Journal of Applied Artificial Intelligence
https://jaai.sabapub.com
ISSN: 2709-5908

2023 Volume 4, Issue 2: 1-11

DOI: https://doi.org/10.48185/jaai.v4i2.867

Understanding the Impact of Requirements Evolution and Reaction on Evolution of Software: a Survey and Comparison

Ahmed Mahdi Salih[1,2,*], Mazni Omar[3], Osamah Mohammed Alyasiri[2,4], Pantea Keikhorakani[2], Sharifah Mashita Syed Mohamad[2]

[1]Department of Mathematics, College of Education for Pure Sciences, Tikrit University, Tikrit, Iraq
[2]School of computer sciences, University Sains Malaysia, Penang, Malaysia
[3]College of Art and Science, School of Computing, University Utara Malaysia, Kedah, Malaysia
[4]Karbala Technical Institute, Al-Furat Al-Awsat Technical University, Karbala 56001, Iraq

Received: 17.10.2023 • Accepted: 25.12.2023 • Published: 29.12.2023 • Final Version: 29.12.2023

Abstract: In software systems, the continuous changing of requirements, known as requirements evolution, is considered one of the significant issues. Requirements' evolution denotes the post-deployment changes in the requirements. This article reviews the most related requirements evolution approaches. Different approaches have been presented in modelling requirements evolution, managing requirements evolution, and relevant analysis techniques, like inconsistency detection and change impact analysis. The relevant approaches of requirements evolution can be generally classified into the impact of evolution and reaction on evolution. The article also has given a comparison among those approaches. The approaches that have been surveyed in this article exhibited many limitations. These limitations need to be addressed and coped with for the approaches to be more effective in managing the evolution of software requirements. One of the solutions to these limitations is to develop a framework that addresses the modelling and reasoning behind software requirements evolution. The framework will include evolution rules to capture evolution in the requirements model, particularly observable rules for capturing potential changes and their uncertainty and controllable rules for capturing different reactions from the designing aspect.

Keywords: Continuous changing, Requirements' evolution, Evolution uncertainty

1. Introduction

In software systems, the continuous changing of requirements, known as requirements evolution, is considered one of the significant issues. Practically, requirements evolution is still a major problem since the constant change makes the traceability and monitoring of requirements complicated and unreliable. However, it is an unavoidable activity since useful and successful software motivates users to demand new and improved requirements [1, 2].

Requirements' evolution occurs throughout the development life cycle due to continuous changes. Requirements' evolution denotes the post-deployment changes in the requirements. These changes happen when the system starts operating due to many factors, such as operational environments, changing technologies, and business needs [3]. These changes may involve qualitative and/or quantitative features of requirements. For instance, the specifications can be increased or more precise;

* Corresponding Author: ahmad.ballu@tu.edu.iq

tacit requirements have to be more explicit, or specifications are no longer needed and may be totally discarded [4]. Requirements' evolution is unavoidable throughout the software project lifecycle. It can make the systems faster, more reliable, and more efficient [5].

Additionally, requirements could be evolved to increase the understanding of the problem by the designers and the users themselves [6].

After this introduction of requirements evolution, an explanation of requirements evolution perspectives is presented. Then, requirements evolution approaches and their two classes, the impact of evolution and reaction on evolution, are illustrated, followed by a comparison and discussion of them. Finally, the conclusion of this article is derived.

