A Study of the Energy Consumption of Databases and Cloud Patterns

Béchir Bani(✉), Foutse Khomh, and Yann-Gaël Guéhéneuc

SWAT Lab - Ptidej Team, Polytechnique Montréal, Montreal, QC, Canada
{bechir.bani,foutse.khomh,yann-gael.gueheneuc}@polymtl.ca

Abstract. Nowadays databases have become the backbone of cloud-based applications. Cloud-based applications are used in about every industry today. Despite their popularity and wide adoption, little is still known about the energy footprint of these applications and, in particular, of their databases. Yet, reducing the energy consumption of applications is a major objective for society and will continue to be so in the near to far future. In this paper, we study the energy consumption of three databases used by cloud-based applications: MySQL, PostgreSQL, and MongoDB, through a series of experiments with three cloud-based applications (a RESTful multi-threaded application, DVD Store, and JPetStore). We also study the impact of cloud patterns on the energy consumption because databases in cloud-based applications are often implemented in conjunction with patterns. We measure the energy consumption using the Power-API tool to keep track of the energy consumed at the process-level by the variants of the cloud-based applications. We report that the choice of the databases can reduce the energy consumption of a cloud-based application regardless of the cloud patterns that are implemented.

Keywords: Energy consumption · Relational databases · NoSQL databases · Performance · Sharding · Priority Message Queue

1 Introduction

With the continuous development of the Internet and cloud computing, companies use databases to store and perform analyses on large data-sets in cloud environments. These companies demand high performance databases when reading and writing data. In addition, they want to benefit from best practices encoded in the form of cloud patterns [6], which are general and reusable "good" solutions to recurring design problems for cloud-based applications. Design patterns have been applied to all fields of software engineering, including cloud computing. These patterns were refined to take into account the specificities and requirements of the cloud. To the best of our knowledge, none of previous works, that have benchmarked cloud applications [5], investigated the combined impact of databases and cloud patterns on the energy consumption of cloud-based applications. Consequently the benefits and trade-offs of different databases and combinations of cloud patterns are mostly intuitive and not validated. In this paper, we evaluate the impact on energy consumption of three cloud patterns: Local Database Proxy, Local Sharding-Based Router, and Priority Message Queue, with three databases: two relational databases, PostgreSQL and MySQL, and one NoSQL database, MongoDB. To achieve this goal, we use three versions of three cloud-based applications (a RESTful multi-threaded application, DVD Store, and jPETStore) that use respectively MySQL, PostgreSQL, and MongoDB databases. We also implement the three studied patterns in each version of these applications. We choose these databases because they are the most popular relational databases in the last few years [2,4]. We measure energy consumption using the Power-API [3], which estimates the energy consumed by an application at the process-level. Our results show that MySQL database is the least energy consuming among the three databases and PostgreSQL is the most energy consuming among them. MongoDB consumes more energy than MySQL but less than PostgreSQL. We also show that various combinations of patterns impact energy consumption. The rest of the paper is structured as follows. Section 2 provides the most closely related works to our study. Section 3 presents the cloud-based applications used by our study and the design of our experiments. Section 4 discusses the results of our experiments. Section 5 discusses our results and possible threats to their validity. Section 6 concludes with some future works.

2 Related Work

The most closely related work to ours is by Abtahizadeh et al. [1]. They conducted an empirical study that aimed to compare the energy efficiency of the same three cloud patterns performed in our study. However, they only considered MySQL database in their work. Their results show that cloud patterns can reduce the energy consumption of a cloud-based application but only in some specific cases. In the same direction, Manotas et al. [7] conducted an empirical study in which they investigated the impact of four web servers on the energy consumption of a web application. They showed that the energy consumption of a web application depends on the web server used to handle requests, where each web server can increase or decrease the energy consumption of the web application, depending on the features for which it is executed. Sahin et al. [8] investigated the energy efficiency of 15 structural, behavioral and creational design patterns, implemented in an application. For each pattern, they examined the energy consumption of the versions of the application before and after applying design pattern. Their results show that design patterns have a significant impact on energy consumption, where certain design patterns like `Decorator` can increase the energy usage of an application by up to 700 %.

