# A Novel Web Publishing System Architecture for Statistics Data Using Open Source Technology

Md Mostafizur Rahman(✉), Hans Dicken, and Dirk Huke

German Centre for Research on Higher Education and Science Studies (DZHW), Hannover, Germany
mostafiz.de@gmail.com
https://www.dzhw.eu

**Abstract.** Web applications have become the primary source of information and transactions over the internet. The number of statistical research organizations, with focused interests in publishing result data on the web, is increasing rapidly in recent years. Statistical report publishing is typically requested to publish reports using different documentation formats. In this research, we examine different web publishing frameworks and effects of saturation trends that appear during implementation. The existing publishing solutions sometimes may or may not cover these requirements. We introduce a simple architectural model that describes the basic steps and overall model to publish statistical reports coming from tabular data sources on the web. Finally, we describe the implementation of such a model and indicate what technologies can be used to enforce it in order to increase productivity and work quality when comparing it with other solutions.

**Keywords:** Web publishing · Statistics data · XML · Java · Open source technology

## 1 Introduction

The World Wide Web (WWW) is one of the greatest technologies to have ever come into present existence. In the early 90s, it served as a medium of communication with static content, and later became the medium of rendering web-based software applications as a user requirement.

Publishing refers to the production and distribution of media products. Media content is in the form of images, text, video and audio. A medium is used in storage, transmission or reproduction of this content. Such as, print media (analog) and Internet (digital media). Digital publishing on the Internet has numerous advantages (cost, topicality, availability, added value) [1]. Nowadays, publishing organizations has been dramatically changed in order to digitize the production and distribution of information on the web. In addition, depending on the requirement, information published on the web has different prospective. Contrarily, the question of how multiple uses of content within a publishing framework can be possible is often asked. For these multi-use purposes, a markup language is required, which separates content from a concrete representation, then converts it to any destination document format.

The presentation of content requires the selection of a suitable presentation format. In this regard, there are numerous formats for different media types. HTML is a standardized language of the W3C (World Wide Web Consortium) [2]. The Portable Document Format (PDF) is a data exchange format created by Adobe Systems [AdobJa]. It is suitable for a template-faithful transformation of print files and a distribution of electronic documents on the internet [3]. A Microsoft Excel spreadsheet (XLS) is a proprietary file format that manages data in a tabular form. Furthermore, an XLS document can be handled by many different applications. One example is Open Office, a free Office product, which is available for use on different operating systems. XLSX is another file extension, similar to the XML spreadsheet file format used by Microsoft Excel. Microsoft introduced this XML standard excel file format for Microsoft Office 2007 and upper as an open file format to transfer data between applications and other working areas [4].

This research work is organized as, first, we address the basic functionality at the heart of the publishing system. In the second layer, we study findability via the following question: In a web populated with a number of existing solutions, how can we identify the right one, the framework that provides the right service for our unique publishing system? To answer this question, we address the existing solution limitations, depending upon the general project requirements. In the third layer, we propose a platform architecture that leverages these graphs encapsulated by real life experience. We demonstrate how this helps to publish reports in a straightforward way, by building a user-friendly web-based publishing system. Our primary goal is to propose publishing system architecture for statistical data, using open source technologies that facilitate their integration into composite applications. Finally, to test our architectural model, we apply them to a real life statistical project (FOSTAT) [5]. After implementation, we are able to study the qualitative benefits of applying this proposed model. Then, to better understand how the architecture can facilitate the development of a real-world application, we illustrate its benefits with respect to this prototype.

## 2 Research Motivation

When developing a publishing system, the most important point needing consideration is the accommodation of collaborative development. A good framework allows collaborative development along with the smooth integration between independent development components. The idea to develop a more robust model for web publishing emerged when I started to work in DZHW (German Centre for Research on Higher Education and Science Studies) on one project (FOSTAT) [5], which leveraged to publish statistical research data on the web. The FOSTAT project is working to publish statistical report data for the Federal Ministry of Education and Research (Germany). One of the main requirements was to develop a web-based publishing system to publish different presentation formats by using the same data, and while following the open source technologies. The idea to bring a new architecture to this manner has hitherto been applied to web-based statistical report publishing.

