Reengineering legacy document information systems: Challenges and solutions

Delfina Ramos-Vidal
delfina.ramos@udc.es
University of A Coruña, CITIC Research Centre, Database Laboratory
A Coruña, Spain

ABSTRACT

Since internet applications have reached a satisfactory level of maturity, large information systems have been developed to manage and facilitate access to documents. Simultaneously, there was an enormous international effort to digitise documents, enabling access via the internet. This endeavour facilitated the access of researchers to extensive document repositories and libraries, while also aiding companies in organising their documents. Two decades later, these vast databases are reasonably clean and well-organised, although the software used to manage and feed them is gradually becoming obsolete. Therefore, it is imperative to continuously reengineer the software to maintain optimal functionality. Furthermore, after the initial effort to digitise documents and create the initial metadata, it is reasonable to augment the metadata information pertaining to the documents. As such, two necessities are apparent: improving support for reengineering legacy document information systems and enabling data model updates and schema evolution to accommodate new information. Our goal is to automate the reengineering process as a whole.

CCS CONCEPTS

• **Software and its engineering → Reusability**; • **Information systems → Extraction, transformation and loading**;

KEYWORDS

document information systems, software reengineering, schema evolution, automated development

ACM Reference Format:
Delfina Ramos-Vidal. 2023. Reengineering legacy document information systems: Challenges and solutions. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE '23), June 14–16, 2023, Oulu, Finland.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3593434.3593436

Advisor: Nieves R. Brisaboa, University of A Coruña, Centre for Information and Communications Technology Research CITIC, Database Laboratory.
E-mail: nieves.brisaboa@udc.es

EASE '23, June 14–16, 2023, Oulu, Finland
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE '23), June 14–16, 2023, Oulu, Finland*, https://doi.org/10.1145/3593434.3593436.

1 INTRODUCTION

The predominance of technology in our culture has significantly altered the way we deal with information, making it more accessible, reproducible, and massive. As our technology needs continue to evolve, so do the tools and software we employ to satisfy them. Nevertheless, the rapid rate of technological advancement has left many organisations struggling to keep pace with these changes. Many organisations rely on legacy software systems developed years or even decades ago, and as time passes, these systems may become outdated and inadequate for current business needs.

The incessant evolution of technology has engendered a significant challenge for organisations: the reengineering of legacy software. This process entails the updating and modernising of existing software systems to align with modern corporate requirements and the rapid pace of technological innovation. This challenge is especially pronounced in the context of legacy document information systems and their attendant databases, which frequently constitute the core of an organisation's operations. As a result, the necessity of enhancing such systems has become a top priority for organisations across a wide range of sectors.

We understand a Document Information System (DIS) to be an information system that is massive in the storage of digital documents, whether textual or multimedia, with a vast volume of metadata, and occasionally even the digitised documents themselves, which may have been acquired in the context of corporations or libraries. Our research group has extensive experience developing Document Information Systems, like digital libraries. Most of the platforms were developed two decades ago, such as the [1], and are in need of on update. When it comes to upgrading a legacy document information system, two critical challenges arise: developing a new platform and migrating the existing data from the legacy system to the new system.

Developing a DIS from scratch may be quite expensive in terms of both time and financial investment. However, the reengineering process introduces the difficulty of working with pre-existing software that ideally would be utilised to cut costs and the transfer from the old system to the new platform, which should be seamless to avoid service interruptions to users. Most document information systems share a core set of features. In order to study the variability of DIS, we focused on the domain of digital libraries, analysing current platforms such as the Galician Virtual Library[2] or the Miguel de Cervantes Virtual Library [3]. Most of the digital libraries we have researched include features like digital resource management and visualisation, the integration of a search engine

[1]Publishing in Galicia during the Franco Era: https://ediciongalizafranquista.udc.gal
[2]Galician Virtual Library: https://bvg.udc.es/
[3]Miguel de Cervantes Virtual Library: https://www.cervantesvirtual.com/

capable of conducting complicated queries over massive datasets, and metadata interchange utilising standards such as Dublin Core or MARC. Because there is a set of common features, it is feasible to automate software development using variability management approaches such as Software Product Line Engineering.

