A First Step to Cloud Infrastructure Cost Estimation in Early Stages of Web Development

Rubén Martín, Juan Carlos Preciado, Roberto Rodriguez-Echeverria,
José María Conejero and Fernando Sánchez-Figueroa
QUERCUS Software Engineering Group, School of Technology. University of Extremadura, Spain

Keywords: Web Engineering, Web Design, Costs, CRUD.

Abstract: Currently, the cost of cloud computing infrastructures for Web applications is calculated in deployment and production phases. Recently, the scientific community is offering several methodologies to calculate the most suitable infrastructure at these stages. On the other hand, the Model Driven Web Engineering is taking advantages of code generation from Design level. With both concepts in the scene, in this work we show the first steps toward an approach to estimate the production costs in cloud computing infrastructures at Design phase.

1 INTRODUCTION

In the last years, cloud computing infrastructure has become the most used solution for deploying Web applications, mainly due to the flexibility, agility and availability capabilities that it offers (Fu, Cang, Zhu, and Deng. 2015). Cloud computing enables Web developers to use remote hosting services to improve onsite infrastructure. In that sense, the organization systems can be specified at a deep level of detail according to the service and quality level agreements that determine the kind of servers, its arrangement and its scalability options. In that sense, there are many factors that impact the pricing structures, mainly related to the resources used in each moment. As an example, high availability, high data volume and high number of concurrent users are key factors to be considered in data intensive Web applications deployment (Suresh and Sakthivel. 2016). Thus, since cost is of utmost importance for the infrastructure, getting an optimal cloud infrastructure is critical, especially when this kind of Web applications came into this scenario.

Concretely, the definition of a deployment infrastructure for a Web application, previously implemented, is based on a general specification of a service level agreement (SLA) that is usually defined in terms of (Andzrejak, D. Kondo, and S. Yi, 2010) (Cheng Tian, Ying Wang, Feng Qi, and Bo Yin, 2012): (1) Network latency, (2) host CPU and throughput, (3) memory and (4) storage.

On the other hand, data intensive Web applications development has been widely studied by Model Driven Web Engineering community (MDWE). Among the many benefits provided by MDWE approaches, such as IFML/WebRatio (Brambilla, M., Fraternali, P., 2014) or, OOHDM (Rossi, G, Pastor, O, Schwabe, D, Olsina, L., 2007), it is worth to mention the automatic generation of the final application code from the design so that this phase (design) becomes the most important step in the software development process. Other advantages provided by these approaches include productivity improvements, an important increase in software quality or a reduction in costs to adapt the system to changes in requirements. However, these approaches do not support the identification or estimation of production costs for the Web application in a selected cloud infrastructure yet. This identification needs to be made in post-design software development lifecycle stages, such as performance-testing phase (Huihong He, Zhiyi Ma, Xiang Li, Hongjie Chen and Weizhong Shao. 2012) neglecting the benefits of an earlier identification.

As an example, the identification at design level of design decisions that may have a relevant impact in the infrastructure costs may involve changes that

Martín, R., Preciado, J., Rodriguez-Echeverria, R., Conejero, J. and Sánchez-Figueroa, F.
A First Step to Cloud Infrastructure Cost Estimation in Early Stages of Web Development.
DOI: 10.5220/0006393904370443
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 437-443
ISBN: 978-989-758-246-2

would be accomplished before the final system has been generated. The identification of these decisions once the final system has been generated would imply higher cost changes (Barry W. Boehm, 1981).

Given this opportunity, we have formulated the following research question: *can we estimate at design phase the costs of production for a given Web System, developed with a specific MDWE approach and given a service quality requirements specification, to determine a Cloud deployment infrastructure?*

The main goal of this paper is to present a first step towards an approach for the definition of a cloud computing cost estimation model for Web applications during the design phase. In other words, a design-time evaluation of the infrastructure needed at the next production stage to cover a certain quality of service for the application. This approach would reduce the impact of changes due to decisions about the capabilities and costs of the cloud infrastructure on the development process. As a first step to carry out this estimation, this work presents an analysis of the throughput times of an application, based on different design decisions, in order to have more information to anticipate the impact of the design on the infrastructure.

This paper is structured as follows. In Section 2 we gather the steps and data used to carry out this first estimation as a function of throughput time. Section 3 analyses the data obtained. Finally, Section 4 presents the conclusions and future work.

