A Document-oriented Web-based Application for Supporting Collaborative Product Development

Filipe Rocha, Leonilde R. Varela, S. Carmo-Silva
Department of Production and Systems, School of Engineering, University of Minho
Campus de Gualtar, 4710-057 Braga, Portugal
filiperocha@gmail.com, leonilde@dps.uminho.pt, scarmo@dps.uminho.pt

ABSTRACT

Innovation is a creative process strongly associated with development and deployment of new products, i.e., goods and services. This is essential in the global economy of today for sustainability and success of companies. Good collaboration between those involved in new product development is an essential requisite for this success. Such collaboration, which involves not only company stakeholders but also suppliers and customers, requires easy access to necessary data and agile communication and sharing of relevant information, which is distributed in a network of resources and users. Hence, web-based applications, decentralized repositories and databases are used to store and manage product and process development information. For meeting these new product development requirements Internet based collaborative tools and services must be applied. The use of web services is important in product development, helping the integration of data and knowledge bases and also processes and application interactions.

This paper reports on work for managing product and process information, as well as documentation generation, throughout the product development process cycle in an Internet based collaborative environment. The information concerned includes product and process information, as well as product development history. One important aspect deals with web based restoring decisions and options made along the product design and development cycle, including product revisions and/or versions manipulation. The research work, focus is on the development of a document-oriented web-based application using Apache CouchDB technology and REST web services. The proposed application is described and the main functionalities are illustrated through some examples of use.

INTRODUCTION

Innovation is closely associated with new product development (NPD). This is an activity that can benefit very much from collaboration between stakeholders involved. This can be enhanced through the use of agile tools and technology for efficient and fast information sharing and the carrying out of collaboration tasks. Such tools and technology are mostly available through the Internet. In fact, nowadays the Internet can be considered the best channel for collaboration and knowledge exchange and un inevitable path for companies sustainability and competitiveness [1]. This is true not only for a company itself, but also for suppliers and customers that contribute for product innovation in a cooperative and collaborative environment [1-4].

Zhang et al. [5] enhances and describes a web-based collaborative design platform. This platform provides a product catalogue, a communication system between peers, knowledge access restrictions and project management.

The use of web services [6] can be seen as a feasible interface between involved parties in product innovation, helping the integration of data and knowledge bases, processes and application interactions [1, 3, 5, 7].

One important aspect of this work is to access product and process information, in the collaborative distributed context of NPD.

Access to information for product design and development is not an easy task when using relational databases (RDB). Tables are referencing other tables and to retrieve little information, extreme computational work needs to be done by selecting and joining data from this structure.

RDB are designed to store and report on highly structured, interrelated data. Moreover, they include schemas and storage of the existing data that need frequent update. This often causes problems not anticipated in the initial database design. Each change in the initial database schema is a challenge for local and distributed upgrades. When dealing with paperwork, this is common and happens when we need to adapt predefined paper forms by adding or removing information and fitting its strict form to the needs of the moment.

In this paper, we propose a new approach to manage new product development information. This new concept is based on Document-oriented Databases (DoDB) with native distributed properties, instead of the usual Relational Database Managements System (RDBMS). So, instead of the strict schema provided by tables we get to work with the documents and free schemas. This new concept is based on an open source Apache project named Apache CouchDB, commonly referred just as CouchDB [8].

This paper, in addition to this introduction follows with a literature review and the description of a proposed approach for managing product information. Based on this a system architecture is described and illustrated. Finally a conclusion is presented.

LITERATURE REVIEW

Companies use proprietary software, most of the times centralized, having high costs with licenses and high risk with information losses. Aziz et al. [9] talk about the shortcomings of centralized architectures and proposes a decentralized one based on open standards and open source tools, stating that systems like this are implementable at low cost.

Sharma et al. [3] referring collaborative product innovation presents a theoretical framework identifying tools and processes for people and teams to collaborate on product development.

Zhang et al. [5] reviewed the state of the art on Internet-based product information, concluding that despite several Internet web-based technologies which have been applied to product development still none has been applied to real industrial applications. They also found a number of practical problems remaining unsolved, including dynamic product information collection and update, real-time collaboration, scalability and interoperability. These problems still occur today despite several attempts and approaches to solve them [10-14]. This is probably due to the existence of bottlenecks on the database back-ends [14]. Relational databases have very strict schemas and enhance those problems with dynamic information, i.e., a change on the initial schema represents a change in the whole system. Moreover, dynamic product information is not what we get with schemas because there is no predefined schema for each product or service.