The second critical aspect of upgrading legacy document information systems is data migration, which involves moving data from the database of a legacy system to the database of the modern system. This can be a complex and challenging task, requiring careful planning and execution to ensure that data is accurately transferred and that there is no loss of information during the migration process. Commercial applications have risen to the top of the market in recent years, setting the standard for extract, transform, and load (ETL) operations. Most solutions are intended for the general public and support generic transformations, but the scope of the operations they provide is limited, and in most cases, the user is not allowed to supervise the decisions made by the tool, making it necessary to perform a final check-up of the migration to refine any mistakes made during the process.

In this paper, we will explore the challenges and opportunities of upgrading legacy document information systems and their databases. We will discuss the feasibility of automating the development of document information systems and the importance of data migration and the best practices that can be employed to ensure a successful migration. The paper is structured as follows: Section 2 describes the motivation of this thesis in detail; Section 3 defines the research questions that will be addressed during the paper; Section 4 describes the work plan to be followed throughout the thesis; and Section 5 provides an overview of the previous work related to the topic.

2 MOTIVATION

After conducting extensive research and development of software solutions, both for academic and commercial purposes, it has been observed that many companies often develop applications with similar characteristics. This hypothesis has been tested by a research group in two distinct domains, namely Document Information Systems and Geographic Information Systems. Over the years, the group has collaborated with partners to create new software solutions, some of which have been operational for more than two decades. However, the need to upgrade these previous developments by evolving their technologies, updating requirements, and refining several features has become apparent. In addressing these evolution projects, it has become evident that the reengineering process when migrating from a legacy system needs optimization. The reengineering process is confronted with two main challenges.

There is a need for efficient cost-reduction approaches to modernise technology and refactor implementation with the minimum impact on the workflow of present platforms. Previous research by Cortiñas et al. [4] indicates that using variability management techniques, particularly Software Product Line Engineering (SPLE), while building new software can dramatically lower costs and time to market. A research group with extensive expertise in designing and deploying SPLE in real-world use cases has created two primary SPLs, one for geospatial information systems (GIS) [5] and the other for digital libraries [11, 13]. Recent research [14] has demonstrated that employing an SPL as the foundation for the reengineering process, rather than only as a tool for generating new software, is feasible. This approach has yielded positive outcomes for the researchers.

The challenge with reengineering lies in the need to adapt the entire product to new technology, redesign the data model, and ensure that it still meets the demands of the domain. In addition, it involves migrating all previous data into the new database. This process is further complicated by the presence of legacy products with their respective models and updated products with updated models. During the migration of original data from legacy products into the updated database, there are frequent instances of incompatibilities and inconsistencies that arise. The research group has practical experience working hand-in-hand with humanities research groups, developing several document information systems and digital libraries for them. During the years of partnership, it has become apparent that experts in the humanities require a fine degree of detail in their data and, in most cases, they must be included in the data migration process to make decisions over the data, although they do not possess the technical knowledge needed to do it independently using commercial ETL tools, which are designed with developers in mind. As a last resource, developers export datasets into Excel sheets, which are known and easy to use for experts in the humanities, and then import the refined data into the database, but this process is slow and impractical. In addition, in most cases, document information systems use relational databases, making it possible to define general transformation rules for relational data.

The purpose of this thesis is to design and implement a framework capable of aiding the reengineering process, with a special focus on these two stages. First of all, aiding the software generation by implementing an automated development platform for document information systems, and then integrating a tool capable of generalising relational transformation rules in such a way that they can be shown in a user-friendly way, with graphical interfaces that allow making decisions at any step of the data migration process.

