Quality in use evaluation of a GraphQL implementation

Antonio Quiña-Mera[1,2], Pablo Fernández-Montes[2], José María García[2], Edwin Bastidas[1], and Antonio Ruiz-Cortés[2]

[1] Faculty of Engineering in Application Science, eCIER Research Group, Universidad Técnica del Norte, Ibarra 100105, Ecuador,
aquina@utn.edu.ec
[2] SCORE Lab., I3US Institute, Universidad de Sevilla, Sevilla 41012, Spain

Abstract. The software development trend uses service-oriented software architecture (SOA), which provides efficiency, agility, and ease of growth. The architectural design most commonly used in SOA application development is REST (Representational State Transfer); however, some data management problems have been identified in its Application Programming Interface called API-REST. Several technological options have emerged to appease these problems, such as SPARQL, Cypher, Gremlin, and the most popular GraphQL. GraphQL was developed by Facebook in 2012 and released in 2015 to the community as an open-source project, used by companies such as GitHub, Airbnb, Amazon, Apollo, IBM, and Facebook. The goal of this research is to demonstrate whether GraphQL implementations work. Therefore, we based the research design on Design Science Research (DSR) to evaluate the quality-in-use of a GraphQL implementation that automated the systematic mapping studies (SMS) process for technology researchers at Universidad Técnica del Norte - Ecuador. We used the ISO/IEC 25000 series of standards to evaluate the quality in use; the results showed that the implementation met 84.11% of the established quality model's expected value. The detailed evaluation by quality characteristics was: Effectiveness 96.62%, Efficiency 78.90%, and Satisfaction 70.26%.

Keywords: GraphQL, SOA, Quality in use, ISO/IEC 25000

1 Introduction

A technological trend is the consumption of services through the Internet; this practice is called Cloud Computing; its benefits are cost savings, efficiency, agility, growth opportunities, and innovation [1]. Cloud Computing offers three areas of services: software, platform, and infrastructure [2]. We focus on the software as a service (SaaS) model, which distributes computer applications hosted by cloud service providers and made available to users through a network without the need to download or install them [3]. The most popular software architectures for the development of applications that support SaaS are the Service-Oriented Architectures (SOA) and Microservice Oriented Architectures (MSA), the latter being a modification of SOA [4]. REST (REpresentational State Transfer) is the most widely used SOA and MSA architectural design to develop Application Programming Interfaces called API-REST or API-RESTFUL [5]. Despite the acceptance of REST in the scientific and technological community, it has presented some problems in data handling, such as over-fetching (occurs when the data provider delivers more information than the client requires in a request). And under-fetching (occurs when the data provider does not offer in a query all the information that the client needs, therefore it must make more requests to obtain the complete information) [6]. Several technological options have emerged to improve the REST problems, such as SPARQL, Cypher, Gremlin, and the most popular of these GraphQL [7]. GraphQL is a query language and execution engine for data in client-server applications that has been accepted in the technology community because it was developed and used in the products of the company Facebook [8]. Created in 2012 and released in 2015 to the community as an open-source project and used by technology and application development companies such as 8base, Airbnb, Amazon Web Service, Apollo, Dgraph Labs, Elementl, Facebook, Fauna, Gatsby, Hasura, HomeAway, IBM, Intuit, Neo4j, PayPal, Prisma, Salsify, Shopify, Solo.io, Twitter [9].

Accordingly, we base this study on the Design Science Research (DSR) approach and pose the research question Does the GraphQL implementation work?. The research aims to answer the research question by evaluating the quality-in-use of a GraphQL component's implementation. We evaluated the quality-in-use using the ISO/IEC 25000 series of quality standards known as SQuaRE (System and Software Quality Requirements and Evaluation). The GraphQL implementation consisted of automating Systematic Mapping Studies (SMS) management proposed by technology researchers. We evaluated the quality in use using the ISO/IEC 25000 series of quality standards known as SQuaRE (System and Software Quality Requirements and Evaluation) at the Software Department of the Universidad Técnica del Norte - Ecuador. The rest of paper is structured as follows: Section 2) Research Design: we establish the research activities based on DSR, theoretical foundation, and artifact design and build (software implementation). Section 3) Results: results of the evaluation of the artifact quality in use. Section 4) Discussion: discussion of the research. Section 5) Conclusions.

