4GL Code Generation: A Systematic Review

Abdullah A H Alzahrani[1]
Department of Computer Sciences, Computing College -Alqunfuda
Umm Al Qura University, Makkah, Saudi Arabia

***Abstract*—Code generation is longstanding goal in software engineering. It allows more productivity of computer programming as it aims to provide automation of transformation of models into actual source code. This process has been covered adequately in many programming languages. However, this topic has not been covered sufficiently with regards to Fourth Generation Languages (4GL) which have a high specialized nature. The goal of this paper is to represent a systematic literature review of 4GL Code generation. The paper focuses on reviewing systemically the studies published in the past 20 years on the topic. This is to investigate the trends in the topic and the approaches introduced in order to identify potential new research lines.**

Keywords*—Software engineering; code transformation; 4GL; code generation; Model Driven Development (MDD); Extraction Transform Load (ETL); Model Driven Engineering (MDE); Rapid Application Development (RAD)

I. Introduction

Fourth-generation languages (4GL) are a class of high-level programming languages. The idea of 4GL is to make programming as simple as possible by the mean of programming in natural human language [2], [3]. 4GLs aim to make programming easier, more efficient and more effective for users with less programming skills [4].

However, most of 4GLs are proprietary or designed and developed for a very specific scope and purpose. This has led to a hinder in progressing research and development in the area of 4GL. These difficulties are derived from highly-specialized nature of 4GL. In addition, the limited target users pose accessibility issues. Furthermore, insufficient tool support causes a problem as 4GL needs in-house development of tools support. Consequently, this introduces high costs for 4GL evolvements in all areas of research [5]. The followings summarize the reasons behind the recessions in progressing research and development: proprietary developments, complexity, modularity, and integrations with other interfaces (web browsers) issues [5], [6].

A number of 4GL exist and are used in many areas from research and industry, for example: Query languages SQL, Oracle, Report Generator, Magic, Informix, Advanced Business Application Programming (ABAP), MathWorks, MATLAB, SPSS, etc. [5]. Some of these languages are popular as they are open to use and some are not as they are proprietary.

Code generation is longstanding goal in software engineering [1]. However, it is not a straightforward process nor an easy to achieve optimal solution for it [1]. Code generation is the process of transforming a model of high-level representation to a source code that can be read and understood by a computer. Usually, this is done via a use of Computer-Assisted Software Engineering (CASE) tools [5], [7], [8]. A CASE tool can generate initial software or database code directly from system models. Examples of 4GL CASE tools are: Oracle2Java, Evo, Jheadstart, Pitss, Ormit [9]. Code generation is an actual practice of forward engineering.

Several problems are often attached to the CASE code generations. These issues are related mainly the complexity of the models or the capability of the target language. In addition, in the case that models gained by reverse engineering legacy systems, code generations CASE tools are often regarded as less useful especially if they are independent from the reverse engineering CASE tools.

In general, the existing approaches and tools offered for 4GL code generation suffer from several limitations. First, the majority focuses on auto-transformation of Oracle forms, whereas many 4GL languages are not considered. Second, majority of these approaches are semi-automated approaches which often need experts to be involved. Finally, immaturity is a nature for these approaches as only proof-of-concept models and tools are presented. However, mature approaches might be developed but for proprietary purposed and not shared for researchers.

The systematic review is a process of addressing a research question then finding and evaluating all available research done with relation to it [10]. It helps highlighting the major work conducted in the area of the research question. From such a review, research gaps can be found in a way that assure a satisfactory coverage of area of research.

A number of researchers [11]–[13] have considered reviewing the topic of transformation between 4G languages from different points of view. In addition, they have concluded that it is non-trivial process and needs a considerable manual effort and knowledge from all of the people involved. However, sufficiency when authors are reviewing related work of others is still an issue. In addition, up to the date of writing this paper, there is no systematic review on the topic of 4GL code generation. This leads to the importance of investigating objectively the current state of arts in the topic of 4GL code generation. This paper aims to provide an objective and systematic review of the topic of 4GL code generation. Introduction of such a review allows evolving the research in this area and highlighting the current research gaps that are not resolved.

This paper has been structured as follows. Section 1 introduced the topic of 4GL and code generation. In addition, it highlights the importance of systematic review on the

considered topic. Section 2 illustrates the methodology used in this paper and formulate the research question. Section 3 discusses the trends in the area of the 4GL code generations. Section 4 categories the main works conducted in the 4GL code generations and discussed the main findings. Finally, Section 5 concludes the review with the identification of trends and new research areas.

II. Methodology

This paper employs a systematic review methodology demonstrated in [10], [14]. The methodology allows a structural and objective review of a topic under consideration. In addition, it allows providing a broad view on works which are primary and related to the topic under consideration.

A. Research Question

What are the initiatives undertaken in relation to 4GL code generations in the last 20 years? This research question can be answered by identifying the approaches, models, and CASE tools introduced, Thereafter, highlighting the major difficulties and issues faced in order to achieve the goal of code generations. Therefore, keywords leading the search have been listed. These keywords are: code generation, fourth generation languages, and 4GL. It is important to mention here that 4GL is often referred to fourth generation languages [5].

B. Sources Selection

Having the aforementioned keywords, a search string has been made from a combination of these keywords and been used in the search by the search engines of the selected digital libraries. Table I shows the formulated search string with an OR logical operator which is one of useful operators offered by the search engines in the digital libraries.