3 Study Design

In this section, we introduce our research questions, describe the objects and the design of our study, and analysis method. Our research questions are:

- **RQ1**: Does the choice of MySQL, PostgreSQL and MongoDB Databases affect the energy consumption of cloud applications (when no cloud patterns are implemented)?
- **RQ2**: Does the implementation of Local Database Proxy, Local Sharding Based Router or Priority Message Queue patterns affect the energy consumption of cloud applications using MySQL, PostgreSQL and MongoDB Databases?
- **RQ3**: Do the interactions between Local Database Proxy, Local Sharding Based Router and Priority Message Queue patterns affect the energy consumption of cloud applications using MySQL, PostgreSQL and MongoDB Databases?

3.1 Objects and Design

In our experiments, we use a combination of databases and cloud patterns encoded using a letter and a number. The Local Database Proxy pattern has three implementation strategies: Random Allocation (P1), Round-Robin (P2), and Custom Load Balancing (P3). The Local Sharding Based Router pattern also has three strategies: Modulo Algorithm (P4), Consistent Hashing (P5), and Lookup Algorithm (P6). The Priority Message Queue pattern is called P7. The databases are named: MySQL (D1), PostgreSQL (D2), and MongoDB (D3). We performed each experiment on three different systems, because one system could be intrinsically more complex to understand. We deployed them on 10 virtual machines (2 master nodes and 8 slaves nodes) in a private cloud. At first, for Experiment 1, we implement and deploy a multi-threaded distributed application that communicates through REST calls. The application interacts with one of the three chosen databases. *Sakila sample database* is used as it contains a large number of records, making it interesting for experiments. We adapted The schema of the *Sakila database* to PostgreSQL and MongoDB databases. For Experiment 2 and 3, we use DVDStore and JPetStore systems. DVDStore[1] is provided with the implementation of MySQL and PostgreSQL databases. We refactor the code of DVD Store to allow it to connect with a MongoDB database. Similarly, we also modified the code of JPetStore[2] to implement connections to MySQL, PostgreSQL and MongoDB databases. We perform our experiments using different numbers of clients, which are simulated using a multi-threaded architecture. The number of clients simulated varies from 100 to 1500 clients. Each execution is done using different databases and different cloud patterns.

3.2 Independent and Dependent Variables

MySQL, PostgreSQL and MongoDB databases are the independent variables of our study. Also, the three studied cloud patterns, as well as the strategies of these patterns are considered as independent variables. The application response time (measured in *milliseconds*) and the energy consumption measured by Power-API (measured in joules) are considered as dependent variables.

[1] http://linux.dell.com/dvdstore/.
[2] https://github.com/mybatis/jpetstore-6.

3.3 Hypotheses

To answer our research questions, we formulate the following null hypotheses, where P0 is the experiment consisting in comparing the energy consumption and response time of the three versions of each application using respectively MySQL, PostgreSQL, and MongoDB databases. Px (x $\in$ {1 ... 6}), and P7 are the different patterns. In each experiment we compare two versions of a same application implementing two different databases Dy, Dz (y, z $\in$ {1, 2, 3} and y $\neq$ z), with the same (combination) of patterns.

– H^1_{0yz}: There is no difference between the average amount of energy consumed by applications implementing databases D_y and D_z (without any cloud pattern).
– H^1_{xyz}: There is no difference between the average amount of energy consumed by applications implementing databases D_y and D_z in conjunction with patterns Px.
– H^1_{xyz7}: There is no difference between the average amount of energy consumed by applications implementing databases D_y and D_z in conjunction with the combination of patterns Px and P7.

To have more clear comprehension regarding the trade-offs between the energy consumption and the performance of a cloud-based application measured in terms of response time, we also formulate the following null hypotheses:

– H^2_{0yz}: There is no difference between the average response time of databases D_y and D_z by applying the design P0.
– H^2_{xyz}: There is no difference between the average response time of databases D_y and D_z by applying the design Px.
– H^2_{xyz7}: There is no difference between the average response time of databases D_y and D_z by applying the combination of designs Px and P7.

3.4 Analysis Method

To analyze our collected data (i.e., response time and energy consumption measurements), we performed the Mann-Whitney U test [9] to test the aforementioned hypotheses. Mann-Whitney U test is a non-parametric statistical test where its relevance is reflected in the assessment of two independent distributions. We also computed the Cliff's δ effect size because effect sizes are very important to understand the magnitude of the difference between 2 distributions.

