Decision Drivers for Persistence Technology on Software Development: From Traditional Systems to Big Data Systems

Rodrigo Nunes Laigner

Informatics Department  
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)  
Rio de Janeiro, Brazil  
rlaigner@inf.puc-rio.br

Abstract

Context: Software development has evolved to be a process well supported by a myriad of software assets, such as frameworks, build and dependency management tools, and IDEs. However, these tools cannot replace the human role on choosing the appropriate technology that best fit a software project.

Objective: This work aims at collecting evidence on persistence technology selection process in the context of software development as well proposed methods and techniques to support the same. The objective is to build an overview on the current practice and the means of which a persistence technology is selected.

Method: A systematic mapping study was performed to identify and analyze current research and practice on identification of suitable persistence technology in software development project, covering publications from 2010 to 2017.

Results: To be defined

Conclusion: To be defined

Keywords - data-intensive systems, big data systems, software engineering, database systems, nosql, sql, persistence technology, software development

1. Introduction

The decision of the persistence technology of a software system is a fundamental part of software development cycle, primarily on web-based systems and scientific applications. … software solutions, hadoop, new technologies and architectures…

Software development has evolved to be a process well supported by a myriad of software assets, such as frameworks, build and dependency management tools, and IDEs. These tools enable faster prototyping and automation of tasks (compiling and dependency resolve).   
  
Particularly, frameworks withdraw efforts when it comes to provide general functional requirements such as authentication, UI design, data persistence, and error log. These way, developers can focus on specific domain problems.   
  
However, these tools cannot replace the human role on choosing the appropriate programming language, software paradigm (structured, object-oriented, aspect-oriented), software architecture and database technology.  
  
The last pose challenges on selecting a persistence solution that meets software constraints such as fast data retrieval, availability, and transaction management. It is common to face software projects that have to fundamental changes when engineers realized that the persistence technology chosen (or the set of database solutions) was/were not appropriate due to functional and non functional requirements not met (i.e. http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/)  
  
Along with software tools, software engineering field of study has increased in maturity along the years and the literature presents models and techniques for software construction. Patterns for modeling and representing systems (UML), studies on detection of defects on code (code smells), agent-based approaches, domain-driven design, and also study on agile methods are some of the contributions.  
  
Software engineering is continuing to evolve. Needs that were important in a moment of time, can become inappropriate for challenges that arise. For example, new technologies and concepts, such as microservices, REST architecture, polyglot persistence, and mapreduce are found on job listings and new software architectures. With data-driven application (known as data-intensive applications), which can make use of big data technologies, such as Hadoop, Apache Spark, and NoSQL systems that store and process data in large scale, the myriad of solutions grow. Chen et al. [11] asserts that “there are a large number of emerging big data technologies (both open source and proprietary), many of which are provided by new vendors or open source projects”. According to a survey by Infochimps and SSWUG.ORG [12], in the context of big data projects, organizations strive on finding tools, professionals and understand the technologies.

Professionals can be positioned on situations where a persistence technology should be selected and a vast range of material should be read and no window to experiments or tests are given. The context expressed above add complexity to this process. They regularly have to consider sources (non peer reviewed, grey literature) that do not go through a systematic process of review on the exposed information.   
  
Most studies focus on addressing issues on the selected database. Our study focus on the steps before database selection. There is a need to provide an overall blueprint of decision-making process for persistence selection. Madhavji et al. [16] also address this concern in the context of big data systems for requirements specification and architectural decisions.

Since there is no comprehensive overview of strategies and software characteristics that drive persistence technology selection, this study aims at surveying existing research on software engineering with the objective to build a knowledge on selecting appropriate persistence technologies based on the requirements and characteristics of an application and also find gaps and window for improvement of the process of selecting appropriate technology for persistence.

2. Background and Related Work

Since 2011, the literature started a process of departing the called traditional systems from big data systems [2][7][13]. As this work intends to oversee the current practice on development of software systems, it is important to research both perspectives. Next topics cover the differences between traditional systems and big data systems, and their formal definitions. In addition, the last topic address database systems in terms of paradigms such as NoSQL and NewSQL, domain-specific databases and non-functional requirements. For last, related work will present research that address difference between traditional and big data systems.

