A Continuous Delivery Pipeline for EA model Evolution

Simon Hacks[1], Andreas Steffens[1], Peter Hansen[2], and Nikhitha Rajashekar[2]

[1] Research Group Software Construction, RWTH Aachen University, Aachen, Germany
{hacks,steffens}@swc.rwth-aachen.de
[2] RWTH Aachen University, Aachen, Germany
{peter.hansen,nikhita.rajashekar}@rwth-aachen.de

Abstract. The pace of changing structures and complexity within enterprise architecture (EA) models is expected to increase. This will challenge existing maintenance processes of EA models. To tackle this challenge, we propose to adapt the well-known concept of continuous delivery (CD) from the agile software development domain. We propose to automate the necessary steps to ensure EA model quality by applying multiple validation and analysis steps. Therefore, this results shorter feedback loops and helps to uncover possible conflict as early as possible.

Keywords: EA model Evolution · Continuous Delivery · EA model Maintenance

1 Introduction

Since it beginnings in the 1980's [25], Enterprise Architecture (EA) has developed to an established discipline [32, 34]. The ISO 42010:2011 defines architecture as the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution [20]. As this definition implies, the EA model, comprised by the organizations elements and relationships, is a central artifact of EA. Additionally, EA has to provide important and up-to-date information of the organization to its stakeholders.

There are a many different sources for changes of the EA model [9], which contribute to a continuous evolution of the EA model. As our research assumes a project-driven environment, we will refer to projects as example for the main source of changes. However, our approach sticks not solely to projects, which can also be replaced by teams working in a more agile environment.

EA models are currently mostly modeled manually and changes require notable manual efforts. This is especially true when complex organizational structures need to be covered and the organization is constantly changing. The pace of changing structures and complexity is expected to increase and this makes it even more challenging [40]. In recent years, the field of enterprise architecture management already adopted techniques to reduce model maintenance effort.

However, there are still challenges in regards to conflicting changes, different semantics and responsibilities [8].

In the field of software engineering, changing requirements are also very common. Software engineering deals with this by becoming as agile as possible and uses various social and technical techniques to improve towards this direction [14].

Examples for social techniques are the ongoing adoption of agile process models like scrum or kanban and even techniques directly related to the development itself like pair programming. Technical examples are the rise of continuous integration and delivery. All of these techniques lead to the same shared goal: Shorten feedback loops [17]. Techniques used for software engineering are also being adopted for other parts of organizations: With the DevOps movement, which emphasizes on the collaboration of development and operations, infrastructure is being covered using techniques typically used in the context of software engineering and processes are also adopted [6].

To overcome the aforementioned problems of EA modeling, we proposed already an architecture roundtrip process [13]. However, this process is still abstract and needs to be instantiated. To do so, we facilitate the well-known technique of continuous delivery (CD) and realize the architecture roundtrip process. Accordingly, we formulate our research question:

Can a continuous delivery help to overcome the challenges of manual EA model's maintenance?

So far, existing research on EA model maintenance automatizing has focused either on collecting information from different external sources (e.g. [4, 18]), trying to bring contradictory information together (e.g. [22, 38]), or proposing an overall process for maintenance (e.g. [9, 13]). To the best of our knowledge, there is no research around trying to adapt the technique of CD to the domain of EA model maintenance. Our results contribute to the existing body of knowledge by enhancing the proposed processes with the benefits of CD and offering new possibilities to connect further sources of information to the central EA model.

In the rest of this paper, we will elaborate on this question. First, we will present work related to automatic maintenance/evolution of EA models. Second, we sketch our research design, before we give insights into the design and implementation of our pipeline. Next, we demonstrate our pipeline by a fictitious example and discuss the findings of the experiment. Last, we conclude our work and give an impression of future research.

2 Related Work

EA is used in large organizations and different departments often own information, which is used within the EA. This makes it hard for a central enterprise architecture team to gather all information and keep them up-to-date. Fischer et al. proposed a semi-automated federated approach for the maintenance of EA models [10]. The main idea is that the data is kept within specialized architectures and linked to a central EA repository.

Other approaches to automatize EA model maintenance are presented e.g. by Buschle et al. [4], who facilitate an ESB (Enterprise Service Bus) to extract EA models automatically. In contrast, Holm et al. [18] concentrate more on technically observable components as they map the output of a network scanner to ArchiMate. An extension of this work is presented by Johnson et al. [22], who incorporate uncertainty into the mapping. The work of Välja et al. [38, 37] focuses on uniting different information from contradictory sources. Hence, they try to estimate the trustworthiness of the sources.