Major problems with scalability with relational databases become a point of failure. In particular, replication is not trivial. To understand why, consider the problem of having two database servers that need to have identical data. Having both servers for reading and writing data makes it difficult to synchronize changes. Having one master server and another slave is bad too, because the master has to take all the heat when users are writing information [15]. This statement leads to wish that there was a replication system that would come with an automatic conflict detection and resolution, by making it easy to synchronize data in both directions whenever wanted after being able to work offline independently [16].

REST (Representational State Transfer) technology is considered to have great potential for solving scalability problems [17-20]. Fielding [19] states that “The REST architectural style has been defined to describe the web architecture and to guide its future evolution, preserving the fundamental characteristics of scalability”. It promotes software evolvability, efficiency, performance and reliability [16] using the protocols already available for the web.

Liu and Xen [21] reviewing some traditional technologies and systems for management of product information and discuss the need for their integration on a web environment as a means of having a more adaptive and flexible infrastructure. They also mention bandwidth and security concerns with data availability over the Internet. Unfortunately they do not say how the reviewed systems handle changes made to predefined data management models, leaving therefore, important questions unanswered.

Frutos and Borenstein [22] talk about mass customization and the role that customers should have in new product development. They report a web-based information system for flats' customization through interactions between building company web site and costumers. Customers can generate a variety of customized solutions, i.e new products, with control on costs, despite flat variants' limitations.

One concept that has been used in production areas on several fields is the Document-Oriented Databases (DoDB). This concept is implemented in Apache CouchDB DBMS (Database Management System) [23] and can be of great utility in NPD due to the functionalities that DoDB offer .

The next sections will focus on explaining how Apache CouchDB works in the proposed approach to manage new product development information, followed by a description of our solution for the system's architecture.

PROPOSED APPROACH

In this work we propose the CouchDB for managing NPD information. CouchDB is an open source "distributed, fault-tolerant and schema-free DoDB, accessible via a RESTful HTTP/JSON API" [24]. Data is stored in documents, presented in key-value maps using the data types from Javascript Oriented Notation (JSON) [25].

CouchDB is not a relational database or a replacement for it. Moreover, it should also not be seen as an object-oriented database, despite relationships between documents being possible. CouchDB is designed to store and report on large amounts of semi-structured data, simplifying the development of document-oriented applications, which is the bulk of collaborative web applications. A CouchDB database can be described as a flat collection of JSON (JavaScript Object Notation) documents, identified by a universal unique identifier (UUID). Each document is an object that consists of named fields. Field values can be strings, numbers and dates, ordered lists and associative maps.

With CouchDB, no schema is enforced, so new document types with new meaning can be safely added alongside the old ones. **Error! Reference source not found.**Figure 1 illustrates a JSON document including some of the substructures stated before.

```
{
    "_id": "1287072474",
    "_rev": "50-3224511275",
    "name": "Round Table Anderatti Rosso",
    "category": "Table",
    "updated_at": {
        "json_class": "Time",
        "data": "2009/07/24 10:55:52 +0100"
    },
    "leg_material": "Steel",
    "leg_height": "1.20",
    "kind": "Round",
    "type": "Product",
    "number_of_legs": "1",
    "top_diameter": 1.279,
    "revisions": [
        {
            "accepted": "Yes",
            "date": {
                "json_class": "Time",
                "data": "2009/06/24 10:50:58 +0100"
            },
            "author": "Filipe Rocha",
            "doc_revision": "1",
            "description": "Created product number 1287072474"
        },
```


Figure 1: Excerpt of a JSON Document in CouchDB

CouchDB is a fully Atomic Consistent Isolated Durable (ACID) [18] storage engine, never overwriting committed data or associated structures, ensuring that the database file is always in a consistent state. Thus it guarantees that database transactions are reliably processed. Document updates are serialized and there are no locks. It uses a Multi-Version Concurrency Control (MVCC) model, meaning that any number of clients can be reading documents without being locked out or interrupted by concurrent updates, even on the same document [15]. Documents are indexed in B-Trees by their DocID and a sequential number. Since we are talking about append-only, each update to the database generates a new sequential number to identify the document state in its history. This is implemented in the *_rev* field, i.e. the field containing the identification of document revision. of the document as shown in Figure 1.