1. Traditional Software Development

The literature started addressing traditional software development in the context of introducing techniques and strategies for supporting the development of big data systems [1].

Chen et al. [9] address that traditional software development is characterized as

“structured, batch-oriented, [and] relational small data [volume]”. In another work, Chen et al. [7] argues that traditional systems are primarily based on relational database systems, an established set of feasible system development life cycle (SDLC), and straightforward architecture design.

The author of this work does not agree that architecture design for systems before the advent of NoSQL databases and tools for data processing such as Hadoop were straightforward, since software architecture historically presented challenges on integration, selection of middleware technologies and distributed applications [7]. MAYBE CITE WORKS ON BIOINFORMATICS

Madhavji et al. [16] argue that big data systems “are complex solutions with many  
dynamic components, such as distributed computation nodes, networks, databases, middleware, and business intelligence layers”.

It is not clear, based on the above definition, that big data systems drastically differs from traditional systems, since systems have been characterized by the same attributes mentioned by Madhavji [16], although not being coined as big data systems.

However, Gorton & Klein [13] present a different perspective, once their argument remains on explaining that data-intensive systems “have long been built on SQL database technology, which relies primarily on vertical scaling—faster processors and bigger disks—as workload or storage requirements increase.”

It means that handling large volume of data and acceptable response time as access requests increase is not a novel problem. For instance, distributed database systems have been employing a set of mechanisms for these problems, such as: optimization of distributed queries, data replication, parallelism, and also maintaining ACID properties (covered on topic C in this section) [15].

In addition, Gorton & Klein [13] argue that traditional business systems are “relatively well constrained in terms of data growth, analytics, and scale”.

Based on the works presented, I would argue that traditional software development is related to systems on which scalability, despite possibly being considered as a major concern on non-functional requirements, presents a immutable characteristic, since data growth is at a constant rate along the time. For instance, this characteristic can be applied to business systems such as ERPs and CRMs, which are solutions fitted for a specific business characteristic (e.g., number of employees and number of business processes).

In addition, regarding structure of data (data model), I would assert that in traditional systems data ingestion and processing regularly obeys a predefined format. Occasionally, an Extraction-Transform-Load (ETL) tool, such as Pentaho Data Integration [17] could be employed in order to process input data.

Recently, business systems and applications with a defined data model, started considering document-oriented databases, such as MongoDB, for tasks related to storage of logs and legacy data (with an old schema) [14]. This way, it is not possible to address traditional software development limited to the usage of relational databases.

(My assumption is that we can define big data systems as solutions that employ multiple database systems, often handling different data models and unstructured data, real-time data processing, scalability as a major concern, and more….)

1. Big Data Systems (maybe change order with traditional software development)

According to Anderson [2], software engineering field is facing new challenges on the process of generation and collection of large volume of data. This context have led to the design and development of software systems targeted to tackle these challenges, the so called big data systems.

Gorton & Klein [5] define big data systems as “distributed systems that include redundant processing nodes, replicated storage, and frequently execute on a shared cloud infrastructure … employing a heterogeneous mix of SQL, NoSQL, and NewSQL technologies”

In another work [13], they characterize big data applications based on four requirements:

* Must be able to sustain write-heavy workloads
* Deal with variable request loads (avoid the costs of overprovisioning to handle occasional spikes, application-specific strategies to detect increased loads,  
  rapidly add new resources, and release them as necessary)
* Support computation-intensive analytics (diverse query workloads, rapid responses, long-running requests, varying latency demands)
* High availability (replicating data across geographical regions, stateless services, and application-specific mechanisms)

HELP These four requirements are not different from any other system. My point here is that these requirements fit a normal system where

As argued on last topic, distributed database systems (DDBS) have the ability to replicate data across distinct sites in order to enable acceptable response time and have long been used for commercial systems since 1990s [18].

However, it is important to note that Gorton & Klein [13] address the use of a cloud infrastructure and polyglot persistence as an important feature for enabling big data systems (BDS).

Polyglot persistence is the presence of different database technologies working/arranged in a systematic conjecture, that can possibly exchange information, within a software system.

In addition, [13] argues that BDS enable data access (through queries from users) to the system from different types of devices and applications. These applications may be within the system or an independent cell of system, which is integrated through an endpoint mechanism.

