A Tool for Supporting the Co-Evolution of Enterprise Architecture Meta-models and Models

Nuno Miguel Silva *nuno.miguel@tecnico.ulisboa.pt*
Universidade de Lisboa/INOV Inesc Inovação
Lisboa, Portugal

Miguel Mira da Silva *mms@tecnico.ulisboa.pt*
Universidade de Lisboa/INOV Inesc Inovação
Lisboa, Portugal

Pedro Sousa *pedro.manuel.sousa@tecnico.ulisboa.pt*
Universidade de Lisboa/Link Consulting SA
Lisboa, Portugal

Abstract

Enterprise architecture models capture the concepts and relationships that together describe the essentials of the various enterprise domains. This model of the enterprise is tightly coupled to a domain-specific modeling language that defines the formalisms for creating and updating such model. These languages are described as meta-models by the model-driven engineering field. Results from surveys on enterprise architecture tool analysis showed a lack of support concerning the co-evolution of enterprise architecture meta-model and models. This paper presents a tool that automates enterprise architecture models co-evolution according to a set of meta-model changes. A Portuguese governmental organization used and validated the tool using observational, analytical and descriptive evaluation methods.

Keywords: Enterprise architecture, Meta-model, Model, Co-Evolution, Tool.

1. Introduction

A model captures the concepts and relationships within a given domain. The Enterprise Architecture (EA) field uses models to express, through diagrammatic descriptions, the various organizational domains, from the business domain to the information systems and the information technology domains. The structure and semantics of these models must obey a set of formalisms defined by a domain-specific modeling language. These languages, known as meta-models by the model-driven engineering community [5, 6, 10, 11, 20], tend to continuously evolve due to new modeling requirements that enforce changes to these meta-models [9].

In the EA context, meta-models and models are also the targets of the evolutionary pressure with meta-model changes often occurring due to the iterative nature of the meta-model and model construction process [9]. A typical reason for these changes has to do with the EA modeling language no longer being able to express the stakeholders' needs in its models. In this case, the evolution of a meta-model drives the call for addressing those needs and changing requirements. The evolution of the meta-model is not always adding new elements. Often involves specializing, redefining or revising existing elements. One of the risks associated with this redefinition is naturally dependent on the existing model and reports, as they may no longer reflect the needs of the different stakeholders. Thus, the evolution of the EA meta-model, when there is already an existing repository containing the EA model, reports, and analysis made of such model, is always complex because typically it is intended on the one hand to maintain compatibility with the old model, while on the other to support a more expressive model.

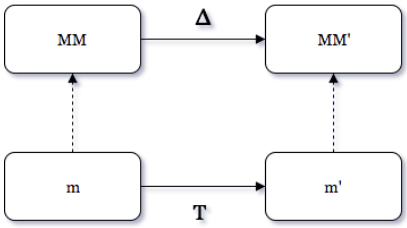

Fig. 1. MM-M co-evolution.

As referred above, changes on the meta-model have a high probability of impacting all models that conform to it. Meta-model-Model (MM-M) co-evolution occurs when an element from the meta-model changes and the model no longer conforms to the new meta-model, as Figure 1 illustrates. After performing an evolution Δ of a meta-model MM into MM', the goal is to co-evolve model m that conforms to MM, to m' that conforms to MM', by applying a set of model transformations T aligned with evolution Δ [5, 9].

Co-evolution of models is strictly related to the notion of information preservation [20] from which additive, subtractive, and refactoring meta-model changes are distinguished. Therefore, changes that occur on a meta-model may have different effects on the related models. These changes are classified as follows [5, 6]: *Non-breaking changes* - This type of changes occurring on the meta-model does not break model conformance to the corresponding meta-model; *Breaking and resolvable changes* - This type of changes occurring on the meta-model break model conformance to the corresponding meta-model, however, they can be automatically resolved; and *Breaking and unresolvable changes* - This type of changes occurring on the meta-model break model conformance to the corresponding meta-model and cannot be automatically resolved, therefore human intervention is required.

In model-driven engineering approaches, the meta-models MM and MM' are two different versions of the meta-model. When replacing MM with MM', model m must be transformed into m', otherwise MM-M conformance ceases to be valid. The authors theorize a different approach. Instead of being two different versions, MM and MM' are two states in the life-cycle of a single meta-model. Meta-model elements can have one of three states in their life-cycle: "conceived", "alive", or "dead". The born date property establishes the transition from "conceived" to "alive" and the death date property sets the transition from "alive" to "dead". In these conditions, the evolution from state MM to state MM' at a time instant *t*, consists of marking *t* as the death date of elements that no longer exist in MM' and as the born date of the elements that did not appear in MM' but do exist in MM.