Transactional commits provide consistent state to the database, i.e., in case of failure during a commit transaction, the commit is discarded and maintenance operations are run. From time to time, some jobs are run in order to compact data, freeing space on disk.

CouchDB also integrates a view model to add structure back to unstructured data. Views are the method of aggregating and reporting on the documents. They are built dynamically never affecting the document and providing different representations of the same data, as shown in Figure 2.

Figure 2: Concurrency with CouchDB – Documents concurrently trying to access Apache CouchDB

CouchDB uses Google's MapReduce programming model [26] for views.

CouchDB is a peer based distributed database system. Here, any number of CouchDB hosts, servers and off-line clients, can have independent "replica copies" of the same database, where applications have full database interactivity (query, add, edit, delete). When back on-line or on a schedule, database changes are replicated bi-directionally on the several players (CouchDB hosts), as seen on Figure 3.

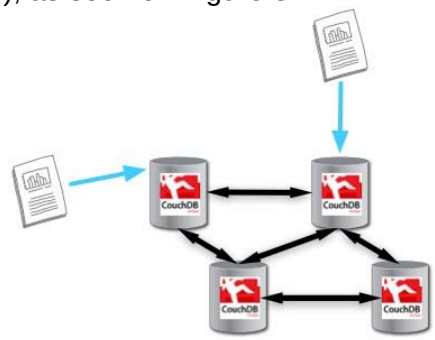

Figure 3: Apache CouchDB distributed System

CouchDB also is characterized by having a built-in conflict detection and management and the replication process is incremental and fast, copying only documents and individual fields changed since the previous replication.

Figure 4 shows an updated document that is inserted on one database and should be replicated through the entire distributed database backend.

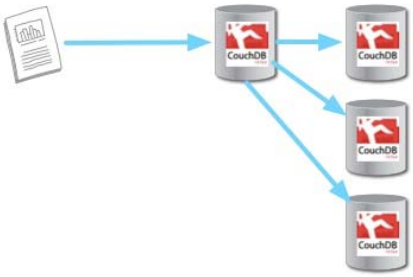

Figure 4: CouchDB Replication

Most applications require no special planning to take advantage of distributed updates and replication.

"Unlike cumbersome attempts to bolt distributed features on top of the same legacy models and databases, it is the result of careful ground-up design, engineering and

integration" [27]. The document, view, security and replication models, the special purpose query language, the efficient and robust disk layout are all carefully integrated for a reliable [27] and efficient system.

With very little database work, it is possible to build a distributed document management application with granular security and full revision histories. Updates to documents can also be easy and interactively carried out by enabling incremental fields that hold the revision number and also replication of fields that hold digitized information and replications including text, images, audio or video.

The main purpose of this work is to manage NPD information. Hence, in the next section we present a system architecture aimed at supporting this purpose.

SYSTEM ARCHITECTURE

A schematic representation of how the system has been implemented is shown in Figure 1. The system is based on distributed CouchDB DoDB. The figure shows the main system functionalities in two parts. The first part includes a web-based product catalog, available to customers and fed by final versions of products created by NPD teams inside the company's walls. The second part deals with information handling and storage, arising from interaction among NPD stakeholders involved, particularly from NPD teams, and applications used.

Internet feedback from customers can be seen as an advantage. Customers' views, comments and reviews on a product usually lead to new versions of products. So, new product ideas can arise from customers' feedback. Thus, one major feature of this system is to take the views and ideas from customers and from other stakeholders for generating new ideas and new products. A NPD idea may originate a new document that can be tagged to call attention e.g. to the idea and to resulting product versions. Then, taking advantage of CouchDB's, product development information, including all external attachments, design information, reports, specifications and concepts in all product development iterations should be stored.