4 Study Results

This section presents and discusses the results of our research questions.

4.1 Results and Answers to RQ1

Tables 1 and 2 summarizes the results of Mann-Whitney U test and Cliff's δ effect sizes for the energy consumption and the response time.

Table 1. Energy consumption p-value and Cliff's δ

Pattern	MySQL	PostgreSQL	p-value	Cliff's δ	MySQL	MongoDB	p-value	Cliff's δ	PostgreSQL	MongoDB	p-value	Cliff's δ
P0	262.5	568.2	0.01	medium	262.5	354.7	0.24	small	568.2	354.7	0.09	small
P1	490.2	1391.1	$< 10e{-6}$	large	490.2	890.0	$< 10e{-6}$	large	1391.1	890.0	0.09	small
P2	495.2	1529.9	$< 10e{-6}$	large	495.2	915.9	$< 10e{-6}$	large	1529.9	915.9	0.04	medium
P3	495.0	1476.5	$< 10e{-6}$	large	495.0	904.5	$< 10e{-6}$	large	1476.5	904.5	0.04	medium
P4	1331.9	6330.2	$< 10e{-6}$	large	1331.9	5826.4	$< 10e{-6}$	large	6330.2	5826.4	0.23	small
P5	611.6	4245.1	$< 10e{-6}$	large	611.6	3821.8	$< 10e{-6}$	large	4245.1	3821.8	0.23	small
P6	824.1	4929.4	$< 10e{-6}$	large	824.1	4194.4	$< 10e{-6}$	large	4929.4	4194.4	0.23	small
P1+P7	442.7	1379.8	$< 10e{-6}$	large	442.7	814.3	$< 10e{-6}$	large	1379.8	814.3	0.03	medium
P2+P7	468.8	1482.5	$< 10e{-6}$	large	468.8	891.9	$< 10e{-6}$	large	1482.5	891.9	0.03	medium
P3+P7	490.2	1391.1	$< 10e{-6}$	large	490.2	890.0	$< 10e{-6}$	large	1391.1	890.0	0.09	small
P4+P7	1255.5	5777.4	$< 10e{-6}$	large	1255.5	5622.9	$< 10e{-6}$	large	5777.4	5622.9	0.82	negligible
P5+P7	492.2	3884.5	$< 10e{-6}$	large	492.2	3386.6	$< 10e{-6}$	large	3884.5	3386.6	0.23	small
P6+P7	775.9	4526.8	$< 10e{-6}$	large	775.9	4127.4	$< 10e{-6}$	large	4526.8	4127.4	0.23	small

Table 2. Response time p-value and Cliff's δ

Pattern	MySQL	PostgreSQL	p-value	Cliff's δ	MySQL	MongoDB	p-value	Cliff's δ	PostgreSQL	MongoDB	p-value	Cliff's δ
P0	262.5	568.2	0.01	medium	262.5	354.7	0.24	small	568.2	354.7	0.09	small
P1	490.2	1391.1	$< 10e{-6}$	large	490.2	890.0	$< 10e{-6}$	large	1391.1	890.0	0.09	small
P2	495.2	1529.9	$< 10e{-6}$	large	495.2	915.9	$< 10e{-6}$	large	1529.9	915.9	0.04	medium
P3	495.0	1476.5	$< 10e{-6}$	large	495.0	904.5	$< 10e{-6}$	large	1476.5	904.5	0.04	medium
P4	1331.9	6330.2	$< 10e{-6}$	large	1331.9	5826.4	$< 10e{-6}$	large	6330.2	5826.4	0.23	small
P5	611.6	4245.1	$< 10e{-6}$	large	611.6	3821.8	$< 10e{-6}$	large	4245.1	3821.8	0.23	small
P6	824.1	4929.4	$< 10e{-6}$	large	824.1	4194.4	$< 10e{-6}$	large	4929.4	4194.4	0.23	small
P1+P7	442.7	1379.8	$< 10e{-6}$	large	442.7	814.3	$< 10e{-6}$	large	1379.8	814.3	0.03	medium
P2+P7	468.8	1482.5	$< 10e{-6}$	large	468.8	891.9	$< 10e{-6}$	large	1482.5	891.9	0.03	medium
P3+P7	490.2	1391.1	$< 10e{-6}$	large	490.2	890.0	$< 10e{-6}$	large	1391.1	890.0	0.09	small
P4+P7	1255.5	5777.4	$< 10e{-6}$	large	1255.5	5622.9	$< 10e{-6}$	large	5777.4	5622.9	0.82	negligible
P5+P7	492.2	3884.5	$< 10e{-6}$	large	492.2	3386.6	$< 10e{-6}$	large	3884.5	3386.6	0.23	small
P6+P7	775.9	4526.8	$< 10e{-6}$	large	775.9	4127.4	$< 10e{-6}$	large	4526.8	4127.4	0.23	small