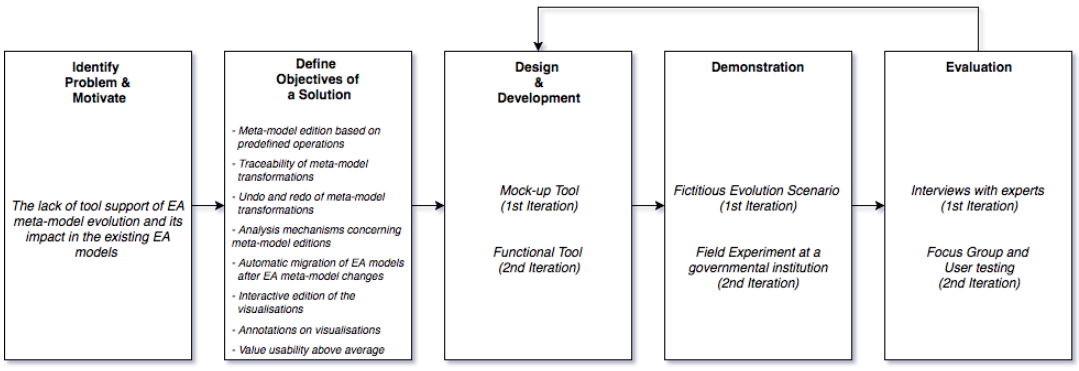

Fig. 1. DSR method steps applied to this research – adapted from [15].

In this paper the authors present a functional version of a tool prototype that fully automates EA model migration when breaking and resolvable meta-model changes occur, complementing preliminary descriptions of this tool [16]. A set of operations adapted from the meta-model breaking and resolvable changes presented in [5, 6, 12, 18, 20] enable the automation. In the presented tool, MM and MM' are two different life-cycle states of the same meta-model against

which all existing models must conform. Hence, models that are not yet conformant with the MM' state can still exist and be used in the repository. The Design Science Research (DSR) method was used throughout this research as shown in Figure 1.

The remainder of the paper is structured as follows. Section 2 describes the current EA tool support and limitations regarding MM-M co-evolution. Then, Section 3 presents the requirements and a detailed description of the EA MM-M co-evolution tool. Afterward, Section 4 describes a performed field experiment within a Portuguese governmental organization to demonstrate the tool in practice with Section 5 presenting the results of the evaluation phase. Finally, Section 6 presents related work on the topic and Section 7 concludes the paper and enumerates future efforts.

2. EA Tool Meta-Model Flexibility and MM-M Co-Evolution Support

EA tools provide model visualizations and support strategic decisions concerning the enterprise. Existing solutions restrict the means of extending and refactoring the EA meta-model, to commit data flexibility and data management, thus simplifying the process of managing data visualization and reporting.

Table 1. Analyzed EA tools with a focus on meta-model flexibility and MM-M co-evolution.

Company	Tool	Meta-model Flexibility	MM-M Co-Evolution
Niche Players			
Erwin	Erwin EA	YES	NO
Unicom Systems	System Architect	YES	NO
BOC Group	ADOIT	NO (ArchiMate)	NO
Visionaries			
Planview	Troux	YES	NO
Challengers			
Sparx Systems	Enterprise Architect	YES	NO
Orbus Software	iServer	YES	NO
Leaders			
BIZZdesign	Enterprise Studio	YES	NO
Avolution	Abacus	YES	NO
QualiWare	QualiWare EA	YES	NO
MEGA	Hopex EA	YES	NO
Software AG	ALFABET	YES	NO
Other			
Obeo	SmartEA	YES	NO
EA Composer	WhiteCloud Software	YES	NO

This approach has the downside of forcing the organization's structure to fit the EA meta-model and not the opposite. Gartner's magic quadrant for EA tools [3] also pinpoints as a minimum requirement of an EA tool a *robust yet flexible repository and metamodel(s) that support often-changing relationships between objects within and between multiple viewpoints or architectures, as well as capturing temporal relationships and changes*. Nonetheless, the reviews of the vendor's EA solutions lack a detailed explanation of the tool's capabilities concerning meta-model evolution and management [3].

Table 1 shows the results of an EA tool survey with a focus on two metrics: meta-model flexibility and MM-M co-evolution features. The considered tools constitute Gartner's magic quadrant for EA tools plus other two meta-model-based EA tools. The authors planned the conduction of the survey in two parts. The first part consisted of an extensive read of the available free online documentation regarding each tool's features with emphasis on meta-model customization, extensibility, and MM-M co-evolution. The second part of the survey was to use available demo versions of each tool to corroborate the available meta-model extensibility and customization features, as well as MM-M co-evolution features. Results show that all EA tools surveyed do provide meta-model customization and extensibility to give architects control over the meta-model they use, and the ability to customize and adjust such

meta-model as the business and IT requirements evolve. Nonetheless, to the best of the authors' knowledge and according to the surveyed information, none of the analyzed tools provide MM-M co-evolution features together with their meta-model customization and extensibility features.