2 WORKING ENVIRONMENT

In order to make concrete the conceptual framework that the work presented here relies on, we have analyzed different MDWE approaches that were mature enough to be used by industry (Toffetti, G., Comai, S., Preciado, J. C., Linaje, M., 2011). Among the current approaches, it is worth to mention IFML (Interactive Flow Modelling Language) (Brambilla, M., Fraternali, P., 2014), an OMG standard that allows the design and development of data intensive Web applications. This standard has also become a reference for the industry in the data intensive web applications development area. WebRatio, the case tool that supports the development of applications by using IFML, allows managing and validating IFML models but also the automatic generation of the final code of the application based on a particular J2EE target platform.

Thus, based on the usage of IFML, our main research question, introduced in previous section, was refined as follows: *can we estimate at design phase the costs of production for a given Web System, developed with IFML/WebRatio and given a service quality requirements specification, to determine a Cloud deployment infrastructure?*

As a first step to answer this question, in this work we focus on assessing how different design and production parameters (independent variables of our study) impact the response time (dependent variable) of a Web application.

To analyze the impact of the independent variables treatment in the study, we have defined a canonical design that will be used in all the assessments. This design consists of an IFML navigation model composed of a set of CRUD operations (Create, Read, Update, Delete). Note that CRUD operations represent the tasks that are more frequently repeated in IFML designs and, thus, the operations with a higher activity load in data intensive Web applications (Rodriguez-Echeverria R., M. Conejero J., Preciado J. and Sanchez-Figueroa F., 2016). The model that has been designed follows the next pattern: first, a Create operation is executed; second, a Read is performed; next, an Update; and, finally, a Delete.

Once the core design has been specified, the independent variables related to the design aspects that may affect its response time have been defined. Concretely, the variables are the next: (a) number of attributes in the data entity that the operation is performed over (in this case, we have considered values of 1, 10, 20, 30, 40, 50); (b) persistence type considered for the operation (data stored into the data base, data as session variable or data as application variable at memory level). For the sake of simplicity, the attributes of the data entities of the design have been defined just as string. For the persistent entities database, the mapping has been defined for a PostgreSQL 9.5.4 database.

On the other hand, the production independent variables refer to the technical characteristics of the deployment infrastructure that may affect the response time of the application. In this case, the

independent variables considered are: (c) computational capacity of the server instance were the tests are performed (two different machines are used: an EC2 t2.micro Amazon Web Services with a 2,5 GHz Intel Xeon processor and 1 GB of RAM memory; an EC2 t2.small Amazon Web Services with a 2,5 GHz IntelXeon processor and 2 GB of RAM memory); (d) the number of concurrent users that may launch a particular operation in the Web application (in our case, this parameter may have the values 1, 5 and 10).

Table 1 summarizes the independent variables considered in the study together with the values that they may have.

Table 1: Independent variables for the study.

Category	Variable	Range of values
Design	CRUD operation	2000 operations per type
Design	Number of attributes of the entity	1, 10, 20, 30, 40 or 50
Design	Persistence	Application, session or data base
Production	Computational capacity	EC2 t2.micro Amazon Web Services EC2 t2.small Amazon Web Services
Production	Number of concurrent users	1, 5 or 10

In order to have a relevant set of data and to be able to dismiss abnormal results, each combination of values for the independent variables has been repeated 2000 times. Thus, we have executed 2000 create operations, then, 2000 reads and so on. So, the behavior of the design with the CRUD operations has been studied in each infrastructure (the two machines used in the study) with a different number of users (3 different sets) and by using different set of attributes (6 options) and persistence types (3 options).

3 FIRST ANALYSIS

Given the great amount of data and the existence of different combinations for the testing groups, all the results have been represented by means of a 5-dimension ROLAP (Konstantinos Morfonios, Stratis Konakas, Yannis Ioannidis, and Nikolaos Kotsis., 2007) cube. Those 5 dimensions are: operation type (C, R, U, D), persistence type (data base, session, or application), attribute number (1, 10, 20, 30, 40, 50), simultaneous user number (1,5), deployment machine (AWS t2.micro, or AWS t2.small) and theirs respectably RDS storage using PosgreSQL 9.5.4 with 20GB (AWS db.t2.micro, or db.t2.small).

Once finished the different tests, all the resulting data have been processed by R to assess their reability/confidence, i.e., how similar the execution times of the same test throughout its 2.000 repetitions are. We have applied k-means (MacQueen, J, 1967) to identify that the behavior was homogeneous in the great majority of the data (>91%) and to be able to discard not relevant outliers. Therefore, given a homogeneity coefficient greater than 91%, we can derive a relevant mean execution time for every operation. Then, these mean times can be used in the design phase of an application to estimate its production costs. By example, the Tables 1, 2, 3 and 4 show the mean times (in miliseconds) for the CRUD operations considered, given 1 user.