EA related research did not only elaborate solely on the technical aspects of EA model maintenance. For example, Kirschner and Roth [24] rely on a human component to solve arising conflicts from different sources. Further, Khosroshahi et al. [23] investigated the social factors influencing the success of federated EA model maintenance. A slightly different point of view is taken by Hauder et al. [15] as they focused on the challenges of a federated EA model maintenance.

Further related research to our work can be identified in the field of continuous delivery. Humble and Farley [19] define continuous delivery as a set of practices, which enables to speed-up, automate and optimize the delivery of software artifacts to the customer with higher quality and lower risks in a continuous manner. Continuous delivery uses an automated development infrastructure, called deployment pipeline, which automates nearly every step of the delivery process. Each commit of a developer enters the deployment pipeline and an automated process is started, which produces a new software increment as a result artifact.

The deployment pipeline incorporates all activities known from continuous integration [7] as automatic build, unit testing, and static code analysis. In addition to these, the pipeline performs testing activities like integration, performance, and security testing. All these tasks are executed in a defined order of stages. After each stage, the test results are evaluated at a quality gate, which stops the processing if the quality conditions are not met. If all quality gates are passed, the software artifact is stored and can be accessed and used from external clients; it is released.

In recent research many challenges of adopting continuous delivery have been found [26, 5, 28] and coping with software evolution and heterogeneity can be identified as the major technical obstacles for a continuous delivery system. To overcome many of these obstacles, we proposed a generalized model and architecture for a new generation of continuous delivery systems [35].

Lastly, our process relies heavily on the quality of the EA model. Regarding to ISO/IEC 25010 quality "is the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use" [21]. In the context of EA research Ylimäki states that "a high-quality EA conforms to the agreed and fully understood business requirements, fits for its purpose [...] and satisfies the key stakeholder groups' [...] expectations" [42, p. 30]. In general, research regarding EA quality agrees that it is defined by the ability to meet the EA users' requirements [30, 27]. Most of the related work divides quality aspects

of EA into the quality of EA products, its related services, and EA processes [30, 27].

In the discipline of enterprise modeling there are approaches that discuss model quality in general, without focusing on a certain modeling structure. Becker et al. [2] define six principles that have to be considered when assessing an enterprise model's quality. Sandkuhl et al. [33] apply these principles to evaluate the quality of their modeling language 4EM and further depict concrete quality attributes.

3 Research Design

Design science research (DSR) is a widely applied and accepted means for developing artifacts in information systems (IS) research. It offers a systematic structure for developing artifacts, such as constructs, models, methods, or instantiations [16]. As our research question indicates the development of means, the application of a DSR is appropriate. We stick to the approach of Peffers et al. [31], since it transpired as effective in former research. It is split up into six single steps and two possible feedback loops:

- **Identify Problem & Motivate:** As previous research has shown, reasons to change the EA model are manifold [9] and raise many different challenges [15]. One of them is to handle different sources and another to design a suitable process for EA model maintenance. We believe that the principle of continuous delivery offer efficient means to support the EA model maintenance process.
- **Define Objectives:** Based on our research problem stated before, we identified mainly three sources for objectives: First, Farwick et al. [8] identified a set of 23 requirements on automated EA model maintenance grouped into categories like architectural, organizational, or data quality. Those requirements should be incorporated into a feasible solution. Second, Fischer et al. [10] describe an EA model maintenance process comprised by activities that mainly are related to data collection, quality checks, and delivery of the new information. Additionally, Fischer et al. define four roles, which are either related to process coordination (EA Coordinator, EA Repository Manager), data delivery (Data Owner), or quality checks (EA Coordinator, EA Stakeholder). Last, we presented a process for a distributed EA model evolution [13] describing different tasks and their sequence focusing on a continuous evolution of the EA model.
- **Design & Development:** To realize an artifact in accordance to the beforehand identified objectives, first, we align the input of the three objectives' sources. Then, we design an abstract process model using Business Process Model and Notation (BPMN) [39] and implement it using JARVIS [35]. Our derived integrated EA maintenance process consists of activities, which will be implemented as microservices following JARVIS's architectural framework. In addition to the activities defined in our objectives, we include

additional steps inspired by principles found in the continuous delivery domain.
- **Demonstration:** The demonstration is put into practice by applying the proposed means to a single fictitious case study. Single case studies gain a first, in-depth reflection on means in real life scenarios [41]. Moreover, single case studies are a feasible instrument to show applicability. Our case study is based on an EA model illustrating an airport. Within this case study, we show that a CD pipeline can reduce the manual effort in EA model maintenance.
- **Evaluation:** We identified 54 equivalence classes of possible actions, which should be considered in our pipeline. Therefore, we created for each class an exemplary test case as a representative for this class [3, p. 623].
- **Communication:** The communication is done with this paper itself and its presentation on a conference.