2. Requirements Evolution Perspectives

Requirements' evolution is studied within a specific period. This period might be short (a few years) or long (10 years and above), depending on the lifetime of the software system [7]. Different evolution perspectives depend on the scenarios of how the study may evolve. Lund et al. [8] defined three evolution perspectives related to risk analysis: maintenance, before-after, and continuous evolution.

i. Maintenance evolution: This perspective is concerned with updating the documents for an available system. This perspective will not be considered in this paper as it mainly concentrates on the solution in advance of requirements model evolution in the early stages.
ii. Before-After evolution: This perspective expects future contexts through predicting the unplanned and planned changes in existing models of requirements when the study is complete. Several evolutions' probabilities will be considered, and each probability can happen. For instance, the Air Traffic Management (ATM) 2000+ Strategic Agenda [9] and "European Single European Sky ATM Research Initiative (SESAR)" [10] have defined the direction of the ATM improvements in the period from 2010 until 2020 to get one or more alternatives of novel tools of queue management. This management comprises Departure Management (DMAN), Arrival Management (AMAN), and Surface Management (SMAN).
iii. Continuous evolution: This perspective expects existing contexts' evolution over the study period, relying on gradually planned changes. The whole period of study will be divided into many milestones. Within each milestone, the possible changes in the requirements model can be predicted. Numerous evolution possibilities are also considered.

The critical distinction between the before-after and the continuous evolution is that before-after evolution is concerned with only one evolution possibility at definite points of time in the future. In contrast, continuous evolution enables numerous possibilities at definite points [8]. The former can be collapsed to the latter when there will be a single evolution possibility at each point. Thus, the definition of continuous evolution here can be considered as the generalization described in the study of Lund et al. [8].

Furthermore, the changes that Lund et al. [8] have addressed from the perspective of continuous evolution are considered expectable and gradual evolution that can be defined as time functions.

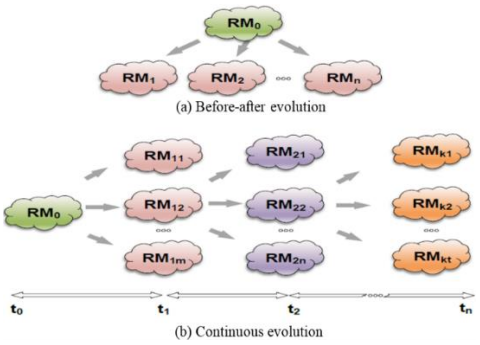

Figure 1. Requirements Evolution Perspectives

The expectations can be based on planned developments or well-founded predictions. This means that they are related to the situation in which they were designed to evolve over time, or, in other ways, they can expect continuous changes over time. Figure 1 shows the perspectives of evolution. In this figure, the requirements models are represented as clouds. Figure 1 (a) demonstrates before-after requirement evolution. This evolution examines requirements in a finite and determined period throughout the study. It can be analyzed throughout this period concerning how the requirement model will look at the beginning (before the model) and possibly at the end (after the model). Only one after model will happen [11]. On the other hand, Figure 1 (b) illustrates the continuous requirement evolution that at the time (t_0), RM_0 represents the original requirements model that can be evolved to one of RMi at t_1. The evolution constantly occurs at the end of the study (t_n). Consequently, the original requirement model can be one of RM_{kj}. The main distinction between the before-after and the continuous requirement evolution is that before-after evolution refers to one evolution possibility at specific points in the future. On the contrary, continuous requirement evolution enables many possibilities at specific points [12].

3. Requirements Evolution Approaches

The relevant approaches to requirements evolution can be generally classified into 1) the impact of evolution and 2) the reaction to evolution. Approaches to the impact of evolution aim to identify possible effects on artefacts (like models and specifications) and security properties or consistency violations. In contrast, approaches in the evolution reaction propose reactions to requirements evolution [13].

3.1. Approaches in Impacts of Requirements Evolution

As mentioned before, approaches to the impact of evolution focus on recognizing possible consequences on artefacts (like models and specifications) and violating security or consistency properties. There are many approaches related to the impact of requirements evolution. The most related of them are explained in Table 1.

Table 1. Summary of the approaches in impacts of requirements evolution

No	The Approach	Aims	Main Contributions	Main Limitations
1	"Viewpoint-Oriented for Restructuring Requirements Specification [14]"	Revision and analysis approach for restructuring software requirements. Discover inconsistency and manage changes. Decomposing requirements specifications into parts	Track and analyze evolutionary changes in the original requirements specification. Allow evolutional changes to occur and verify their impact on requirements satisfaction.	Manually performed. Depends on personal experiences (not so accurate). The domain-dependent rules are hard-coded into the approach.