Four well known digital libraries were selected in order to perform the search for related studies in. These libraries are: Springer Link, IEEE Digital Library, ACM Digital Library, and ScienceDirect. These libraries offer variety ways of searching for journal articles, conference papers, books, and other publication types. In addition, these libraries offer the use of logical operators in the searching.

C. Inclusion and Exclusion Criteria

Inclusion criteria, for studies to be considered relevant, are the outcome of analyzing the title, research keywords, abstract, and the conclusion of a paper. In addition, as this research aims to investigate the work done towards 4GL code generation in the past 20 years, an exclusion criterion of year of publication has been applied to retrieve only the work published between the years 2000 and 2020. Another exclusion criterion was a non-English publication items which have been found in the retrieved list of items from the digital libraries.

As can be seen in Table II, in the first iteration of applying the search string in all digital libraries, the search resulted in a total of 1925 items. Repetition of items in some between libraries was noticed, so, it was inevitable to eliminate repetitions. After removing repetitions, the total was 1770 items. The total become 593 when applying exclusion criterion of publication before the year of 2000. Finally, 187 of the publication items were relevant to this research topic by inclusion criterion.

TABLE I. Search String

Search string	"4G languages" OR "4G languages code generation" OR "4GL code generation" OR "4GL" OR "fourth-generation-languages" OR "fourth-generation-languages code generation" OR "fourth generation languages code generation" OR "fourth generation languages"

TABLE II. Sources and Studies Found

Sources	Studies					
	Found	Not repeated	Since 2000	Relevant	Primary	%
Springer Link	793	694	216	64	6	21.43%
IEEE Digital Library	35	35	15	12	7	25.00%
ACM Digital Library	302	274	99	41	6	21.43%
ScienceDirect	795	767	263	70	9	32.14%
Total	1925	1770	593	187	28	100%

Primary studies were, as shown Table II, 28 studies. Complete list of primary studies is shown in (Appendix A). The primary studies were nominated from the relevant studies after in-depth reading and analysis of the entire list of the relevant studies. The primary studies are only the studies which mainly consider the 4GL code generation by introduction new approaches, model development, evaluation of current approaches, and branches of code generation with relation 4GL such as code transformation. It is important to mention that studies that are related to the 4GL code generation have been considered to be relevant studies. Examples of which are code quality, effort estimation, optimization, refactoring, and code maintenance.

III. Publication in 4GL Code Generation in the Past 20 Years

This section shows an analysis of primary and relevant studies in general and the main aspects and remarks identified during the reviewing process. Obviously, from the number of relevant studies, it can be concluded that the topic of 4GL code generation has not been considered sufficiently in the past 20 years. Fig. 1 illustrates publications trends of primaries and relevant studies of 4GL code generation over the past 20 years in the four digital libraries namely Springer, IEEE, ACM, and ScienceDirect.

From Fig. 1, it can be noticed that the years of 2003, 2005, 2011 were the years were most studies were published with a number of around 14 publications in all digital libraries investigated. Then, the years of 2008 and 2019 come in almost close number of the aforementioned. In addition, it can be obvious that the average of publications related to the topic of 4GL code generation is 9 studies each year over the past 20 years. Most of these primaries and relevant studies have been published in Springer and ScienceDirect. This highlights a lack of studies in this topic.

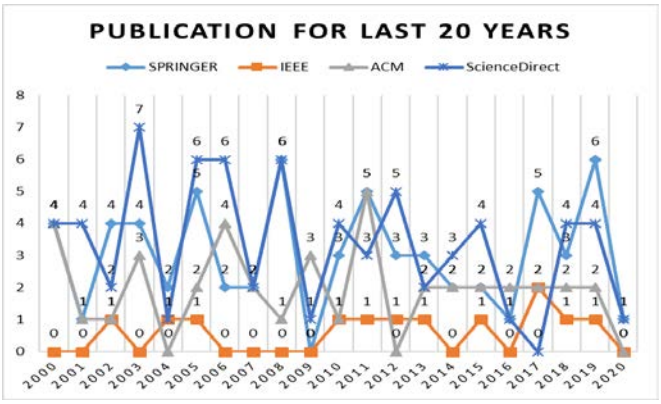

Fig. 1. Run graph of publications (Relevant Studies Published in the Past 20 Years based on Number of Publications).

Table III shows the categories of the publications based on the types of work. The main types are conference papers, journal articles, and books and technical reports. The majority of primaries and relevant studies are conference papers and journal articles with the total of 153 studies. In addition, as illustrated in Fig. 2, 59% of journal articles which are primaries and relevant studies published in ScienceDirect library. Furthermore, 74% of books and technical reports are published in same library. However, 45% of conference papers which are primaries and relevant studies published in Springer library with 0% published in ScienceDirect library of this category of the publications.