### 2.1 Requirement of Publishing System

The functional requirements are derived from the application business process, which consists of several contiguous tasks that are performed by an actor to create a desired result.

Access to the application system: Users access the application system via an internet-connected personal computer and a web browser installed on it.
Selection and display of a table in a defined data format: Users have the option to select a table displaying, the home page of the application forms via the UI.
Import of databases in the system: Administrators can import databases with appropriate functions in the system.

Data format:

– HTML, for presentation in the browser.
– PDF, to view in the Adobe Application Compatible
– XLS, to display the Microsoft Excel application
– XLSX, to display the Open XML spreadsheet file format
– XML, represent the XML format in the browser
– XSL-FO, display the XML Formatting Object format

To resolve this research problem via providing a way to integrate design science and object oriented principles. To address this problem, we consider our research approach as "Design Science" [6]. This approach is widely used in many research disciplines, especially in engineering and computer science, but also in information systems [7, 8]. Java Server Faces (JSF) framework was chosen to solve this research question - which is a Java standard and satisfy the MVC pattern.

## 3 Related Work

Several papers and news articles relevant to the web publishing framework were studied during the course of development of publishing architecture. There are number of business intelligence (BI) and reporting tools available in the marketplace. The Eclipse Business Intelligence and Reporting Tools (BIRT) and the Jasper soft Community Edition are mostly using open source tools to generate the report in a variety of document formats [9–12]. Well, there is inauspicious news against these tools [13]. Withal, this is out of the scope to consider these reporting tools in this research work because; the dissertation focuses on the publishing.

### 3.1 Apache Cocoon

The Cocoon is a top-level project of the Apache Software Foundation (ASF) [14]. Apache Cocoon is an XML publishing framework based on the servlet API model. It was built around Separation of Concerns (SoC) and component based development (COP), providing pipeline SAX processing [15, 16].

Apache Cocoon uses an integrated configuration mechanism called the site-map, as a declarative XML document describing a set of pipelines that will be invoked, depending upon a URI (Uniform Resource Indicator) pattern match. An XML document is pushed through a pipeline that exists in several transformation steps. The pipeline consists of three main components, beginning with a generator, zero or more transformers, and a serializer [17, 18].

A sitemap is shown schematically in Fig. 1. Here, a matcher defines each pipeline. Within the illustrated pipeline, data is generated from an XML document using XSLT and an HTML document. The supported output formats include PDF, XML, VRML, etc., which can be controlled by changing transformation rules. The latest stable release of Apache Cocoon is version 2.2.0, released in May of 2008, which is a spring-based framework. Cocoon 1 version 2.2, however, has the advantageous compatibility with Apache Maven 2, which allows for much more efficient building management, as well as perfect integration with the spring framework [19]. Apache Cocoon uses the Apache FOP API to leverage the PDF output format. However, the last stable release of Cocoon framework supports the older version 1.0 of FOP [20], while the recent version of Apache FOP is 1.1. The lack of documented serializer, there is a considerable gap of information explaining the methods for creating Microsoft Excel (.xls) and Open XML spreadsheet (.xlsx) file formats.