4 A Pipeline for EA model Evolution

Following, we will sketch our pipeline for an EA model maintenance. Fischer et al. [10] contribute two main findings to our pipeline. First, they propose an EA model maintenance process, which we unite with our work from [13]. Second, they offer a fine-grained role concept, which we incorporate in the pipeline as well.

To implement our deployment pipeline for EA model maintenance, we opt for our prototype JARVIS [35]. It allows integrating the proposed processes into a deployment pipeline and we create a BPMN version of the process as JARVIS is equipped to use BPMN as a modeling language. From this model, we derive the necessary activities, which needs to be implemented as microservices. During this, we transform the process model to reflect better the principles of JARVIS and continuous delivery in general. Figure 1 shows the resulting model for EA model maintenance.

The first process steps from Fischer et al. and Hacks et al. of initializing and collecting the necessary data of the EA model evolution can be omitted. We assume, that in an environment following the principles of continuous delivery from Humble et al. [19] all artifacts like the global and the special EA models are under version control and stored in an appropriate system like subversion or git. Each change to one of these models by the responsibles within in projects needs to be committed to the repository. A change is resulting in a new version of the model. Whenever a change is committed to the repository for the special architecture our deployment pipeline is triggered automatically.

The technical infrastructure of the SCM and the deployment pipeline ensure the automatic processing of the first process steps of both proposed maintenance processes. Necessary notifications can be sent by the system if we need to be compliant to the overall process, but effectively we want the stakeholders only to be involved if really necessary.

The pipeline starts by first checking out the new models versions from the repository and provide both to the first transformation activity. This activity is