TABLE III. TYPES OF PUBLICATIONS OF PRIMARIES AND RELEVANT STUDIES

Type / Library	Conference Paper	Journal Article	Book and technical reports	Total
SPRINGER	35	20	9	64
IEEE	12	0	0	12
ACM	30	11	0	41
ScienceDirect	0	44	26	70
Total	77	75	35	187

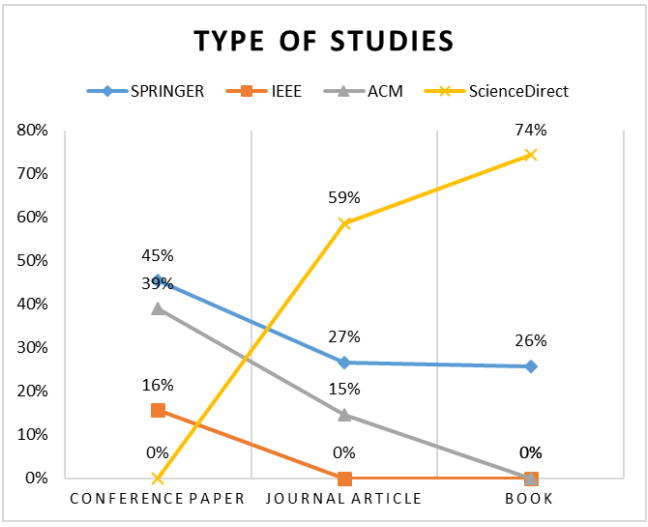

Fig. 2. Studies Types (Only Relevant Studies Published in the Past 20 Years.

IV. PRIMARY STUDIES AND DISCUSSIONS

In this section the main findings of this systematic review regarding 4GL code generation are presented and discussed. The findings have been categorised into three main sections: 1) Transformation between 4GL languages; 2) End user computing; 3) other related studies. In this review, the related studies are the studies which indirectly consider the code generation in 4GL, such as studies considering effort estimation in producing, manually re-engineering, refactoring, and maintaining 4GL software systems.

A. Transformation between 4GL Languages

Kicsi et al. [15] have introduced a semi-automatic approach which extracts features from a 4GL language namely Magic language. They have tested their approach on 2000 programs written in Magic. However, the completed stage up-to-date is extraction stage and the project is still ongoing. In addition, experts are needed to provide design decisions.

In addition, Kicsi et al. [16] have introduced an approach that extracted the structural and conceptual feature of legacy systems built in Magic language. The approach aims to provide two level of views on the legacy systems. The first level views are for expert which show the conceptual features. The second level views are for developer which shows structural feature. Although, this work is a promising in the reengineering of 4GL systems as it helps is the stage of design discovery, the work currently provides information for different level of stakeholders to make decisions in the re-designing of the legacy systems when the adopting Software product line (SPL) architecture.

Mendivelso et al. [9] have introduced an approach that relies on Model-Driven Reverse Engineering (MDRE) in order to reverse engineer programs in 4GL languages such as Oracle Forms, Visual Basic and Delphi. The resulting outputs are two different levels of views on Sirius graphical editor. The first level is for the end user who are the developers, architects, and testers. The second level is for MDRE experts. The approach seems promising, however, experts are needed to validate and verify the output model of a given source code. In addition, no forward engineering (complete code generation) is completed by the approach.

Newcomb et al. [17] have reported on a project called Pilot Project which aims to transform 4GL software systems into more standardized and modernized platforms namely Java and JavaScript. The work has a feature of considering conversion of non-functional requirements as well as security requirements. Moreover, tests were done on small scale programs to prove the concept. However, an important point is that 4GL software are often large scale ones. So, in order to generalize the findings this could be a point of weakness. In addition, further manual tuning was needed as performance issues occurred.

Sneed et al. [18] have introduced an approach that consider re-implementing legacy systems built by 4GL languages into object-oriented 3GL languages. This approach aim is to avoid the risks of automated conversion of such systems and taking into account the preserving functionalities. The approach was tested on two 4GL languages programs, namely,

VisualAge/PL/I-DB2 and COBOL-IMS applications. However, they have reported a number of side effects including comprising design and possibility of re-developing the whole architecture. This stands against the idea of preserving the original architecture.

Garcés et al. [19] have introduced a new semi-automated approach that transformed 4GL program to modernized platforms namely from Oracle Forms to Java programs. The approach has been tested on 5 medium scale 4GL programs. Although, the approach has a tool supports and seems promising, it has some backwards which are worth to mention here. Firstly, the approach is semi-automated and needs a human intervene. This might, as author reported, time consuming and error-prone. In addition, migration is a manual process. Finally, although, the results showed a reduction in time with comparison with other transformation processes, time overhead and code defects are still a considerable issue.

Salvatierra et al. [20] have introduced an indirect and semi-automatic approach for migration of legacy systems in COBOL into Service-Oriented Architecture (SOA). The approach is called Assisted Migration and has been tested on a legacy system of an Argentinean government agency. The aim of the approach is to enhance the quality of direct migrated version of the legacy system. However, the approach introduces a need for human experiences to perform as intended. In addition, accuracy is still an issue. Furthermore, legacy systems are not transformed or replaced which means adding more layers to use these systems.

Sánchez Ramón et al. [21] have introduced a new approach based on Model Driven Engineering (MDE). The approach aims to automated re-engineering process of the interfaces of programs built in 4GL namely Oracle Forms or Borland Delphi. Currently, the approach allows detecting the main elements in GUI and generate a tree to represents the arrangement of element on the GUI window. The resulting outputs is a model in Concrete User Interface (CUI) which can be used for further forward engineering.