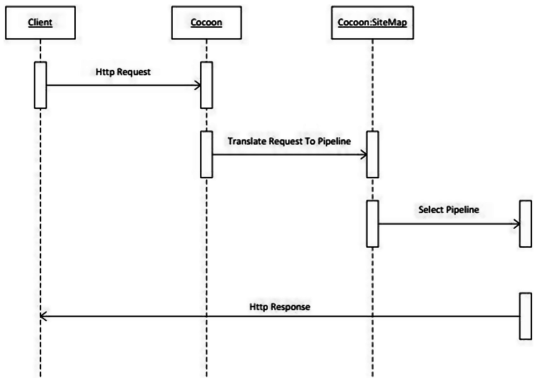


**Fig. 1.** Cocoon pipeline with Sitemap

## 3.2 Oracle XSQL Framework

The Oracle XSQL page publishing framework is a publishing platform capable of publishing XML documents in any format using SQL, XML and XSLT. The Java-based XSQL servlet is the controller port that provides a declarative interface to publish web content dynamically with relational data. However, the biggest limitation of this publishing framework, it is a proprietor base initiative, not a completely open source commencement [21]. It has only little available documentation and a bit cumbersome to implement it. In addition, the XSQL servlet can connect to any JDBC-supported database. However, the object-relational functionality only works when using an Oracle database coupled with the Oracle JDBC driver [21].

### 3.3 Maverick Framework

Maverick is an MVC framework for web publishing using Java technology. This framework focuses solely on MVC logic to allow for generating presentations using a variety of templates and transformation technologies [22]. The most recent update of the Maverick framework was released in 2006. The documentation of this framework is very poor, that makes it difficult to find all the features relative to the frameworks specific requirements.

### 3.4 Apache AxKit

Another Apache initiative was Apache AxKit, which is also an XML Application Server for data publishing. AxKit provides XML document conversion on-the-fly to any format such as HTML, WAP or text [23]. However, this framework is retired in August 2009.

Most of the framework described in this section has some limitations when considering for statistical report publishing. Despite of these, our initiative is novel that can manage both general and domain-specific statistics report publishing.

## 4 Technology and Software Selection

To tests our architecture how compatible it is for statistical report publishing, we select different technologies and software to demonstrate the implementation of the prototype. We use XML as middleware a medium-neutral data format that is not tied to a specific purpose or special software due to its separation of content and layout. After generating XML document, it is used as an input for the report publishing. JAXB (Java Architecture for XML Binding) API is used to generate the XML document [24]. JAXB is a Java mapping standard that defines how Java objects are converted from and into XML. For the persistence layer, an Object Relational Mapping (ORM) persistence framework was chosen. To achieve this functionality, we used the Hibernate persistence tool. Hibernate is an open source high performance object relational mapping (ORM) tool and the query service provider. After considering these aspects, the freely available RDBMS PostgreSQL database server version 9.4.4 version chosen for our application prototype. Higher versions are also usable. However, the concept of persistence offers a switch to any other RDBMS with very little effort.

To develop the prototype, we select the Apache Tomcat server, version 7.0,54. Note, however, our prototype application can be run using any of the 7.x subversions of this software. For the middle and front layers of the application, we chose JSF framework version 2.0. JSF is distributed as an open source under the Oracle Standard Web Framework software license.

## 5 Architecture of the Prototype

To manage the complexity of a software application, the application architecture should be structured with loosely coupled subsystems. This means that each subsystem defines clear responsibilities, minimizing the dependencies between these subsystems. Our architecture is divided into multiple logical layers: a presentation layer, application layer and persistence layer. Figure 2: shows different layer structures and demonstrates the relationships and interactions between applications components contained within those layers. Each individual layer has no knowledge the internal structure of the other layers, but provides services that are used exclusively by the neighboring layers. The use of these services is carried out via a Java class, which is located in the respective underlying layers and provides a narrow interface with fewer methods.