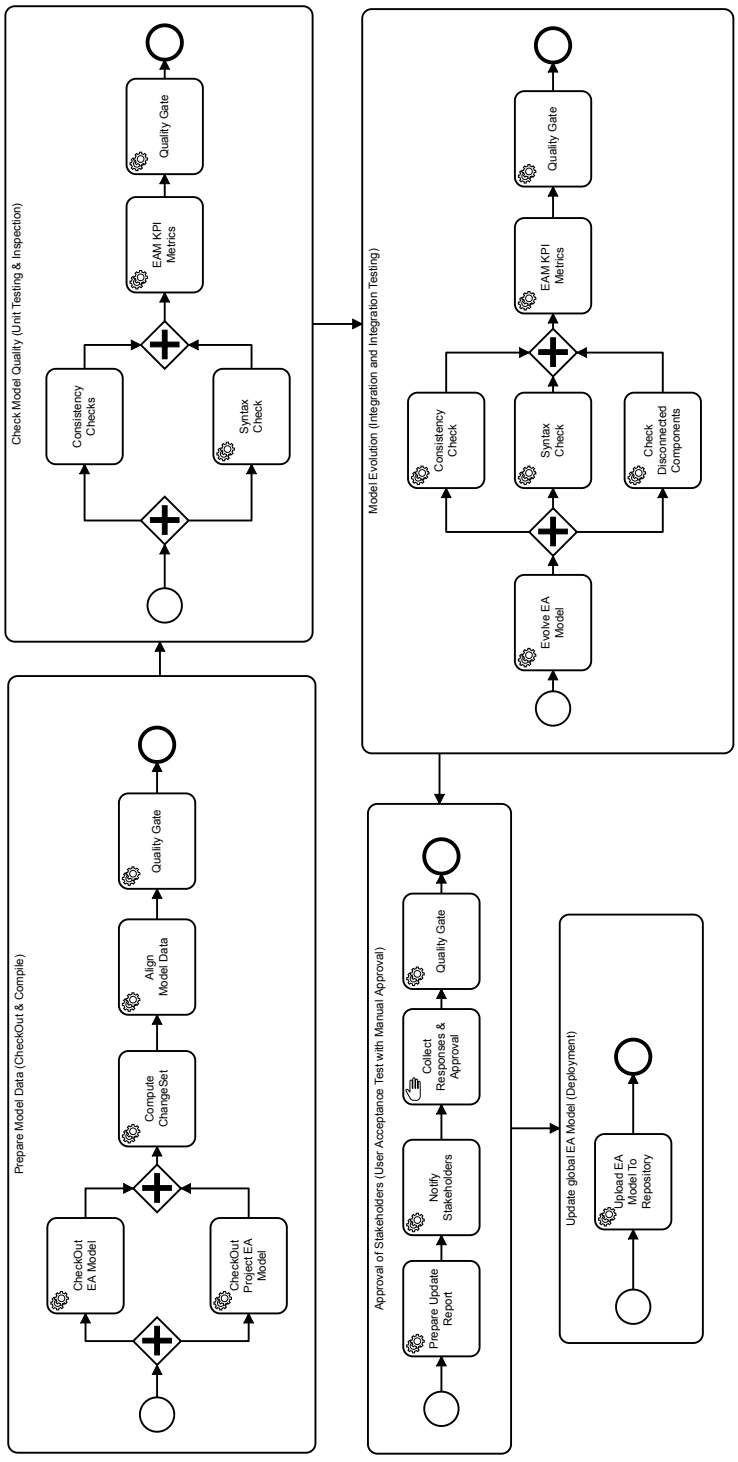

Fig. 1. EA Maintenance Deployment Pipeline as BPMN

called "Compute Change Set" and uses the provided input models to compute the existing deviations between both and provide these as a new artifact called "Change Set".

All existing artifact are now processed in the next transformation activity "Align Model Data". Hacks and Lichter [13] argue that a specific project may contain more detailed information than the more general global EA model. Therefore, the model provided by the project has to be aligned in order to be effectively compare- and merge-able to the central model. This includes a necessary meta-model transformation as well as an adaption of the provided model to the same level of detail presented in the central model. The following quality gates check the successful execution of the proceeded activities and the existence of the three artifacts. Afterwards, the first stage of our deployment pipeline is finished. This stage corresponds to the checkout and compile stages in classic software delivery pipelines.

Fischer et al. and Hacks and Lichter both incorporate steps to check the model quality like consistency or correctness of syntax. In our pipeline, we model these as assessments, which are performed on the model singular artifacts of the proceeding stage and which produce a report for each assessment. This stage corresponds to static analysis for software source code. Paul Duval et al. [7] incorporate an inspection phase into his continuous integration model in which relevant metrics for software quality are measured and evaluated. We adopt this by applying well-known EAM KPIs [29] to models inside the pipeline.

In the next stage, the artifacts are integrated to produce a new and updated candidate for the EA model by reproducing the changes made by the project on the central EA model. This candidate is then examined by the same assessments as before. The modular architecture allows us to integrate even more sophisticated assessments, which can be performed on EA models. We integrated a check for disconnected components, which checks if parts of the resulting EA model candidate has components, which are not connected to the rest of the model. Based on the assessment reports the quality gate decide if the pipeline should continue to the next stage where the candidate is presented to the stakeholders of the overall process.

Up to this point the pipeline is performing its tasks completely autonomous, so the stakeholder are only involved if the model candidate has reached a certain degree of quality due to the assessments performed before. The manual approval of the stakeholders corresponds to the User Acceptance Test (UAT) stage in classic pipelines. Bass et al. [1] define the UAT stage as the last one before going to production and are meant to ensure these aspects of the delivery process which cannot be automated.

If this stage is successfully executed, the EA model candidate is promoted to the final stage where it is deployed to the EA model repository. The next run of the pipeline will use this new version of the EA model and so the roundtrip is completed.