These characteristics expose the most difference from traditional systems. First, types of devices are related to cell phones, desktops, tablets, and more, which can promote concerns over data integrity, security and query workload, since the options for accessing the system grow. Second, independent applications may concern the development of microservices and API, which also adds complexity on systems integration.

Often identified as data-intensive systems [2][13], Anderson [2] asserts that BDS are “systems that generate, collect, store, process, analyze, query, and visualize large sets of data”.

Chen et al. [7] assert that in big data context, volume of data requires distributed and parallel architectures, variety of data leads to polyglot persistence [8], real-time needs, such as event processing, require specialized tools for ingestion, collection and preparation of raw data. For last, Chen et al. [7] highlights the challenge on extending “existing architectural methods and integrate polyglot data modeling, architecture design, and technology orchestration techniques” for big data systems.

Madhavji et al. [16] assert that software engineering (SE) tasks associated with big data applications are characterized by:

* Real-time data processing
* Large volume of historical data
* Scalability in order to cope with increasing data
* Data variety (structured or unstructured)
* Process of cleaning data streams

Chen et al. [10] asserts that architecture design is a critical process in big data systems, incurring in risks for the development process. This assumption is confirmed by Laigner et al. [1], which found that most of the work in big data systems research concern software architecture.

TODO Some big data systems support research on Smart Cities (citar trabalho do fabio kon), event-processing (citar do ken anderson), twitter, uber, etc

1. Database Systems

The choice of the persistence technology in a software project has always been a fundamental concern. A non suitable choice of persistence technology may drastically impact on time and cost [2][19].

However, the decision not only remains on choosing the best persistence paradigm among relational, NoSQL and NewSQL.

On software engineering, according to Evans [3], a domain can influence on constraints such as risk and quality depending on the complexity incurred on requirements engineering and design.

The same follows for database systems, which for given domains, such as spatial data representation, present specific data types in its data model and query language.

Data modeling must be intrinsically linked to the domain that needs to be represented in an application. However, modeling process can introduce extra efforts on current context of software development. Anderson [2] address that:

“Getting data modeling wrong in data-intensive systems is painful since it is easy to generate hundreds of gigabytes of data in the “wrong” format ... and then face the unenviable task of reformatting that data into new forms and/or migrating the data into a new persistence technology”

Last years have brought a myriad of solutions for persistence technology. Relational database management systems (RDBMS) no longer represent the unique feasible solution to a software project.

According to Moniruzzaman & Hossain [20] asserts that problems on data modeling and constraints of horizontal scalability (the authors do not cite any...) have led to the development of non-relational databases, which are able to store unstructured data.

On the other side, Gorton & Klein [13] address that “vertical-scaling limitations of SQL databases have led to [non-relational] that relax many core tenets of relational databases”.

Scalability is addressed by both works in different manners. In this work we also intend to understand the reasons behind an adoption of NoSQL databases in order to undercover this misleading concept.

In addition, [13] points out that:

“Strictly defined normalized data models, strong data consistency guarantees, and the SQL standard have been replaced by schema-less and intentionally denormalized data models, weak consistency, and proprietary APIs”

It means that data modeling was a major playing driving persistence technology in that occasion.

EXPL

D. Software Architecture

This work is particularly concerned over architectural decisions in software development, once we believe that software architecture plays a major role on decision of persistence technologies for a software project [7].

In addition, Laigner et al. [1] found that most studies on big data systems present an approach concerning the development of software architecture.

D. Related Work

Tao & Gao [4] address differences between tests in conventional applications and big data applications, asserting that in a conventional testing, non-functional requirements such as performance, reliability, availability, and security, are objects of main interest, whereas in big data systems tests rely most on supporting large-scale data input, complex data models and integrations.

3. Systematic Literature Revision

Literature present misleading concepts concerning differences between big data systems and traditional systems, as noted on last section. This work intends to help consolidate the differences, once data is a central point on every definition found, along with a systematic literature revision (SLR) with the objective to build a knowledge on how practitioners and researchers choose persistence technology in the context of software development.

In this context, a SLR is carried out in order to fully characterize existing research and current practice in the area of software engineering.

According to Kuhrmann et al. [23], “in contrast to mapping studies, systematic reviews usually cover a smaller, more specific range of publications while the analysis focuses on the details of the published contributions”.