3. The EA Meta-model and Model Co-Evolution Tool

This section describes relevant aspects of the developed tool prototype, starting from requirements gathered from surveys and interviews with experts. Section 3.1 enumerates each requirement as well as the rationale behind the definition of each requirement, followed by a description of the tool (Section 3.2).

3.1. Requirements

Two surveys [13, 14] and two rounds of seven interviews with both practitioners and researchers were the foundation in which the following tool requirements were defined (objectives of a solution in DSR – see Figure 1):

1. A solution should allow meta-model edition based on predefined operations;
2. A solution should allow traceability of metamodel transformations, hence maintaining a meta-model change history;
3. A solution should allow the undo and redo of meta-model transformations;
4. A solution should provide analysis mechanisms concerning meta-model editions, namely the impact of altering a meta-model element on other existing elements;
5. A solution should allow automatic migration of EA models after EA meta-model changes;
6. A solution should enable interactive edition of the visualizations;
7. A solution should allow annotations on visualizations to add semantic;
8. A solution should have value usability above average (Bangor et al. 2008b).

To understand the current state of progression regarding EA transformation support, analysis, and management within the EA tool's industry, the authors gathered information from two tool surveys [13, 14]. In the first analysis, nine EA tools were evaluated by three different teams [13]. The evaluation process was based on the analysis of functional criteria and EA task criteria. For each criterion, different test scenarios were created, complemented by online questionnaires. The findings were then processed, presented, and rated for each task. The rating scale range from 0 to 7, being seven the highest score possible. Later the same analysis was complemented, using the same evaluation process, with four additional EA tools [14]. Regarding the EA meta-model, the functional criteria assessed the meta-model's flexibility. The analysis showed that four tools scored above five, incorporating a solution to manage the meta-model. These data binds to the types of tools analyzed in [13], categorized according to flexibility and guidance: meta-model driven, methodology driven, and process driven.

Meta-model driven tools allow the user to change the underlying meta-model, giving more flexibility and semantics to the models. On the other hand, process driven tools focus on the process and guidance concerning the management of EA instead of flexibility, thus not allowing the edition of the underlying meta-model. Nevertheless, even though most EA tools showed concerns regarding the meta-model flexibility, any considerations towards impact analysis of EA models, migration support, and visualizations affected by those changes were mentioned. The main points and issues that the authors took from this analysis are as follows: lack of interaction with EA artifact visualizations; annotating visualizations is relevant since it provides more semantic to the model's visualizations; and the need for manual changes to EA model visualizations to access changes to the EA metamodel. These points suggest the need for automatic migration of models (Requirement 5) and for annotating the visualizations with more semantic to better share information (Requirement 7). There is also the need for more interaction with EA artifacts visualizations, allowing to create a more interactive modeling task (Requirement 6).

To support the definition of the solution's requirements, we performed two rounds of interviews with four practitioners and three researchers. The practitioners had 2 to 15 years of EA experience in both public institutions and private companies. The researchers with expertise in the modeling field had 2 to 5 years of experience in modeling and participation on European projects, with focus on EA.

Results gathered from both rounds suggested that organizations evolve the EA meta-model and co-evolve the existing EA model in an iterative basis, usually done by a team, with a team element or client being responsible for validating the meta-model. In this case, visualization and traceability of all changes are required to share information between different members (Requirement 2).

When meta-modelling, the users need to perform impact analysis to understand which elements are impacted by a particular change (Requirement 4), allowing the roll-back of actions if some changes need to be undone (Requirement 3). One of the main aspects also pointed by the interviewees was a description of typical meta-modeling errors: wrong relations between meta-model elements; properties with wrong names; and elements with wrong properties. These typical errors reinforce the need for a tool capable of providing proper edition support regarding meta-model relations, properties, or elements (Requirement 1), in a straightforward and user-friendly manner (Requirement 8).

3.2. Tool Description

The tool prototype features a set of co-evolution operations that alter the meta-model component's life-cycle. The developed tool is agnostic to the meta-model specificities, i.e., is not configured to fully process a specific EA modeling language, such as ArchiMate, UML, and BPMN. The meta-model elements processed by the tool are classes, relations, and relation types expressed in a graph-based structure. This design option allows for more language flexibility and broader coverage at the cost of more language-specific expressiveness, meaning that not all characteristics of a specific EA modeling language are accounted. The authors leave this assessment for future reference.