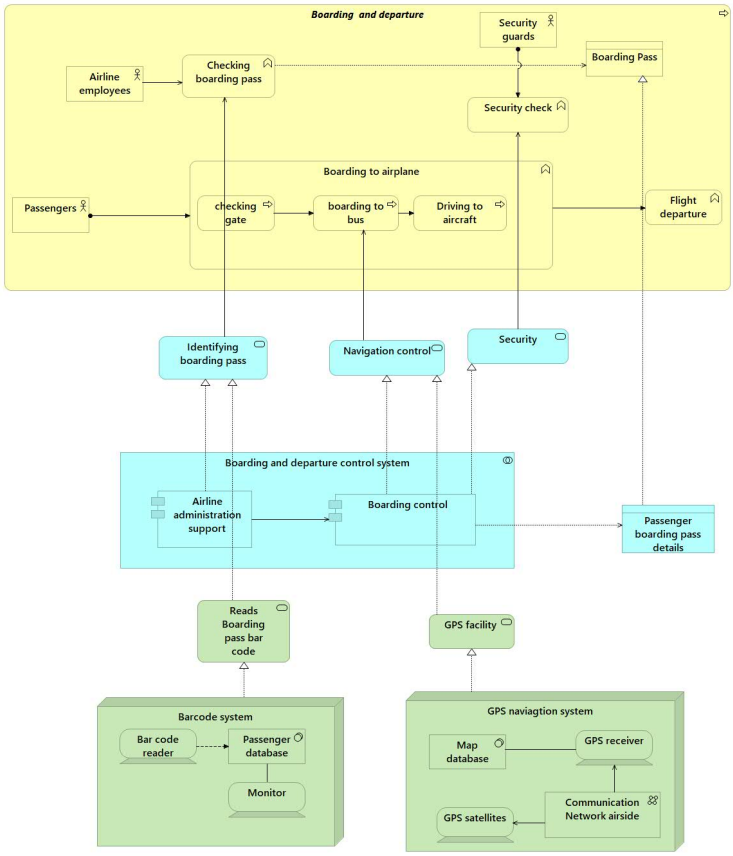

Fig. 2. Excerpt of the boarding and departure process

5 Demonstration

5.1 Exemplary EA model

To demonstrate and evaluate our artifact, we conduct a fictitious case study. Therefore, we facilitate the example of an airport departure system. This example was originally developed to illustrate the realistic use of ML and graph analytic methods in the context of analyzing EA models. Following, we illustrate a scenario of airport departure system, which depicts the functionality of the passengers before boarding to an aircraft.

This example is modelled as an EA model based on ArchiMate 3.0.1 [36]. The model incorporates all the ArchiMate layers beginning with business, application, and technology architectures. It consists of 171 different elements and 250 relations. We will further discuss in detail the core layers of the system. An excerpt of the model is presented in Figure 2.

The business layer depicts the business services offered to the customers, which is mainly used to build business architecture [12]. In this example, the active entities of the business layer are airline employees, passengers, and security guards. Four functions are provided by boarding and departure process. The function boarding to airplane is internally divided into three sub-processes.

The application layer includes for example the airline administration support, which is responsible for handling check-in process, and the boarding control, which handles boarding process. Furthermore, two application components collaborate in boarding and departure control system, i.e., the airline administration component and the boarding control to provide application level services like identifying boarding pass, security and navigation control support.

The technology layer offers several components to the application layer. E.g., there is a barcode system offering the needed means to validate the barcodes of the boarding tickets and a GPS navigation system guiding the bus drivers to the right plane on the airfield.

5.2 Facilitated Metrics

To simulate the "Check Model Quality" step of the pipeline, we check the EA model against KPIs from the EAM KPI Catalog [29]. As we do not want to implement all KPIs of the catalog, we randomly chose three of them as representatives for all KPIs. Those KPIs are only exemplary and can be replaced by any other calculable metrics. Nonetheless, we have to keep in mind that it can be quite challenging to assess the necessary input parameters (e.g., if interviews have to be conducted).

PM guideline adherence checks if IT projects adhere to the stated PM guideline [29, p. 28]. As the information model of the KPI catalog is not directly reflected in ArchiMate, we identify work package as an IT project and business object with a property PMguideline as PM guideline. To compute this KPI, the project managers, first, answer the degree the project adheres to every guideline. Second, we compute the average for every project along all guideliness. The catalog defines three categories of adherence. If a project adheres to 100% to the guideline it is full adherence. Between 100% and 75% it is a minor deviation which will cause a warning in our pipeline. With less than 75% it is a major deviation causing a fail of the pipeline.

Application continuity plan availability [29, p. 19] measures the degree how completely IT continuity plans for business critical applications have been drawn and tested for the IT's application portfolio. To reflect the information model in ArchiMate, we map the application to application component and continuity plan to business object with the property ContinuityPlan. The responsible for the operation of the applications answer if there exists a continuity plan for a certain application and if it is tested. The KPI is then computed by the number of critical applications where a tested continuity plan is available divided by the total number of critical applications. The value is good above 80%. Normally, the value will between 60% and 80% resuming in a warning to related stakeholders. If the value drops beyond 60% the value is problematic and the pipeline fails.

IT process standard adherence [29, p. 33] checks if a certain application (application component in ArchiMate) adheres to the IT standard processes (application process in ArchiMate). This is answered by the process responsible and then calculated by the number of applications, which adhere to an IT standard process, divided by the total number of applications. The value is good at 100%. Normally, the value will between 80% and 100% resuming in a warning to related stakeholders. If the value drops beyond 80% the value is bad and the pipeline fails.

Besides the EAM KPI Catalog, we check also the *connectivity* of the graph representing the EA model, where the elements of the EA model represent nodes within the graph and their connections represent edges. A graph is connected if there is a path between every pair of nodes. We assume that the model of an EA should be always connected. If the model contains isolated elements or sub-graphs, there are parts in the organization, which are not related to the other parts. So to say, there are parts in the EA pursuing different goals and, therefore, different organizations within the organization. Nevertheless, for two organizations there would be two EAs. Consequently, we expect the model of the EA to be connected. Otherwise, the pipeline should fail.