This way, SLR are more suitable for this work, since it enables a more comprehensive aggregation and concluding results for building a knowledge, which is our intent.

The standard process defined by Kitchenham et al. [20] is employed in this work. The protocol used is shown on Figure 1 and the effort in each phase is depicted in the following sections.



**Figure 1.** Components of a systematic literature review protocol (adapted from [20])

A. Definition of Research Question

A primary research question was defined: “RQ. What drives persistence technology selection on software applications?”. Based on the primary question, complementary questions were derived in order to characterize the decision process.

* RQa. Which functional and nonfunctional requirements are take into consideration on persistence technology selection?

Requirements engineering is a fundamental life-cycle phase of software development, since approaches on agile or non-agile address this phase as the primary factor for understanding user needs. This way, it is expected that requirements play a major role on defining persistence technology. We expect to find what are the most relevant requirements that are take into consideration during the decision process on persistence.

* RQb. On which degree framework, platform and programming language are take into consideration on persistence technology selection?

Frameworks have long supported the development of applications, enabling fast prototyping, diminishing time spent, allowing the developers work on the core functions of a system. Johnson & Foote [23] define framework as “a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes”.

Some frameworks have historically evolved to support a wide range of functionalities, such as Spring Framework for Java platform. The Spring “ecosystem” presents data persistence, testing, dependency injection, and transaction capabilities, for example.

It would be better to break down this question???

In addition, platforms for development such as Java and .NET support a variety of technologies, such as programming languages, server components, and frameworks. However, it is possible that some quality attributes, such as performance, does not fulfill what is expected for an application, having an direct impact on persistence technology.

Limitations, in terms of maturity and lack of testing, in drivers for establishing connections to a database is an example that might trigger developers consider choosing another technology for persistence in their applications.

In this context, we would like to find out patterns on the selection of platform, language and framework and its relation with persistence technology chosen.

* RQc. How domain have importance on persistence technology selection?

According to Evans [30], application domain can impact in constraints such as risk and quality of a software project, according to the complexity related to the design of the solution.

We expect to find correlations between the application domain, the data modeling patterns and the chosen persistence technology.

* RQd. How software architecture (systems integration, data availability, API, ETL, BI) plays a role on persistence technology selection?

Architectural styles, software architecture ...

B. Definition of Search Strategy

We aim at characterizing, with the advent of multiple persistence solutions over the last years, how modern software development tackles the problem of selecting an adequate solution to store data processed handled by a software system.

The search strategy should contemplate two aspects:

* Studies proposing strategies/techniques/tools for selecting persistence technology (database) in the context of software development
* Studies that address persistence selection process in the context of software development

As shown on Table 1, a set of studies, targeted to the aspects above, were previously identified in order to obtain an overview of main keywords, their reference and related studies. In order to accomplish this task, Google Scholar and Scopus datasources was used.

These previously studies are generally called control papers. In other words, these control studies support the process of defining the search string for retrieval of studies. It means that the search string should be adjusted to return the control papers and other relevant studies. However, it is important to note that the control papers are necessarily included in the selected papers.

Table 1. Control Papers Selected

|  |  |
| --- | --- |
| Article | Source |
| MySQL to NoSQL Data Modeling Challenges in Supporting Scalability | [24] |
| Application-Specific Evaluation of NoSQL Databases | [25] |
| Choosing the right NoSQL database for the job: a quality attribute evaluation | [26] |

Next, exploratory searches with different search strings were employed. This strategy is depicted in [27], and suggests that “he purpose aims at iteratively narrowing down the list of potential candidates by checking whether … a search query returns a (potentially) meaningful result set [and] keyword or a combination thereof returns hits”.

The first tentative involved the following keywords for submission to digital libraries:

* Population:
  + Database; persistence;
* Intervention:
  + Selection, choice, transition;;

Table 2 presents the search string derived from this first tentative.

Table 2. Search string for first tentative

|  |
| --- |
| ( database OR persistence )  AND |
| ( selection OR choice OR choosing OR select OR selecting )  AND |
| LIMIT-TO ( SUBJAREA , "COMP " ) |

This general search string was first applied due to author’s experience on search string definition, which found out that more general strings lead to better results.