No	The Approach	Aims	Main Contributions	Main Limitations
2	"Change Impact Analysis Using Use Case Maps (UCMs**)** [1]"	Change impact analysis. Applies both slicing and dependency analysis at the Use Case Map specification level. Illustrate the approach's applicability with a case study conducted on a telephony system.	Identify potential consequences of a change. Estimate what needs to be modified to accomplish a change. Adapt functional and nonfunctional requirement changes without destroying the integrity of the underlying system architecture.	Requires dynamic information to minimize the size of UCM's slices. Lack of measurements of impact analysis prediction at the UCM level.
3	"Formal Concept Analysis-Based (FCA) [15]"	Addressed the problem of controlling the evolution of requirements during the software development process. Deal with requirements stated in natural language and address the inconsistency among requirements belonging to various evolutionary stages.	Requirements of various evolutional stages have traceability connections to the others. Detect errors in evolution from one requirement to another in the different stages.	It is hard to analyze more extensive systems. The difficulties of modelling software entities as components of Formal Concept Analysis (FCA). Concept interpretation is a hard and time-consuming task.
4	"UML 4PF Profile and Tool Assistance of Evolutionary Requirements [16]"	Performing requirements evolution is achieved by identifying the rules of each requirements engineering step. It was developed based on the eclipse modelling framework.	The requirements' evolution method is embedded in an improvement process. The determined number of operators and rules helps conduct software evolution.	It lacks quantitatively systematic reasoning for supporting decision makers to select the system design alternatives that can be more evolution-resilience. Do not consider requirements evolution uncertainty.
5	"SeCMER Method for Managing Requirements Evolution [17]"	SeCMER is a tool for requirements evolution management developed in the context of the SecureChange project. Addresses the before-after evolution perspective. Manage evolution developed in the context of secure change projects. The tool allows modelling the evolution of the requirement model as the effect of introducing the (System Wide Information Management).	Provides support for: Modelling requirements evolution in Si* modelling language. Managing Changes based on rules of evolution. Argumentation-based security analysis The tool supports the automatic detection of requirement changes and violation of security properties using change-driven transformations.	Some concepts cannot map from one formalization to another. Some modelling elements map to many corresponding modelling elements in other formalizations.
6	"Creative Strategic Scenarios for Preparation to Requirements Evolution [4]"	Focus on Creative Strategic Scenarios as predictive models of software evolution for sociotechnical systems in organizations.	Combine Strategic Planning and Creativity theories to generate strategic scenarios that could predict Organizational Changes. Integrate scenarios and i* modelling mechanism to analyze the impacts of organizational change.	The i* goal modelling language is considered complex and hard to understand its graphical representation by practitioners. Requires an iterative methodological approach because not all the changes can be anticipated from the beginning.