5.3 Implementation of the Pipeline

Our designed pipeline for EA model maintenance was implemented for the continuous software delivery system JARVIS [35] and each activity in the pipeline model was implemented as an independent microservice following the architectural framework of JARVIS. From JARVIS, we reused the complete infrastructure and general activities like the git checkout activity and the quality gate activity.

6 Evaluation

To evaluate our pipeline, we conduct an equivalence class test [3, p. 623] where the values of the metrics serve as input parameters (three respectively two classes per KPI) and the behavior of the pipeline (i.e., successful, warning, and fail) represents the output. To test the pipeline, we combine each possible input class and determine the expected output for each combination resulting in 54 test cases. We choose always the worst expected output (fail<warning<successful) if different outputs would be possible. An extract of four exemplary test cases is presented in Table 1.

For the execution, the presented example model from section 5.1 was stored in a git repository and used as the global EA model. A variant of the model was stored in a second repository and was used as the repository for a simulated specific project model. This variant was altered for each test case to represent the different test cases and resulting behavior of the pipeline is checked against its expected behavior.

Table 1. Exemplary test cases

Test case ID	Input				Expected output
	PM guideline adherence	Application continuity plan availability	IT process standard adherence	Connected	
1	100%	100%	100%	TRUE	fail
6	100%	70%	100%	TRUE	warning
7	100%	70%	90%	TRUE	warning
8	100%	50%	100%	TRUE	success

The execution of the test cases by triggering the pipeline with different inputs showed that our approach is feasible. The expected behavior of our pipeline could be observed. In case of a failing pipeline the execution always stopped at the first KPI assessment in the Model Quality stage. The reason for this is, that we already test the KPIs on each model in this stage, so the assessment of the project model results in a fail. By deactivating this assessment, the pipeline performs the Model Execution stage and fails at this point. Both behaviors are correct. Due to this phenomenon we recognize, that our pipeline is already performing a simpler inspection process for the project mode, it is embedded in the global EA model maintenance process.

7 Discussion

Before, we presented our pipeline and its application on a fictitious example. Our results show that the existing approaches are missing certain steps, which we incorporated into our pipeline. For example, our roundtrip process lacks a step for an evaluation of EA KPIs, which are represented in the Model Quality stage and in the Model Evolution stage of our pipeline. As the KPIs can be easily computed and automatically evaluated, we can naturally apply it inside in a continuous delivery pipeline. Besides, it has to be mentioned that the calculation of a KPI is only easy as long as the basic measures are provided, which can be quite challenging.

Furthermore, the pipeline incorporates a simple inspection process of the project model, which is presented by its own independent pipeline and is executed during the project solution development. This leads to a similar result as with Continuous Integration and Continuous Delivery. Continuous Delivery can be seen as an extension of Continuous Integration as Fowler argued [11]. The project pipeline would only consider the single project model as our maintenance process also considers the global EA model and an EA model candidate, which integrates changes from the project model into the EA model.

In addition, the roundtrip approach lacks the incremental and iterative nature of an agile development process. The project solution delivers its model only one time to the maintenance process. With incorporating continuous delivery,

the project can deliver the changed model every time to the overall maintenance process. Therefore, the project will get feedback on the compatibility with the global EA model earlier and can adopt to this feedback more easily. The deviations between global and specific model are therefore minimized.

On the other hand, changes to the EA model are much earlier distributed to other projects in the organization, as there maintenance process will use the adapted EA model also for other active projects. So the deviations between the various projects are minimized. In result, the automation of the maintenance process may lead to more relevant EA model, which represents the current state of the organization and its enterprise architecture in a much more accurate way. Furthermore, the whole process is completely transparent and most important traceable, which supports further requirements regarding compliance and security.

The process of Fischer et al. [10] lacks the roundtrip approach. As we count on short feedback cycles as typical for agile development, we overcome this shortcoming. In addition, our proposed means reduces the involvement of stakeholders and the necessary manual work to a minimum. Stakeholders only assess EA model candidates, which has achieved a certain degree of quality.

Lastly, we introduced a new metric to measure the connectivity of the EA model represented by a graph. For our case study, we assume that the complete graph needs to be connected. However, depending on the needs of the organization under observation multiple connected components are desired. Another organization's need could be for a metric to assess the certain degree of connectivity for the whole EA or its sub-graphs. As our case study is only fictitious, it does not offer further insights into these aspects and need to be investigated in future research.

8 Conclusion

EA models are currently mostly modeled manually and changes require huge manual efforts. This is especially true when complex organizational structures need to be covered and the organization is constantly changing. The pace of changing structures and complexity is expected to increase and this makes it even more challenging [40]. In recent years, the field of enterprise architecture management already adopted techniques to reduce model maintenance effort. We contribute to this field of research by adapting the means of continuous delivery to shorten feedback cycles and providing a higher degree of automatizing.