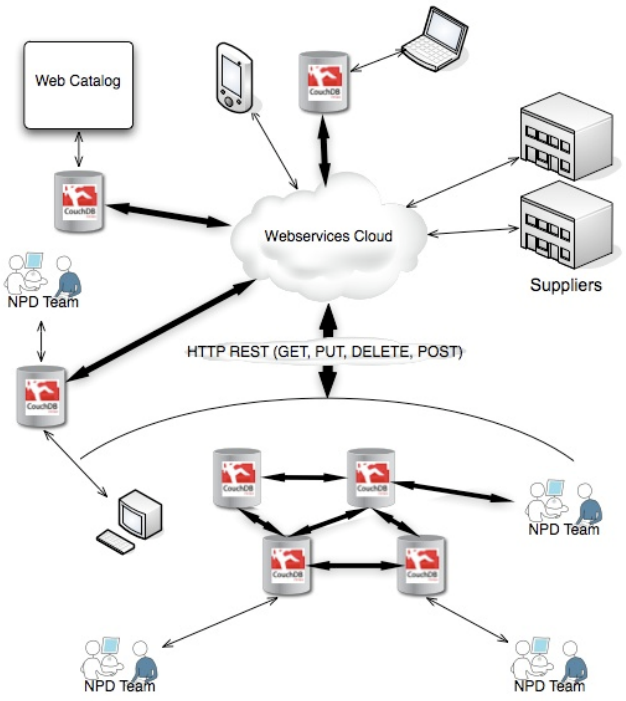

Figure 5: System architecture.

There are several technologies being considered for system implementation. CouchDB

manages the distributed environment keeping all peers synchronized on demand and/or on schedule. Since we are using REST, operations on the database are simple. Catalog and product information management web interfaces will be created using Ruby programming language [28] and the web application framework Ruby on Rails (RoR) [29], usually referred simply as Rails. Ruby is a robust object-oriented programming language and Rails is a Model/View/Controller (MVC) web framework for creating dynamic websites coded in Ruby. RoR also uses REST [30], Therefore, RoR and CouchDB just need a small wrapper to exchange information. All of these technologies are open source and thus available for free.

At this stage of the work, we managed to produce a very simple prototype of a web application that is able to interact with CouchDB. The application is designed to allow users to add their own product specifications freely and register and browse revisions to the shared documents.

Figure 6 shows the prototype at work. It presents some product information, including the last revision of it. Attachments are also shown when they are images, or links to the files are provided for images' retrieval in an easy and user-friendly way.