2 Research design

We designed the research based on the guidelines of the Design Science Research approach [10, 11], see Table 1.

2.1 Population and sample

The population of professors of the Software Department of the Universidad Técnica del Norte is twenty-two. The sample calculation with a margin of error of 5%, a heterogeneity of 99%, and a confidence level of 99% is thirteen subjects.

Table 1. Research design

Activity	Components	Paper Section
Problem Diagnosis	Problem; Objective. Population and sample; Acceptance survey.	Introduction Research design
Theoretical foundation	Software Architecture; SCRUM Framework; GraphQL; Systematic Mapping Study; ISO/IEC 25000 Standards: Quality in Use.	Research design
Artefact Design: GraphQL implementation	Requirements; Design (Process and Architecture); Development and Deployment.	Research design
Artifact evaluation	Quality in use evaluation of a GraphQL implementation	Results

However, in the quality-in-use model's measurement instruments (practical workshop and satisfaction survey), the non-probabilistic convenience sampling technique was applied. The sample consisted of 40 subjects (15 professors and 25 students) who had adequate knowledge to perform the practical workshop.

Acceptance survey. To establish the automation topic's acceptance (SMS Management) for GraphQL implementation, we surveyed the study population. The survey consisted of 6 questions validated by three expert researchers in Software Engineering before the survey. The survey results indicated that 45.45% of the respondents knew about literature review methods (SLR[3]/SMS), but no one knew a tool to carry them out; furthermore, 100% indicated that they would like training about SMS management.

2.2 Theoretical foundation

Software architecture. Software architecture has evolved in different architectural approaches over the years. In the '80s, structures were vertical and isolated; in the '90s, horizontal models focused on business processes appeared, and nowadays, software architecture is focused on continuous delivery of different services. This evolution leads companies to migrate their monolithic architectures (combining user interface, business logic, and data in the same application towards Service Oriented Architecture (SOA) and Microservices (small services that operate independently) [13] [4]. SOA and microservices present an approach to building distributed systems that deliver web services (self-describing, modular business applications that export business logic as a service via the Internet [14];

[3] A systematic literature review (SLR) is a means of identifying, analyzing, and interpreting reported evidence related to a set of specific research questions [12].

its implementation has also had an evolution ranging from building web services based on WSDL and SOAP protocol to building REST and GraphQL Web APIs [13].

GraphQL. Created in 2012 by Facebook as an internal project and then released in 2015 to the community as an open-source project [15]. GraphQL is a runtime and API data query language [8]. It arises with the need to reduce the overhead of transferred data and the number of API query requests; reduce the number of invalid client query errors; support the evolution of the data model without an API version [16]. GraphQL service is composed of a type system, executable operations, or by extensions of external type systems. The type system defines the schema of the data capability of the GraphQL API service. The schema is defined in terms of types (objects, scalars, Enums, input, interfaces, and unions), types of operations (queries, mutations, and subscriptions), and directives (describes alternate runtime execution) [8].

SCRUM Framework. Establishes processes to manage product development with an interactive and incremental life cycle that promotes adaptive planning, development, and evolutionary delivery of the software product that drives rapid response to change. Scrum provides a usage guide comprised of phases, roles, events, artifacts, and best practices [17].

Systematic Mapping Study (SMS). Provides an overview of a research topic through a classification [18]. The SMS focuses on conducting a literature search that answers research questions usually are what, when, and where have been published about a research topic [19]. The main phases of SMS are review planning, study identification, and data extraction and classification [20].