The authors implemented seven co-evolution operations adapted from [18]: *Create EA Class, Rename EA Class, Create EA Relation, Change EA Relation Type, Move EA Relation, Delete EA Class and Delete EA Relation*. Table 2 shows an example of a co-evolution operation. This operation changes the life-cycle state of a meta-model class from "alive" to "dead" and then propagates that same change to all model elements in the EA repository that are instances of that class. The implementation details and correctness of each co-evolution operation can be found in [18].

Table 2. Delete EA Class Specification.

Params	`name: String, deathDate: Date, retirementDate: Date`
Pre-conditions	`this.gestationDate != null && this.birthDate != null && this.deathDate = null && this.retirementDate = null`
Post-conditions	`this.gestationDate != null && this.birthDate != null && this.deathDate != null && this.retirementDate != null`

Statements	1. `this.deathDate -> deathDate`
	2. `this.retirementDate -> retirementDate`
	3. `forAll(instance | this.getInstances())`
	`  instance.deathDate -> deathDate`
	`  instance.retirementDate -> retirementDate`
	4. `forAll(relation | this.getRelations())`
	`  Delete_EA_Relation(deathDate, retirementDate, this)`

Different meta-model evolution scenarios can be created and tested by creating multiple projects using the same EA Repository; meaning each project can then apply different sets of operations to the same EA meta-model. After executing the operations, the user should choose which project to apply to the EA Repository in use. The tool's user interface is composed of two main screens:

- **Projects Screen.** This screen depicts a list of all projects created by the user. This screen provides feedback to the user concerning the state of the project. The possible states are: **Not Submitted -** no editions submitted for approval; **Submitted -** meaning that the project was submitted for approval; **Accepted/Rejected -** if approved, the state of the project becomes "Accepted", otherwise "Rejected"; and **Sent -** when accepted, the user can then send both the EA meta-model and model updates to the EA tool's repository integrated with the tool.

- **Edition Screen.** Allows for the creation of a new project ("New project"), opening a specific project ("Open project"), or the submission of a working project for approval ("Submit"). These actions can be performed by clicking on the respective buttons on the top-left corner of the screen. Changes to the meta-model can be performed as follows: On the screen's left side, one can apply three different operations: create/edit/delete relation types. Each operation allows the user to create the relation types that are best suited to each scenario. On the center of the screen, an interactive view illustrates a comparison between the *as-is* meta-model state, i.e., before the execution of the operations, and the *to-be* meta-model state, i.e., after applying the co-evolution operations. The view is dynamic, meaning that the addition or removal of operations will update the viewpoint, thus displaying the impact of new operations in real time. Here it is also possible to switch the view to a graph-based visualization (see Figure 2) of the EA meta-model; however, this visualization is not interactive like the previous one. On the screen's right side, an activity list of all performed operations is shown as an edition history, aggregating all the operations made by the user. This activity list is shown as a tree, where inside each operation is possible to observe all the sub-operations made.

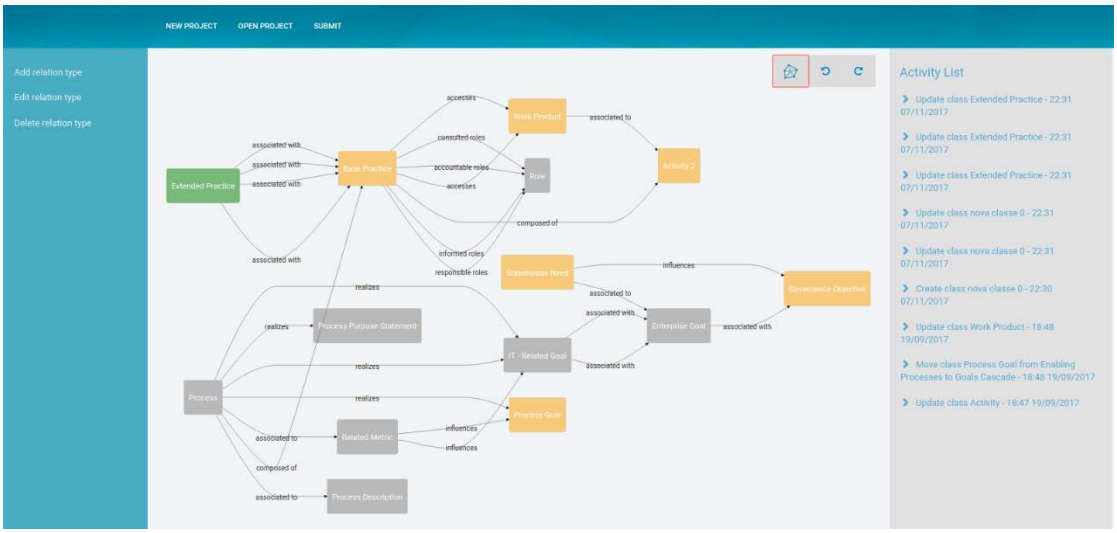

Fig. 2. Tool's graph-based view of the enterprise architecture meta-model.