To do so, we facilitated existing EA model maintenance processes and implemented them within our tool JARVIS. Our first evaluation shows that existing maintenance processes benefit from the ideas of the agile domain leading from a model maintenance to a model evolution perspective. Additionally, we could show that the interaction between stakeholder and enterprise architects can be further reduced. Consequently, both can concentrate more on the essential parts of EA than on technically related issues.

However, our research includes still some limitations. First, we were not able to test our approach in a natural environment. Such a field evaluation may raise additional issues, especially related to the influence of our approach on the sociological environment. So far, we focused only on technical aspects, but internal resistance might hinder our approach.

Second, we just took a single project as data provider for our pipeline into account. A plenty of distributed data provider might cause issues, we did not consider thus far. In particular, we encourage short feedback cycles, which might cause problems as well if the mindset of the involved employees is missing.

Third, today most EA models are maintained in a central EA model tool, which apply version control mainly internally. To apply our approach to those environments, the tools need to provide an interface providing model information for interaction with our pipeline. However, this needs a change of thinking at EA tool providers from a single, closed tool to an integrated tool, which is part of a bigger environment.

Fourth, we took a very technical view on the problem. For instance, we assumed for simplicity reasons that the needed input for the KPIs we facilitate for our quality gate can be computed easily. However, the assessing of certain inputs for the KPIs can be quite challenging, which needs to be further evaluated in future research. Additionally, there might be not only one perception of a KPI as multiple stakeholders with a diverse background and possibly different expertise and expectations contribute to its assessment and interpretation, which has to be taken into account.

References

1. Bass, L., Weber, I., Zhu, L.: DevOps: A Software Architect's Perspective. Addison-Wesley Professional, 1st edn. (2015)
2. Becker, J., Probandt, W., Vering, O.: Grundsätze ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement. BPM kompetent, Springer Berlin Heidelberg, Berlin Heidelberg (2012)
3. Burnstein, I.: Practical software testing: a process-oriented approach. Springer Science & Business Media (2006)
4. Buschle, M., Ekstedt, M., Grunow, S., Hauder, M., Matthes, F., Roth, S.: Automating enterprise architecture documentation using an enterprise service bus. 18th Americas Conference on Information Systems (2012)
5. Chen, L.: Continuous Delivery: Overcoming adoption challenges. Journal of Systems and Software **128**, 72–86 (jun 2017)
6. Debois, P.: Agile Infrastructure and Operations: How Infra-gile are You? Agile 2008 Conference pp. 202–207 (2008)
7. Duvall, P., Matyas, S.M., Glover, A.: Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series). Addison-Wesley Professional (2007)
8. Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., Hanschke, I.: Requirements for Automated Enterprise Architecture Model Maintenance - A Requirements Analysis based on a Literature Review and an Exploratory Survey. In: ICEIS (2011)

9. Farwick, M., Schweda, C.M., Breu, R., Voges, K., Hanschke, I.: On Enterprise Architecture Change Events. In: Aier, S., Ekstedt, M., Matthes, F., Proper, E., Sanz, J.L. (eds.) Trends in Enterprise Architecture Research and Practice. pp. 129–145. Springer, Berlin, Heidelberg (2012)
10. Fischer, R., Aier, S., Winter, R.: A Federated Approach to Enterprise Architecture Model Maintenance. In: EMISA (2007)
11. Fowler, M.: Continous integration (2006), `http://martinfowler.com/articles/continuousIntegration.html`
12. Guild, B.A.: A Guide to the Business Architecture Body of Knowledge (BIZBOK Guide), vol. V04 (2014)
13. Hacks, S., Lichter, H.: Towards an Enterprise Architecture Model Evolution. In: Czarnecki, C., Sultanow, E., Brockmann, C. (eds.) Workshops der Informatik 2018. Lecture Notes in Informatics, Gesellschaft für Informatik e.V, Bonn (2018)
14. Hanssen, G.K., Smite, D., Moe, N.B.: Signs of agile trends in global software engineering research: A tertiary study. In: 2011 IEEE Sixth International Conference on Global Software Engineering Workshop. pp. 17–23 (Aug 2011)
15. Hauder, M., Matthes, F., Roth, S.: Challenges for Automated Enterprise Architecture Documentation. In: TEAR/PRET (2012)
16. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly **28**(1), 75–105 (2004)
17. Highsmith, J., Cockburn, A.: Agile Software Development: The Business of Innovation. IEEE Computer **34**, 120–122 (2001)
18. Holm, H., Buschle, M., Lagerström, R., Ekstedt, M.: Automatic data collection for enterprise architecture models. Software & Systems Modeling **13**(2), 825–841 (2014)
19. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional, 1st edn. (2010)
20. ISO, IEC, IEEE: Systems and software engineering – Architecture description (01122011)
21. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, ISO/IEC, vol. 25010. ISO, Geneva (2011)
22. Johnson, P., Ekstedt, M., Lagerström, R.: Automatic Probabilistic Enterprise IT Architecture Modeling: A Dynamic Bayesian Networks Approach. In: Franke, U., Lapalme, J., Johnson, P. (eds.) 20th International Enterprise Distributed Object Computing Workshop (EDOCW). pp. 123–129 (2016)
23. Khosroshahi, P.A., Aier, S., Hauder, M., Roth, S., Matthes, F., Winter, R.: Success Factors for Federated Enterprise Architecture Model Management. In: Persson, A., Stirna, J. (eds.) Advanced Information Systems Engineering Workshops, pp. 413–425. Lecture Notes in Business Information Processing, Springer International Publishing (2015)
24. Kirschner, B., Roth, S.: Federated Enterprise Architecture Model Management: Collaborative Model Merging for Repositories with Loosely Coupled Schema and Data. In: Multikonferenz Wirtschaftsinformatik 2014 (2014)
25. Kotusev, S.: The History of Enterprise Architecture: An Evidence-Based Review. Journal of Enterprise Architecture **12**(1), 31–37 (2016)
26. Laukkanen, E., Itkonen, J., Lassenius, C.: Problems, causes and solutions when adopting continuous deliveryA systematic literature review. Information and Software Technology **82**, 55–79 (feb 2017)