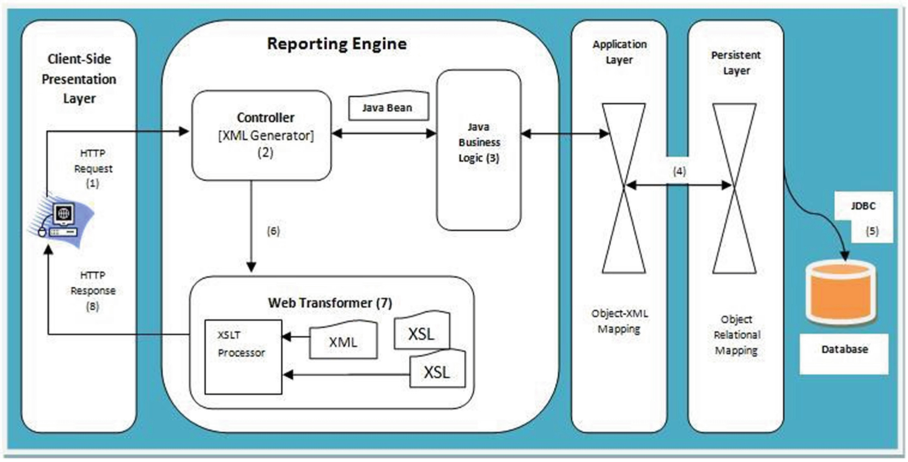


**Fig. 2.** The application architecture of the prototype

The application architecture of the prototype is presented. As the first step of the process, the Presentation layer asks for a request to the Reporting Engine. Then, within the Reporting Engine, the Controller takes this request and calls upon the respective Java Business Logic. The resulting Java object is annotated as an XML tree structure by JAXB API. Using Hibernate API, the Java object is mapped to the corresponding database table. The database (PostgreSQL) is connected with a JDBC driver. The XML document, or response, goes to the Web Transformer engine. This XML document is used as the input. Afterwards, the XSLT parses this XML document along with the respective XSL style sheet. Finally, the desired output result is displayed to the user.

Figure 5: PDF and XSL-FO Report Publishing Architecture, Apache FOP parser to generate the final PDF document sent to the user. Figure 4: HTML Report Publishing Architecture, the HTML Transformer parser which processes the data and presents it in the final HTML document for user view. Figure 3: Excel (.xls, .xlsx) Report Publishing Architecture, with the help of the Apache POI API, to generate the final Excel

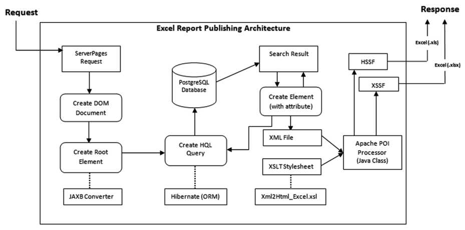


**Fig. 3.** Excel (.xls, .xlsx) report publishing architecture

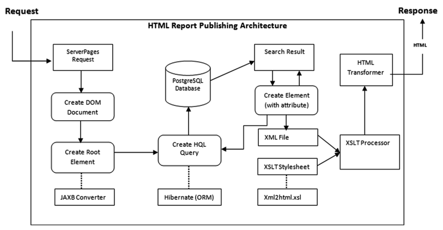


**Fig. 4.** HTML report publishing architecture

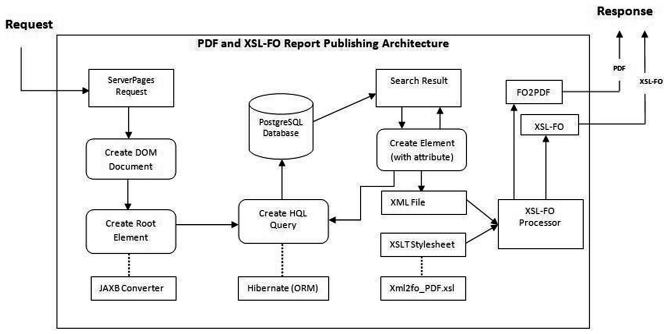


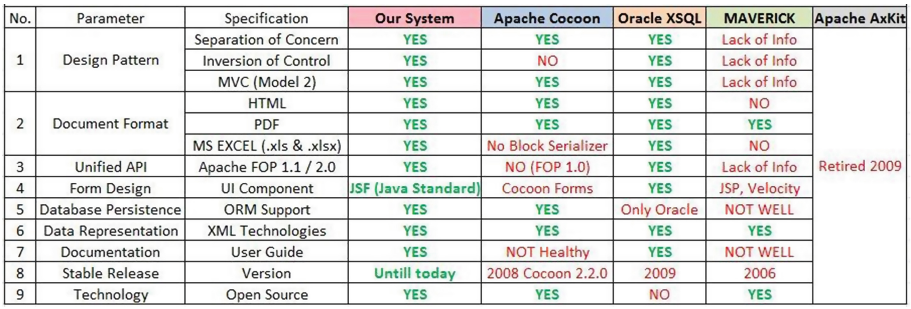
**Fig. 5.** PDF and XSL-FO report publishing architecture