Table 2: Mean execution times CREATE.

CREATE	Attributes 1	10	20	30	40	50
Application_small	0,72	0,93	0,64	1,00	1,02	1,09
Session_small	0,82	0,94	1,26	0,82	0,89	1,00
Persistent_small	23,76	98	190,53	273,89	264,91	304,00
Application_micro	0,69	1,91	1,96	2,00	2,08	2,60
Session_micro	0,75	1,26	1,88	2,01	2,24	2,77
Persistent_micro	32,58	330,89	332,09	361,58	471,87	620,99

Table 3: Mean execution times READ.

READ	Attributes 1	10	20	30	40	50
Application_small	0,00	0,00	0,00	0,00	0,00	0,00
Session_small	0,00	0,00	0,00	0,00	0,00	0,00
Persistent_small	1,10	1,20	1,22	1,30	1,32	1,36
Application_micro	0,29	1,37	1,45	1,42	1,43	1,49
Session_micro	0,65	1,32	1,41	1,49	1,50	1,60
Persistent_micro	0,89	1,09	1,09	1,18	1,27	1,38

Table 4: Mean execution times UPDATE.

UPDATE	Attributes 1	10	20	30	40	50
Application_small	5,21	14,60	17,52	23,83	29,28	35,43
Session_small	7,11	16,32	18,69	25,43	32,72	38,27
Persistent_small	18,62	79,26	102,54	176,55	216,88	220,98
Application_micro	3,22	209,58	271,14	272,82	339,31	498,04
Session_micro	2,92	217,25	265,79	289,68	329,91	498,75
Persistent_micro	22,37	169,28	197,04	242,97	289,94	307,57

Table 5: Mean execution times DELETE.

DELETE	Attributes 1	10	20	30	40	50
Application_small	0,00	0,98	1,05	1,05	0,99	1,10
Session_small	0,00	1,03	1,08	1,08	1,10	1,11
Persistent_small	31,22	101,44	124,00	134,96	151,72	275,00
Application_micro	1,01	12,47	10,18	11,55	22,77	36,63
Session_micro	1,10	12,51	10,00	11,41	22,76	36,64
Persistent_micro	41,18	303,30	314,36	372,20	416,23	443,78

Figures 1, 2, 3 and 4 visually present the results for the four CRUD operations, considering just 1 user, for all the different combinations of values from the dimensions: machine, persistence and attribute number.

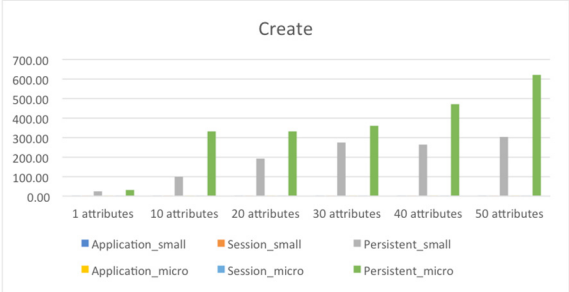

Figure 1: Mean time values plot for every case. CREATE.

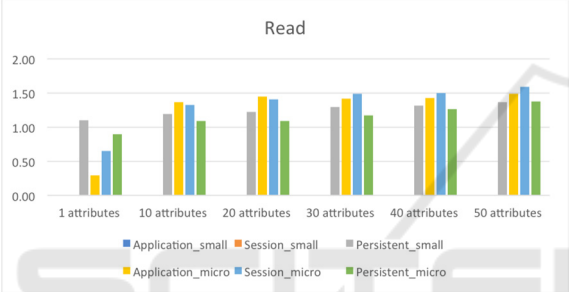

Figure 2: Mean time values plot for every case. READ.

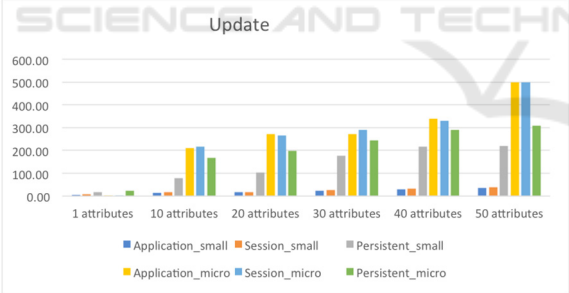

Figure 3: Mean time values plot for every case. UPDATE.

