Evaluation categorizes, technologies decision for a Composite Environmental Performance Indicators (LWC-EPI) solution

Naoum Jamous[1], Frederik Kramer[1], Gamal Kassem[1]

Abstract

Rapid environmental changes have necessitated a closer and more critical review of current environmental policies. In light of a myopic tendency to focus on short- to medium-term objectives within organizations, environmental issues are usually not given full attention. In this regard, recommendation- or dashboard-like systems are useful to increase the awareness of management as to the impact of their decisions on the environment on a more frequent and cost-effective basis. However, the role of current environmental management information systems (EMIS) in providing organizations with the information enabling them to assess the current impact of their processes and operations on the environment has been given more prominence.

Information technology is changing business in ever shortening cycles. Especially with the success of the Java platform, the standardization through the i386 hardware platform and last but not least the rivalry introduced by the Free, Libre and Open Source Software (F/LOSS) paradigm a sheer inconceivable amount of programming languages, integrated development environments, design patterns and programming frameworks have been disseminated.

In this paper, we will start with a state of the art of the available tools support providing dashboard-type solutions to control and monitor the EPI's in an organization, which may lead to reduce the gap between estimated values and current running values for environment on a more frequent basis than a mere yearly review. Furthermore, we will propose an evaluation categorizes to be used as a methodology to take the technologies decisions.

1. Introduction

Preserving the environment is a real global issue. Our ecosystem does not have borders, so we have to know that wherever we are, our actions have repercussion on the whole earth. Our planet has one atmosphere that we share and we have to protect it. Environmental issues have become an important public policy goal throughout the world, and preservation of environmental sustainability and energy efficiency are becoming new challenges for today's companies (Günther, 1998). Corporate environmental management information systems (CEMIS) can play a major role to change attitudes by providing information that enables users to assess the current environmental impact of their processes and operations.

In this paper we will investigate the state of the art of different software development approaches with respect to the LWC-EPI platform under development. In other words, we will investigate the available tools support providing dashboard-type solutions to control and monitor the EPI's in an organization, which may lead to reduce the gap between estimated values and current running values for environment on a more frequent basis than a mere yearly review. Therefore it will not recite the broad variety of terms and technologies that historically emerged but instead concentrate on those that are useful to be considered within the LWC-EPI solution. Furthermore, we will propose an evaluation categorizes to be used as a methodology to take the technologies decisions.

[1] Faculty of Informatics, Otto-von-Guericke-Universitaet, Germany.

1.1 Motivation

Information technology is changing business in ever shortening cycles. Especially with the success of the Java platform, the standardisation through the i386 hardware platform and last but not least the rivalry introduced by the Free, Libre and Open Source Software (F/LOSS) paradigm a sheer inconceivable amount of programming languages, integrated development environments, design patterns and programming frameworks have been disseminated.

The main objective of the LWC-EPI is to provide an efficient environmental information system that can help any small or medium sized enterprise "SME" in selecting, creating, calculating, comparing and reporting its environmental performance indicators "EPIs". Therefore, our challenge is to find a cost-effective solution taking into consideration the size and type of an organisation and its needs and priorities (Jamous, et al., 2010). Proposing a way that helps the enterprise to find the appropriate EPIs which address primarily those environmental impacts that are most significant and which the enterprise can influence by its operations, management, activities, products and services (Jamous, et al., 2011).

Such a tool can be considered as an analysis and reporting system which is characterized by:

1. Read-only operation.
2. Inquiring over huge amounts of data with different dimensions, characteristics and measures (Key Performance Indicators (KPI)) over the time.
3. Ad hoc querying to allow the users themselves to create specific and customized queries.
4. A flexibility to allow the users to create and change the data models. For example structures of an Environmental Performance Index (EPI) and its indicators and relationships.

There are many specific concepts and techniques which could help to develop such a tool:

- Data warehousing (cover point 1 and 2)
- Multidimensional modelling (cover point 1 and 2)
- Online Analytical Processing (OLAP) (cover point 3)
- Meta models / repositories (cover point 4)
- Service-oriented architecture (SOA) (cover point 4)

This technological diversity does not shape the challenge to determine the "state of the art" easier. In other words the appropriate state of the art is depending on project preconditions such as:

- The already existing software.
- Information system assets.
- The strategy of the underlying business.
- The scale of the intended application.
- The available developer skills.