document. In the Java class, HSSF method is used to generate .xls format and XSSF method is used to generate .xlsx format Excel report. This architecture is designed to publish an Excel format report. The first request to create a DOM document is received via a JAXB converter. The resulting Hibernate query is used to create the XML document, including the element and attribute and their respective values. XSL file is used as the XSL declaration. These two files are then used as input for the Excel parser Java Class with the help of Apache POI API, to generate the final Excel document. In this Java class, HSSF method is used to generate .xls format and XSSF method is used to generate .xlsx format Excel report.

## 6 Evaluation

The publishing architecture was evaluated, with a reference of standard web application requirements, to test the social acceptability, maintenance, scalability and operational feasibility of the prototype. Measurements were considered based on functional requirements of statistical report publishers. The significant differences in the measurements favoring this architecture will portray the advantages of report publishing.

Comparison between the existing solution and our proposed architecture is based upon the requirements of a report publishing system, as well as the list of parameters necessary to develop a common web-based publishing application. Figure 6 shows the comparison between different system architecture.



| No. | Parameter | Specification | Our System | Apache Cocoon | Oracle XSQL | MAVERICK | Apache AxKit |
|---|---|---|---|---|---|---|---|
| 1 | Design Pattern | Separation of Concern | YES | YES | YES | Lack of Info | |
| | | Inversion of Control | YES | NO | YES | Lack of Info | |
| | | MVC (Model 2) | YES | YES | YES | Lack of Info | |
| 2 | Document Format | HTML | YES | YES | YES | NO | |
| | | PDF | YES | YES | YES | YES | |
| | | MS EXCEL (.xls & .xlsx) | YES | No Block Serializer | YES | NO | |
| 3 | Unified API | Apache FOP 1.1 / 2.0 | YES | NO (FOP 1.0) | YES | Lack of Info | Retired 2009 |
| 4 | Form Design | UI Component | JSF (Java Standard) | Cocoon Forms | YES | JSP, Velocity | |
| 5 | Database Persistence | ORM Support | YES | YES | Only Oracle | NOT WELL | |
| 6 | Data Representation | XML Technologies | YES | YES | YES | YES | |
| 7 | Documentation | User Guide | YES | NOT Healthy | YES | NOT WELL | |
| 8 | Stable Release | Version | Untill today | 2008 Cocoon 2.2.0 | 2009 | 2006 | |
| 9 | Technology | Open Source | YES | YES | NO | YES | |

**Fig. 6.** Functionality comparison between different system architecture

## 6.1 Design Pattern

In the proposed architecture, we have considered the Model-View-Controller (MVC) design principal. Most of the other frameworks discussed previously follow the same MVC design. In addition, the Apache Cocoon framework follows the Separation Of Concerns (SoC) principal, while our architecture considers both the Separation Of Concerns (SoC) and Inversion of Control (IoC) 6design patterns. Notably, we used JSF framework, which supports both design patterns.

## 6.2 Document Publishing Format

A statistical report is required to publish in various publication formats, such as PDF, Excel, HTML, and other possible file formats. To publish these formats, the publishing architecture needs to allow the respective parser API. Our publishing architecture applies the separation of the Concern (SoC) principle, where individual format architecture is used to generate an individual document format. The prototype shows the individual format model that gives the flexibility to introduce numerous document formats. Sequentially, Oracle XSQL page publishing architecture also supports different documentation formats, while Apache Cocoon framework does not carry the serializer to support Excel 2007 or Open XML spreadsheet formats, and Maverick framework also has lack of declaration to support the Excel format output.