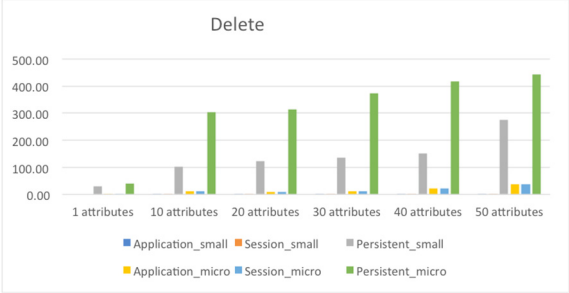

Figure 4: Mean time values plot for every case. DELETE.

From the data obtained, here textually and visually given, we can observe several interesting facts. On the one hand, CREATE and DELETE operations (Figures 1 and 4) behave properly with persistence at memory level (Session and Applications variables), i.e., both operations take really low portion of time for its execution, the memory size seems to be not relevant to perform them.

On the other hand, for both CREATE and DELETE operations when working on persistence at data base level the time became a key factor and we can also noticed that the size of the memory plays a relevant role in these cases, being a little bit higher in the AWS t2.micro/db.t2.micro case. In addition, when the number of attributes increases, it has a significant impact on performance.

Regarding READ operations (Figure 2), the scenario range comprises from 0 up to 1,6 milliseconds. We can observe that the execution of READ operations involves a really low portion of time but, in a particular way, the memory size is really relevant when reading from memory level. The number of attributes from 1 up to 50 does not significantly impact executions times. For all cases, but mainly in READ case, millisecond seems to be a measure with not enough level of detail to perform the study, at least nanoseconds level should be considered.

As also expected, better hardware features set (more RAM, in this case) of the deployment machine implies a relevant reduction on the mean execution time for every considered case of the UPDATE operation (Figure 3). Conversely, a greater number of attributes implies higher execution times. Additionally, it may be noted the high impact of the RAM availability when data entities are stored inside session or application scopes.

From the first data here obtained we can make decisions in order to optimize the execution times of the CRUD operations groups at design phase and we can give the first step to estimate the throughput time to convey the general services level specification. For it, once we have the operation times estimated from this initial study for the different combination, it is possible to assign a concrete value to each particular operation that conforms an operation chain in the business logic of an application at design phase. We define an Operation Chain as the set of all of the CRUD operations that are launched sequentially and that must be executed from the first to the last, as if it

was a transaction. We do it by adding up all the operation times of each of the operations involved in such operation chain.

$$\max_{j:C_j \epsilon OC} \sum_{i:O_i \epsilon C_j} time(O_i)$$

The formula above obtains the estimated execution time of the longest operation chain (C_j) by selecting the maximum value from the set of estimated execution times of all the Operation Chains (OC) in the design. The function time returns the mean time of every operation (O_i) inside the Operation chain C_j given its type, the number of attributes in the involved data entity, and its type of persistence.

For instance, suppose the application needs two operation chains. In the first one (OP1 – Figure 5) the whole chain is composed sequentially by a CREATE (10 attributes at data base) + CREATE (40 attributes at session) + READ (40 attributes at session) + UPDATE (10 attributes at data base) + DELETE (40 attributes at session). For the second one (OP2 – Figure 6) the whole chain is composed sequentially by a CREATE (20 attributes at data base) + UPDATE (20 attributes at data base) + DELETE (20 attributes at data base).

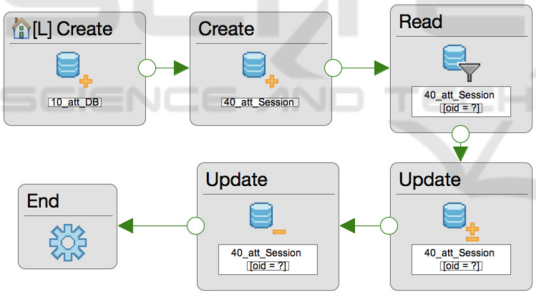

Figure 5: Operation chain OP1.

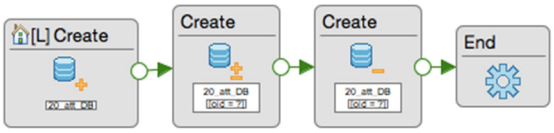

Figure 6: Operation chain OP2.