Our primary research objective is to determine the state of the art on software development architectures for a specific development project and its constraint. In order to achieve that goal we need to model a general comparison framework containing criteria to evaluate technologies based on concrete project requirements. The following chapter will thus introduce those criteria.

2. State of the Art Platform Solutions

In order to determine the state of the art we have to briefly specify what the term "state of the art" means. It stems from jurisdiction and more specifically patent law. It is used to determine whether an invention is new with respect to what already existed before. The European Patent Convention only indirectly defines the term state of the art however. It says in its 56 article (EU-Patent-Convention, 2007): "An invention shall be considered as involving an inventive step if, having regard to the state of the art, it is not obvious to a person skilled in the art."

Following this definition it is obvious that every programming language that has been disseminated or proposed in the internet is already part of that construct. Given the huge amount of programming languages and paying no attention to patterns, tool support and alike, it is immediately obvious that we need to shrink in the meaning of "state of the art" within the scope of this article. We will do so by adding a new part to the phrase such as: "State of the art within the scope of this research project means whatever technology is available and can be used to build up enterprise ready web applications". Within this improved specification of the term we put two additional constraints.

First we would like an application developed with state of the art technologies to be enterprise ready. Enterprise readiness means that any used technology can match important enterprise criteria by its own. It must be clearly said that enterprise readiness depends at least on project if not company preconditions. Furthermore selecting enterprise ready software technologies does not necessarily mean the resulting software is enterprise ready as well. Since we are targeting basically SMEs, a second precondition is that the technology must allow developing a pure web application without putting many restrictions on the client side of the application. This preconditions stem from the fact that an ideal application should comply with the aims of having as few as possible restrictions on the users accessing the platform. In order to achieve this goal we define evaluation categories in the next paragraph.

To enhance our state of the art, we started with comparing the web application frameworks products' available in the market. We selected 13 products "or tools" of web application frameworks based on different technologies. The tables 1 to 13 summarize the result of this comparison.

2.1 PHP framework

Product Name	**Zend**
Programming Language	PHP 5
Architecture Pattern	Adopts MVC architecture
Ajax	Support Ajax, PDF generation, e-mail communication, and search.
i18n/ l10n	Yes
ORM	Table and Row data gateway.
SOA	Web services are an integral part of Zend Framework.
Cashing	Support data Caching
License	BSD License
General remarks	• Support Rapid Application Development (RAD) • ACL-based and RBCA-based, plugins • Authentication, Authorization, and Session management

Table **1:** Zend (Zend, 2006)

Product Name	**CakePHP**
Programming Language	PHP 5
Architecture Pattern	Adopts MVC architecture
Ajax	Helpers AJAX, Javascript, XML, RSS
i18n/ l10n	Yes
ORM	• Active record pattern (CakePHP 1.x) • Data Mapper Pattern (CakePHP 2.x)
SOA	Helpers for HTML, Forms, Pagination, AJAX, Javascript, XML, RSS and more

Cashing	Flexible Caching
License	BSD License
General remarks	• Security, Session, and Request-Handler Components • HTTP Authentication via Security Component • Access Control Lists and Authentication • CSRF protection via Security Component • Router for mapping urls and handling extensions • Utility classes for working with Files, Folders, Arrays and more

Table **2:** CakePHP (cakephp, 2005)

Product Name	**Drupal**
Programming Language	PHP 5
Architecture Pattern	Mini-MVC / Push & Pull MVC
Ajax	Yes
i18n/ l10n	Yes
ORM	Optional modules
SOA	Web services
Cashing	Memcache, APC, Varnish
License	GNU GPL
General remarks	• DB migration framework • Simple Testing framework

Table **3:** Drupal (Drupal, 2001)