Nagy et al. [22] in 2011 have investigated the lack of work on 4GL same language version transformation. In addition, the authors have offer a new approach to automatically transform code from Magic older to Magic version 5. However, the focus of the work was only on version 5 of Magic. In addition, performance issues have been raised. In addition later, Nagy [23] in 2013 have introduced an automated approach to recovering architecture of data-intensive applications developed in Magic 4GL. However, the approach only support static reverse engineering from SQL no forward engineering or round-trip engineering.

Martin et al. [20] introduced a new approach to transform source code between two different 4GL platforms. The approach aims to overcome the incompatibilities between 4GL. Therefore, a tool, called OctaveToR, was developed to automate the transformation from a source code written in Octave to a target code in R language. The approach employed TXL transformation language and was tested in a medium size source code. Although, the approach provides almost an instant code transformation, a number of issues are reported. These issues are performance, readability, and information loss. In addition, a use of TXL cannot be an ideal solution as it does not offer a feature abstraction representation.

Yafi et al. [24] introduced a new method that allow overcoming a problem that occurs when parsing Uniface 4GL languages source code embedded in XML format. Although the tool provides an automated way for reverse-engineer a 4GL code, the work was only to improve the readability and the work is in progress.

Nandivada et al. [25] introduced a framework that translate 4GL program in ARAP to java equivalent. This is for the purpose of debugging fault in 4GL programs. However, the work still incomplete and suffer from incorrectness of transformation with some statements in selected 4GL syntax as well as some overhead issues.

Bimonte et al. [26] introduced a new Model Driven Development (MDD) method which combines the use of ETL (Extraction, transform, load) and their Business Process Modeling Notation (BPMN) approaches to transform source from ETL to Oracle MetaBase (OMB) scripting language code [27]. However, efficiency in resources and time is an issue.

Reus et al. [28] have introduced an approach which aids in reverse engineer legacy systems to model-driven architecture (MDA). In specific, reverse engineer a 4GL program to language-independent models in UML, namely, Class, State-chart, Collaboration diagrams. Code generation then is offered to transform the models to Java classes. The approach has been tested to an Oracle's PL/SQL program of an insurance company in Netherlands. Although, the authors have introduced a promising approach which aims to automate completely the re-engineering trip from a 4GL source code to another platform, a number of pitfalls are there such as scalability issues. Another open challenge is the representation of business logic. In addition, the code generation is an uncompleted goal as the approach only generate code stubs (no functionally).

Cleve et al. [29] introduced an approach for transformation of date structure from 4GL language in legacy system to a modular structure for other platforms. However, the main concern was the data migration and data structure re-factoring. In addition, it took 10 days for re-engineering which is an obvious overhead. This can be linked to the previous work by Canfora et al. [30] where they in the same manner introduced their approach and noted the re-engineering risks increase beside the costs and performance issues.

Andrade et al. [12] have introduced a tool called Forms2Net which aims to transform the Oracle Forma and PL/SQL code to .NET C# program bearing in mind semantics and similarity and differences. Authors have investigated semantics and functionality in such transformations and offer the tool based on this. Forms2Net is a promising tools to facilitate an automatic code transformation, however, a number of shortcomings are impotent to mention here. First, complex transformations decisions are not made and left to the developer to re-engineer which introduced the human intervention. Second, only one output architecture is allowed which is the Model View Controller (MVC) architecture. Third, migration process needs to be simplified as currently

Forms2Net is attached with a number of guides for explaining it. Finally, runtime calls in the Oracle Forms are not sufficiently represented in C# outputs program.

B. End user Computing

Waszkowski [31] have introduced Aurea BPM low-code platform that allows users to draw in BPMN diagrams which will be transformed into working web pages with XML and supplementary files. The main goal is to automate the generation of application for business processes. However, authors stated that low-code platform is hard in manufacturing and it raises the risk of verifications.

Related to this topic, a number of authors [4], [32] have published a book in which they describe a number of examples of 4GL languages for End-user programing. The books show an exploring view on the available tools for such an approach. Others [33] have looked at it from different viewpoints such as the risks of privacy and errors which might be posed. Furthermore, bridging the knowledge gap between engineers and business users [34].

C. Other Related Topics

A number of related topics have been a focus for 4GL community, for example, Effort Estimation, quality assurance, testing, and distributed programming. However, it is important to mention here that Effort Estimation in producing, manually re-engineering, refactoring, and maintaining 4GL software systems has gained a considerable amount of attention form 4GL community.

Although, many researchers [35]–[45] have considered Effort Estimation, other have considered useful topics as well. For example, Shasharina et al. [46] have considered a model that offer ability of automating the linking the Grid Technology and Web services for 4GL legacy systems in Interactive Data Language (IDL). Furthermore, many researchers [47]–[49] have introduced their models and methods on measuring quality and have offers a number of matrices for this. However, quality assurance for 4GL suffer from a lack of work on it. Other have considered different related topics to 4GL. For instance, Zaytsev [50] has reported onto a new tool which generates test codes for a 4GL programs in specific C# programs. It has been tested on a large scale code of a company where the author is working. However, this is an ongoing project of generating code of testing for 4GL languages compiler with focus to C#. Furthermore, Albizuri-Romero [51] discussed different factors influenced organizations when choosing CASE tools.

V. Conclusions

In conclusion of this systematic review, four well-known digital libraries have been used to search for research studies that are related to the topic of 4GL code generation over the past 20 years. These libraries are Springer, IEEE, ACM, and ScienceDirect. Total of 593 studies were found. After applying the criteria of inclusion and exclusion employed in this paper, a total of 187 studies were found relevant studies. Out of these relevant studies only 28 were found primaries studies.