The tool also implements a transactional feature composed of two operations: *undo* and *redo*. Both operations refer to each co-evolution operation stored in a persistent activity list. The logic is analogous to word-processing or spreadsheets editors undo and redo features. When operating the EA meta-model, that operation is stored in the activity list using a *first-in-first-out* approach. For example, if the user wants to undo the last three operations, (s)he could do so by clicking three times on the undo button on the upper-right side of the system's user interface. The undo feature will roll back the last operation made by the user and then update the activity list accordingly (see Figure 3). If the operation is composed of more than one change, all changes in that operation are also undone. The same rationale applies to the inverse (redo) operation. By clicking on the redo button (next to the undo button), the user can roll back

to the previous action made. So, assuming the user only wanted to undo a single operation instead of two operations, (s)he can redo the last undo operation.

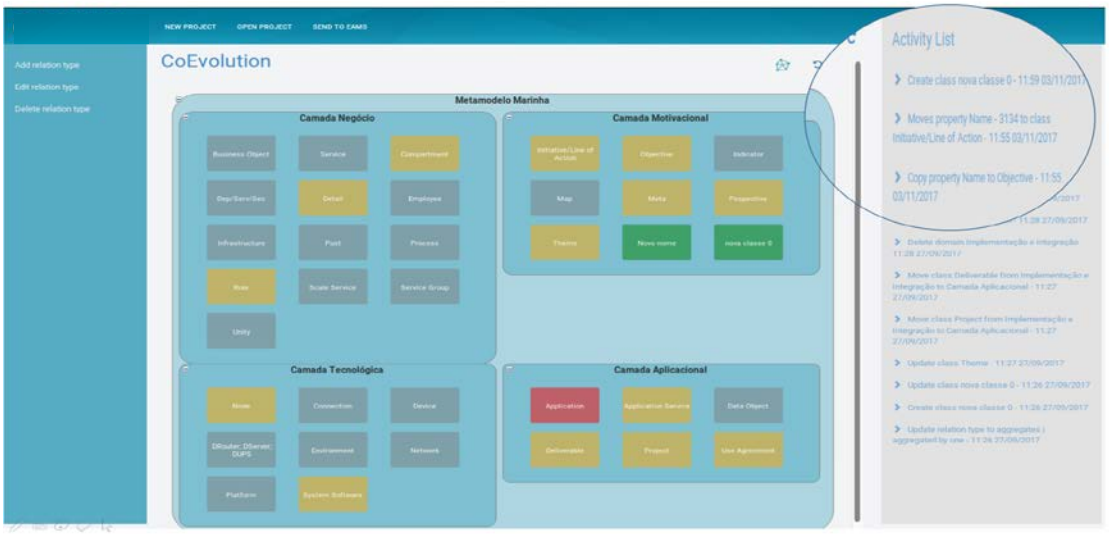

Fig. 3. Tool's view with emphasis on the activity list presenting the next three co-evolution operations to undo.

The element's color follows the scheme defined in [17], where different colors represent different life-cycle states of the meta-model elements.

4. Demonstration

The authors applied the developed tool inside a Portuguese governmental organization with the purpose of assessing its effectiveness in addressing the research problem. The organization recognizes that EA can benefit its enterprise transformation initiatives. However, the numerous EA projects conducted by the organization have created architectural silos that hinder overall analysis, thus diminishing the value of EA. An EA project was conducted with the collaboration of the organization's EA team to support the organization towards the consolidation of its architecture. The project aimed to achieve two primary goals:

- Evolving the organization's EA proprietary meta-model, which at the time was scattered into three different meta-models, by unifying all elements and relationships into one single meta-model while extending it with any missing information.
- Perform an impact analysis in the existing EA models before migration to estimate the migration effort.

The followed approach consisted of dividing the evolution task into two stages. The first stage was composed of two steps: 1) mapping the common elements of the three meta-models, and 2) creating a single meta-model containing all the necessary elements and relationships presented in each one, as well as extending the new meta-model. The concept mapping allowed us to understand the overlapping elements of all three meta-models and from there apply the list of co-evolution operations required to co-evolve both the meta-model and existing models. Table 3, on the second column, contains all the operations applied to the meta-model (and models) on stage one.

Table 3. Number of co-evolution operations used on each stage.

Co-Evolution Operations	#	
	First Stage	**Second Stage**
Create EA Class	6	7
Rename EA Class	8	10
Create EA Relation	14	59
Delete EA Relation	16	3
Change EA Relation Type	1	1
Number of co-evolution operations executed (per type): 5 Number of co-evolution operations executed: 125 Number of failed co-evolution operation executions: 0		

The second stage consisted of further extending and refactoring the meta-model while performing a model impact analysis based on the updated version of the EA meta-model. Table 3, on the third column, contains all the operations applied to the meta-model and models on stage two.