3 RESEARCH QUESTIONS

The following three key research issues will be addressed in this study:

Research Question 1: How can we create a tool based on variability for the automatic generation of document information systems? The primary focus of our research centres on the extraction of common and variable features from a corpus of document information systems, to employ them as foundational components of our solution's variability model. Following the delineation of potential variability in these systems, we intend to undertake a thorough analysis of the most effective methodologies for implementing a tool that can harness this variability to its fullest extent.

Research Question 2: How can we automate the data migration process based on the legacy data model, the updated data model to be generated, and even data models employed to introduce unprecedented data? The data migration process entails several challenges of its own. Foremost among these is the need to integrate data from heterogeneous data sources, such as relational databases or document files with little to no structure, and examine numerous data migration cases in order to effectively identify and generalise common data migration issues. This generalisation serves as the foundation for the formulation of data transformation rules, which are subsequently employed to automate the data migration process. In addition to this, it is necessary to study several automated development and low-code approaches to determine the optimal solution for addressing data evolution concerns. By doing so, it is possible to identify the most efficient and effective means of managing the challenges that arise during the data migration process.

Research Question 3: How to abstract the current navigation model so that it serves as the basis for the new application? Whilst the primary objective of a reengineering process is typically to modernise existing technology, update data models, or incorporate new functionalities to meet current demands, certain features may remain unaltered throughout this process, such as the navigational workflow of the application. Hence, it would be highly advantageous to automatically capture the navigational workflow of the legacy system in an abstract representation, which can be utilised as input by the automated development tool. The resultant navigation model derived from the legacy system can subsequently serve as a reference point for the variability of the front-end of the reengineered system.

4 WORK PLAN

The work plan established is described in the next subsections, related to the research approach, threats to validity, and the research summary, which covers both the current state of the research and the next steps.

4.1 Research approach

The present doctoral thesis addresses two distinct challenges, as outlined at the outset of this paper. Specifically, it outlines two distinct research lines about automated development and automated data migration, both of which converge in addressing the overarching challenge of automated reengineering. The doctoral thesis will employ an approach based on the classical software development methodology, which adopts a sequential approach to software development. This methodology typically encompasses distinct phases, including requirements gathering, design, implementation, testing, and maintenance, which are executed linearly. An overview of the intended phases for this research can be seen in Figure 1

The first phase of the research approach involves gathering requirements, where the researcher will engage with experts in the domain of document information systems and the humanities to understand their needs and expectations. This phase involves conducting interviews, surveys, and focus groups to collect and analyse data to gain insights into the software requirements that must be included in the feature model that will, later on, serve as the base for our automated development platform. A feature model is a modelling technique used in software product line engineering (SPLE) to represent the commonalities and variabilities of a family of related software products. In addition, during the first phase, the research will perform the manual data migration process of real use cases with the objective of extracting recurrent data transformations and common issues from this process. Then, a detailed analysis of migration problems that may arise must be carried out to design an automated development tool in which you can indicate

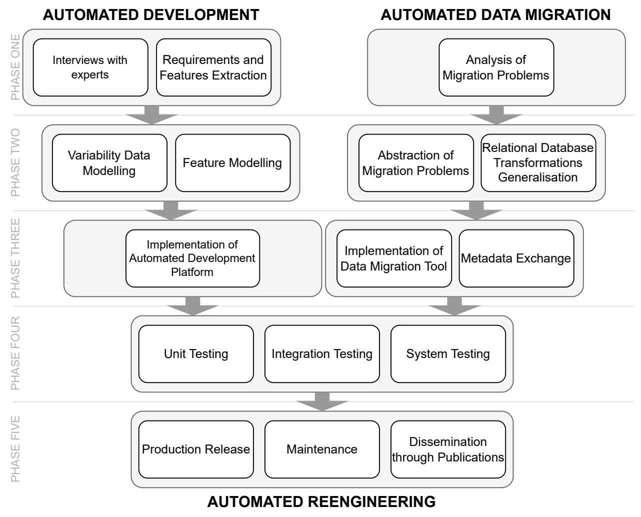
AUTOMATED DEVELOPMENT
AUTOMATED DATA MIGRATION
PHASE ONE
Interviews with experts
Requirements and Features Extraction
Analysis of Migration Problems
PHASE TWO
Variability Data Modelling
Feature Modelling
Abstraction of Migration Problems
Relational Database Transformations Generalisation
PHASE THREE
Implementation of Automated Development Platform
Implementation of Data Migration Tool
Metadata Exchange
PHASE FOUR
Unit Testing
Integration Testing
System Testing
PHASE FIVE
Production Release
Maintenance
Dissemination through Publications
AUTOMATED REENGINEERING