The following summarizes the main findings of the primary studies published over the past 20 years:

1) In general, there is a lack of studies in 4GL code generation.

2) The topic of 4GL languages code generation is often focused on transforming 4GL source codes to different 4GL or 3GL languages.

3) Lack of studies is variety of 4GL languages as the majority focuses on auto-transformation of Oracle forms.

4) The majority of the studies introduce semi-automated approaches which often need experts to be involved.

5) The majority of studies are Immature studies which might be due to the specialized nature of 4GL and that most of 4GL are proprietary.

Reaching this point of this paper, the previously mentioned research question of this paper can be answered. 4GL code generation is a topic that has not been considered sufficiently over the past 20 years. The offered studies and approaches are inadequate, and more work is needed in this topic. Furthermore, the previous section shows a detailed answer of the research question.

For future work, this paper can be basis for researchers interested in 4GL code generation and code transformation as the paper aimed to cover the work done on the topic for the past 20 years. As for authors for this paper, the future work direction is to fill the research gap of transforming Uniface 4GL to C#.net as the limitation of coverage for this direction is clear. In addition, a data set that allow constructing the transformation model has been shared from one of institute interested in the direction.

Acknowledgment

I would like to express my gratitude to my University (Umm Al Qura University) which gave me the opportunity to do this research. Secondly I would also like to thank family for their unlimited support. In addition, I would like to thank Saudi Digital Library (SDL) for giving me the membership to allow accessing digital libraries used in this paper.

References

[1] J. Krogstie, A. L. Opdahl, and S. Brinkkemper, Eds., Conceptual Modelling in Information Systems Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[2] W. H. Inmon, D. Strauss, and G. Neushloss, ‘Chapter 1 - A brief history of data warehousing and first-generation data warehouses’, in DW 2.0, W. H. Inmon, D. Strauss, and G. Neushloss, Eds. Burlington: Morgan Kaufmann, 2008, pp. 1–22.

[3] C. Combe, ‘Chapter 2 - E-business technology’, in Introduction to e-Business, C. Combe, Ed. Oxford: Butterworth-Heinemann, 2006, pp. 21–52.

[4] C. Shipley and S. Jodis, ‘Programming Languages Classification’, in Encyclopedia of Information Systems, H. Bidgoli, Ed. New York: Elsevier, 2003, pp. 545–552.

[5] B. Selic, ‘Personal reflections on automation, programming culture, and model-based software engineering’, Autom. Softw. Eng., vol. 15, no. 3–4, pp. 379–391, 2008.

[6] J. Jong, ‘History: How Did We Get Here?’, in Vertically Integrated Architectures: Versioned Data Models, Implicit Services, and Persistence-Aware Programming, J. Jong, Ed. Berkeley, CA: Apress, 2019, pp. 15–28.

[7] J. Whitten and L. Bentley, Systems Analysis and Design Methods, 7th edition. Boston: McGraw-Hill/Irwin, 2005.

[8] F. J. Budinsky, M. A. Finnie, J. M. Vlissides, and P. S. Yu, 'Automatic code generation from design patterns', IBM Syst. J., vol. 35, no. 2, pp. 151–171, 1996.

[9] L. F. Mendivelso, K. Garcés, and R. Casallas, 'Metric-centered and technology-independent architectural views for software comprehension', J. Softw. Eng. Res. Dev., vol. 6, no. 1, p. 16, Dec. 2018, doi: 10.1186/s40411-018-0060-6.

[10] D. Budgen and P. Brereton, 'Performing systematic literature reviews in software engineering', in Proceedings of the 28th international conference on Software engineering, 2006, pp. 1051–1052.

[11] S. M. F. Ali and R. Wrembel, 'From conceptual design to performance optimization of ETL workflows: current state of research and open problems', VLDB J., vol. 26, pp. 777–801, 2017.

[12] L. Andrade, J. Gouveia, M. Antunes, M. El-Ramly, and G. Koutsoukos, 'Forms2Net–migrating oracle forms to microsoft. NET', in International Summer School on Generative and Transformational Techniques in Software Engineering, 2005, pp. 261–277.

[13] A. Kicsi, V. Csuvik, L. Vidács, Á. Beszédes, and T. Gyimóthy, 'Feature level complexity and coupling analysis in 4GL systems', in International Conference on Computational Science and Its Applications, 2018, pp. 438–453.

[14] B. Kitchenham, 'Procedure for undertaking systematic reviews', Comput. Sci. Depart-Ment Keele Univ. TRISE-0401 Natl. ICT Aust. Ltd 0400011T 1 Jt. Tech. Rep., 2004.

[15] A. Kicsi et al., 'Feature analysis using information retrieval, community detection and structural analysis methods in product line adoption', J. Syst. Softw., vol. 155, pp. 70–90, 2019, doi: https://doi.org/10.1016/j.jss.2019.05.001.

[16] A. Kicsi, L. Vidács, V. Csuvik, F. Horváth, A. Beszédes, and F. Kocsis, 'Supporting product line adoption by combining syntactic and textual feature extraction', in International Conference on Software Reuse, 2018, pp. 148–163.