Product Name	**CodeIgniter**
Programming Language	PHP 5
Architecture Pattern	• Adopts MVC architecture • Modified active record pattern
Ajax	Prototype "jQuery/jQuery UI"
i18n/ l10n	Yes
ORM	Full Featured database classes Active Record Database Support
SOA	XML-RPC Library
Cashing	Full Page Caching
License	BSD License
General remarks	• Security and XSS Filtering • Flexible URI Routing • File Uploading Class • Support for Hooks, Class Extensions, and Plug-in

Table **4:** CodeIgniter (codeigniter, 2001)

Product Name	**Yiiframework**
Programming Language	PHP 5
Architecture Pattern	Adopts MVC architecture
Ajax	AJAX-enabled widgets
i18n/ l10n	Yes
ORM	DAO and Active Record
SOA	Supports automatic generation of complex WSDL service specifications and management of Web service request handling.
Cashing	Supports data caching, page caching, fragment caching and dynamic content.
License	BSD License
General remarks	• Built-in authentication support. It also supports authorization via hierarchical RBAC. • Purely object-oriented.

Table **5:** Yiiframework (yii, 2008)

2.2 ASP.Net framework

Product Name	**ASP.NET MVC**
Programming Language	ASP.NET
Architecture Pattern	Web application framework that implements the MVC
Ajax	Support Ajax
i18n/ l10n	No
ORM	Independent ORM
SOA	Web application framework
Cashing	Cashing framework
License	MS-PLicense
General remarks	• Unit testing framework • ASP.NET Forms Authentication.

Table **6:** ASP.NET MVC (ASP.NET, 2007)

Product Name	**MonoRail**
Programming Language	ASP.NET
Architecture Pattern	MVC: Active record pattern
Ajax	Prototype
i18n/ l10n	Yes
ORM	Active record pattern
SOA	--
Cashing	Cashing framework
License	Apache License
General remarks	• Formerly called Castle on Rails • Unit testing framework • ACL-based , ASP.NET Forms Authentication.

Table **7:** MonoRail (MonoRail, 2010)

2.3 Web application framework

Product Name	**Grok**
Programming Language	Python
Architecture Pattern	Use Pure MVC Python version
Ajax	Yes
i18n/ l10n	Yes
ORM	OODBMS
SOA	--
Cashing	Cashing framework
License	ZPL
General remarks	• Based on Zope Toolkit • OODBMS called ZODB, SQLAlchemy, Storm

Table **8:** Grok (Grok, 2008)

Product Name	**Camping**
Programming Language	Ruby
Architecture Pattern	Organized in MVC application
Ajax	No
i18n/ l10n	No
ORM	Active record pattern
SOA	web application framework
Cashing	No
License	MIT
General remarks	• Light weight code “less than 4kB” • No security framework • Testing via Mosquito

Table **9:** Camping (Camping, 2009)

2.4 Java-based web application framework

Product Name	**Spring**
Programming Language	Java
Architecture Pattern	• Has its own MVC framework called "Spring MVC". • Spring MVC is considered the better solution in many developer forums compared to struts
Ajax	Yes
i18n/ l10n	Yes
ORM	Abstraction layer for all kind of persistence frameworks (incl. Hibernate)
SOA	• Spring Web Services. • Supports SOAP-based and RESTful Web services.
Cashing	ehcache.etc.
License	Apache

General remarks	• Inversion of Control concept • Independent code entities • Has its own security framework • Decoupled configuration

Table **10:** Spring (Spring, 2002)

Product Name	**Apache Struts**
Programming Language	Java
Architecture Pattern	• Provides its own MVC architecture • Presentation code is written with JSP templates
Ajax	Yes
i18n/ l10n	Yes
ORM	Supported
SOA	--
Cashing	--
License	Apache
General remarks	• Java-based open-source web-application framework. • Focus on MVC pattern.