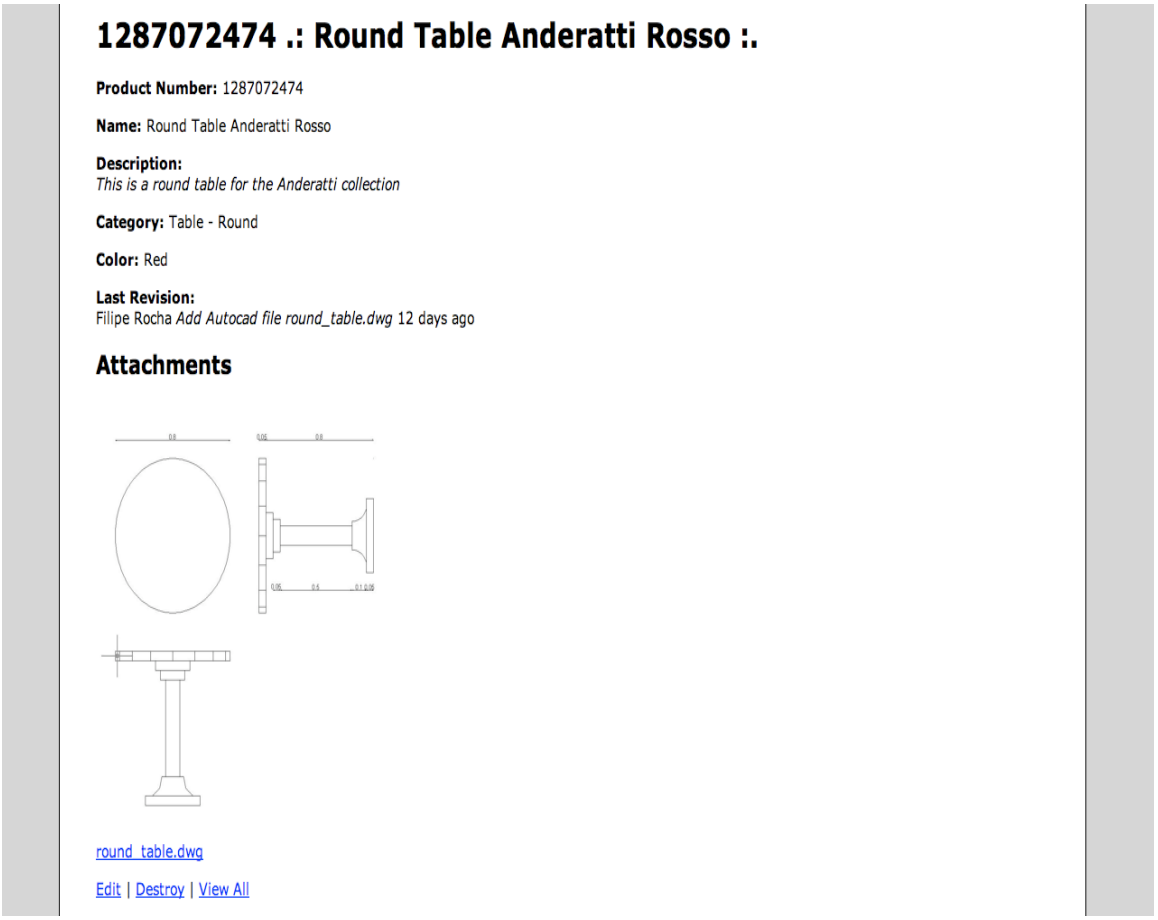

Figure 6: Showing basic product information.

Another interesting system feature is related to suppliers. They may dynamically propose supplying resources for a product. In order to preserve suppliers' confidential information, data should be filtered by web services, becoming a subpart of the product information. Figure 7 is a graphical representation of this idea.

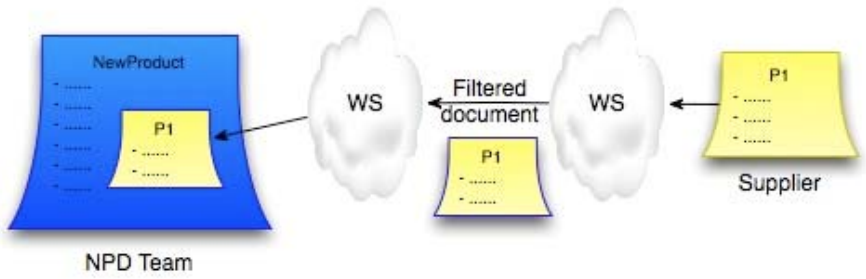

Figure 7: Use of web services to filter confidential information.

CONCLUSION

New product development (NPD) can be enhanced trough collaboration of stakeholders involved including customers and suppliers. This collaboration requires easy access to information generated in the NPD process. This process and collaboration can benefit very much from the use of recent advances on Internet technology, including web services and Document-oriented Databases (DoDB).

In this work we explore the use of some recent Internet technologies, namely Apache CouchDB DBMS and REST towards the development of an environment for dynamic information handling, updating and storage, in the process of NPD, carried out by the local or remote interaction and collaboration of several stakeholders, including customers product development teams and suppliers.

As a first incursion on the collaboration and easy access of information in the process of NPD, a small prototype of a web application that is able to interact with CouchDB has been developed and described here. It is designed to allow users to freely add their own product specifications, register and browse revisions to be shared. Moreover, document revisions and attachments can be easily made available through revision properties provided by DoDB.

Shared catalogs, provided by distributed properties, can be used in this context, since NPD teams may have online and offline access to suppliers' catalogs. Hence, further work will be carried out in the near future enhancing the system by adding several features. These include sharing catalogs that can provide parts of the new product directly from suppliers using an API, improving abstraction of the prototype by letting users define all product requirements and functional specifications freely and interactively choosing what revisions created should be adopted in the design of the final product.

REFERENCES

[1] W. Yan, C. Chen, Y. Huang *et al.*, "A data-mining approach for product conceptualization in a web-based architecture," *Computers in Industry,* vol. 60, no. 1, pp. 21-34, Jan 1, 2009.

[2] B. Dong, G. Qi, X. Gu *et al.*, "Web service-oriented manufacturing resource applications for networked product development," *Advanced Engineering Informatics,* vol. 22, no. 3, pp. 282-295, Jul 1, 2008.

[3] A. Sharma, "Collaborative product innovation: integrating elements of CPI via PLM framework," *Computer-Aided Design,* vol. 37, no. 13, pp. 1425-1434, Nov 1, 2005.

[4] L. Wang, "Collaborative conceptual design—state of the art and future trends," *Computer-Aided Design,* vol. 34, no. 13, pp. 981-996, Nov 1, 2002.