No	The Approach	Aims	Main Contributions	Main Limitations
7	"Visualizing the Effects of Requirements Evolution [18]"	Employ a Requirements Evolution Chart (REC), a graphical representation of requirements evolution generated from issue tickets. Examine whether a Requirements Evolution Charts (REC) helps software engineers conduct an impact analysis.	The study revealed that engineers in the REC group identified the affected artefacts more accurately and quickly than the non-REC group. Identify requirements evolution events based on combinations of operations in the issue tickets. The REC visualizes a series of these events identified by the mapping rules and the issue ticket list.	Analyzed only one part of an extensive document management system regulated by laws and regulations (lack of generalizability).
8	"How do requirements evolve over time? A case study investigating the role of context and experiences in the evolution of enterprise software requirements [12]."	Conducted a longitudinal, exploratory single-case study of the life cycle of cloud-based enterprise software (ES) in a medium-sized organization. Isolate nine mechanisms that explain how contextual factors and experiences are intertwined and shape the evolution of requirements.	The developed process theory sheds light on mechanisms that shape the evolution of ES requirements. Sourcing cloud-based ES changes the influence of business divisions in: Acquisition and configuration activities, the role of upgrade and customization procedures, and the influence of the ES' ecosystem.	The research followed a single-case study methodology (limitation of results generalizability).
9	"Readiness model for requirements change management in global software development [19]."	Develop requirements change management readiness model (RCMRM) for Global Software Development (GSD) Global Software Development organizations	The developed readiness model can help organizations reduce their requirements change management (RCM) implementation challenges to produce and maintain quality software. Researchers can use this model for further enhancement to reflect the ever-dynamic nature of industry practices. The proposed RCMRM effectively accesses and improves RCM activities in the context of GSD.	The developed model does not consider the non-GSD context. Small number of participating organizations in the study (a limited generalization of the case study results).
10	Effect of Human-Related Factors on Requirements Change Management in Offshore Software Development Outsourcing: A theoretical framework [20].	Identify related success factors (HSFs) and human-related challenges (HCHs) that could influence the RCM RCM (requirement change management) process in GSD (Global software development) organizations.	The study reveals that five out of ten HSFs and 4 out of ten HCHs are critical for RCM process implementation in GSD. Develop a theoretical framework of the identified factors concerning process implementation. The results of this research can help tackle the complications associated with the RCM in the GSD environment, which is vigorous to the success and progression of GSD organizations.	Numerous articles are missing enough information about organization size, while the data extraction, e.g., from 25 articles, only 14 discussed organization size in detail. Cannot utilize every last one of assessable advanced libraries, e.g., Scopus. The study needs to conduct a real-world practitioners' survey to identify the more challenges and success factors of the RCM process in GSD.

No	The Approach	Aims	Main Contributions	Main Limitations
11	Managing The Uncertainty of The Evolution of Requirements Models [21].	• Capture the requirements evolution and its uncertainty. • Provide a set of metrics with formal semantics for reasoning about evolution uncertainty. • Automate (with formal analysis and tool support) the reasoning that can enumerate and quantitatively assess individual design alternatives.	Propose a framework that could capture the requirements evolution and evolution uncertainty. To facilitate the evaluation of the proposed framework, the study identifies several success criteria concerning research questions.	The study is limited to a single domain and a particular requirement of engineering language. Evolution uncertainty is a kind of subjective probability; there is a need to evaluate evolution uncertainty based on the interpretation of uncertainty and the Analytic Hierarchical Process (AHP), which is used in the literature to prioritize requirements.

3.2. Approaches in Reaction to Requirements Evolution

There are several approaches that aim to support the requirements evolution of the systems. Some of those approaches concentrate on early design phases, while others focus on the phases of deployment and implementation. There are many approaches related to the reaction of evolution. The most related of those approaches are explained in Table 2.

Table 2. Summary of the approaches in reaction to requirements evolution

No	The Approach	Aims	Main Contributions	Main Limitations
1	"Logical Framework for Modelling and Reasoning of Requirements Evolution [22]"	Capturing intuitive aspects to manage changes happen to requirements models. Referring to Fig. 8, RE starts with the expression of a set of incomplete goats, the incomplete Requirements Model (RMo). Requirements engineers use defaults and assumptions to convert these incomplete sentences into a complete requirements model (RMo, RM_1, etc). By a series of revisions, the model is refined and completed.	Modelling requirements as theories. Reasoning on changes through mapping changes among models.	Did not commit to a specific modelling language. Lack of analysis requirements evolution consequences during the operations.
2	"Problem Frames for Change [23]"	Providing tools can help analyze and synthesize change that impacts an organization's sociotechnical systems. Problem Frames inspire the proposed tools.	Introduce a manual process of change analysis; that is, the situation before-the-change is changed to the situation after-the-change.	The approach did not use specific reasoning on changes but instead captured the before model part, which is changed.
3	"Incremental Solutions for Evolving Requirements [24]"	Focus on unknown-unknown requirements evolution, which is an evolution that is not known and cannot determine when it will occur. Study some algorithms by utilizing ATMS *(*AI's Truth Maintenance Systems*)*.	Discover new solutions by using as many old ones as possible the old solutions. Minimizing the task number that needs to be implemented.	Considers only unanticipated evolution changes; thus, these changes cannot be modelled. Requires reducing assumptions number to be more accurate.