## 6.3 API Upgradetion

When designing system architecture, flexibility of API upgradetion is an important concern. Regarding this, the proposed architecture is designed with a sub-function strategy, meaning the individual output format is a sub-function with the respective API. Using this process, it is very easy to upgrade the API version to get the latest API facility. For example, when considering the PDF output format, Apache FOP parse the data with the help of XSL Formatting Object. The Cocoon framework uses Apache FOP version 1.0 while the latest version is FOP 2.0. In our system, the individual sub-function is independent of the others, making it very easy to update the latest version of the API. The Oracle XSQL framework also has this flexibility.

### 6.4 Form Design

In this architecture, we used JSF to design the form. The JSF UI component provides very simple functionality to design a form while the Apache Cocoon Form design concept is much more difficult. For example, suppose you want to design 10 forms. When using the Cocoon platform, you have to define 10 form definitions, plus 10 JXTemplates with controller logic, as well as the pipeline definitions in the sitemap [.xmap] file. This process raises the productivity scalability issue. On the other hand, when using JSF, the process of form design has been simplified to negate the added step in the process. The Oracle XSQL framework also supports JSF to design user interface.

### 6.5 Database Persistence

Persistence delivers the ability of an object to remain alive throughout the lifetime of the OS (Operating System) process in which it resides. In our proposed architecture, we considered Hibernate as an object relational mapping (ORM) tool. However, Oracle XSQL only supports the Oracle database system to give the ORM flexibility. This is a big limitation for the Oracle XSQL publishing framework.

### 6.6 Documentation (User Guide)

A User Guide is an important way to understand all steps of the system. We preferred standard framework and APIs to design the architecture. Our considered framework and APIs are Java and W3S standard, which are well documented within their respected areas. In contrast, Apache Cocoon and Oracle XSQL systems documentation are not healthy enough to understand their functionality easily. The last stable Apache Cocoon release 2.2.0 came out in 2008, whereas the latest version of Oracle XSQL6 was introduced in 2009. Consequently, the Maverick systems documentation is very poor in the sense of availability, especially since the latest release occurred in 2006 and the Apache AxKit was retired in 2009.

### 6.7 Open Source Technology

Many authors have referred to the advantages of using open source software [25–27]. The first perceived advantage of open source software is a low cost and in gratis availability. With us, all publishing frameworks that we considered are open source initiatives except the Oracle XSQL Publishing framework. It is a collaboration system following the parameters of the Oracle database, therefore preventing the framework from being a truly open-sourced tool.

## 7 Conclusion

As presented through this work, we studied various demands towards statistical report publishing on the web. We proposed an architectural direction to most efficiently meet these demands. We also built a prototype implementation of a real life project (FOSTAT) in order to report our resulting solutions. The prototype has all the essential characteristics and satisfies demands by the arrangement. The parameters resulted in a reduced learning curve, which were discussed in the evaluation section, proving that this architectural model is most suitable for a developer desiring to implement a web based statistical report publishing system with much flexibility. In addition, when compare to existing solutions, this publishing architecture is a clear winner in all aspects.

**Acknowledgements.** I gratefully acknowledge the financial support towards my thesis from the German Center for Research on Higher Education and Science Studies (DZHW GmbH) [www.dzhw.eu] under the grant FOSTAT Project, Funded by The Federal Ministry of Education and Research (BMBF), Germany.