Figure 1: Overview of the intended research phases

the characteristics that can be given and that must be considered to generate the necessary import services.

In the second phase, the design phase, the researcher will create detailed design specifications based on the requirements gathered in the previous phase. This phase involves creating diagrams that represent the variable data model of a Document Information System, and other design artefacts that provide a comprehensive overview of the software system, such as the Feature Model describing the variability covered by our solution. This phase will also cover the abstraction of the migration problems observed during the first phase and the theoretical definition of general transformations for relational databases.

The third phase aligns with the implementation phase, which involves developing solutions based on the design specifications, testing, and debugging said solutions to ensure that they meet the design specifications and requirements. During this phase, two main artefacts must be developed: on one hand, the automated development platform that automates the generation of document information systems, and, on the other hand, the data migration assistance tool that will guide the migration process. This tool will display the tables of the databases as listing, although the input can be heterogeneous, either a database connection or an Excel file, so that we can implement quality management systems that prevent the user from changing persistent data without being aware of it. Ideally, both systems will establish a metadata exchange so that the migration tool is conscious of the data model generated by the automated development platform in such a way that it can implement restrictions based on compliance with the resulting schema, avoiding inconsistencies between the database schemas generated by both systems.

Phase four will focus on testing the proposed solution, which involves evaluating the software system to ensure that it works as intended and meets the requirements by means of different types of testing such as unit testing, integration testing, and system testing to ensure that the software functions correctly. The main goal of this phase is to make sure that the system is capable of generating a contemporary document information system that covers all the requirements gathered by the feature model by validating that we can generate the real use cases that were used during the first phase for the requirement extraction. During this phase, the data migration tool should be tested by the initial humanities experts to make sure it covers the original requirements and the interface is intuitive enough for users with little to no expertise in software development.

Finally, the fifth phase will cover two different aspects of the thesis. First of all, during this phase, the final solutions developed should be included in the actual development process of new projects arising while providing ongoing support and updates to the software system to ensure that the software remains up-to-date with changing user needs and technological advancements. The second goal of this phase is to disseminate research findings and promote our solution by means of scientific publications.

Although the classical engineering methodology is sequential and linear, we are aware that unforeseen circumstances and exigencies may require adaptations to the planned phases of the research process to facilitate the timely convergence of the intended outcomes. Notably, it is noteworthy to mention that dissemination through publication, though traditionally reserved for the final phase, may not be limited to this stage alone. The dissemination of preliminary results may also be warranted, and such publications could provide valuable insights into the research process while serving as precursors to the ultimate research output.

4.2 Validity threats

The current section aims to examine the potential threats to construct and external validity, as categorized by Wohlin et al. The four categories proposed by Wohlin et al. [16] are well-established in empirical research and widely utilised by researchers to identify and mitigate potential threats to the validity of their studies.