[17] P. H. Newcomb, D. Henke, J. LoVerde, W. Ulrich, L. Nguyen, and R. Couch, 'Chapter 6 - PowerBuilder/4GL Generator Modernization Pilot**© 2010. The Software Revolution, Inc. All rights reserved.', in Information Systems Transformation, W. M. Ulrich and P. H. Newcomb, Eds. Boston: Morgan Kaufmann, 2010, pp. 133–170.

[18] H. Sneed and C. Verhoef, 'Re-implementing a legacy system', J. Syst. Softw., vol. 155, pp. 162–184, Sep. 2019, doi: 10.1016/j.jss.2019.05.012.

[19] K. Garcés et al., 'White-box modernization of legacy applications: The oracle forms case study', Comput. Stand. Interfaces, vol. 57, pp. 110–122, 2018, doi: https://doi.org/10.1016/j.csi.2017.10.004.

[20] G. Salvatierra, C. Mateos, M. Crasso, and A. Zunino, 'Towards a computer assisted approach for migrating legacy systems to SOA', in International Conference on Computational Science and Its Applications, 2012, pp. 484–497.

[21] Ó. Sánchez Ramón, J. Sánchez Cuadrado, and J. García Molina, 'Model-driven reverse engineering of legacy graphical user interfaces', in Proceedings of the IEEE/ACM international conference on Automated software engineering, 2010, pp. 147–150.

[22] C. Nagy, L. Vidács, R. Ferenc, T. Gyimóthy, F. Kocsis, and I. Kovács, 'Solutions for Reverse Engineering 4GL Applications, Recovering the Design of a Logistical Wholesale System', in 2011 15th European Conference on Software Maintenance and Reengineering, Mar. 2011, pp. 343–346, doi: 10.1109/CSMR.2011.66.

[23] C. Nagy, 'Static Analysis of Data-Intensive Applications', in 2013 17th European Conference on Software Maintenance and Reengineering, Mar. 2013, pp. 435–438, doi: 10.1109/CSMR.2013.66.

[24] M. Z. Yafi and A. Fatima, 'Syntax Recovery for Uniface as a Domain Specific Language', in 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim), Mar. 2018, pp. 61–66, doi: 10.1109/UKSim.2018.00023.

[25] V. K. Nandivada, M. G. Nanda, P. Dhoolia, D. Saha, A. Nandy, and A. Ghosh, 'A framework for analyzing programs written in proprietary languages', in Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, 2011, pp. 289–300.

[26] S. Bimonte, É. Edoh-Alove, H. Nazih, M.-A. Kang, and S. Rizzi, 'ProtOLAP: rapid OLAP prototyping with on-demand data supply', in Proceedings of the sixteenth international workshop on Data warehousing and OLAP, 2013, pp. 61–66.

[27] Z. El Akkaoui, E. Zimànyi, J.-N. Mazón, and J. Trujillo, 'A model-driven framework for ETL process development', in Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP, 2011, pp. 45–52.

[28] T. Reus, H. Geers, and A. Van Deursen, 'Harvesting software systems for MDA-based reengineering', in European Conference on Model Driven Architecture-Foundations and Applications, 2006, pp. 213–225.

[29] A. Cleve, J. Henrard, and J.-L. Hainaut, 'Co-transformations in Information System Reengineering', Electron. Notes Theor. Comput. Sci., vol. 137, no. 3, pp. 5–15, Sep. 2005, doi: 10.1016/j.entcs.2005.07.001.

[30] G. Canfora, A. Cimitile, A. De Lucia, and G. A. Di Lucca, 'Decomposing legacy programs: a first step towards migrating to client–server platforms', J. Syst. Softw., vol. 54, no. 2, pp. 99–110, Oct. 2000, doi: 10.1016/S0164-1212(00)00030-3.

[31] R. Waszkowski, 'Low-code platform for automating business processes in manufacturing', IFAC-Pap., vol. 52, no. 10, pp. 376–381, 2019, doi: https://doi.org/10.1016/j.ifacol.2019.10.060.

[32] J. Stigliano and M. Bruni, 'End-User Computing Tools', in Encyclopedia of Information Systems, H. Bidgoli, Ed. New York: Elsevier, 2003, pp. 127–139.

[33] R. R. Panko and D. N. Port, 'End User Computing: The Dark Matter (and Dark Energy) of Corporate IT', in 2012 45th Hawaii International Conference on System Sciences, Jan. 2012, pp. 4603–4612, doi: 10.1109/HICSS.2012.244.

[34] G. Baster, P. Konana, and J. E. Scott, 'Business components: a case study of bankers trust Australia limited', Commun. ACM, vol. 44, no. 5, pp. 92–98, 2001.

[35] P. A. Whigham, C. A. Owen, and S. G. Macdonell, 'A baseline model for software effort estimation', ACM Trans. Softw. Eng. Methodol. TOSEM, vol. 24, no. 3, pp. 1–11, 2015.

[36] L. Song, L. L. Minku, and X. Yao, 'A novel automated approach for software effort estimation based on data augmentation', in Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, pp. 468–479.

[37] S. Amasaki and C. Lokan, 'A replication study on the effects of weighted moving windows for software effort estimation', in Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, 2016, pp. 1–9.

[38] L. L. Minku and X. Yao, 'An analysis of multi-objective evolutionary algorithms for training ensemble models based on different performance measures in software effort estimation', in Proceedings of the 9th international conference on predictive models in software engineering, 2013, pp. 1–10.