## References

1. Renear, A.H., Palmer, C.L.: Strategic reading, ontologies, and the future of scientific publishing. Science **325**(5942), 828–832 (2009)
2. Maglott, D., Ostell, J., Pruitt, K.D., Tatusova, T.: Entrez gene: gene-centered information at NCBI. Nucleic Acids Res. **33**(suppl 1), D54–D58 (2005)
3. Massand, D.: System and method for reflowing content in a structured portable document format (pdf) file. US Patent App. 12/413,486, 27 March 2009
4. Neyeloff, J.L., Fuchs, S.C., Moreira, L.B.: Meta-analyses and forest plots using a microsoft excel spreadsheet: step-by-step guide focusing on descriptive data analysis. BMC Res. Notes **5**(1), 52 (2012)
5. Dicken, H.: Data Warehouses für die Forschungsstatistik. In: HIS (2006)
6. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf. Syst. **24**(3), 45–77 (2007)
7. Salminen, A., Jauhiainen, E., Nurmeksela, R.: A life cycle model of XML documents. J. Assoc. Inf. Sci. Technol. **65**(12), 2564–2580 (2014)
8. Zhang, F., Zhang, R.: The research and application of JasperReports in project management system. In: 2009 IEEE International Workshop on Open-Source Software for Scientific Computation (OSSC), pp. 56–59. IEEE (2009)
9. Layka, V., Judd, C.M., Nusairat, J.F., Shingler, J.: Beginning Groovy, Grails and Griffon. Springer, Berlin (2013)
10. Sapre, B.S., Ulhe, P.R., Meshram, B.: Report generation system using JasperReports and SQL stored procedure. Int. J. Eng. Res. Appl. (IJERA) **2**, 984–988
11. Chlouba, T., Kminek, D.: Building open-source based architecture of enterprise applications for business intelligence
12. Innovent Solutions: Open Source Reporting Review-BIRT, Jaspersoft, Pentaho (2015). http://www.innoventsolutions.com/open-source-reporting-review-birt-jasper-pentaho.html. Accessed 20 May 2016

13. Gonzalez, E.J., Hamilton, A., Moreno, L., Mendez, J., Marichal, G., Sigut, J., Sigut, M., Felipe, J., et al.: Intelligent agents and apache cocoon for a CV generation system. In: IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2007, pp. 9–15. IEEE (2007)
14. Mazzocchi, S.: Adding XML capabilities with Cocoon. In: ApacheCon Europe (2000)
15. Introduction apache Cocoon. https://cocoon.apache.org/2.1/introduction.html. Accessed 20 May 2016
16. Noels, S.: Standards applied: using apache Cocoon and forrest. In: XML Europe (2003)
17. Singh, A.: Web based system architecture
18. Mohammed, S., Orabi, A., Fiaidhi, J., Passi, K.: Developing a Web 2.0 restful Cocoon web services for telemedical education. In: International Symposium on Applications and the Internet, SAINT 2008, pp. 309–312. IEEE (2008)
19. Apache Cocoon 2.2.0 released. http://cocoon.apache.org/1445_1_1.html. Accessed 20 May 2016
20. Eisenblatter, K., Deckarm, H., Scherer, R.: Context-sensitive information spaces for construction site applications. In: eWork and eBusiness in Architecture, Engineering and Construction, ECPPM 2006: European Conference on Product and Process Modelling 2006, Valencia, Spain, 13–15 September 2006, p. 421. CRC Press (2006)
21. XML developers kit programmers guide. http://docs.oracle.com/cd/B19306_01/appdev.102/b14252/adx_j_xsqlpub.htm. Accessed 20 May 2016
22. Melis, E., Siekmann, J.: ActiveMath: an intelligent tutoring system for mathematics. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 91–101. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24844-6_12
23. Traffic Accounting System Foundation: Apache AxKit. http://www.axkit.org/. Accessed 20 May 2016
24. Acampora, G., Loia, V.: A proposal of ubiquitous fuzzy computing for ambient intelligence. Inf. Sci. **178**(3), 631–646 (2008)
25. Lapa, J., Bernardino, J., Figueiredo, A.: A comparative analysis of open source business intelligence platforms. In: Proceedings of International Conference on Information Systems and Design of Communication, pp. 86–92. ACM (2014)
26. Bitzer, J., Schroder, P.J.: The economics of open source software development: an introduction. Econ. Open Source Softw. Dev. 1–13 (2006)
27. Sauer, R.M.: Why develop open-source software? The role of non-pecuniary benefits, monetary rewards, and open-source licence type. Oxford Rev. Econ. Policy **23**(4), 605–619 (2007)