External Validity: The objective of this research is to test the outcome against several known test projects that show variability management and software product lines are feasible for the development of document information systems since we have previously worked with these platforms and have a deep knowledge of their behaviour. However, the results could differ when generating different Document Information Systems that are unknown to us. Second, in order to properly compare the performance of the automated development platform to that of custom development, it would be necessary to give the specifications of the products to several teams with similar levels of experience, ensuring that the generated ones are thoroughly adapted to achieve a refinement level comparable to that of custom-made. Nonetheless, because of the cost, our experiment does not include said stage, but we do attempt to create each product and compare their sizes. Even if it is not as accurate as performing the actual experiment, it gives relevant information when estimating that the effort for each KLOC (thousands of lines of code) written by the SPL and custom-made is more or less uniform.

Construct validity: Our solution aims at automating the process of re-engineering legacy software but it can be difficult to measure the improvement in terms of time dedicated to the re-engineering or the effort put in the process. To compare the effort dedicated to the migration of the database, we could have a single developer program the migration script manually while the solution performs the migration, and record the time dedicated to said task. However, when it comes to the re-engineering of the application, it is not feasible to have a development team building a new platform from scratch due to the economic costs, making it difficult to properly compare both approaches. In this case, the metric based on Lines of Code (LOC) is applied, measuring the LOC produced by the solution versus the LOC manually coded beforehand.

Conclusion Validity: When it comes to measuring the performance of our approach, it may be difficult to find reliable measures to assess the improvement provided by our solution. Lines of Code (LOC) or Kilo Lines of Code (KLOC) are common metrics to measure the size of software programmes and are the most used in cost estimation. However, these metrics depend on the syntax of the programming language at stake, which makes a metric such as LOC reliable when measuring a language like Java, with a standard structure, but can be misleading when measuring an artefact programmed in a Domain Specific Language, whose syntax is defined by the researcher. Other metrics, such as Function Points, could be taken into consideration.

In this study, internal validity is not considered since there is no causal relationship.

4.3 Research summary

Preceding research has been undertaken in the domain of this doctoral thesis, giving us the opportunity to present preliminary findings and a succinct overview of the current state of research.

In the first instance, an initial version of the automated development platform was developed, LPS-BIDI. This platform was developed by applying Software Product Line Engineering to manage variability, following the steps of the SPLE methodology presented by Cortiñas et al. [4], and employing the spl-js-engine[4] [3] as the derivation engine to build the SPL. However, this first approach was focused solely on the domain of digital libraries and fell short of fulfilling the requirements of a comprehensive Document Information System. Despite these limitations, LPS-BIDI yielded promising results in terms of development effort after undergoing testing in three distinct real use cases, generating three unique products. Table 1 shows the characteristics of each generated product in terms of the size of the resulting generated code, in terms of the number of files generated, the number of classes generated, the number of entities generated, and the number of lines of code (LOC). This experiment demonstrated that after producing three projects, we were able to generate 27,666 LOC, whereas we only had to manually implement the 13,583 LOC that compose the initial application. This supports the assertion by Pohl et al. [12] that the advantages of adopting the SPL approach over single-system development become apparent following the third software system development. Quality and maintainability are two additional aspects of the created code that warrant further investigation.

[4]spl-js-engine: https://github.com/AlexCortinas/spl-js-engine

Table 1: Characteristics of the generated products

Project	Files	Classes	Lines of code (LOC)		
			Total	Back-end	Front-end
Project 1	186	130	9,190	6,436	2,754
Project 2	161	110	8,396	5,641	2,755
Project 3	196	127	10,080	6,608	3,472

In addressing the data migration challenge, various approaches are currently being explored. Our initial strategy involves the creation of a domain-specific language (DSL) that enables the specification of the data transformation and migration processes between two software products' data models. The DSL utilised for database transformation is declarative in nature and comprises statements that facilitate the expression of a mapping from an existing data model to a target data model, without requiring the inclusion of implementation details or the corresponding control flow. To generate the parser, we employ ANTLR (ANother Tool for Language Recognition), and the syntax of our DSL has been designed to comply with SQL standards, given that our primary goal is to transfer data between relational databases.