Table **11:** Apache Struts (Struts, 2000)

Product Name	**Tapestry**
Programming Language	Java
Architecture Pattern	Provides its own MVC architecture.
Ajax	Yes
i18n/ l10n	Yes
ORM	tapestry-hibernate module
SOA	--
Cashing	--
License	Apache
General remarks	• Java-based open-source web-application framework. • Focus on MVC pattern • Implements Inversion of Control concept • Encourages the implementation of small code entities

Table **12:** Tapestry (Tapestry, 2010)

2.5 Java-based application server

Product Name	**JBoss**
Programming Language	Java
Architecture Pattern	Supports JSP as well as JSF

Ajax	Yes
i18n/ l10n	Yes
ORM	Supports JPA and Hibernate
SOA	Support JAX-WS and JAX-RS specification
Cashing	JbossCache, EHCache
License	GNU LGPL
General remarks	• Integrated Hibernate • Supports JPA specification

Table **13:** JBoss (Jboss, 2010)

Product Name	**WebObjects**
Programming Language	Java
Architecture Pattern	• Visual editor for creating the UI • Based on MVC
Ajax	Yes
i18n/ l10n	Yes
ORM	Enterprise Objects Framework (EOP)
SOA	--
Cashing	Supported
License	Proprietary
General remarks	• Java-based application server and web-application framework. • Closely related to Apple products • Introduces its own mapping from database tables to business objects

Table **14:** WebObjects (webobjects, 2005)

3. Evaluation categories

3.1 Programming Paradigm

Even if that question has been discussed in ample scientific disputes since its dissemination in the 1960s and almost all modern software development technologies focus on the object oriented paradigm it has to be kept as an important decision category in the model.

This is because object orientation is not just a matter of technology selection but also a software development concern itself. While not being easy it is possible to develop procedural code instead of object oriented code in principle even in environments that focus on pure object orientation. Moreover there are arguably some cases were procedural programming is more suitable to solve problems than object orientated programming is. However almost all of those cases are related to memory usage, run time performance and embedded technologies instead of playing a visible role in enterprise web application development.

3.2 Architectural and Design patterns

In the history of software engineering it became obvious that similar software engineering problems that arose from different types of business domains can often be grouped into and addressed by applying the same or similar processes and behaviour (in short patterns) in the design phase of an application.

Using appropriate design patterns supports developers to develop more maintainable and also more secure software. Patterns like the famous Model, View, Controller pattern hence try to methodologically separate concerns such as database representation, business logic and user presentation from each other. Patterns like Service-oriented architectures (SOA) address organisation wide and even inter-organisational exchange and reuse of business process portions (in this case known as "services").

3.3 Programming Language

With the rise of the internet an unconceivable competition of programming languages and paradigms has begun. Whereas in the history of information technology the programming language was often predetermined by the hardware platform used, the standardisation towards the i386 platform widely opened the battlefield of technology to software and more specifically to programming languages.

As already mentioned in the introduction, today there is a zoo of programming languages, some of which addressing a generally broad user-community (like PHP or Java) and variety of potential applications (e.g. Java, C#), some others rather special needs (e.g. Prolog, Smalltalk, Ruby). While determining the state of the art with respect to the defined constraints the subset of programming languages has been stripped down. We only consider technology to comply with our state of the art term, if they are in principle well suited to develop enterprise ready application and have strength in web application development.

3.4 Development Environment

Tooling support is crucial. If there might be some sort of fancy design paradigm, that could be best implemented in programming language "A" to match the goals of LWC-EPI but the tooling for this combination is weak or not existing, one might end up with lots of duplicate and inefficient work.

Higher cost of implementation and maintenance would be a severe economic drawback. Last but not least missing tooling support seriously hampers the dissemination of the technology to create SISE. This situation must be avoided by using a weighted criteria model.

3.5 Strategic perspective

In cases where technology selection is necessary often the strategic perspective is disregarded. This often happens if technicians decide on their own about technology without shedding an eye on strategic implications. Regularly this hampers the success of enterprises or its projects. With regard to the LWC-EPI's constraints we define the following strategic criteria to be evaluated and support the technology decision before it can finally be taken.

A technology must be sustainable and enough personnel must be available that is able to develop on the basis of that technology. The technology must be mature and the maturity must be proven in industry scale web application projects. The learning curve must be reasonably steep. A technology that requires an unreasonably costly training shall hence not be selected. The technology must be flexible. Flexibility for example requires good and comprehensive documentation of basic concepts of a technology. Technological boundaries must thus be unveiled by appropriate investigations. An ideal technology must be portable. This means it should not be delimited to a special set of vendor specific it-infrastructure for example.