No	The Approach	Aims	Main Contributions	Main Limitations
4	"Qualitative Reasoning for Deferential Relations [25] "	Focusing on run-time. The system's dynamic behaviour is governed by a group of (in) equations named "qualitative differential constraints."	Characterize the controllability space of software systems in terms of variation points, requirements models, indicators, and control variables.	Lack of essential information that controllers require of a feedback loop to adapt their target systems. The limitation of omitting the effects of patterns on adaptation flexibility.
5	"Event Condition Action Rules for Modelling Requirements Evolution [26]."	Described new requirements families: *EvoReqs (Evolution Requirements) and AwReqs (Awareness Requirements).*	Identify changes in other requirements when specific situations are applied. Allow modelling changes of requirements models precisely and explicitly.	Large sets of rules are difficult to evolve. The approach is hard to apply to systems dependent on third-party services/components or legacy systems.
6	"Looking into the Crystal Ball: Requirements Evolution over Time [11]."	Presented a method for specifying changes in intentions over time and a technique that uses simulation for asking a variety of 'what if' questions about goal models.	Understand trade-offs in selecting development technologies over the goals to have a functional, practical, useable, and maintainable tool. Determine which tasks must be completed in a prescribed order and which were independent. Overall, these strategies were effective in understanding possible evolutions of the requirements. Goal modelling for early-phase requirements engineering can be improved by explicitly modelling and analyzing intention evaluations over time.	Operationalized intentions' changing evaluations with the dynamic functions, but did not establish that this was the best representation. Given the research bias discussed in the paper, there is a risk that the model may not be representative of other iStar goal models. This paper dealt with only "relative" times and can be extended by adding "wall clock time" to the analysis.
7	"Sentiment Analysis-Based Requirement Evolution Prediction [27]."	Propose a framework that combines a supervised deep-learning neural network with an unsupervised hierarchical topic model to analyze user reviews automatically for product feature requirements evolution prediction.	The results of this study contribute to efforts toward automatic text-mining analysis for product requirements engineering. The approach detected product features mentioned in the user review text for different granularities with sentiment orientation. Distributed word embedding can differ from the training objectives and language models. Therefore, the quality of the word embedding could impact the efficacy of the sentiment classification results.	The text analysis-based approach to product requirements evolution detection should be adapted to the implicit context to identify implicit product features and sentiment. Used hierarchical Latent Dirichlet (LDA) to extract software features. However, LDA is not suitable for analyzing shorter texts such as tweets due to the sparsity of word co-occurrence patterns in the individual document [28].
8	"Evaluating Mutual Requirements Evolution of Several Information Systems [29]."	Proposed and exemplified a method of eliciting and evaluating requirements of several systems together.	The proposed method can help analysts discover unaware requirements for each system, improving each system and its supported activities. In this method, analysts can use any modelling language to focus on different aspects of systems. Therefore, this method may improve the efficiency of human, time, and space resources and system development efforts.	The method primarily depends on the insight of analysts (subjective views). There is a limitation in managing the links between model elements. Therefore, the method must develop a traceability management tool based on an existing modelling editor.