[5] S. Zhang, W. Shen, and H. Ghenniwa, "A review of Internet-based product information sharing and visualization," *Computers in Industry,* vol. 54, no. 1, pp. 1-15, May 1, 2004.

[6] "Web Services @ W3C," 20 May 2009; http://www.w3.org/2002/ws.

[7] H. Wang, G. Liu, B. Han *et al.*, “Collaborative simulation environment framework based on SOA,” *Computer Supported Cooperative Work in Design, 2008. CSCWD 2008. 12th International Conference on*, pp. 416 - 419, Mar 31, 2008.
[8] http://couchdb.apache.org. "Apache CouchDB: The Apache CouchDB Project," 20 May 2009, 2009; http://couchdb.apache.org/.
[9] H. Aziz, J. Gao, P. Maropoulos *et al.*, “Open standard, open source and peer-to-peer tools and methods for collaborative product development,” *Computers in Industry,* vol. 56, no. 3, pp. 260-271, Apr 1, 2005.
[10] W. Shen, Q. Hao, and W. Li, “Computer supported collaborative design: Retrospective and perspective,” *Computers in In*, no. 59, pp. 855-863, 2008.
[11] K. Rodriguez, and A. Al-Ashaab, “Knowledge web-based system architecture for collaborative product development,” *Computers in Industry*, no. 56, pp. 125-140, 2004.
[12] K. C. Tseng, H. Abdalla, and E. M. Shehab, “A Web-based integrated design system: its applications on conceptual design stage,” *The International Journal of Advanced Manufacturing Technology,* vol. 35, no. 9-10, pp. 1028-1040, 2006.
[13] B. Dong, and D. Zhao, “Service-oriented design part information semantic modeling and applications,” *Computer Design and Applications (ICCDA), 2010 International Conference,* vol. 5, pp. 174-177, 2010.
[14] S. Xie, H. Huang, and Y. Tu, “A WWW-Based Information Management System for Rapid and Integrated Mould Product Development,” *The International Journal of Advanced Manufacturing Technology,* vol. 20, no. 1, pp. 50-57, 2002.
[15] G. Decandia, D. Hastorun, M. Jampani *et al.*, "Dynamo: amazon's highly available key-value store." pp. 205-220.
[16] J. C. Anderson, J. Lehnardt, and N. Slater, "CouchDB: The Definitive Guide," O' Reilly, 2009.
[17] L. Richardson, and S. Ruby, *RESTful web services - Webservices for the real world*: O'Reilly, 2007.
[18] J. Gray, and A. Reuter, *Transaction Processing : Concepts and Techniques (Morgan Kaufmann Series in Data Management Systems)*: {Morgan Kaufmann}, 1992.
[19] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” PhD Thesis, 2000.
[20] P. Mazzetti, S. Nativi, and L. Bigagli, “Integration of REST style and AJAX technologies to build Web applications; an example of framework for Location-Based-Services,” *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pp. 1-6, 2008.
[21] D. T. Liu, and X. W. Xu, “A review of web-based product data management systems,” *Computers in Industry,* vol. 44, no. 3, pp. 251-262, 2001.
[22] J. D. Frutos, and D. Borenstein, “A framework to support customer-company interaction in mass customization environments,” *Computers in Industry,* vol. 54, no. 2, pp. 115-135, 2004.
[23] "CouchDB Wiki: CouchDB in the Wild," 20 May 2009; http://wiki.apache.org/couchdb/CouchDB_in_the_wild.
[24] "CouchDB wiki: FrontPage," 29 May 2009; http://wiki.apache.org/couchdb.
[25] http://www.json.org. "JSON," 29 May 2009, 2009; http://www.json.org.
[26] J. Dean, and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Commun. ACM,* vol. 51, no. 1, pp. 107-113, 2008.
[27] "Apache CouchDB: The Apache CouchDB Project," 20 May 2009; http://couchdb.apache.org/.
[28] http://www.ruby-lang.org. "Ruby Programming Language," 20 May 2009, 2009; http://www.ruby-lang.org/.
[29] http://www.rubyonrails.org. "Ruby on Rails," 20 May 2009, 2009; http://www.rubyonrails.org.
[30] "Ruby on Rails Guides," September 2010; http://guides.rubyonrails.org/getting_started.html#rest.