The meta-model was based on a subset of the ArchiMate 3.0 meta-model. The initial meta-model consisted of 18 classes (8 from Technology Layer, 4 from Application Layer, 4 from Business Layer, one from Implementation and Migration Layer, and Location), 81 relations, and 14 relation types. At the end of stage two, the final meta-model had 31 classes (8 from Technology Layer, 4 from Application Layer, 12 from Business Layer, 2 from Implementation and Migration Layer, 4 from Motivation Layer, and Location), 135 relations, and 16 relation types.

The application of the developed tool allowed the EA team to discuss and test the different implications of the meta-model operations to the existing models without potentially compromising the EA repository. In the end, the co-evolution rules applied to the EA repository resulted in the migration of 287 model elements and 537 model relations.

5. Evaluation

In DSR, artifact validation concerning the objectives of a solution is done by applying design evaluation methods. The authors applied the following evaluation methods: *Observational* – EA project inside a governmental organization; *Analytical* – dynamic Analysis with user tests (scholars and practitioners); and *Descriptive* – two focus groups; one with practitioners and another with the organization's EA team (Section 4), as well as, interviews with 10 EA practitioners.

5.1. Observational

A field experiment is typically done with the goal to empirically examine an intervention in the real world rather than in the laboratory. The EA Project presented in Section 4 provided a real experimental case to validate the tool's applicability and efficacy towards solving the research problem.

5.2. Analytical

The authors applied user testing as an analytical method to assess the tool's performance and usability. A sample of 20 EA academic and four practitioner subjects were used, each one performing a total of 12 tasks. The user tasks incorporated the use of different types of operations, as well as exploring the tool by using the supported information visualization features. Afterward, the subjects were asked to answer a survey regarding the tool's usability. The survey consisted of 10 questions using a scale of 1 to 5 (1 meaning that the subject strongly disagrees and five meaning he/she strongly agrees with the statement) and one more question to evaluate the overall user perception concerning the tool's usability.

The survey was based on [4] questionnaire with the addition of a seven-point adjective-anchored Likert scale question from [2]. From the entire user sample, 75% of users gave a score

above 80 points which correspond to a B grade in the SUS survey [1]. The analysis of each question concluded that: 87.5% of users (strongly) agree that they would like to use the tool; 83.3% of users (strongly) disagree that the tool is unnecessarily complicated; 79.1% of users (strongly) favor the tool as easy to use; 91.6% of users (strongly) disagree they would need the support of a technical person to be able to use this tool; 87.5% of users (strongly) agree on the various functions in the tool were well integrated; 95.8% of users (strongly) disagree there was too much inconsistency on the tool; 87.5% users (strongly) agree with that most people would learn to use the tool quickly; 83.3% of users (strongly) disagree the tool is very cumbersome to use; 70.8% of users (strongly) agree they fell very confident using the tool; 91.6% of users (strongly) disagree they would need to learn many things before they could get going with the tool; 75% of the users rated the user-friendliness of the tool as "Excellent" or "Best Imaginable".

5.3. Descriptive

Two focus groups and interviews with practitioners were also conducted to evaluate the effectiveness of the tool. The first group was composed of five elements of the organization's EA team (see Section 4) while the second group included three EA practitioners with six, ten, and fifteen years of experience respectively. Results from the first focus group suggested that future improvements should focus on the tool's impact analysis feature. The impact analysis should be more visually comprehensive regarding the impact of deleting a relation and a class in the meta-model to the model. Moreover, the group also suggested that would be interesting to understand the reason behind a particular model change (or set of changes), such as a form of the rationale of what is impacting what. Despite those issues, the group agreed on the tool's effectiveness in supporting the meta-model and model co-evolution by providing a straightforward and interactive edition of the meta-model elements, a comprehensive visualization of the meta-model changes, and a mechanism for automating model migration.

The second focus group also mentioned the tool's lack of a more thorough impact analysis feature. In particular, it was not intuitive for each practitioner to understand which architectural elements were indeed impacted from a holistic multi-artifact perspective; although, consensus was also reached regarding the tool's effectiveness in addressing the co-evolution problem by enabling a safe modeling environment with minimal modelling effort and no errors which could potentially lead to architectural inconsistencies. Moreover, they found visualization support for meta-model changes to be helpful in supporting meta-model changes over time.

The authors conducted interview sessions with ten different EA practitioners from different companies and organizations, from consultancy and audit to telecommunications companies. The tool was presented to each practitioner followed by a discussion of the tool's features and how they adequately address the research problem and comply with the defined objectives of a solution. Eight out of ten practitioners acknowledged the practical relevance of the tool in addressing the co-evolution problem. Table 4 presents both the pros and cons of the system according to the interviews' results.