ISO/IEC 25000 Standards. Called SQuaRE (System and Software Quality Requirements and Evaluation), focuses on the specification, measurement, and evaluation of software product quality requirements. SQuaRE establishes a model for measuring the internal, external, and in-use quality of software. The quality-in-use requirements specify the required level of quality from the end user's point of view; it is a validation of the software product from the needs of a context of use [21]. For quality-in-use evaluation, we define a quality model, measure the model, and evaluate it according to its measurement. We determined the quality-in-use model based on the ISO/IEC 25010 standard, establishing the quality characteristics and sub-characteristics. In the measure of the model, we use the metrics defined in ISO/IEC 25022. For model evaluation, we use the recommendations of ISO/IEC 25040.

2.3 Software implementation (Artefact Design)

Requirements. The requirements backlog for the Systematic Mapping Study (SMS) management automation we established in 15 user stories.

Design (Process and Architecture). We used the Petersen et al. guide to establish the three-phase process for conducting SMS: Phase 1: Review planning, Phase 2: Study identification, Phase 3: Data extraction and classification. We designed a client-server application based on SOA service-oriented architecture using the provider, an API-GraphQL. The client, a web application integrated with the API-Mendeley for bibliographic reference management, see Figure 1.

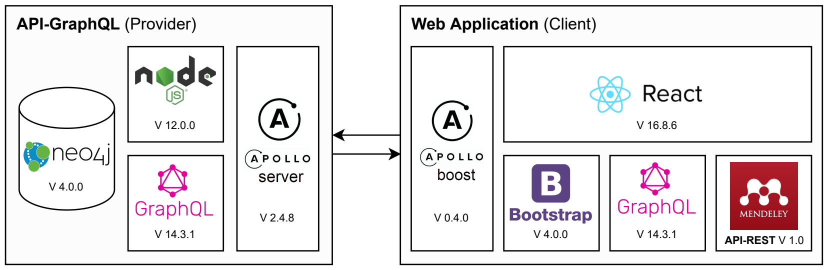


Fig. 1. Software product architecture design.

Development and Deployment. The application development was performed from 03/06/2019 to 06/12/2019 (220 hours) using the SCRUM framework. The application (named "SMS-Online") automates the Systematic Mapping Study process. The work team consisted of two stakeholders, one product owner, one scrum master, and one developer. Below is a summary of the phases and artifacts used in the software product implementation, see Table 2.

Table 2. Development and implementation summary

Phase	Sprint (duration)	Deliverables
Pre-Game	Sprint 0 (20 hours)	Requirements: - 15 user stories - Product Backlog Design (Architecture): - Process Diagram - Software Architecture - Initial Database Diagram
Game	Sprint 1 (42 hours) Sprint 2 (42 hours) Sprint 3 (42 hours) Sprint 4 (42 hours)	- Spring Backlog - Software Product Increment
Post-Game	Sprint 5 (32 hours)	- 15 Acceptance Tests - Deployment of the software product - Delivery-Receipt Act

In the development, we verified the GraphQL components implemented in SMS-Online. Besides, we noticed that using the neo4j-graphql.js package (A JavaScript package to make it easier to use GraphQL and Neo4j together) translates GraphQL queries to a single Cypher[4] query, eliminating the need to write queries in GraphQL resolvers and to perform batch queries [23]. Table 3 shows the components implemented manually and generated by neo4j-graphql.js.

Table 3. GraphQL components implemented

Component clasification	Component	Implemented	Provider		Client
			Generated	Developed	Components
Executable Definition	Query	✓	12	0	27
	Mutation	✓	62	1	40
	Subscription	X	0	0	0
Type System Definition	Schema	✓	0	1	0
	Scalars	✓	0	132	132
	Objects	✓	0	12	12
	Interfaces	✓	0	1	1
	Unions	✓	0	1	1
	Enums	✓	0	2	2
	Input Object	✓	63	0	0
	Directives	✓	0	14	0

Access to the SMS-Online application: https://appsms.utn.edu.ec:3000

3 Results (Artifact evaluation)

As a result, we show the quality in use evaluation of the GraphQL implementation developed in the previous section. Next, we offer the evaluation process.