However, the search retrieved few papers addressing database selection (from a set of 303 results) and most of them were studies older than 10 years. It is important to observe that it was necessary to limit to Computer Science area due to several studies out of the scope and only to titles, due to a huge amount of papers out scope returned.  
After an auxiliary tentative using the keywords “data modeling”, “sql”, and “nosql”, the following keywords were used:

* Population:
  + Database; persistence;
  + Software system, software engineering, software architecture, software infrastructure, software development;
* Intervention:
  + Selection, choice, transition;

In addition, as we seek to address the current context of software development, it was important to limit studies within a date range. In order to define the cut-off date, it was observed the emergence of new technologies considered in software development projects, such as MongoDB, Hadoop, and Cassandra. This way, the results were limited to studies published starting from 2008 until 2017.

Table 3 presents the search string derived after the Trail-and-Error Search [27].

Table 3. Final search string for submission

|  |
| --- |
| ( database OR persistence )  AND |
| ( "software system" OR "software engineering" OR "software architecture" OR "software infrastructure" OR "software development" )  AND |
| ( selection OR choice OR choosing OR select OR selecting OR transition OR transitioning ) |

The search string was applied to titles, abstracts and keywords. Since it is known that full text search are likely to retrieve studies from domains outside software engineering field, full text search were not applied. The results retrieved a set of 596 results and in general, they are consistent with the characterization of the work.

* Data Sources

According to Mourao et al. [28] “available digital libraries (e.g., ACM DL, IEEExplore, Scopus, Science Direct, and Web of Science) are not designed to support SLRs. They contain significant content overlaps, provide different interfaces, and present search limitations (e.g. on logical operators and number of terms).”

This way, we opt to choose, based on Mourao et al. [28], a hybrid search strategy employing a single digital library with backward and forward snowballing, since it was observed that snowballing is an efficient method for retrieving relevant studies for inclusion

The digital library used for our search string submission was Scopus, since it is a data source that index work on most relevant digital libraries.

For forward snowballing, it is important to observe that I am going to use Google Scholar, once it provides a consistent index for citing studies.

C. Definition of Selection Criteria

As the results showed a large number of studies, many of them did not meet the relevance requirements for this work. Therefore, it was necessary to define a set of rules in terms of inclusion and exclusion criteria.

Inclusion Criteria

* Studies providing guidelines or reporting experience on persistence technology selection in the context of software development

Exclusion Criteria

* Papers where focus on software development is lacking
* Studies not reported on English language
* Non peer-reviewed studies (white papers, thesis, conference review, article in press, short survey)
* Abstracts and presentations

D. Selection Process

The Data Sources topic have depicted that our selection process will be based on Mourao et al. [28]. In addition, intermediary tasks will be accomplished in order to guarantee a set of final relevant papers. This way, the selection process is divided into 5 stages, as shown on Figure 3.



Figure 3. The filtering process of the studies

The first step consists on searching the papers using the previously defined search string on Scopus.

In the second step, the first filter is applied, in order to remove non peer-reviewed publications, including abstracts and powerpoint presentations, for example. Also, papers not written in English are also excluded.

In the third step, filtering is carried out by reading the title, abstracts and introductions of each work. In case of non compliance to our study objective, the study is discarded. When the decision incur in dubiety, the paper is considered on the next step.

In the fourth step, the papers are read in full in order to respond to the doubt that arose in last step.

For last, the snowballing is applied by checking the references of each selected study and verifying which studies cite them.

E. Study Quality Assessment

Assessment of selected studies can indicate limitations and provide directions on current challenges in the field of software development. This way, I intend to use score based review as a procedure to assess quality.

Based on Alves et al. [29], a set of questions regarding the level of evidence and quality of extracted data on each study will be defined.

Q1: How much evidence supports the claims related to the decision process of persistence technology in the study?

Level 0: No evidence

Level 1: Evidence obtained by toy examples

Level 2: Evidence based on expert opinions or observations

Level 3: Evidence obtained from academic studies

Level 4: Evidence obtained from industrial studies

Level 5: Evidence obtained from industrial practice

Q2: Is there a clear line of reasoning about the set of options available in the decision process?

Q3: Are the limitations of the study clearly discussed?