The SLA establishes a maximum unitary user performance time of 400 milliseconds –Time (ml)1u– with a runtime growth, following a logarithmic scale depending on the number of concurrent users, with a maximum performance peak of 4 seconds for 1.600 users –Time (ml)nu–. Table 2 shows the operations for OP1 and OP2 and the values for each operation regarding persistence type and number of attributes. The column named as Time (ml)1u adds up the whole operation for each case.

If we analyze the data of this example, collected in Table 2, we can observe that the level of 1 user in SLA is fulfilled for both operations if we use a machine t2.small (179ml in OP1 and 417 ml in OP2). However, when we plan the results up to 1,600 users we can appreciate that the first operation chain (OP1) could be executed in a t2.small within the required parameters (3,604 seconds) but the second operations chain (OP2) does not reach it (4,192 seconds) and therefore should be redesigned in order to use a t2.small infrastructure.

4 RELATED WORK

Optimization of cloud computing infrastructre in the Web production phase has been addressed recently by several works. Concretely, in (Andzrejak, D. Kondo, and S. Yi,, 2010) and (Cheng Tian, Ying Wang, Feng Qi, and Bo Yin., 2012) the authors

Table 6: SLA coverage in OP1 and OP2.

OP1	Create	Create	Read	Update	Delete	Time (ml)1u	ET (ml)un
	10 attributes	40 attributes	40 attributes	10 attributes	40 attributes		
	Data Base	Session	Session	Data Base	Session		
t2.small	98	0,89	0	79,26	1,10	179	3604,43419
t2.micro	330,89	2,24	1,50	169,28	22,76	526,67	4354,45935
OP2	**Create**	**Update**	**Delete**			**Time (ml)1u**	**ET (ml)un**
	20 attributes	20 attributes	20 attributes				
	Data Base	Data Base	Data Base				
t2.small	190,53	102,54	124,00			417	4192,33568
t2.micro	332,09	197,04	314,36			843,48	4681,72125

propose diferrents methods for estimating the cloud infrastructure pricing, once the application has been developed. In (Andzrejak et al., 2010), a probabilistic model is introduced to determinate the pricing, performance and reliability given a set of service requeriments. This information is combined with the real cloud provider prices to find the most suitable cloud infrastructure.

A wide range of executions with different values for the parameters used must be performed over the final application to identify the most suitable infrastructure combination. Similarly, (Cheng Tian et al., 2012) presents a model based on the characteristics of three purchasing options provided by Amazon EC2, which can be used for guiding the capacity planning activity once the application is ready to be deployed.

On the other hand, in (Huihong He, Zhiyi Ma, Xiang Li, Hongjie Chen and Weizhong Shao. 2012) the authors describe also an approach to calculate operating cost and performance needs to suggest a suitable cloud computing infrastructure but at design phase in this case. It can be performed by means of a UML extension that collects the cloud computing infrastructure capabilities for designing the infrastructure combinations. This approach incorporates a cost estimation algorithm to calculate the production pricing, that uses previously known values for factors like load, storage, concurrency, peaks, and so on.

In (Fu, Cang, Zhu, and Deng. 2015), the authors propose a heuristic algorithm to help the developer in the decisions related to the placement of the tasks when deploying a web application into a cloud infraestructure. The algorithm deals with the placement of the subtasks in the different nodes of the virtual machines in order to reduce data transmission and communication traffic. The authors claim that the algorithm provides important benefits in terms of completion time of the web applications. Unlike our work, this algorithm would be executed once the system (and its resources) has been completely generated.

5 CONCLUSIONS AND FUTURE WORK

In this paper, a first approach for estimating production costs and cloud infrastructure for data intensive Web applications has been presented.

As a first step of the approach, an experiment has been developed where the execution times of different CRUD operations have been measured based on a series of design and production parameters. This first analysis shows interesting and promising results regarding the possibility of establishing a first infrastructure estimation based on the significant (independent) variables considered in the study. That is, in light of the results, this first step would help the designer to anticipate, in the early stages of design, the computing needs and cloud infrastructure that the application will need for its later deployment.

To do this, we have already identified the following immediate steps for our research. First, it would be useful to modify the WebRatio *TimeUnit* to achieve a finer grain level of detail in term of nanoseconds. It would allow a more objective assessment of the results of the study, specially when the service load increases. Secondly, we plan to monitor the behavior of the processor, RAM and storage capacity at each point in the execution of the operation chain in order to evaluate the quality requirements in these aspects. Thirdly, we are going to modify the operation chain so that the sequence of tests and data collection when repeating the test can be heterogeneous, that is, the execution of the operation that is launched and measured at any time can execute CRUD operations randomly.