A lot of programming languages, tools, as well as architectural models (e.g. RCP) have been released under F/LOSS licenses, others follow an almost entirely closed stack (e.g. MS Visual Studio). This has to be addressed and investigated as well.

4. Technology decision

4.1 Scoring model

A Scoring model is a formula that assigns credits, points, or weights based on known information to predict an unknown future outcome (scoringmodels, 2007) or supports a decision making process (Kleijnen, 1980). In our case, we will use the scoring model to set the elements and factors which will help in choosing the technologies and solutions to be used in the LWC-EPI platform.

As we mentioned before, the main objective of the LWC-EPI is to provide an efficient environmental information system that can help any small or medium sized enterprise "SME" in selecting, creating, calculating, comparing and reporting its environmental performance indicators "EPIs".

4.2 Value benefit analysis

In our decision making process, we will use the value benefit analysis which is a useful method for preparing decisions systematically (Schulze, 2006). The two main advantages of using the value benefit analysis are:

- We can take non-quantitative criterions into consideration,
- It could cover many point of views (developers, users, stakeholders, etc.), so it's a flexible multidimensional method.

To apply the value of benefit analysis we need five steps:

1. Listing all alternatives.
2. Setting the comparison measures.
3. Establish weights for the measures.
4. Establish factors to rate the alternatives based on different measures.
5. Determine the value of benefit.

4.2.1 Listing all alternatives

In this step we list most of possible solutions for developing the LWC-EPI platform. Starting to develop the platform, we should decide what and which solutions or technologies we want to use. As a first step we should decide which programming language we will use. As alternatives we will take six well known programming languages, table 15 shows the six alternatives:

Programming languages	Java	PHP	C++	.Net	Python	Ruby

Table **15:** Programming languages for LWC-EPI

4.2.2 Setting the comparison measures

In this step, we set the measures that we want to use for the evaluation. Here we select the most important criterions or measures. To do so, we assign an importance label for each potential measure that can be used like "high" for very important measures, "middle" for important ones and "low" for not very important measures for our case. Table 16 shows nine measures with the importance label for each.

Measure ID	Measure	Importance label
01	Platform independency	high
02	Data base connector / Persistency / DAO	high
03	Web application readiness	high
04	RAD "Rapid Application Development" framework	high
05	Libraries	middle
06	Documentation	middle
07	Tooling	middle
08	Enterprise readiness "maturity"	low
09	Future Development	low

Table **16:** measures' importance labels

4.2.3 Establish weights for the measures

After selecting the measures to be used, we have to evaluate the relations among them. To do that we arranged a matrix table in which the rows and the columns represent the measures. After assessing all measures in pairs based on the importance label, we calculated each measure weight factor by summing up all the numbers we assigned for each measure and divide this sum by the total sum over all single sums in the matrix lines. As matrix table filling's criteria, we give "0" if the compared measure is less important than the other measure; we put "1" if they have the same importance label and "2" if the compared measure is more important than the other. Table 17 shows the measures' matrix.

Measure ID	01	02	03	04	05	06	07	08	09	Sum	**Weight factor**
01	-	1	1	1	2	2	2	2	2	13	0.180
02	1	-	1	1	2	2	2	2	2	13	0.180
03	1	1	-	1	2	2	2	2	2	13	0.180
04	1	1	1	-	2	2	2	2	2	13	0.180
05	0	0	0	0	-	1	1	2	2	6	0.084
06	0	0	0	0	1	-	1	2	2	6	0.084
07	0	0	0	0	1	1	-	2	2	6	0.084
08	0	0	0	0	0	0	0	-	1	1	0.014
09	0	0	0	0	0	0	0	1	-	1	0.014
Total sum										72	1

Table **17:** measures' matrix

4.2.4 Establish factors to rate the alternatives based on different measures

In this step we evaluate the fulfilment of the measures for each alternative. To do so we will use the presented scale in table 18:

Grade	NA	1	2	3	4	5
Rate	No answer	Very good	Good	Satisfactory	Sufficient	Not sufficient

Table **18:** Rating scale

Such a rating scale allows us to rate the alternatives we listed in step 1. Table 19 shows the result of the rating which was done by 10 experts from different organizations "Enterprises, Universities and research centres". To get more accurate result, we excluded all "NA's" answers after asking the experts to use it for the alternatives or measures which they are not familiar with.

