. *Data-centric.* Such applications use two or more services to create an integration point towards a business process goal. Usually the used services provide information feeds. Such applications does not focus on any presentation layer (i.e. they may not provide any presentation too). The main activities are related to automating data extraction, data migration and data integration by consuming SOA services.
2. *Presentation-centric.* Application related to presentation of some data. It takes two different resources to create something which is more useful than the standard sum of its parent parts. This business value should be seen on the user's screen. Presentation-centric mashups could also be intra-enterprise (e.g. representing sales with a graphical enterprise logistic system).

According with the content they process we have:

1. *Republishing HTML.* This is an old technique used for a long time: the application retrieves HTML content from specific web sites and then re-publishes a customized content.
2. *Re-syndication.* The simplest form of mashup is taking RSS feeds [18], and either combining it with another feed or embedding it in another location. There are many ways of doing this.
3. *Customized Search.* Nowadays, all search engines offer public APIs, therefore building a customized search engine is no longer a difficult task. Such customized engines are particularly interesting on top of the data provided by social networks or search engines that are customized by using the users' profiles in the social network.
4. *Personalized Portal.* Such application uses available services to define a "custom page" where the user finds its needs. Usually, such a page interacts with the user and is dynamically changed according with the user's profile.
5. *Business Mashup.* They use the enterprise platform to enrich the collaborative action among businesses, between employees, as well as between the business and its customers.

2.2 Rule-Based Intelligent Mashups

When we refer to the term *intelligent mashup* we understand *a mashup enriched with reasoning capabilities usually provided by an inference engine.* In computer science, and specifically the branches of knowledge engineering and artificial intelligence, see an inference engine as a computer application that tries to derive answers from a knowledge base. Our approach on rule-based intelligent mashups focuses on data integration into a single presentation and allows for artificial reasoning and collaborative action among businesses and users.

This approach uses the JSON Rules language introduced in one of our previous works ([[5, Giurca and Pascalau, 2008]]) with the goal to empower Web 2.0 applications, particularly mashups, with rule-based inference capabilities. At the modeling layer we will use an UML based modeling language, designed in [20, Wagner at al., 2006] which was successfully used to model business rules (see [14, Pascalau and Giurca, 2008]).

We argue more to using rules for mashup modeling since the cloud is service-oriented and rules showed their capabilities to model Web Services (see for example, [4, Giurca et al.], [10, Lukichev et al., 2007], [15, Ribaric et al., 2009])