No	The Approach	Aims	Main Contributions	Main Limitations
9	Formal reasoning for analyzing goal models that evolve over time [30].	Formalize the Evolving Intentions framework for specifying, modelling, and reasoning about goals that change over time.	Specify a set of functions that define how intentions and relationships evolve. Use path-based analysis to ask various "what if" questions about requirements changes.	Unable to represent all possible behaviors of model intentions. Goal models, in general, are considered to be open-world artefacts; this means that it is assumed that a decomposition relationship can have an additional source, i.e., a child, that is not yet present in the model. The Evolving Intentions framework requires additional information in the specification of the evolving functions and is therefore limited by the modellers' ability to express anticipated changes.
10	How do requirements evolve during elicitation? An empirical study combining interviews and App Store analysis [13]	Study how requirements get transformed from initial ideas into documented needs and then evolve based on the inspiration from similar products. Select 30 subjects that act as requirements analysts and perform interview-based elicitation sessions with a fictional customer.	The study empirically showed that requirements are not elicited in the strict sense but co-created through interviews, with analysts playing a crucial role in the process. The study also showed evidence that app store-inspired elicitation could be particularly beneficial to completing the requirements.	Only focusing on the requirement evolution after a software product is deployed might make the tags less fitting for requirements before a product has been deployed [31].
11	Union Models: Support for Efficient Reasoning About Model Families Over Space and time [32]	Proposes union models as a paradigm supporting the representation of model families (for time and space dimensions) using one generic model.	Demonstrate empirically the usefulness of union models for analyzing a family of models all at once, compared to individual models, one model at a time. Suggests that the use of union models facilitates efficient analysis in several contexts.	The study is limited to simple type graphs, where attributes of model elements have to be expressed structurally with named nodes and edges. The study also limited to language-independent, syntactic properties (which describes the structure of models) other than semantic properties (which describe the behaviour of models, e.g., traces).

3.3. Comparison and Discussion

From the surveys of Tables 1 and 2, it can be noticed that there are limitations in evolution uncertainty and systematic and quantitative reasoning of many existing approaches of requirements evolution modelling. These limitations can restrict their utilization in managing the uncertainty of software requirements evolution. For instance, studies such as Russo, Nuseibeh, & Kramer [14], Hassine et al. [1], Côté & Heisel [16], Bergmann et al. [17], Schneider [12], and Mehmood [20] have focused on the issue of management and consistency of requirements without addressing the modelling and reasoning of requirements evolution uncertainty.

Furthermore, the studies of Zowghi & Offen [22], Brier et al. [23], Ernst, Borgida, & Mylopoulos [24], Souza et al. [26], and Ferreira [13] proposed the implementation of management or testing consistency on the evolution of requirements; however, they did not develop an evident approach for the reasoning of requirements evolution.

It is observed that the reviewed studies lack a comprehensive framework that can be used for modelling and reasoning requirements evolution. This modelling and reasoning for the requirement is essential to predict the most accurate and low-cost requirements for the software systems that suffer from continuous changes over time.

4. Conclusion and Future Work

Software requirements change and evolve continuously. The evolution of software requirements is an unavoidable phenomenon during the operation of long-lived software systems because of the dynamic nature of their operating environments. Therefore, software systems may be unstable or non-operational. Requirements' evolution occurs throughout the development life cycle due to continuous changes. If this evolution is not appropriately managed, it can cause costly repairs and time-consuming inconsistencies. The approaches of requirements evolution surveyed in this article exhibited many limitations. These limitations need to be addressed and coped with for the approaches to be more effective in managing the evolution of software requirements.

One of the solutions to these limitations is to develop a framework that addresses the reasoning behind software requirements evolution. This framework will be based on continuously changing requirements within the software lifecycle. Furthermore, this framework can be designed with factors like time, cost, and behaviour. Moreover, the suggested framework will be modelling and reasoning about the evolution of the requirements model in long-lived software systems. It will aim to provide a means for studying requirements evolution. The framework will include evolution rules to capture evolution in the requirements model, particularly observable rules for capturing potential changes and their uncertainty and controllable rules for capturing different reactions from the designing aspect (i.e., design choices or design alternatives) to these changes. Incorporating evolution in requirements models will allow designers to have a global view of the potential evolution of the system in future.