Average Amount of Consumed Energy: Results presented in Table 1 show that, without using any pattern (in other words, by applying the design P0), there is a statistically significant difference between the average amount of energy consumed by application using MySQL and application using PostgreSQL. The effect size in this case is medium. Therefore, we reject H^1_{0yz} for D_y, D_z (y = 1, z = 2). However, there is not a statistically significant difference between the average amount of energy consumed by application using MySQL and application using MongoDB. Therefore, we cannot reject H^1_{0yz} for D_y, D_z (y = 1, z = 3). Similarly, there is not a statistically significant difference between the average amount of energy consumed by application using PostgreSQL database and application using MongoDB database. In these two cases the effect size is small. Therefore, we cannot reject H^1_{0yz} for D_y, D_z (y = 2, z = 3).

Average Response Time: Results presented in Table 1 show that, by applying the design P0, there is not a statistically significant difference between the average response time of application using MySQL database and application using PostgreSQL database. Therefore, we cannot reject H^2_{0yz} for D_y, D_z (y = 1, z = 2). However, there is a statistically significant difference between the average response time of application using MySQL database and application using MongoDB database. Similarly, there is a statistically significant difference between the average response time of application using PostgreSQL database and application using MongoDB database. Therefore, we cannot reject H^2_{0yz} for D_y, D_z ((y = 1, z = 3), (y = 2, z = 3)).

4.2 Results and Answers to RQ2

Average Amount of Consumed Energy: These results show that by applying the Local Database Proxy pattern, there is a statistically significant difference between the average amount of energy consumed by application using MySQL database and application using PostgreSQL database. Similarly, also, between application using MySQL and application using MongoDB. Similarly also by application using PostgreSQL database and application using MongoDB database (where the effect size is large). But, except for the case where the proxy pattern is implemented using the random strategy, there is not a statistically significant difference between application using PostgreSQL database and application using MongoDB database. Therefore we reject H^1_{xyz} for P_x, D_y, D_z (x $\in$ {2, 3}, (y = 1, z = 2), (y = 1, z = 3)), but we cannot reject H^1_{xyz} for P_x, D_y, D_z (x = 1, y = 2, z = 3). By applying the Local Sharding Based Router, there is a statistically significant difference between the average amount of energy consumed by application using MySQL database and application using PostgreSQL database. Similarly also between application using MySQL and application using MongoDB (the effect size is large). But, there is not a significant difference between application using PostgreSQL database and application using MongoDB database. Therefore, we reject H^1_{xyz} for P_x, D_y, D_z (x $\in$ {4, 5, 6}, (y = 1, z = 2), (y = 1, z = 3)), but we cannot reject H^1_{xyz} for P_x, D_y, D_z (x $\in$ {4, 5, 6}, y = 2, z = 3).

Average Response Time: Results show that by applying the Local Database Proxy pattern, there is not a statistically significant difference between the average response time of application using MySQL database and application using PostgreSQL database. Therefore, we cannot reject H^2_{xyz} for P_x, D_y, D_z (x $\in$ {1, 2, 3}, (y = 1, z = 2)). However, there is a statistically significant difference between the average response time of application using MySQL database and application using MongoDB database. Similarly, there is a statistically significant difference between the average response time of application using PostgreSQL database and application using MongoDB database. Therefore, we reject H^2_{xyz} for P_x, D_y, D_z (x $\in$ {1, 2, 3}, (y = 1, z = 3), (y = 2, z = 3)). Further results, by applying the Local Sharding Based Router, there is not a statistically significant difference between the average response time of application using MySQL database and application using PostgreSQL database. Therefore, we cannot reject H^2_{xyz} for P_x, D_y, D_z (x $\in$ {4, 5, 6}, (y = 1, z = 2)). However, there is a statistically significant difference between the average response time of application using MySQL database and application using MongoDB database. Similarly, there is a statistically significant difference between the average response time of application using PostgreSQL database and application using MongoDB database. Therefore, we reject H^2_{xyz} for P_x, D_y, D_z (x $\in$ {4, 5, 6}, (y = 1, z = 3), (y = 2, z = 3)).