2.3 Benefits and Drawbacks of Rule-Based Mashups

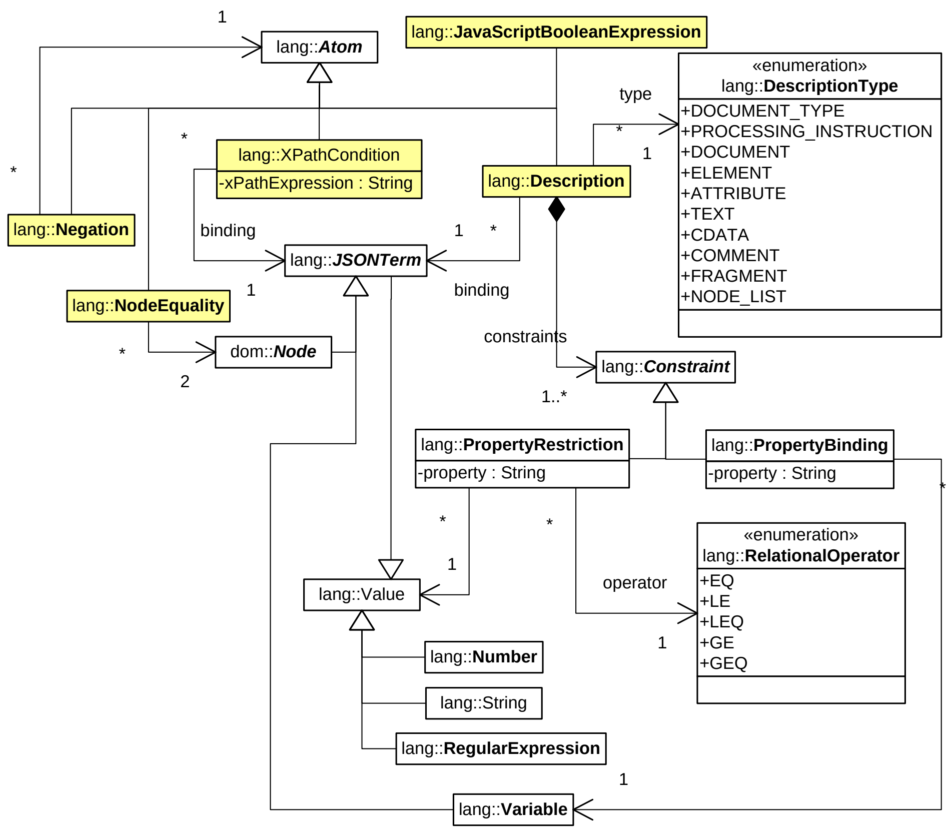

Fig. 2. Basic JSON Rules Conditions

JSON Rules are JavaScript-based reaction rules triggered by DOM Events. Using Ajax technologies, they are able to handle any XML-based format such as Atom [13], [6], RSS 2.0[18], and RDF [9, Klyne and Caroll, 2004]. The language uses a condition language similar with other rule systems and employs any JavaScript function call as actions. The syntax was influenced by the JSON Notation [2, Crockford, 2006] a well known notation to express JavaScript objects.

JSON rules operates on a specific knowledge base with facts obtained from the content. Rules conditions are based on atoms defined on top of Document Object Model (DOM). Figure 2 depicts the UML model of supported atoms. While the metamodel is large we can provide straightforward examples for such conditions. For example, considering an Atom entry (Google) such as:

```
<entry gd:etag="DUEMQno6fyp7ImA9WxVaF08">
  <id>tag:blogger.com,1999:blog-10861780.post-6717232825138410541</id>
  <published>2009-04-13T17:40:00.000-07:00</published>
  <updated>
   2009-04-14T09:48:03.417-07:00
  </updated>
  <app:edited xmlns:app="http://www.w3.org/2007/app">
   2009-04-14T09:48:03.417-07:00
  </app:edited>
  <category scheme="http://www.blogger.com/atom/ns#" term="accessibility"/>
  <title>An ARIA for Google Moderator</title>
  <content type="html">
   ...
  </content>
  <link rel="edit" type="application/atom+xml"
    href="http://www.blogger.com/feeds/..."/>
  <author>
   <name>A Googler</name>
   <email>noreply@blogger.com</email>
  </author>
  <feedburner:origLink>
   http://googleblog.blogspot.com/2009/04/aria-for-google-moderator.html
  </feedburner:origLink>
 </entry>
```

Our rules can handle various conditions such as:

```
// Variable $C is bound to all <category> elements
// in the DOM  and $V retrieve the value of the attribute term i.e.
// ?V == 'accessibility' for the above excerpt
$C:category($V:attributes['term'])

// check if the content of <email> element is a valid email address
// if so, bound the element  to variable $E
$E: email(nodeValue=="match(^[0-9]{4}-(((0[13578]|(10|12))-(0[1-9]|[1-2][0-9]
|3[0-1]))|(02-(0[1-9]|[1-2][0-9]))|((0[469]|11)-(0[1-9]
|[1-2][0-9]|30)))$)")

// check if the node bound to the variable $X is in the
// node list obtained by evaluating the corresponding xPath expression
"$X in "entry//category[@term='accessibility']"
```

A simple analysis on JSON Rules proves that they provide a platform which is able to deal with all kinds of mashups discussed above. However there are drawbacks too. The Table 1 shows rule-based mashups benefits and drawbacks:

Feature	Status	Comments
Easy Modeling	$\Uparrow$	See [20, Wagner at al., 2006]
Support for any XML data formats	$\Uparrow$	Induced by using DOM
Allows re-syndication	$\Uparrow$	DOM combination
Search based on public Web services	$\Uparrow$	via Ajax
Service aggregation	$\Uparrow$	Using ECA rules. See for example, [4, Giurca et al., 2006]
Side effects	$\Uparrow$	Rule actions consist of any JS function call
Any presentation layer	$\Uparrow$	Induced by CSS
Support for public and private mashups	$\Uparrow$	
Declarative programming	$\Uparrow$ / $\Downarrow$	Some people like rules some others not. Newcomers should learn basics about rules.
Speed	?	The engine runs in the browser. Viability tests should be performed. Some JS frameworks are quite fast.
Stability	$\Uparrow$ / $\Downarrow$	As much as any other JavaScript-based application
Security	$\Uparrow$ / $\Downarrow$	As much as any other JavaScript-based application

Table 1. Rule-Based Mashups Benefits and Drawbacks

3 Using Rule-based Mashups on the Cloud

[3, Foster and Tuecke, 2005] analyzes terms such as *software as a service*, *software on demand*, *adaptive enterprise* and *mashups* and concludes that they are overlapping to many extents. We share this view and consider mashups at the SaaS level in the cloud. As a consequence, intelligent mashups should be able to use the PaaS layer as well as resources available in its layer.

Coming back to Spewak approach (Section 1.1) we argue that intelligent mashups provide solutions to the mainstream questions i.e.

Answering Q1: Intelligent mashups act both at SaaS and PaaS level therefore they are able to handle various businesses. Using SalesForce services (CRM) or Zoho Services (e.g. invoicing) put the mashup in the center of our business.

Answering Q2: Intelligent mashups handle all kind of XML content. content as data was proved as being enough in a large number of business applications. Such content is delivered by various platforms from the PaaS level in the cloud (e.g. Enables teams, divisions, partners and vendors to work together effectively by accessing and sharing centralized information.).

Answering Q3: The tendency to use the browser as a client are already for a long time and they are very successful. Google (e.g. Google Apps) is one of the notable leaders and Amazon Web Services is the most powerful e-commerce service-based infrastructure.

Answering Q4: Running mashups in the browser entitle us to believe that a JavaScript engine is the most suitable one.

4 Conclusion and Research Opportunities

We have argued that some of the issues of modeling and executing mashups on the cloud can be addressed by using intelligent mashups based on existing literature, our experience, and an observational case study. Future WWW programming will be strongly oriented on the cloud since it provides various data sources (from the main data creators) and a complex infrastructure using all kinds of services, together with a powerful level of application build on top of strongly established platforms. Inside of this *"cloud of data, infrastructure, technologies and services"*, some issues remain open and offer research opportunities:

Business Level: *How can intelligent mashups handle the legal agreements of using cloud resources?* ENISA (the European Network and Information Security Agency) is conducting a security risk assessment on cloud computing. For the Cloud Risk Assessment, they focus on scenarios including: (a) A user perspective on Cloud Computing (i.e. Small and Medium Enterprises), and (b) Cloud Computing in a eGovernment environment (i.e. national health service). While we find the most of mashups applications related to the first scenario, the second one is also possible. We see necessary to define standard legal agreements for publicly offered data and services. It is to be investigated if legal policies can be exchanged here as in B2B solutions.

Business Level and Technological Level: *How the cloud can guarantee the integrity of data and services offered to mashups?* It is clear that this is one of the key issues on the cloud. If your data is not available to you, for whatever reason, then it is no good for your mashup. Therefore we probably should investigate exception mechanisms. In addition, many service providers on the cloud will provide at least one back up resource, maybe more. Any subscriber should check what provisions are made and choose the data provider accordingly. Finally we see interoperability issues: For example using the CRM from Salesforce.com, you may be limited to its data proprietary format. If you want to move to another service implies how would you get your data back? This shows the necessity for a standard on data interoperability in the cloud (May be OWL 2?).

Conceptual level: *What is the conceptual model of a mashup?* Recently, [8, Jarrar and Dikaiakos, 2009] defines a data mashup language for the "Data Web". By contrary, Google deprecates its mashup editor which was XML based. Therefore we consider that researching a conceptual model of mashups will improve chances to obtain a better definition of this paradigm and will contribute to the cloud computing mainstream too.

Tools Level: *Do we need development tools to model, build and debug intelligent mashups?* The actual mashup market includes many visual tools such as Yahoo Pipes, DERI Pipes, Intel Mash Maker, Microsoft Popfly, or IBM Mashup center.