Table 4. Tool's pros and cons identified during the interviews and focus groups.

Pros	Cons
• The activity list, working as a log of all transformations and changes made. • Good user interaction. • Diagram with the meta-model relations. • Workflow with administration review, thus reducing the probability of errors. • Number of impacted objects when deleting a class.	• Lack of a more descriptive model impact analysis. • Visualization of impacted objects. • Not keeping a standard color pattern associated with class changes to all elements. • Inability to resume and quantify changes. • Unable to filter by class' neighbors or domain in the meta-model relationship view.

6. Related Work

Model-driven engineering approaches have addressed the MM-M co-evolution challenge with a focus on two strategies [5]: identifying the differences between the baseline and target meta-models or applying a set of transformations on the model to be conformant to the new meta-model. The first strategy uses a declarative evolution specification to define a difference meta-model which can be calculated from identified changes in the meta-model [5] whereas the second approach specifies meta-model evolution and model co-evolution through a sequence of operations in which each operation is then applied on meta-model and model level [6, 12].

One approach to model co-evolution classifies the changes in atomic changes and defines the process of co-evolution [5]. Then, creates a differential meta-model with the identified changes, and it is classified into two new meta-models, the ones that are breaking and resolvable and the ones that are breaking and unresolvable. If there are no relations between the two meta-models, each one is executed independently. If relations between the two meta-models do exist, the co-evolution is done stepwise using user intervention.

COPE is a language used to satisfy two requirements: 1) reuse of recurring migration knowledge and 2) expressiveness to support domain-specific migrations. COPE allows creating coupled transactions that are a combination of meta-model adaptation and model migration [12]. COPE solves the co-evolution problem by executing those coupled transactions, and its execution does not require user intervention.

Another approach addressed the co-evolution problem by applying a set of automatic transformations thus solving the problem in one of three categories: addition, delete or rename [10]. When a breaking and unresolvable change is found, the user should specify the way that the elements are going to change by creating a set of transformation rules using ETL.

As stated in Section 1, model-driven engineering approaches on MM-M co-evolution, like the ones described above, are built on the premise that MM and MM' are two different meta-models. The developed tool, however, deals with MM-M co-evolution taking into account the life-cycle principle. In this case, MM and MM' are both the same meta-model but with different life-cycle states regarding each element and relation. The use of life-cycle of artifacts to deal with the evolution of EA models was initially developed in [19], allowing the minimization of the necessary effort to maintain EA models. In the tool prototype developed, the same technique applies to the elements of the meta-model.

In the EA context, Florez et al. [9] developed a platform capable of addressing breaking and unresolvable changes in model co-evolution sustained on two hypothesis: i) a meta-modeler that knows the rationale behind meta-model changes and capable of providing guidelines for model co-evolution, ii) a modeler as the only one allowed to make the final decisions about his models. Their proposed language for meta-modelers allows to specify changes in the meta-models and to propose corresponding changes in the models, which in turn are executed by an engine that automatically solves the changes in models that can be automatically solved.

Farwick et al. stated, based on a survey among EA practitioners, that despite the necessity of manual data input as the primary source of changes to EA repositories, the next step to increase productivity, efficiency, and ROI of EA initiatives can be reached by automating the maintenance processes for EA models, with minimal human intervention [8]. The results of the survey reinforced the idea of using automated model maintenance as a means to reduce EA modeling efforts, thus contributing to meet better desired EA element coverages. As a step in that direction, they have proposed several processes for (semi-)automated enterprise architecture model maintenance, based on the requirements mentioned above and research efforts on Living Models, that reduce manual work for EA model maintenance and increase data quality attributes such as consistency and actuality [7].

7. Conclusion

In this paper, the authors presented a tool prototype that supports EA MM-M co-evolution by achieving the eight requirements presented in Section 3.1. The tool prototype was applied in a Portuguese governmental organization to support the conduction of an EA project and validated with observational, analytical, and descriptive design evaluation methods. The analytical evaluation results from SUS questionnaire reported a B score, which regarding usability validates a system as having good usability according to [1], hence achieving requirement eight presented in Section 3.1. The demonstration of the proposed artifact (Section 4), together with the descriptive evaluation methods (Section 5.3) validate the achievement of requirements 1, 2, 3, 5, and 6. Requirement four could not be adequately achieved since the impact analysis feature did not provide all the required model information comprehensively and visually. The application of a color-based code to annotate the semantic of changes to meta-model elements, based on the work from [17], enabled the achievement of requirement seven.