4.3 Results and Answers to RQ3

Average Amount of Consumed Energy: When we combine the Local Database Proxy pattern with the priority Message Queue pattern, results show that there is a statistically significant difference between the average amount of energy consumed by application using MySQL database and application using PostgreSQL database. Similarly also between application using MySQL and application using MongoDB (the effect size is large). The same is true for application using PostgreSQL database and application using MongoDB database (where the effect size is large). However, except applying the combination of the custom strategy with the Priority Message Queue pattern, there is not a statistically significant difference between application using PostgreSQL database and application using MongoDB database. Therefore, we reject H^1_{xyz7} for P_x, D_y, D_z (x $\in$ {1, 2, 3}, (y = 1, z = 2), (y = 1, z = 3)), but we cannot reject H^1_{xyz7} for P_x, D_y, D_z (x = 3, y = 2, z = 3). Also, when we combine the Local Sharding Based Router pattern with the priority Message Queue pattern, results show that there is a statistically significant difference between the average amount of energy consumed by application using MySQL database and application using PostgreSQL database. Similarly also between application using MySQL and application using MongoDB (the effect size is large). However, there is no a significant difference between application using PostgreSQL database and application using MongoDB database. Therefore, we reject H^1_{xyz7} for P_x D_y, D_z (x $\in$ {4, 5, 6}, (y = 1, z = 2), (y = 1, z = 3)), but we cannot reject H^1_{xyz7} for P_x, D_y, D_z (x $\in$ {4, 5, 6}, y = 2, z = 3).

Average Response Time: By applying the Local Database Proxy pattern with the priority Message Queue pattern, there is not a statistically significant difference between the average response time of application using MySQL database and application using PostgreSQL database. Therefore, we cannot reject H^2_{xyz7} for P_x, D_y, D_z (x $\in$ {1, 2, 3}, (y = 1, z = 2)). However, there is a statistically significant difference between the average response time of application using MySQL database and application using MongoDB database. Similarly, there is a statistically significant difference between the average response time of application using PostgreSQL database and application using MongoDB database. Therefore, we reject H^2_{xyz7} for P_x, D_y, D_z (x $\in$ {1, 2, 3}, (y = 1, z = 3), (y = 2, z = 3)). Besides that, when we combine the Local Sharding Based Router pattern with the priority Message Queue pattern, results show that there is not a statistically significant difference between the average response time of application using MySQL database and application using PostgreSQL database. Therefore, we cannot reject H^2_{xyz7} for P_x, D_y, D_z (x $\in$ {4, 5, 6}, (y = 1, z = 2)). However, there is a statistically significant difference between the average response time of application using MySQL database and application using MongoDB database. The combination of the Lookup strategy and the Priority Message Queue pattern there is not a significant difference. Similarly, there is a statistically significant difference between the average response time of application using PostgreSQL database and application using MongoDB database. The combination of the Lookup strategy and the Priority Message Queue pattern there is not a significant difference.

Therefore, we reject H^2_{xyz7} for P_x, D_y, D_z (x ∈ {4, 5}, (y = 1, z = 3), (y = 2, z = 3)), and we cannot reject H^2_{xyz7} for P_x, D_y, D_z (x = 6, (y = 1, z = 3), (y = 2, z = 3)).