We already observe in this first analysis that there are aspects can influence the effectiveness of cost estimation such as data base type, cache capabilities, the ability to scale the application on several machines by elastic growth or resources sharing by other applications, as well as the correlations between these aspects and the throughput time. Notwithstanding, we are currently defining the experiments to monitor and then evaluate the impact of those aspects and cloud service providers offer basic monitoring capabilities for controlling computation and data transfer costs.

Complementary, we are planning to evaluate the impact for each type of attribute placed in an entity (float, integer, date, time, text, etc.) and to identify how their different combinations may affect the results. Regarding the data model, we need also to identify the perfomance data when using 1:N (one to many) and N:M (many to many) relationships.

Once we advance in the study of all these variables, we will work on a plugin for WebRatio that will provide the designer with automatic suggestions about what cloud infrastructure is estimated as necessary based on the identification of the design parameters discussed in this work. Morever, by means of connecting to the AWS (Amazon Web Services) infrastructure cost calculator (AWS Calculator, 2017), the plugin could also automatically calculate infrastructure pricings based on the selected operating parameters. Even, the plugin could suggest a first visual infrastructure proposal by connecting to Cloudcraft (Amazon CloudCraft, 2017).

Finally, our future research lines include the application of the study to other MDWE proposals or even to Web development frameworks commonly used in software factory environments. This advance would allow us to be able to estimate at design time the infrastructure costs of different frameworks and to be able to compare them, supporting the decision making regarding the chosen infrastructure.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of TIN2015-69957-R (MINECO/FEDER, UE) project and Homeria Open Solutions, S.L. to the work here presented.

REFERENCES

Amazon CloudCraft, 2017. https://cloudcraft.co/

AWS Calculator, 2017. https://calculator.s3.amazonaws.com/index.html

Andzrejak, D. Kondo, and S. Yi, “Decision Model for Cloud Computing under SLA Constraints,” in IEEE Int. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOT), 2010

Barry W. Boehm. Software Engineering Economics 1st Edition. Prentice Hall HTR. New Jersey. 1981. ISBN-13: 978-0138221225

Brambilla, M., Fraternali, P. Interaction Flow Modeling Language – Model-driven UI Engineering of Web and Mobile Apps with IFML. Morgan Kauffman, USA, 2014.

Cheng Tian, Ying Wang, Feng Qi, and Bo Yin. 2012. Decision model for provisioning virtual resources in Amazon EC2. In Proceedings of the 8th International Conference on Network and Service Management (CNSM '12), International Federation for Information Processing, Laxenburg, Austria, Austria, 159-163.

Fu, X., Cang, Y., Zhu, X. and Deng, S., 2015. Scheduling Method of Data-Intensive Applications in Cloud Computing Environments, Mathematical Problems in Engineering, vol. 2015, doi:10.1155/2015/605439

Huihong He, Zhiyi Ma, Xiang Li, Hongjie Chen and Weizhong Shao. 2012. An approach to estimating cost of running cloud applications based on AWS. 19th Asia-Pacific Software Engineering Conferencce. ISSN 1530-1362, ISBN 1467349305 (1) pp. 571-576.

Konstantinos Morfonios, Stratis Konakas, Yannis Ioannidis, and Nikolaos Kotsis. 2007. ROLAP implementations of the data cube. ACM Comput. Surv. 39, 4, Article 12 (November 2007). DOI: https://doi.org/10.1145/1287620.1287623

MacQueen, J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281--297, University of California Press, Berkeley, Calif., 1967.

Rodriguez-Echeverria R., M. Conejero J., Preciado J. and Sanchez-Figueroa F. (2016). AutoCRUD - Automating IFML Specification of CRUD Operations.In Proceedings of the 12th International Conference on Web Information Systems and Technologies - Volume 1: APMDWE, ISBN 978-989-758-186-1, pages 307-314. DOI: 10.5220/0005923003070314

Rossi, G, Pastor, O, Schwabe, D, Olsina, L., 2007. Web Engineering: Modelling and Implementing Web Applications. Human-Computer Interaction Series, Springer-Verlag, London.

Suresh, S. and Sakthivel, S, 2016, System modeling and evaluation on factors influencing power and performance management of cloud load balancing algorithms, Journal of Web Engineering, Vol. 15, 5,6, pages 484-500

Toffetti, G., Comai, S., Preciado, J. C., Linaje, M., 2011. State-of-the Art and trends in the SystematicDevelopment of Rich Internet Applications. In Journal of Web Engineering, 10, 1, 70-86.