3.1 Definition of the quality in use model

The Product Owner and the Scrum Master of the project defined the quality in use model according to the ISO/IEC 25010 guidelines and the implementation context's quality needs. The model was structured with the expected valuation (percentage of the total) of the quality characteristics and sub-characteristics established in the model, up to complete 10% [24], see Table 4.

[4] Cypher is Neo4j's graph query language [22].

Table 4. Quality in use model

Characteristics	Sub characteristics	Percentage Sub-characteristic	Percentage Characteristic
Effectiveness	Tasks completed	16%	42%
	Objectives achieved	16%	
	Errors in a task	16%	
Efficiency	Task time	16%	32%
	Time efficiency	16%	
Satisfaction	Usefulness	8%	26%
	Trust	8%	
	User experience	8%	

3.2 Measurement of the model

We identified a set of metrics described in ISO/IEC 25022 [25] associated with measuring the sub-characteristics of the quality-in-use model defined in the previous section. The metrics describe the measurement functions and their elements, see Table 5.

Table 5. Quality in use measures

Characteristic	Sub characteristic	Measurement function	Measurement elements
Effectiveness	Tasks completed	X=A/B	A=Number of unique tasks completed. B=Total number of unique tasks attempted.
	Objectives achieved	$X = 1 - \sum Ai$ $\lvert X \geq 0$	Ai = Proportional value of each missing or incorrect objective in the task output (maximum value = 1).
	Errors in a task	X = A	A = Number of errors made by the user during a task.
Efficiency	Task time	X = T	T = Task time.
	Time efficiency	X = A/T	A = Number of objectives achieved. T = Time.
Satisfaction	Usefulness	X = N/T	N=Number of satisfied users. T=Number of users surveyed.
	User trust	C = A / T, X=1-C	X=% complaints. C=% Trust. A=Number of complaints filed. T=Total number of survey respondents.
	User experience	X=A	A = Psychometric scale value from a pleasure questionnaire (Likert Scale)

We used two instruments to obtain the measurement elements shown in Table 5: (i) Practical workshop on the use of the SMS-Online application, for data on effectiveness and efficiency of quality in use; and (ii) A satisfaction survey (based on SUS[5] [26]) on the use of the SMS-Online application after the practical workshop. We applied the measurement instruments to the sample of subjects (40 subjects) established in Section 2.1. After applying the instruments, we assessed reliability using statistical methods.

Statistical evaluation of the measurement instruments. We evaluated the instruments' reliability according to their types of variables and the results obtained in the measurement applied to the study subjects, using the RStudio[6] tool for the calculations.

Practical workshop. we used the statistical method Cronbach's Alpha coefficient; the analysis matrix structure contains the following measurement elements: tasks attempted, tasks completed, objectives completed, errors in a task, time tasks, objectives achieved, number of complaints, and confidence. The result obtained is 0.89, indicating that the workshop's reliability is acceptable according [28] [29], see Figure 2.

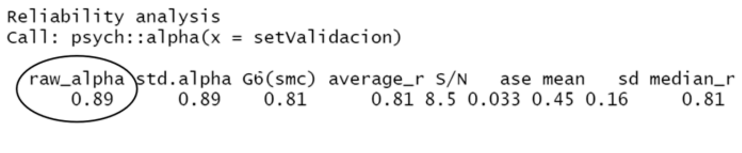
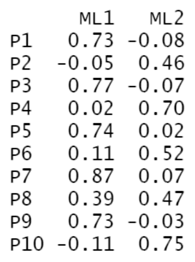

```
Reliability analysis
Call: psych::alpha(x = setValidacion)

 raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd median_r
      0.89      0.89    0.81      0.81 8.5 0.033 0.45 0.16     0.81
```

Fig. 2. Cronbach's alpha of the practical workshop.