Table 2: Lines of code of the migration services

Migration Script	Lines Of Code (LOC)		Character Count	
	Handwritten	DSL	Handwritten	DSL
Project 1	287	126	14 488	7 794
Project 2	240	73	9 542	3 088

Table 2 endeavours to compare the concision and succinctness inherent in manually crafting the migration service versus specifying the migration requirements to the DSL for the purpose of automated migration service generation. While this table does provide a basic understanding of the number of Lines of Code (LOC) associated with each specification, it is not entirely reliable, given the variance in structure arising from a shift from a pure SQL script to the syntax of our DSL.

At the moment, we are focusing on scientific dissemination and preparing a journal paper that details the DSL implementation for data migration. In the near future, our research will focus on conducting interviews with experts in the humanities domain to gain a deeper understanding of real-world use cases for data migration. Additionally, we plan to perform a schema evolution case study on a project currently being developed within our research group, manually executing the data migration process in a traditional manner to identify and extract valuable data migration issues that can be generalised and analysed. As for the automated development aspect of our research, we intend to extract requirements and features from a corpus of document information systems, which will serve as the basis for the feature model in the automated development tool.

5 RELATED WORK.

5.1 Software automatising and variability management

Software development automation and variability management are two important research areas in software engineering that aim to improve the efficiency and quality of software development. In recent years, there has been a significant increase in interest and research in both areas. Software development automation is the process of automating various aspects of the software development life cycle to increase efficiency, reduce costs, and improve quality. Variability management is the process of managing the diversity of software products that share a common base. It allows for the efficient development of software products that can be customised to meet specific requirements while minimising the cost and effort of software development.

Several scholarly works have contributed to the development of various techniques and tools for SPL-based reengineering of legacy systems. Assunção et al. [1] conducted a systematic mapping on the topic and concluded that different strategies based on existing methods present in Software Engineering are used to support the reengineering process, with statistical analysis, expert-driven, and information retrieval being the most commonly employed. Legacy systems, according to Laguna and Crespo [10], may be reconfigured as product lines through the use of reengineering techniques, while existing software product lines can also be refactored to increase their quality, making them more maintainable.

An study regarding software reuse carried out by Krueger [9] identified three alternative techniques for introducing SPLE in firms, one of which, the extractive approach, proposed the use of existing proprietary software systems by extracting common and variable artefacts and migrating them to an SPL. Xue [17] proposed an automated method for reengineering a family of legacy information systems into an SPL, leveraging an analysis of the variability of the legacy products. Similarly, Fleurey et al. [7] have applied Model-Driven Engineering (MDE) methodologies to software modernisation processes such as reverse engineering, translation, and code generation. In a more industry-focused approach, Breivold et al. [2] have introduced a five-step technique for migrating industrial systems to Software Product Lines. This approach includes a novel design proposal and a transition plan. The works in SPL introduction and evolution processes provide well-established frameworks to guide reengineering works.

In conclusion, software product lines offer a powerful approach to variability management and automation in the context of legacy system reengineering. While there are still challenges to be addressed in terms of scalability and tool support, the existing works demonstrate the potential benefits of using SPLs for modernising and evolving legacy systems.

5.2 Schema evolution and data migration

Schema evolution and data migration are two critical processes in software engineering that are essential for the smooth functioning of an organisation's information systems. Over the years, numerous studies have been conducted to investigate these processes, uncover their challenges, and propose potential solutions. The management of schema evolution involves two primary tasks: the discovery or extraction of structural changes to data, and the handling of such changes from the point of view of application development, including data migration.