27. Lim, N., Lee, T.g., Park, S.g.: A Comparative Analysis of Enterprise Architecture Frameworks Based on EA Quality Attributes. 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing pp. 283–288 (2009)
28. Marín, P.R., Haghighatkhah, A., Lwakatare, L.E., Teppola, S., Suomalainen, T., Eskeli, J., Karvonen, T., Kuvaja, P., Verner, J.M., Oivo, M.: Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software **123**, 263–291 (2017)
29. Matthes, F., Monahov, I., Schneider, A.W., Schulz, C.: EAM KPI Catalog v1.0. Garching (2011)
30. Niemi, E., Pekkola, S.: Enterprise Architecture Quality Attributes: A Case Study. In: 2013 46th Hawaii International Conference on System Sciences. pp. 3878–3887. IEEE (2013)
31. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems **24**(3), 45–77 (2007)
32. Saint-Louis, P., Lapalme, J.: Investigation of the lack of common understanding in the discipline of enterprise architecture: A systematic mapping study. In: Franke, U., Lapalme, J., Johnson, P. (eds.) 20th International Enterprise Distributed Object Computing Workshop (EDOCW) (2016)
33. Sandkuhl, K., Stirna, J., Persson, A., Wißotzki, M.: Enterprise modeling. Tackling Business Challenges with the 4EM Method. Springer **309** (2014)
34. Simon, D., Fischbach, K., Schoder, D.: An Exploration of Enterprise Architecture Research. Communications of the Association for Information Systems **32**(1), 1–72 (2013)
35. Steffens, A., Lichter, H., Doring, J.S.: Designing a next-generation continuous software delivery system: Concepts and architecture. In: 2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE). pp. 1–7 (May 2018)
36. The Open Group: ArchiMate 3.0.1 Specification (2017)
37. Välja, M., Korman, M., Lagerström, R., Franke, U., Ekstedt, M.: Automated architecture modeling for enterprise technology manageme using principles from data fusion: A security analysis case. In: Portland International Conference on Management of Engineering and Technology (PICMET) (2016)
38. Välja, M., Lagerström, R., Ekstedt, M., Korman, M.: A Requirements Based Approach for Automating Enterprise IT Architecture Modeling Using Multiple Data Sources. In: 19th International Enterprise Distributed Object Computing Workshop (2015)
39. White, S.A.: BPMN modeling and reference guide: understanding and using BPMN. Future Strategies Inc. (2008)
40. Winter, K., Buckl, S., Matthes, F., Schweda, C.M.: Investigating the state-of-the-art in enterprise architecture management method in literature and practice. In: 5th Mediterranean Conference on Information Systems. AIS (2010)
41. Yin, R.K.: Case Study Research: Design and Methods. Sage Publications, Thousand Oaks and London and New Delhi, 5 edn. (2013)
42. Ylimäki, T.: Potential Critical Success Factors for Enterprise Architecture. Journal of Enterprise Architecture **2**(4), 29–40 (2006)