Satisfaction Survey. we used the Confirmatory Factor Analysis (CFA) technique; the factor structure contains the factors: usefulness (ML1) and comfort (ML2). We analyzed the correlation between the initial survey design and the CFA calculation of the survey execution results (see Figure 3). The initial design of ML1 contains the questions: Q1, Q6, and Q9, of which we discarded Q6 for correlating less than 0.3 (low level to maintain a viable survey). The initial design of ML2 contains the questions: P2, P3, P4, P5, P7, P8, and P9, of which we discarded questions: P3, P5, P7, P9 for having a low correlation. After analyzing and choosing the correlated questions, we calculated the CFI (Comparative Fit Index) and the TLI (Tucker and Lewis) index with the same AFC algorithm. We obtained CFI=0.94 and TLI=0.90, considered acceptable [30], and confirm that the survey is valid and reliable.

[5] The System Usability Scale (SUS) provides a "quick and dirty", reliable tool for measuring the usability [26].

[6] RStudio is an integrated development environment (IDE) for R, a programming language for statistical computing and graphics [27].



```
      ML1   ML2
P1   0.73 -0.08
P2  -0.05  0.46
P3   0.77 -0.07
P4   0.02  0.70
P5   0.74  0.02
P6   0.11  0.52
P7   0.87  0.07
P8   0.39  0.47
P9   0.73 -0.03
P10 -0.11  0.75
```

Fig. 3. Correlation between survey questions (CFA calculation).

3.3 Evaluation of the quality in use model

We evaluated based on the expected values established in the quality model and the model's measurement results, see Table 6.

Table 6. Evaluation of the quality in use model

Characteristics	Sub characteristics	Measurement	Expected	Achieved	Completed
Effectiveness	Tasks completed	0.97	16%	15.50%	96.62%
	Objectives achieved	0.97	16%	15.50%	
	Errors in a task	0.96	16%	9.58%	
Efficiency	Task time	0.78	16%	12.51%	78.90%
	Time efficiency	0.80	16%	12.74%	
Satisfaction	Usefulness	0.71	8%	7.05%	70.26%
	Trust	0.70	8%	5.60%	
	User experience	0.70	8%	5.62%	
			Total	84.11%	

The result obtained from evaluating the quality model in use is ***84.11%***, which is considered an ***"opportunity"*** to improve economic, health, or environmental outcomes, according to the risks and opportunities associated with the quality level established in ISO/IEC 25022 [25].

4 Discussion

Before proposing the automation of the Systematic Mapping Study (SMS) process as an implementation of GraphQL called SMS-Online, we checked in the bibliographic databases Scopus, DBLP, and Springer for similar proposals. We found the studies of Knutas et al. [31] and Kohl et al. [32] that expose systematic literature review (SLR) and SMS automation tools. Knutas proposes the application called NAILS, which connects to the bibliographic database "web of science" to choose a data collection of studies and export in files imported into web-based analysis server HAMMER to display a statistical analysis of the entered data. They also offer an option to download an R language-based application for the study of the data collection. The difference with SMS-Online is the systematization of specific data in the SMS data extraction, the documentation of the entire SMS process, and the integration of studies from various bibliographic databases; this helps researchers organize and document their studies in a maintainable way and other researchers to reproduce.

Kohl reviews Online tools supporting the conduct and reporting of systematic reviews and systematic maps, showing 22 applications, of which two applications (Colandr and CADIMA) support SMS systematization. We reviewed the functioning of Colandr (free access), where we were able to perform Planning, Citation Screening, Full-text Screening, Data Extraction, and import of data collections through flat files. The similarities between Conlandr and SMS-Online are that they can conduct the primary SMS phases; the differences, SMS-Online offers help documentation at each step of the process, import primary-studies API-Mendeley and connects them directly to Mendeley Web. We also reviewed the CADIMA tool (free access), which presented detailed documentation at each step of the process; however, it was difficult to use it, which suggested having previous knowledge of the application to use it correctly. The advantage over SMS-Online is the option of collaborative work of several users in the same SMS study. The functionality of finding duplicate studies in SMS-Online is managed with the functionalities of Mendeley.