Programming Language	Java	PHP	C++	.Net	Python	Ruby
Measure ID	Rating					
01	1.00	0.86	0.52	0.48	0.80	0.80
02	0.96	0.80	0.80	0.85	0.87	0.75
03	0.88	0.94	0.48	0.88	0.84	0.88
04	0.84	0.81	0.50	0.74	0.83	0.84
05	0.98	0.80	0.73	0.77	0.89	0.76
06	0.94	0.86	0.71	0.86	0.77	0.60
07	0.96	0.69	0.84	0.88	0.70	0.65
08	0.96	0.60	0.84	0.95	0.73	0.52
09	0.69	0.74	0.68	0.83	0.77	0.72

Table **19:** Alternatives' rating

4.2.5 Determine the value of benefit

The value of benefit is the result of combining step 4 "rating scale" results and step 3 "measures' weights" results. To calculate the value of benefit we multiply the rating points with the corresponding weights and sum these products up for each alternative. To make it more visible, we will multiply the results by 100. At the end, the alternative with the highest sum (value of benefit) will be suggested to be selected.

Table 20 shows the value benefit of each programming language for each measure and in the last row we can see the value of benefit for each alternative in general.

As a result, Java programming language got the highest value of benefit followed by PHP. So we recommend using one of the two choices for the platform development.

Programming Language		Java		PHP		C++		.Net		Python		Ruby	
Measure ID	Weight factor	Rating	Value of Benefit	Rating	Value of Benefit	Rating	Value of Benefit	Rating	Value of Benefit	Rating	Value of Benefit	Rating	Value of Benefit
01	0.180	1.00	18	0.86	15.48	0.52	9.36	0.48	8.64	0.80	14.4	0.80	14.4
02	0.180	0.96	17.28	0.80	14.4	0.80	14.4	0.85	15.3	0.87	15.66	0.75	13.5
03	0.180	0.88	15.84	0.94	16.92	0.48	8.64	0.88	15.84	0.84	15.12	0.88	15.84
04	0.180	0.84	15.12	0.81	14.58	0.50	9	0.74	13.32	0.83	14.94	0.84	15.12
05	0.084	0.98	8.232	0.80	6.72	0.73	6.132	0.77	6.468	0.89	7.476	0.76	6.384
06	0.084	0.94	7.896	0.86	7.224	0.71	5.964	0.86	7.224	0.77	6.468	0.60	5.04
07	0.084	0.96	8.064	0.69	5.796	0.84	7.056	0.88	7.392	0.70	5.88	0.65	5.46
08	0.014	0.96	1.344	0.60	0.84	0.84	1.176	0.95	1.33	0.73	1.022	0.52	0.728
09	0.014	0.69	0.966	0.74	1.036	0.68	0.952	0.83	1.162	0.77	1.078	0.72	1.008
Total		92.74		83		62.68		76.68		82.04		77.48	

Table **20:** Alternatives' rating

5. Conclusion and future steps

In this paper we investigated the state of the art of different software development approaches with respect to the LWC-EPI platform under development. The was to study and compare the available tools support providing dashboard-type solutions that could be used to control and monitor the EPI's in an organization, which may lead to reduce the gap between estimated values and current running values for environment on a more frequent basis than a mere yearly review. Therefore, we clarified the relevant terms and technologies that are useful to be considered within the LWC-EPI solution. Furthermore, we proposed an evaluation categorizes to be used as a methodology to take the technologies decisions. Afterwards, we used the scoring method to compare and then select the appropriate programming language to be used for the LWC-EPI platform development.