[39] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, 'An effective approach for software project effort and duration estimation with machine learning algorithms', J. Syst. Softw., vol. 137, pp. 184–196, 2018.

[40] T. Tran, V. Nguyen, T. Truong, C. Tran, and P. Le, 'An Evaluation of Parameter Pruning Approaches for Software Estimation', in Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering, 2019, pp. 26–35.

[41] M. Tanrıverdi and Ö. Ö. Tanrıöver, 'An experimental comparison of software effort estimation methods of ORM based 4GL software applications', in 2017 International Conference on Computer Science and Engineering (UBMK), 2017, pp. 239–243.

[42] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobyliński, 'Application of function points and data mining techniques for software estimation-a combined approach', in Software Measurement, Springer, 2015, pp. 96–113.

[43] P. Rijwani and S. Jain, 'Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique', Procedia Comput. Sci., vol. 89, pp. 307–312, 2016, doi: https://doi.org/10.1016/j.procs.2016.06.073.

[44] F. Ferrucci, C. Gravino, and F. Sarro, 'Exploiting prior-phase effort data to estimate the effort for the subsequent phases: a further assessment', in Proceedings of the 10th International Conference on Predictive Models in Software Engineering, 2014, pp. 42–51.

[45] C. Nagy, L. Vidács, R. Ferenc, T. Gyimóthy, F. Kocsis, and I. Kovács, 'MAGISTER: Quality assurance of Magic applications for software developers and end users', in 2010 IEEE International Conference on Software Maintenance, Sep. 2010, pp. 1–6, doi: 10.1109/ICSM.2010.5609550.

[46] S. G. Shasharina, O. Volberg, P. Stoltz, and S. Veitzer, 'GRIDL: high-performance and distributed interactive data language', in HPDC-14. Proceedings. 14th IEEE International Symposium on High Performance Distributed Computing, 2005., Jul. 2005, pp. 291–292, doi: 10.1109/HPDC.2005.1520980.

[47] Z. Tóth, L. Vidács, and R. Ferenc, 'Comparison of static analysis tools for quality measurement of rpg programs', in International Conference on Computational Science and Its Applications, 2015, pp. 177–192.

[48] M.-A. Côté, W. Suryn, C. Y. Laporte, and R. A. Martin, 'The evolution path for industrial software quality evaluation methods applying ISO/IEC 9126: 2001 quality model: example of MITRE's SQAE method', Softw. Qual. J., vol. 13, no. 1, pp. 17–30, 2005.

[49] G. C. Green, A. R. Hevner, and R. W. Collins, 'The impacts of quality and productivity perceptions on the use of software process improvement innovations', Inf. Softw. Technol., vol. 47, no. 8, pp. 543–553, 2005.

[50] V. Zaytsev, 'An industrial case study in compiler testing (tool demo)', in Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, 2018, pp. 97–102.

[51] M. B. Albizuri-Romero, 'A retrospective view of CASE tools adoption', ACM SIGSOFT Softw. Eng. Notes, vol. 25, no. 2, pp. 46–50, 2000..