The study of database migration has been the focus of much academic research, with a variety of methodologies being presented and investigated, ranging from fundamental techniques such as legacy migration and reverse engineering to more complex approaches. Hudicka et al. (1998) proposed a comprehensive approach for the database migration procedure that is divided into seven distinct stages. In a related vein, Elamparithi et al. (2015) conducted a review of existing migration strategies in the area of Relational Database Migration (RDBM) and concluded that existing techniques were inadequate for managing more than one target database or for effectively handling schema and data conversion. Delplanque et al. [6] conducted a study that involved the recording and analysis of the actions of a database architect during a complex evolution of a relational database. The researchers subsequently extracted and generalized the challenges encountered during the migration process and explored possible solutions to these issues. The study identified six main problems that could be addressed through the application of techniques developed by the software engineering community.

In the area of NoSQL databases, Hillenbrand et al. [8] present a methodology of self-adapting data migration, which automatically adjusts migration strategies and their parameters with respect to the migration scenario and service-level agreements, thereby contributing to the self-management of database systems and supporting agile development. Meanwhile, Störl and Klettke [15] have implemented Darwin, a system that supports the whole schema evolution management and data migration lifecycle for different types of NoSQL databases.

ACKNOWLEDGMENTS

CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Department of Culture, Education, Vocational Training and Universities and the Galician universities for the reinforcement of the research centers of the Galician University System (CIGUS); PID2021-122554OB-C33 (OASSIS): partially funded by MCIN/AEI/10.13039/501100011033 and EU/ERDF A way of making Europe; TED2021-129245B-C21 (PLAGEMIS): partially funded by MCIN/AEI/10.13039/501100011033 and "NextGenerationEU"/PRTR; PID2020-114635RB-I00 (EXTRACompact): partially funded by MCIN/ AEI/10.13039/501100011033; GRC: ED431C 2021/53, partially funded by GAIN/Xunta de Galicia; PRE2021-099351, partially funded by MCIN/AEI/10.13039/501100011033 and "FSE+"Fondo Social Europeo Plus".

REFERENCES

[1] Wesley K. G. Assunção, Roberto E. Lopez-Herrejon, Lukas Linsbauer, Silvia R. Vergilio, and Alexander Egyed. 2017. Reengineering legacy applications into software product lines: a systematic mapping. *Empir. Softw. Eng.* 22, 6 (2017), 2972–3016. https://doi.org/10.1007/s10664-017-9499-z

[2] Hongyu Pei Breivold, Stig Larsson, and Rikard Land. 2008. Migrating Industrial Systems towards Software Product Lines: Experiences and Observations through Case Studies. In *34th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2008*. IEEE Computer Society, Parma, Italy, 232–239. https://doi.org/10.1109/SEAA.2008.13

[3] Alejandro Cortiñas, Miguel R. Luaces, and Óscar Pedreira. 2022. Spl-Js-Engine: A JavaScript Tool to Implement Software Product Lines. In *Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume B* (Graz, Austria) *(SPLC '22)*. Association for Computing Machinery, New York, NY, USA, 66–69. https://doi.org/10.1145/3503229.3547035

[4] Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, and Ángeles Saavedra Places. 2017. Scaffolding and in-browser generation of web-based GIS applications in a SPL tool. In *Proceedings of the 21st International Systems and Software Product Line Conference, SPLC 2017, Volume B*, Maurice H. ter Beek, Walter Cazzola, Oscar Díaz, Marcello La Rosa, Roberto E. Lopez-Herrejon, Thomas Thüm, Javier Troya, Antonio Ruiz Cortés, and David Benavides (Eds.). ACM, Sevilla, Spain, 46–49. https://doi.org/10.1145/3109729.3109759

[5] Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, Ángeles Saavedra Places, and Jennifer Pérez. 2017. Web-based Geographic Information Systems SPLE: Domain Analysis and Experience Report. In *Proceedings of the 21st International Systems and Software Product Line Conference, SPLC 2017, Volume A*, Myra B. Cohen, Mathieu Acher, Lidia Fuentes, Daniel Schall, Jan Bosch, Rafael Capilla, Ebrahim Bagheri, Yingfei Xiong, Javier Troya, Antonio Ruiz Cortés, and David Benavides (Eds.). ACM, Sevilla, Spain, 190–194. https://doi.org/10.1145/3106195.3106222