5 Discussions and Threats to Validity

We showed that the implementation of the Local Database Proxy pattern does not impact the behavior of the databases but can significantly improve the energy efficiency of MySQL. Concerning the Local Sharding Based Router pattern, the Modulo strategy has a strong effect on the energy consumption of PostgreSQL and MongoDB databases but a small one for MySQL. Moreover, the Consistent strategy has a strong effect on the energy consumption of PostgreSQL but improves slightly the energy efficiency of MySQL and MongoDB. The Lookup strategy can significantly improve the energy efficiency of PostgreSQL and MongoDB. In addition, we showed that combining Local Database Proxy pattern with the Priority Message Queue pattern has no significant impact neither on the application response time nor on the energy consumed by the application, when it interacts with MySQL. This combination only has a small effect on the energy consumption of PostgreSQL and MongoDB. Interestingly, the implementation of the Local Sharding Based Router pattern with the Priority Message Queue pattern has a strong effect on the response time of the three Databases but without a significant impact on the energy consumption.

Our experiments, as any other experiment, are subject to threats to their validity. We now discuss these threats based on the guidelines provided by Wohlin et al. [10].

Construct validity threats concern the relation between theory and observations. In this study, they could be due to measurement errors. These measurements are subject to variation and perturbations depending of hardware and network. For this reason, we did several experiments, we conducted each experiment five times, and computed average values of these measurements.

Internal validity threats concern our selection of subject systems and analysis methods. Despite of using the three studied databases, the three cloud patterns and the two standard cloud applications, some of our findings may still be specific to our studied application which was designed specifically for the experiments. Future studies should consider using different RDBMS and NoSQL databases, and also other cloud applications implementing the cloud patterns.

External validity threats concern the possibility to generalize our findings. Further validation should be done on different cloud applications and with different relational and NoSQL databases and applying different cloud patterns to these databases can extend our understanding of the impact of databases on the energy consumption of cloud applications.

Reliability validity threats concern the possibility of replicating this study. We attempt to provide all the necessary details to replicate our study.

Finally, the *conclusion validity* threats refer to the relation between the treatment and the outcome. We mainly used non-parametric tests that do not require making assumptions about the distribution of the metrics.

6 Conclusion and Future Work

Nowadays, reducing energy consumption is a challenge for cloud-based applications. We contrasted the performance of various combinations of databases and cloud patterns in terms of energy consumption and response time of the cloud-based applications, with the aim to provide some guidance to software engineers about the usage of databases and cloud patterns for cloud-based applications. We carried on a series of experiments on different versions of a RESTful multi-threaded application implemented with three different databases and three different cloud patterns. We also used two standard cloud applications (DVD Store and JPetStore) because one system could be intrinsically more complex to understand. We showed that MySQL database is the least energy consuming but is the slowest among the three databases. PostgreSQL is the most energy consuming among the three databases, but is faster than MySQL but slower than MongoDB. MongoDB consumes more energy than MySQL but less than PostgreSQL and is the fastest among the three databases. As future work, we plan to examine how a match/mismatch between the selected database and the workload characteristic affects energy efficiency.

References

1. Abtahizadeh, S.A., Khomh, F., et al.: How green are cloud patterns?. In: 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), pp. 1–8. IEEE (2015)
2. Aghi, R., Mehta, S., Chauhan, R., Chaudhary, S., Bohra, N.: A comprehensive comparison of SQL and MongoDB databases (2015)
3. Bourdon, A., Noureddine, A., Rouvoy, R., Seinturier, L.: PowerAPI: a software library to monitor the energy consumed at the process level. ERCIM News 2013(92) (2013)
4. Conrad, T.: PostgreSQL vs. MySQL vs. commercial databases: it's all about what you need (2006)
5. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM symposium on Cloud computing, pp. 143–154. ACM (2010)
6. Fehling, C., Leymann, F., Retter, R., Schumm, D., Schupeck, W.: An architectural pattern language of cloud-based applications. In: Proceedings of the 18th Conference on Pattern Languages of Programs, p. 2. ACM (2011)
7. Manotas, I., Sahin, C., Clause, J., Pollock, L., Winbladh, K.: Investigating the impacts of web servers on web application energy usage. In: 2013 2nd International Workshop on Green and Sustainable Software (GREENS), pp. 16–23. IEEE (2013)
8. Sahin, C., Cayci, F., Gutiérrez, I.L.M., Clause, J., Kiamilev, F., Pollock, L., Winbladh, K.: Initial explorations on design pattern energy usage. In: 2012 First International Workshop on Green and Sustainable Software (GREENS), pp. 55–61. IEEE (2012)
9. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2003)
10. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Heidelberg (2012)