SMS-Online complements its functionalities with Mendeley's SMS tool functionalities; it displays and exports SMS results for use in documents such as scientific articles, scientific posters, or papers. The limitations found in SMS-Online are that it doesn't collaborate with several users in the same SMS and can not perform a check for duplicates in selecting studies. We propose these limitations as future work in updating the implementation of the SMS-Online application.

5 Conclusions

In this paper, we use the DSR approach to formulate the research question, Does the GraphQL implementation work?. We answer by evaluating the quality-in-use of implementation of GraphQL components. The implementation consisted of automating the management of Systematic Mapping Studies (SMS) named SMS-Online, directed to technology researchers of the Software Department of the Universidad Técnica del Norte - Ecuador in the academic period 2019, those who accepted (survey) with 100% the proposal of the automation topic.

GraphQL implementation was performed in a client-server web application using SOA (Service Oriented Architecture) and integrated with the Mendeley API-REST to complement the bibliographic management functionality. We developed using the SCRUM framework and the following technologies: Provider API-GraphQL (Apollo Express, GraphQL, JavaScript, Node.js, Database: Neo4J), Client Web Application (Apollo Scient, GraphQL, React, Bootstrap, and Mendeley API-REST). We implemented 301 GraphQL components in the SMS-Online application: 12 queries, 63 mutations, 1 schema, 12 object, 132 scalars, 1 enums, 63 input objects, 1 interface, 1 union, and 14 directives. We evaluated the implementation using the quality-in-use model based on the ISO/IEC 25000 series of standards. The evaluation had three phases: 1) Definition of the quality-in-use model based on ISO/IEC 25010; 2) Measurement of the quality model based on ISO/IEC 25022; and 3) Evaluation of the quality model based on ISO/IEC 25040. The evaluation result showed that the application's implementation complied with 84.11%, which is considered an "opportunity" to improve economic, health, or environmental outcomes; where, the evaluation by characteristics was: Effectiveness 96.62%, Efficiency 78.90%, and Satisfaction 70.26%. According to these results, on the one hand, we proved that the implementation of GraphQL components is acceptable and work in SOA applications, thus answering the research question. On the other hand, we note that it is useful and convenient to use ISO/IEC standards to evaluate the software product's quality and identify improvement features in the applications.

As future work, we note the need to evaluate the implementation of all GraphQL components; improve the limitations found in the SMS-Online application; evaluate the internal and external quality and replicate the quality-in-use of GraphQL components.

References

1. Wei Tek Tsai, Xiao Ying Bai, and Yu Huang. Software-as-a-service (SaaS): Perspectives and challenges. *Science China Information Sciences*, 57(5):1–15, 2014.
2. Peter M Mell and Timothy Grance. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology, Gaithersburg, 2011.
3. Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing - The business perspective. *Decision Support Systems*, 51(1):176–189, 2011.
4. Justus Bogner and Alfred Zimmermann. Towards Integrating Microservices with Adaptable Enterprise Architecture. In *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW*, volume 2016-Septe, pages 158–163, Toronto, 2016. Springer.
5. Hyuck Han, Shingyu Kim, Hyungsoo Jung, Heon Y. Yeom, Changho Yoon, Jongwon Park, and Yongwoo Lee. A RESTful approach to the management of cloud infrastructure. In *CLOUD 2009 - 2009 IEEE International Conference on Cloud Computing*, pages 139–142, 2009.
6. Maximilian Vogel, Sebastian Weber, and Christian Zirpins. Experiences on migrating RESTful Web Services to GraphQL. In *ICSOC Workshops 2017*, page 283–295, Malaga, 2018. Springer Verlag.
7. Philipp Seifer, Johannes Härtel, Martin Leinberger, Ralf Lämmel, and Steffen Staab. Empirical study on the usage of graph query languages in open source Java projects. In *SLE 2019 - Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, co-located with SPLASH 2019*, pages 152–166, Athens, 10 2019. Association for Computing Machinery, Inc.