[6] Julien Delplanque, Anne Etien, Nicolas Anquetil, and Olivier Auverlot. 2018. Relational Database Schema Evolution: An Industrial Case Study. In *2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018, Madrid, Spain, September 23-29, 2018*. IEEE Computer Society, 635–644. https://doi.org/10.1109/ICSME.2018.00073

[7] Franck Fleurey, Erwan Breton, Benoit Baudry, Alain Nicolas, and Jean-Marc Jézéquel. 2007. Model-Driven Engineering for Software Migration in a Large Industrial Context. In *Model Driven Engineering Languages and Systems, 10th International Conference, MoDELS 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4735)*, Gregor Engels, Bill Opdyke, Douglas C. Schmidt, and Frank Weil (Eds.). Springer, Nashville, USA, 482–497. https://doi.org/10.1007/978-3-540-75209-7_33

[8] Andrea Hillenbrand, Uta Störl, Shamil Nabiyev, and Meike Klettke. 2022. Self-adapting data migration in the context of schema evolution in NoSQL databases. *Distributed Parallel Databases* 40, 1 (2022), 5–25. https://doi.org/10.1007/s10619-021-07334-1

[9] Charles W. Krueger. 1992. Software Reuse. *ACM Comput. Surv.* 24, 2 (jun 1992), 131–183. https://doi.org/10.1145/130844.130856

[10] Miguel A. Laguna and Yania Crespo. 2013. A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. *Science of Computer Programming* 78, 8 (2013), 1010–1034. https://doi.org/10.1016/j.scico.2012.05.003 Special section on software evolution, adaptability, and maintenance. Special section on the Brazilian Symposium on Programming Languages.

[11] Oscar Pedreira, Delfina Ramos-Vidal, Alejandro Cortiñas, Miguel Rodríguez Luaces, and Ángeles Saavedra Places. 2021. Development of Digital Libraries with Software Product Line Engineering. *Journal of Web Engineering* 20, 7 (2021), 2017–2058. https://doi.org/10.13052/jwe1540-9589.2072

[12] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, Berlin, Heidelberg.

[13] Delfina Ramos-Vidal, Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, and Ángeles Saavedra Places. 2020. A Software Product Line for Digital Libraries. In *Proceedings of the 16th International Conference on Web Information Systems and Technologies, WEBIST 2020*, Massimo Marchiori, Francisco José Domínguez Mayo, and Joaquim Filipe (Eds.). SCITEPRESS, Budapest, Hungary, 381–389. https://doi.org/10.5220/0010214903810389

[14] Delfina Ramos-Vidal, Alejandro Cortiñas, Miguel Rodríguez Luaces, Oscar Pedreira, and Ángeles S. Places. 2021. Re-ingeniería y Modernización de la Biblioteca Virtual Galega. In *JISBD2021*. SISTEDES. http://hdl.handle.net/11705/JISBD/2021/061

[15] Uta Störl and Meike Klettke. 2022. Darwin: A Data Platform for Schema Evolution Management and Data Migration. In *Proceedings of the Workshops of the EDBT/ICDT 2022 Joint Conference, Edinburgh, UK, March 29, 2022 (CEUR Workshop Proceedings, Vol. 3135)*, Maya Ramanath and Themis Palpanas (Eds.). CEUR-WS.org. http://ceur-ws.org/Vol-3135/dataplat_short3.pdf

[16] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.

[17] Yinxing Xue. 2011. Reengineering Legacy Software Products into Software Product Line Based on Automatic Variability Analysis. In *Proceedings of the 33rd International Conference on Software Engineering* (Waikiki, Honolulu, HI, USA) *(ICSE '11)*. Association for Computing Machinery, New York, NY, USA, 1114–1117. https://doi.org/10.1145/1985793.1986009