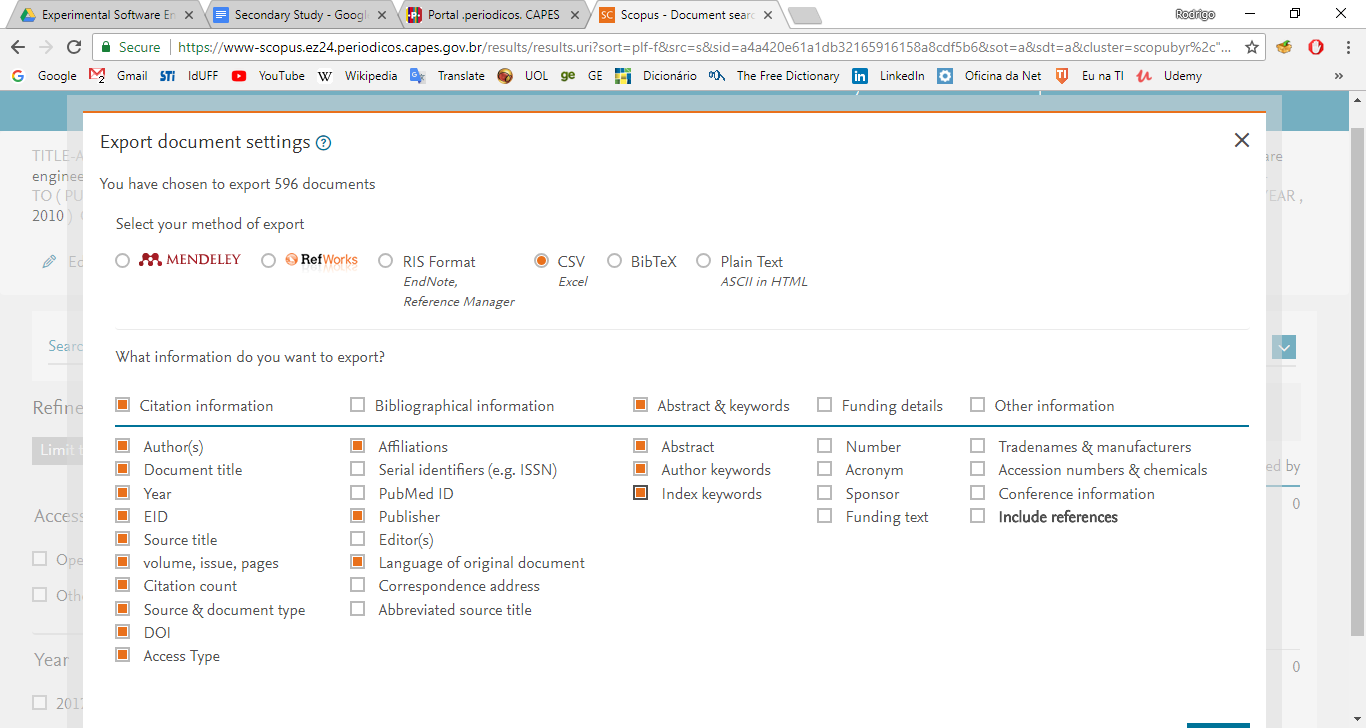
For applying this procedure, I intend to find out if the selected studies presents substantial evidence, once the grade reflects the quality and credibility of the results.

F. Data Extraction Strategy

Scopus provide a functionality to extract a set of results from a search. The extraction will contain the fields shown on Figure 4.

The retrieved studies are managed into a unique online spreadsheet, where all modifications along the time can be traced. It allows researchers to audit the process in a very clear manner.

Also, the spreadsheet will be used to easily identify duplicate results and also score them (quality assessment process).  
  
For last, the spreadsheet must contain all the metadata needed for the execution of the mapping. As metadata, I understand that must reflect the answers obtained from the research questions, such as programming languages, domain, database, context of the study, requirements, and more. It is possible that metadata change over time, due to patterns on data set and knowledge acquired during the selection process.



**Figure 4.** Fields extracted from Scopus datasource

* Classification Scheme

On the current state of progress, 24 articles were analysed. For now, the attributes collected for each study are shown in Table 4.