APPENDIX A

PRIMARY STUDIES

Item	Bibliography	Source
PS1	H. Sneed and C. Verhoef, 'Re-implementing a legacy system', Journal of Systems and Software, vol. 155, pp. 162–184, Sep. 2019, doi: 10.1016/j.jss.2019.05.012.	ScienceDirect
PS2	R. Waszkowski, 'Low-code platform for automating business processes in manufacturing', IFAC-PapersOnLine, vol. 52, no. 10, pp. 376–381, 2019, doi: https://doi.org/10.1016/j.ifacol.2019.10.060.	ScienceDirect
PS3	A. Kicsi et al., 'Feature analysis using information retrieval, community detection and structural analysis methods in product line adoption', Journal of Systems and Software, vol. 155, pp. 70–90, 2019, doi: https://doi.org/10.1016/j.jss.2019.05.001.	ScienceDirect
PS4	L. F. Mendivelso, K. Garcés, and R. Casallas, 'Metric-centered and technology-independent architectural views for software comprehension', J Softw Eng Res Dev, vol. 6, no. 1, p. 16, Dec. 2018, doi: 10.1186/s40411-018-0060-6.	SPRINGER
PS5	M. Z. Yafi and A. Fatima, 'Syntax Recovery for Uniface as a Domain Specific Language', in 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim), Mar. 2018, pp. 61–66, doi: 10.1109/UKSim.2018.00023.	IEEE
PS6	V. Zaytsev, 'An industrial case study in compiler testing (tool demo)', in Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering, 2018, pp. 97–102.	ACM
PS7	A. Kicsi, L. Vidács, V. Csuvik, F. Horváth, A. Beszédes, and F. Kocsis, 'Supporting product line adoption by combining syntactic and textual feature extraction', in International Conference on Software Reuse, 2018, pp. 148–163.	SPRINGER
PS8	K. Garcés et al., 'White-box modernization of legacy applications: The oracle forms case study', Computer Standards & Interfaces, vol. 57, pp. 110–122, 2018, doi: https://doi.org/10.1016/j.csi.2017.10.004.	ScienceDirect
PS9	R. R. Panko and D. N. Port, 'End User Computing: The Dark Matter (and Dark Energy) of Corporate IT', in 2012 45th Hawaii International Conference on System Sciences, Jan. 2012, pp. 4603–4612, doi: 10.1109/HICSS.2012.244.	IEEE
PS10	G. Salvatierra, C. Mateos, M. Crasso, and A. Zunino, 'Towards a computer assisted approach for migrating legacy systems to SOA', in International Conference on Computational Science and Its Applications, 2012, pp. 484–497.	SPRINGER
PS11	C. Nagy, L. Vidács, R. Ferenc, T. Gyimóthy, F. Kocsis, and I. Kovács, 'Solutions for Reverse Engineering 4GL Applications, Recovering the Design of a Logistical Wholesale System', in 2011 15th European Conference on Software Maintenance and Reengineering, Mar. 2011, pp. 343–346, doi: 10.1109/CSMR.2011.66.	IEEE
PS12	V. K. Nandivada, M. G. Nanda, P. Dhoolia, D. Saha, A. Nandy, and A. Ghosh, 'A framework for analyzing programs written in proprietary languages', in Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, 2011, pp. 289–300.	ACM
PS13	Z. El Akkaoui, E. Zimànyi, J.-N. Mazón, and J. Trujillo, 'A model-driven framework for ETL process development', in Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP, 2011, pp. 45–52.	ACM
PS14	Ó. Sánchez Ramón, J. Sánchez Cuadrado, and J. García Molina, 'Model-driven reverse engineering of legacy graphical user interfaces', in Proceedings of the IEEE/ACM international conference on Automated software engineering, 2010, pp. 147–150.	SPRINGER
PS15	P. H. Newcomb, D. Henke, J. LoVerde, W. Ulrich, L. Nguyen, and R. Couch, 'Chapter 6 - PowerBuilder/4GL Generator Modernization Pilot**© 2010. The Software Revolution, Inc. All rights reserved.', in Information Systems Transformation, W. M. Ulrich and P. H. Newcomb, Eds. Boston: Morgan Kaufmann, 2010, pp. 133–170.	ScienceDirect
PS16	T. Reus, H. Geers, and A. Van Deursen, 'Harvesting software systems for MDA-based reengineering', in European Conference on Model Driven Architecture-Foundations and Applications, 2006, pp. 213–225.	SPRINGER
PS17	A. Cleve, J. Henrard, and J.-L. Hainaut, 'Co-transformations in Information System Reengineering', Electronic Notes in Theoretical Computer Science, vol. 137, no. 3, pp. 5–15, Sep. 2005, doi: 10.1016/j.entcs.2005.07.001.	ScienceDirect
PS18	S. G. Shasharina, O. Volberg, P. Stoltz, and S. Veitzer, 'GRIDL: high-performance and distributed interactive data language', in HPDC-14. Proceedings. 14th IEEE International Symposium on High Performance Distributed Computing, 2005., Jul. 2005, pp. 291–292, doi: 10.1109/HPDC.2005.1520980.	IEEE

PS19	C. Nagy, 'Static Analysis of Data-Intensive Applications', in 2013 17th European Conference on Software Maintenance and Reengineering, Mar. 2013, pp. 435–438, doi: 10.1109/CSMR.2013.66.	IEEE
PS20	L. Andrade, J. Gouveia, M. Antunes, M. El-Ramly, and G. Koutsoukos, 'Forms2Net–migrating oracle forms to microsoft. NET', in International Summer School on Generative and Transformational Techniques in Software Engineering, 2005, pp. 261–277.	SPRINGER
PS21	J. Martin and J. Gutenberg, 'Automated source code transformations on fourth generation languages', in Eighth European Conference on Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings., Mar. 2004, pp. 214–220, doi: 10.1109/CSMR.2004.1281422.	IEEE
PS22	J. Stigliano and M. Bruni, 'End-User Computing Tools', in Encyclopedia of Information Systems, H. Bidgoli, Ed. New York: Elsevier, 2003, pp. 127–139.	ScienceDirect
PS23	C. Shipley and S. Jodis, 'Programming Languages Classification', in Encyclopedia of Information Systems, H. Bidgoli, Ed. New York: Elsevier, 2003, pp. 545–552.	ScienceDirect
PS24	A. Arkusinski and E. Green, 'A software port from a standalone communications management unit to an integrated platform', in Proceedings. The 21st Digital Avionics Systems Conference, Oct. 2002, vol. 1, pp. 6B3-6B3, doi: 10.1109/DASC.2002.1067987.	IEEE
PS25	G. Baster, P. Konana, and J. E. Scott, 'Business components: a case study of bankers trust Australia limited', Communications of the ACM, vol. 44, no. 5, pp. 92–98, 2001.	ACM
PS26	G. Canfora, A. Cimitile, A. De Lucia, and G. A. Di Lucca, 'Decomposing legacy programs: a first step towards migrating to client–server platforms', Journal of Systems and Software, vol. 54, no. 2, pp. 99–110, Oct. 2000, doi: 10.1016/S0164-1212(00)00030-3.	ScienceDirect
PS27	S. Barker and A. Monday, 'Business students in information systems: wizards or apprentices?', in Proceedings of the Australasian conference on Computing education, 2000, pp. 6–11.	ACM
PS28	M. B. Albizuri-Romero, 'A retrospective view of CASE tools adoption', ACM SIGSOFT Software Engineering Notes, vol. 25, no. 2, pp. 46–50, 2000.	ACM