Future research can focus on the following issues:

- Replicate the empirical studies in another domain rather than ATM (Air Traffic Management) with different kinds of participants to see whether similar results could be obtained.
- Replicate the empirical studies with different requirement engineering languages rather than i*/Tropos to see whether the chosen requirement engineering languages impact the outcome of the studies.
- Evaluate different aspects of the framework rather than effectiveness, for example, whether the method is easy to use (i.e., Perceived Ease of Use), whether participants want to apply the method in practice (i.e., Intent of Use), and so on. These aspects could be obtained by performing interviews (with and/or without a predefined questionnaire) with individual participants.

Acknowledgement

The authors would like to express their gratitude to the editorial team and to the anonymous reviewers for their useful recommendations and constructive remarks.

References

[1] J. Hassine, J. Rilling, J. Hewitt, and R. Dssouli, "Change impact analysis for requirement evolution using use case maps," in *Eighth International Workshop on Principles of Software Evolution (pp. 81-90). IEEE.*, 2005, pp. 81-90: IEEE.

[2] B. Vogel-Heuser, A. Fay, I. Schaefer, and M. Tichy, "Evolution of software in automated production systems: Challenges and research directions," *Journal of Systems and Software,* vol. 110, pp. 54-84, 2015.

[3] W. Lam and M. Loomes, "Requirements evolution in the midst of environmental change: a managed approach," in *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering (pp. 121-127). IEEE.*, 1998, pp. 121-127: IEEE.

[4] M. G. Ferreira, "Creative Strategic Scenarios for preparation to requirements evolution," in *2014 IEEE 22nd International Requirements Engineering Conference (RE) (pp. 494-499)*, 2014, pp. 494-499: IEEE.

[5] K. Mu, Z. Jin, and R. Lu, "Measuring Software Requirements Evolution Caused by Inconsistency," *Int. J. Software and Informatics,* vol. 6, no. 3, pp. 419-434, 2012.

[6] M. Hamill and K. Goševa-Popstojanova, "Common trends in software fault and failure data," *Software Engineering, IEEE Transactions on,* vol. 35, no. 4, pp. 484-496, 2009.

[7] R. Scandariato, F. Paci, K. Labunets, K. Yskout, F. Massacci, and W. Joosen, "Empirical Assessment of Security Requirements and Architecture: Lessons Learned," in *Engineering Secure Future Internet Services and Systems*: Springer International Publishing., 2014, pp. 35-64.

[8] M. S. Lund, B. Solhaug, and K. Stølen, "Risk analysis of changing and evolving systems using CORAS," in *Foundations of security analysis and design VI*: Springer, 2011, pp. 231-274.

[9] Eurocontrol, " ATM Strategy for the Years 2000+," *Brussels, Belgium,* vol. vol. I and II., 2003.

[10] SESAR, "SESAR definition phase D3: The ATM target concept," *Eurocontrol, Brussels, Belgium, Tech. Rep,* vol. 1550, pp. 0612-001, 2007.

[11] A. M. Grubb and M. Chechik, "Looking into the crystal ball: requirements evolution over time," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 2016, pp. 86-95: IEEE.

[12] S. Schneider, J. Wollersheim, H. Krcmar, and A. Sunyaev, "How do requirements evolve over time? A case study investigating the role of context and experiences in the evolution of enterprise software requirements," *Journal of Information Technology,* vol. 33, no. 2, pp. 151-170, 2018.

[13] A. Ferrari, P. Spoletini, and S. Debnath, "How do requirements evolve during elicitation? An empirical study combining interviews and app store analysis," *Requirements Engineering,* vol. 27, no. 4, pp. 489-519, 2022.

[14] A. Russo, B. Nuseibeh, and J. Kramer, "Restructuring requirements specifications," in *Software, IEEE Proceedings-(Vol. 146, No. 1, pp. 44-53). IET.*, 1999, vol. 146, pp. 44-53: IET.