Regarding identified limitations, according to the feedback gathered from the interviews and focus groups, the over-simplistic way in which the impacted model elements are displayed when evolving the EA meta-model posed a challenge in understanding the impact of meta-model changes to the model. A more comprehensive model impact analysis should be presented concerning the EA model elements. Although this issue was already considered before the interviews and focus groups, the implementation of a detailed impact analysis had to be postponed, due to time restrictions.

In conclusion, the tool's evaluation confirmed the achievement of the solution's requirements by enabling an EA MM-M co-evolution interactive and safe modeling environment, in which co-evolution operations could be executed meta-model-wise, and then propagated to the EA model. Future efforts consist of resolving the limitations stated by the focus groups and interviews with a focus on extending the impact analysis expressiveness and also complementing the tool with other relevant co-evolution operations from literature applied to the EA context, such as Merge Class and Split Class.

Acknowledgments

The authors wish to thank the ISD reviewers for their valuable remarks. This research was supported by the Link Consulting's project IT-Atlas (nº 11419, under the IAPMEI, 2020 Portuguese PO CI Operational Program). The integration of the tool prototype presented here with Link Consulting's Atlas EA tool is ongoing.

References

1. Bangor, A., Kortum, P., Miller, J.: An empirical evaluation of the system usability scale. International Journal of Human-Computer Interaction. 24, 6, 574–594 (2008).
2. Bangor, A., Kortum, P., Miller, J.: Determining what individual SUS scores mean: Adding an adjective rating scale. Journal of Usability Studies. 4, 3, 114–123 (2009).
3. Brand, S.: Magic Quadrant for Enterprise Architecture Tools. Gart. RAS Core Res. Note G. September 2014, 43 (2014).
4. Brooke, J.: System Usability Scale (SUS): A Quick-and-Dirty Method of System Evaluation User Information. Usability Evaluation In Industry. pp. 4–7 (1996).
5. Cicchetti, A., Di Ruscio, D., Eramo, R., Pierantonio, A.: Automating co-evolution in model-driven engineering. In: Proceedings of the 12th IEEE International Enterprise Distributed Object Computing Conference. pp. 222–231 (2008).
6. Cicchetti, A., Di Ruscio, D., Eramo, R., Pierantonio, A.: Meta-model differences for supporting model co-evolution. In: Proceedings of the 2nd Workshop on Model-Driven Software Evolution. (2008).
7. Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., Hanschke, I.: Automation processes for enterprise architecture management. In: Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Workshop. pp. 340–349 (2011).

8. Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., Hanschke, I.: Requirements for automated Enterprise Architecture Model Maintenance. In: Proceedings of the 13th International Conference on Enterprise Information Systems. pp. 325–337 (2011).
9. Florez, H., Sánchez, M., Villalobos, J., Vega, G.: Coevolution assistance for enterprise architecture models. In: Proceedings of the 6th International Workshop on Models and Evolution. pp. 27–32 (2012).
10. Gruschko, B., Kolovos, D., Paige, R.: Towards synchronizing models with evolving metamodels. In: Proceedings of the International Workshop on Model-Driven Software Evolution. p. 3 (2007).
11. Herrmannsdoerfer, M., Vermolen, S., Wachsmuth, G.: An extensive catalog of operators for the coupled evolution of metamodels and models. In: International Conference on Software Language Engineering. pp. 163–182 (2011).
12. Herrmannsdoerfer, M., Benz, S., Juergens, E.: COPE - automating coupled evolution of metamodels and models. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 52–76. Springer, Heidelberg (2009).
13. Matthes, F., Buckl, S., Leitel, J., Schweda, C.M.: Enterprise Architecture Management Tool Survey 2008. (2008).
14. Matthes, F., Buckl, S., Leitel, J., Schweda, C.M.: Enterprise Architecture Management Tool Survey 2014. (2014).
15. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems. 24, 3, 45–77 (2007).
16. Rechau, T., Silva, N., da Silva, M. M., Sousa, P.: A tool for managing the evolution of enterprise architecture meta-model and models. In: Lecture Notes in Business Information Processing. pp. 68–81. Springer, Cham (2017).
17. Roth, S., Matthes, F.: Visualizing Differences of Enterprise Architecture Models. In: International Workshop on Comparison and Versioning of Software Models at Software Engineering. (2014).
18. Silva, N., Ferreira, F., Sousa, P., da Silva, M. M.: Automating the Migration of Enterprise Architecture Models. International Journal of Information System Modeling and Design. 7, 2, 72–90 (2016).
19. Sousa P. et, al, Lima J., Sampaio A., Pereira C.: An Approach for Creating and Managing Enterprise Blueprints: A Case for IT Blueprints. In: Lecture Notes in Business Information Processing. pp. 70-84. Springer, Heidelberg (2009).
20. Wachsmuth, G.: Metamodel adaptation and model co-adaptation. In: The European Conference on Object-Oriented Programming. pp. 600–624 (2007).