Table 4. Current Classification Scheme

|  |  |
| --- | --- |
| Information | Description |
| Problem Target | It describes the main reason of the study, what is the target of the study being conducted |
| Approach | Concerns a type of analysis employed, such as qualitative and performance, or even the usage of engineering approach, such as Model-Driven Engineering |
| Domain | Application Domain on which the study is conducted. |
| Type of Study | Classification of the empirical evaluation, according to Wohlin et al. [X]. It includes categories such as: experiment, case study, or survey. |
| Type of Research | Classification of research, according to Wieringa et al. [X], including the following categories: evaluation research, proposal of solution, validation research, philosophical paper, opinion paper, or experience paper. |
| Type of Author | Industry, Academia or both. The ambient on which the study is conducted. |
| Software Requirements | The requirements involved in the software under development |
| Persistence Technology Impact on Software | Concerns the impact on which the persistence technology involved in the study has on software being developed |
| Persistence Paradigm | The persistence technology paradigm. Divided in: relational, NoSQL and NewSQL |
| Persistence Technology | The set of persistence technology involved in the study. For instance, MySQL, MongoDB, SQL Server, and Cassandra are examples. |

G. Synthesis of Extracted Data

The spreadsheet used was customized to automatically indicate the reviewer which studies are not written in English or a non peer-reviewed publication. It can help the reviewer, since less effort is spend along the process.

I expect to find correlations among domains, platform (.NET, Java), architectural patterns and decisions, functional and nonfunctional requirements, and data modeling.

The characteristics of each primary study will be categorized and patterns on selecting persistence technology can be discovered and gaps uncovered.

Categories will be shown on tables and figures, making use of bars, columns, and bubble-plot. For instance, the Figure 5 and 6 present examples of results.

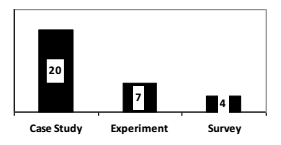


Figure 5. Empirical Study Distribution over Primary Studies (extracted from [1])

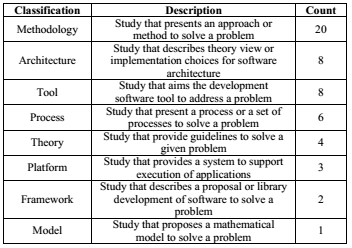


Figure 6. Number of Studies by Type of Contribution (extracted from [1])

H. Dissemination Strategy

The nature of this work presents two opportunities for publishing, once address concerns over the fields of database and software engineering.

As a primary point for collecting insights from local researchers, it would be a good idea to present the first results to a Brazilian Conference. SBBD and SBES are candidates for publishing.

Once the revisions are obtained, I seek to improve this work in order to increase maturity and submit to conference calls to empirical studies on software engineering field, such as:

SEAA  
CIbSE  
ESEM  
EASE

-

Other

Map software characteristics that are most prominent when it comes to define a persistence technology

Gorton et al. defines a model for evaluating distributed databases in the context of big data systems.

At first, it should be noted that relational databases can fit the needs of a big data systems. For example, Uber…. Relational databases also have replication features… Distribution of records for better performance is fully supported by….

Second, our work fit on the SDLCP before Gorton et al. work, which means we try to establish the basis on what relevance information about the software

Their work are fundamentally fixed on NoSQL databases. We want to understand first how to define that a noSQL database is the only possible solution.

Base on the taxonomy proposed by Gorton et al. enrich to introduce sql databases and not just escalable concerns.

References

[1] A Systematic Mapping of Software Engineering Approaches to Develop Big Data Systems. Laigner, R.; Kalinowski, M.; Lifschitz, S.; Salvador, R.; and de Oliveira, D. In 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018, Prague, Czech Republic, Aug 29-31, 2018.

[2] Embrace the Challenges: Software Engineering in a Big Data World - Kenneth Anderson  
2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering - 05 / 2015

[3] Ralf Hartmut Gfiting. An Introduction to Spatial Database Systems. VLDB Journal,3, 357-399. 1994.

[4] Quality Assurance for Big Data Application – Issues, Challenges, and Needs - Chuanqi Tao, Jerry Gao. Proceedings of the 28th International Conference on Software Engineering and Knowledge Engineering - 07 / 2016

[5] Runtime Performance Challenges in Big Data Systems - John Klein, Ian Gorton  
Proceedings of the 2015 Workshop on Challenges in Performance Methods for Software Development - WOSP '15 - / 2015

[6] Developer toolchains for large-scale analytics: Two case studies - Stephanie Rosenthal, Scott Mcmillan, Matthew Gaston 2015 IEEE International Conference on Big Data (Big Data) - 10 / 2015.