[15] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "Controlling requirements evolution: a formal concept analysis-based approach," in *International Conference on Software Engineering Advances, 2007. ICSEA 2007 (pp. 68-68). IEEE.*, 2007, pp. 68-68: IEEE.

[16] I. Côté and M. Heisel, "A UML Profile and Tool Support for Evolutionary Requirements Engineering," in *15th European Conference on Software Maintenance and Reengineering (CSMR) (pp. 161-170).* , 2011, pp. 161-170: IEEE.

[17] G. Bergmann, F. Massacci, F. Paci, T. T. Tun, D. Varró, and Y. Yu, "SeCMER: a tool to gain control of security requirements evolution," in *Towards a Service-Based Internet*: Springer Berlin Heidelberg., 2011, pp. 321-322.

[18] S. Saito, Y. Iimura, H. Tashiro, A. K. Massey, and A. I. Antón, "Visualizing the effects of requirements evolution," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 152-161.

[19] A. Akbar, S. Mahmood, Z. Huang, A. Khan, and M. Shameem, "Readiness model for requirements change management in global software development," *Journal of Software: Evolution Process,* vol. 32, no. 10, p. e2264, 2020.

[20] F. Mehmood and S. Zulfqar, "Effect of Human Related Factors on Requirements Change Management in Offshore Software Development Outsourcing: A theoretical framework," *Soft Computing Machine Intelligence,* vol. 1, no. 1, pp. 36-52, 2021.

[21] L. M. S. Tran, "Managing the Uncertainty of the Evolution of Requirements Models," PhD thesis, University of Trento, 2023.

[22] D. Zowghi and R. Offen, "A logical framework for modeling and reasoning about the evolution of requirements," in *Proceedings of the Third IEEE International Symposium on Requirements Engineering, 1997 (pp. 247-257). IEEE.*, 1997, pp. 247-257: IEEE.

[23] J. Brier, L. Rapanotti, and J. G. Hall, "Problem-based analysis of organizational change: a real-world example," in *Proceedings of the 2006 international workshop on Advances and applications of problem frames (pp. 13-18). ACM.*, 2006, pp. 13-18: ACM.

[24] N. Ernst, A. Borgida, and I. Jureta, "Finding incremental solutions for evolving requirements," in *Requirements Engineering Conference (RE), 2011 19th IEEE International (pp. 15-24). IEEE.*, 2011, pp. 15-24: IEEE.

[25] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, "System identification for adaptive software systems: a requirements engineering perspective," in *Conceptual Modeling–ER 2011*: Springer Berlin Heidelberg., 2011, pp. 346-361.

[26] V. E. S. Souza, A. Lapouchnian, K. Angelopoulos, and J. Mylopoulos, "Requirements-driven software evolution," *Computer Science-Research and Development*, vol. 28, no. 4, pp. 311-329, 2013.

[27] L. Zhao and A. J. F. I. Zhao, "Sentiment analysis based requirement evolution prediction," vol. 11, no. 2, p. 52, 2019.

[28] S. Lim, A. Henriksson, and J. J. S. C. S. Zdravkovic, "Data-Driven Requirements Elicitation: A Systematic Literature Review," vol. 2, no. 1, pp. 1-35, 2021.

[29] H. J. P. C. S. Kaiya, "Evaluating Mutual Requirements Evolution of Several Information Systems," vol. 176, pp. 1251-1260, 2020.

[30] A. Grubb and M. Chechik, "Formal reasoning for analyzing goal models that evolve over time," *Requirements Engineering*, vol. 26, no. 3, pp. 423-457, 2021.

[31] N. v. d. Berg, "From Idea to Product: Requirements Evolution within Software Projects," (Master thesis), Utrecht University, 2023.

[32] S. Alwidian, D. Amyot, and Y. Lamo, "Union Models for Model Families: Efficient Reasoning over Space and Time," *Algorithms*, vol. 16, no. 2, p. 105, 2023.