8. The GraphQL Foundation. GraphQL, 2018.
9. GraphQL Foundation. GraphQL Foundation, 2019.
10. Alexander Maedche, Alan Hevner Eds, May June, and David Hutchison. Designing the Digital Transformation. In *International Conference on Design Science Research in Information System and Technology*, volume 10243, pages 231–246, Kristiansand, 2017. Springer Link.
11. Alan Hevner and Samir Chatterjee. Design Science Research in Information Systems. In *Integrated Series in Information Systems*, chapter 2, pages 9–22. Springer, Boston, 39 edition, 2010.
12. Barbara Kitchenham and Pearl Brereton. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, 55(12):2049–2075, 2013.
13. Pavel Seda, Pavel Masek, Jindriska Sedova, Milos Seda, Jan Krejci, and Jiri Hosek. Efficient Architecture Design for Software as a Service in Cloud Environments. In *10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, ICUMT 2018*, pages 317–322, Moscow, 11 2018. IEEE Computer Society.
14. Uri Klein and Kedar S. Namjoshi. Formalization and automated verification of RESTful behavior. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6806 LNCS:541–556, 2011.
15. Lee Byron. GraphQL: A data query language - Facebook Code, 2015.
16. Mike Bryant. GraphQL for archival metadata: An overview of the EHRI GraphQL API. In *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, volume 2018-Janua, pages 2225–2230, Boston, 12 2017. Institute of Electrical and Electronics Engineers Inc.
17. Ken Schwaber and Jeff Sutherland. *The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org, 2020.
18. Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18, 2015.
19. Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008*, page 11, Bari, 2008. BCS Learning and Development Ltd.
20. José A. Galindo, David Benavides, Pablo Trinidad, Antonio Manuel Gutiérrez-Fernández, and Antonio Ruiz-Cortés. Automated analysis of feature models: Quo vadis? *Computing*, 101(5):387–433, 2019.
21. ISO/IEC. *NTE INEN-ISO/IEC 25000 ISO/IEC 25000:2014 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*. International Organization for Standardization, 2 edition, 2014.
22. Neo4j. Cypher Query Language - Developer Guides, 2020.
23. GRANDstack. neo4j-graphql.js User Guide — GRANDstack, 2021.
24. ISO/IEC. *NTE INEN-ISO/IEC 25010*. International Organization for Standardization, 1 edition, 2015.
25. ISO/IEC. *Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use*, volume 1. International Organization for Standardization, Geneva,, 1 edition, 2016.
26. John Brooke. SUS: A Retrospective. *Journal of Usability Studies*, 8(2):29–40, 2013.

27. Mhairi McNeill. About RStudio - RStudio, 2020.
28. Klaas Sijtsma. On the use, the misuse, and the very limited usefulness of cronbach's alpha. *Psychometrika*, 74(1):107–120, 2009.
29. Juan Mendoza. RPubs - Alfa de Cronbach - Psicometría con R, 2018.
30. István Tóth-Király, Gábor Orosz, Edina Dombi, Balázs Jagodics, Dávid Farkas, and Camille Amoura. Cross-cultural comparative examination of the Academic Motivation Scale using exploratory structural equation modeling. *Personality and Individual Differences*, 106:130–135, 2017.
31. Antti Knutas, Arash Hajikhani, Juho Salminen, Jouni Ikonen, and Jari Porras. Cloud-based bibliometric analysis service for systematic mapping studies. *ACM International Conference Proceeding Series*, 1008:184–191, 2015.
32. Christian Kohl, Emma J. McIntosh, Stefan Unger, Neal R. Haddaway, Steffen Kecke, Joachim Schiemann, and Ralf Wilhelm. Online tools supporting the conduct and reporting of systematic reviews and systematic maps: A case study on CADIMA and review of existing tools. *Environmental Evidence*, 7(1):1–17, 2018.