Therefore we consider that such tools are welcome for intelligent mashups too.

We are looking for groups that may want do cooperative work on these topics but also to collaborate for defining and implementing complex intelligent mashups scenarios. Economic, social, and game production are just some of the domains we are interested in.

References

1. S. Abiteboul, O. Greenshpan, and T. Milo. Modeling the mashup space. In *WIDM'08: Proceeding of the 10th ACM workshop on Web information and data management*, pages 87–94, 2008.
2. D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). `http://tools.ietf.org/html/rfc4627`, July 2006.
3. I. Foster and S. Tuecke. Describing the Elephant: The Different Faces of IT as Service . *Enterprise Distributed Computing*, 3(6):26–34, July/August 2005.
4. A. Giurca, S. Lukichev, and G. Wagner. Modeling Web Services with URML. In *Proceedings of SBPM'2006*, Budva, Montenegro, 11 May 2006.
5. A. Giurca and E. Pascalau. JSON Rules. In *Proceedings of the Proceedings of 4th Knowledge Engineering and Software Engineering, KESE 2008, collocated with KI 2008*, volume 425, pages 7–18. CEUR Workshop Proceedings, 2008.
6. J. Gregorio and B. de hOra. Atom Publishing Format (RFC5023). `http://tools.ietf.org/html/rfc5023`, 2007.
7. M. Jarrar and M. D. Dikaiakos. Mashql: a query-by-diagram topping sparql. In *ONISW '08: Proceeding of the 2nd international workshop on Ontologies and nformation systems for the semantic web*, pages 89–96, New York, NY, USA, 2008. ACM.
8. M. Jarrar and M. D. Dikaiakos. A Data Mashup Language for the Data Web. In *Proceedings of teh workshop Linked Data on the Web (LDOW2009)*, 2009.
9. G. Klyne and J.J. Caroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, February 2004. `http://www.w3.org/TR/rdf-concepts/`.
10. S. Lukichev, A. Giurca, G. Wagner, D. Gasevic, and M. Ribaric. Using uml-based rules for web services modeling. In V. Oria, A. Elmagarmid, F. Lochovsky, and Y. Saygin, editors, *Proceedings of the Second International Workshop on Services Engineering*, pages 290–298, Istanbul, Turkey, 16 April 2007. IEEE Computer Society.

11. A. McAfee. Enterprise 2.0: The Dawn of Emergent Collaboration. *MIT Sloan Management Review*, 47(3):21–28, March-May 2006.
12. C. Morbidoni, A. Polleres, G. Tummarello, and D. Le Phuoc. Semantic web pipes. Technical report, DERI - DIGITAL ENTERPRISE RESEARCH INSTITUTE, November 2007.
13. M. Nottingham and R. Sayre. Atom Publishing Format (RFC4287). `http://tools.ietf.org/html/rfc4287`, 2005.
14. Emilian Pascalau and Adrian Giurca. Can URML model successfully Drools rules? In *Proceedings of 2nd East European Workshop on Rule-Based Applications (RuleApps2008)*, volume 428, Patras, Greece, July 21-22 2008. CEUR Workshop Proceedings.
15. M. Ribaric, S. Sheidaei, D. Gasevic, M. Milanovic, A. Giurca, S. Lukichev, and G. Wagner. *Modeling of Web Services using URML, R2ML and model transformations.* IGI Publishing, 2009.
16. R. G. Ross. *The Business Rule Book: Classifying, Defining and Modeling Rules.* Database Research Group, Inc., Boston (MA), 2nd edition edition, 1997.
17. R. G. Ross. *Principles Of Business Rule Approach* . Addison-Wesley Professional, 2003.
18. RSS. RSS 2.0 Specification, version 2.0.11. `http://www.rssboard.org/rss-specification`, March 2009.
19. S.H. Spewak and S. C. Hill. *Enterprise architecture planning developing a blueprint for data, applications, and technology.* Wiley, New York, 1993.
20. G. Wagner, A. Giurca, S. Lukichev, G. Antoniou, C. V. Damasio, and N. E. Fuchs. Language improvements and extensions. Technical report, Munchen, Germany, April 2006.