[7] Gorton, Ian. Essential Software Architecture. Springer, 2014.

[8] Sadalage P.J., and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Pearson, 2013.

[9] Chen, Hong-Mei, et al. “Agile Big Data Analytics for Web-Based Systems: An Architecture-Centric Approach.” IEEE Transactions on Big Data, vol. 2, no. 3, Jan. 2016, pp. 234–248., doi:10.1109/tbdata.2016.2564982.

[10] Chen, Hong-Mei, et al. “Big Data System Development: An Embedded Case Study with a Global Outsourcing Firm.” 2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering, 2015, doi:10.1109/bigdse.2015.15.

[11] Chen, Hong-Mei, Kazman, R., Haziyev, S. “Strategic Prototyping for Developing Big Data Systems.” IEEE Software, 2016, pp. 1–1., doi:10.1109/ms.2016.65.

[12] “CIOs & Big Data,” http://visual.ly/cios-big-data, retrieved July. 2018

[13] Gorton, Ian, and John Klein. “Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems.” IEEE Software, vol. 32, no. 3, 2015, pp. 78–85., doi:10.1109/ms.2014.51.

[14] “MongoDB Case Study: Craigslist.” MongoDB, www.mongodb.com/post/15781260117/mongodb-case-study-craigslist. Retrieved July. 2018.

[15] Özsu M. Tamer., and Patrick Valduriez. Principles of Distributed Database Systems. Prentice Hall International, 1999.

[16] Madhavji, Nazim H., et al. “Big Picture of Big Data Software Engineering: With Example Research Challenges.” 2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering, 2015, doi:10.1109/bigdse.2015.10.

[17] “Pentaho Data Integration - Accelerate Data Pipeline.” Pentaho Evaluation Support | Hitachi Vantara, www.hitachivantara.com/en-us/products/big-data-integration-analytics/pentaho-data-integration.html.

[18] Ozsu, M.t., and P. Valduriez. “Distributed Database Systems: Where Are We Now?” Computer, vol. 24, no. 8, 1991, pp. 68–78., doi:10.1109/2.84879.

[19] “Why You Should Never Use MongoDB.” Sarah Mei, 12 Nov. 2013, www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb/.

[20] Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. Technical Report. EBSE-2007-01 Keele University

[21] NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison.

[22] Schram, Aaron. Software Architectures and Patterns for Persistence in Heterogeneous Data-Intensive Systems. Thesis directed by Associate Professor Kenneth M. Anderson. 2015

[23] Johnson, R. E. & Foote, B. Designing Reusable Classes. Journal of Object-Oriented Programming June/July 1988, Volume 1, Number 2, pages 22-35.

[24] Schram, Aaron, and Kenneth M. Anderson. “MySQL to NoSQL.” Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity - SPLASH '12, 2012, doi:10.1145/2384716.2384773.

[25] Klen, J. & Gorton, I.Application-Specific Evaluation of NoSQL Databases. Proc. IEEE Big Data Congress. 2015.

[26] Lourenço, João Ricardo, et al. “Choosing the Right NoSQL Database for the Job: a Quality Attribute Evaluation.” Journal of Big Data, vol. 2, no. 1, 2015, doi:10.1186/s40537-015-0025-0.

[27] Kuhrmann, Marco, et al. “On the Pragmatic Design of Literature Studies in Software Engineering: an Experience-Based Guideline.” Empirical Software Engineering, vol. 22, no. 6, June 2017, pp. 2852–2891., doi:10.1007/s10664-016-9492-y.

[28] Mourao, Erica, et al. “Investigating the Use of a Hybrid Search Strategy for Systematic Reviews.” 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2017, doi:10.1109/esem.2017.30.

[29] Alves, Vander, et al. “Requirements Engineering for Software Product Lines: A Systematic Literature Review.” Information and Software Technology, vol. 52, no. 8, 2010, pp. 806–820., doi:10.1016/j.infsof.2010.03.014.

[30] Evans, Eric. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2014.