The next step, after conducting a complete and detailed value benefit analysis on the used programming languages, will be to complete the LWC-EPI platform specifications "server, DBMS, web services and the development environment".

This step will be done by selecting appropriate technologies based on the result we got. In addition, we will take the feedback of experts involved in different type of organizations (Enterprises, Universities and Research Centres) as well as our own diverse experience from various IT-projects into account.

6. Acknowledgment

Part of this research has been funded under the EC 7th Framework Program, in the context of the OEPI project (748735). The authors thank for the support.

7. Bibliography

ASP.NET mvc [Online] // www.asp.net. - 2007. - 2010. - http://www.asp.net/mvc.
cakephp cakephp [Online]. - 2005. - 2010. - http://cakephp.org/.
Camping [Online] // whywentcamping.com. - 2009. - 2010. - http://whywentcamping.com/.
codeigniter codeigniter.com [Online]. - 2001. - 2010. - http://codeigniter.com/.
Drupal Drupal [Online]. - 2001. - 2009. - http://drupal.org/home.
EU-Patent-Convention European Patent Convention, Article 56 [Report]. - 2007.
Grok Grok [Online] // grok.zope.org. - 2008. - 2010. - http://grok.zope.org/.
Günther Oliver Environmental Information Systems [Book]. - Berlin : Springer, 1998. - ISBN: 3-540-60926-1.
Jamous Naoum [et al.] Light-weight composite environmental performance indicators (LWC-EPI) concept. In: Information technologies in environmental engineering [Book Section] // Information Technologies in Environmental Engineering, New Trends and Challenges / book auth. Golinska Paulina and Fertsch Marek / ed. Marx Gómez Jorge. - Berlin : Springer, 2011. - Vol. 3. - http://www.springerlink.com/content/j86665463452kt25/. - ISBN 978-3-642-19535-8.
Jamous Naoum [et al.] Proposed Light-Weight Composite Environmental Performance Indicators (LWC-EPI) Model [Conference] // the 24st International Conference on Informatics for Environmental Protection (EnviroInfo2010). - Bonn & Cologne : shaker Verlag, 2010. - http://www.shaker.de/Online-Gesamtkatalog-Download/2011.01.20-19.39.57-141.44.30.154-rad5B823.tmp/3-8322-9458-9_INH.PDF. - ISBN 978-3-8322-9458-8.
Jboss Jboss [Online] // www.jboss.com. - 2010. - 2010. - http://www.jboss.com/about/.
Kleijnen Jack P C Scoring Methods, Multiple criteria, and utility analysis [Journal] // ACM SIGMETRICS Performance Evaluation Review. - New York : [s.n.], 1980. - 3 : Vol. 9. - ISSN: 0163-5999.
MonoRail monorail [Online] // www.castleproject.org. - 2010. - 2010. - http://www.castleproject.org/monorail/.
Rautenstrauch Claus Betriebliche Umweltinformationssysteme : Grundlagen, Konzepte und Systeme ; mit 8 Tabellen. [Book]. - [s.l.] : Springer, 1999. - ISBN: 3-540-66183-2..
Schulze Lothar Basics planning of logistical systems [Online] // Global campus 21. - 2006. - http://gc21.inwent.org/ibt/en/ilt/ibt/regionalportale/sadc/inhalt/logistics/module_03/61_value_benefit_analysis.html.
scoringmodels www.scoringmodels.com [Online]. - 2007. - 2010. - http://www.scoringmodels.com/.
Spring Spring [Online] // springsource.org. - 2002. - 2009. - http://www.springsource.org/.
Struts Apache struts [Online] // www.apache.org. - 2000. - 2009. - http://struts.apache.org/.
Tapestry tapestry [Online] // www.apache.org. - 2010. - 2010. - http://tapestry.apache.org/.
webobjects webobjects [Online] // developer.apple.com. - 2005. - 2010. - http://developer.apple.com/legacy/mac/library/navigation/index.html?filter=webobjects.
yii yiiframework.com [Online]. - 2008. - 2010. - http://www.yiiframework.com/.
Zend Zend framework [Online]. - 2006. - 2010. - http://framework.zend.com/home.