Implementation of a continuous delivery pipeline for enterprise architecture model evolution

Alex R. Sabau¹ · Simon Hacks² · Andreas Steffens¹

Received: 30 September 2019 / Revised: 19 July 2020 / Accepted: 2 September 2020 / Published online: 21 September 2020
© The Author(s) 2020

Abstract

The discipline of enterprise architecture (EA) is an established approach to model and manage the interaction of business processes and IT in an organization. Thereby, the EA model as a central artifact of EA is subject to a continuous evolution caused by multiple sources of changes. The continuous evolution requires a lot of effort in controlling and managing the evolution of the EA model. This is especially true when merging the induced changes from different sources in the EA model. Additionally, the lack of tool and automation support makes this a very time-consuming and error-prone task. The evolutionary character and the automated quality assessment of artifacts is a well-known challenge in the software development domain as well. To meet these challenges, the discipline of continuous delivery (CD) has emerged to be very useful. The evolution of EA model artifacts shows similarities to the evolution of software artifacts. Therefore, we leveraged practices of CD to practices of EA maintenance. Thus, we created a conceptual framework for automated EA model maintenance. The concepts were realized in a first prototype and were evaluated in a fictitious case study against equivalence classes based on EA model metrics and a set of several requirements for automated EA model maintenance from research. Overall, the concepts prove to be a promising basis for further refinement, implementation, and evaluation in research in an industrial context.

Keywords Enterprise architecture model evolution · Continuous delivery · Enterprise architecture model maintenance

1 Introduction

Since its beginnings in the 1980s [40], Enterprise Architecture (EA) has developed to an established discipline [54,57]. 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” [33]. As this definition implies, the EA model, comprised by the organization’s 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 many different sources for changes of the EA model [18], 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. For example, the projects can be replaced by teams that work 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 [67]. In recent years, the field of EA already adopted techniques to reduce model maintenance efforts. However, there are still challenges regarding conflicting changes, different semantics, and responsibilities [17].

In the field of software engineering, similar challenges are recognized. Applications are developed simultaneously of different developers situated in different locations. The

Communicated by Jelena Zdravkovic and Iris Reinhartz-Berger.

✉ Simon Hacks
shacks@kth.se

Alex R. Sabau
alex.sabau@swc.rwth-aachen.de

Andreas Steffens
steffens@swc.rwth-aachen.de

¹ Research Group Software Construction, RWTH Aachen University, Aachen, Germany

² Division of Network and Systems Engineering, KTH Royal Institute of Technology, Stockholm, Sweden

business, that needs to be reflected in the application, evolves continuously and with increasing acceleration. Accordingly, requirements change often and need to be implemented faster. The established means of waterfall-like processes and monoliths are not able anymore to serve these growing demands. Software engineering deals with this by becoming as agile as possible and uses various social and technical techniques to improve toward this direction [27].

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 these techniques lead to the same shared goal: Shorten feedback loops [30]. Techniques used for software engineering are also being adopted for other parts of organizations: With the DevOps movement, which emphasizes 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 [12].

These techniques became popular in industry and research, and the made experiences show that they can help to overcome the postulated challenges. As one can recognize, the challenges in software engineering and EA modeling are similar. Consequently, we assume that those established means of software engineering might be also helpful in the domain of EA modeling.

To overcome the problems of EA modeling, we proposed already an architecture roundtrip process [24]. As this process is still abstract and needs to be instantiated, we presented a concrete implementation based on the well-known technique of continuous delivery (CD) [26]. However, the presentation of the pipeline was still superficial, due to limited space. Therefore, we like to extend this work and shed light on the challenges we faced in the implementation. Accordingly, we formulate our research question:

How can continuous delivery help to automate EA model maintenance?

So far, existing research on EA model maintenance automation has focused either on collecting information from different external sources (e.g., [8,19,31]), trying to bring contradictory information together (e.g., [23,35,66]), or proposing an overall process for maintenance (e.g., [18,24]). 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 global EA model.

The rest of this paper is structured as follows. First, we 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. In this section, we will also present a DSL for EAM KPI's, which offers a descriptive approach to describe and interpret generic KPI metrics in EA models. Next, we demonstrate our pipeline and the KPI DSL by a fictitious example. We move on to evaluate the prototype we built. Therefore, we conduct an evaluation with five participants who worked with our prototype. We use the experience they gained to grade certain quality factors for the prototype as well as the underlying process model. We finish the evaluation by validating our prototype against well-known EA model maintenance requirements from the literature. 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 [19]. The main idea is that the data are kept within specialized repositories and linked to a central EA repository. In contrast, we try to synchronize the contents of different sources into one central maintained repository. However, the work of Fischer et al. [19] serves as a central input for the creation of our artifact.

Other approaches to automatize EA model maintenance are presented, for example, by Buschle et al. [8], who facilitate an ESB (enterprise service bus) to extract EA models automatically. In contrast, Holm et al. [31] 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. [35] and Bebensee and Hacks [5], who incorporate uncertainty into the mapping. Alegria and Vasconcelos [1] also use network data to get insights into the actual architecture and compare these findings with the existing documentation. The work of Völja et al. [65,66] focuses on uniting different information from contradictory sources. Hence, they try to estimate the trustworthiness of the sources. Due to the fact that these works focus on the collection of data directly at the source, our research can be understood as complementary. Basically, those works can serve as further input sources for our pipeline, respectively, used to rate the trustworthiness of the different input sources.

Further research focuses on the collection of information solely related to the application level. For example, Landthaler et al. [41] present, first, a literature review on facilitated methods to automatically collect data for EA model maintenance and, second, propose a new machine-learning-based means for collection. A concrete prototype is

implemented by Kleehaus et al. [39]. Their work combines runtime data with further relevant information that resides in federated information sources. The introduction of a validation workflow enables fully automated data integration, which minimizes the effort for manual tasks. These works are like our research. However, our work concentrates less on the collection of the data itself, but on the process of negotiating that the collected data are perceived as correct.

EA-related research did not only elaborate solely on the technical aspects of EA model maintenance. For example, Kirschner and Roth [38] rely on a human component to solve arising conflicts from different sources. Further, Khosroshahi et al. [37] investigated the social factors influencing the success of federated EA model maintenance. A slightly different point of view is taken by Hauder et al. [28] as they focused on the challenges of a federated EA model maintenance.

Another way to avoid conflicts is to rely on a collaborative approach. For instance, researchers elaborate on collaborative decision-making in the domain of EA [36,46,49]. Collaborative modeling is also a topic in related areas like enterprise modeling [50,59], business process modeling [3,14], or system modeling [11,52]. However, we take this kind of research not further into account for our work, as we consider a distributed environment in terms of place and time.

Further related research to our work can be identified in the field of continuous delivery. Humble and Farley [32] 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 [16] 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 [10,42,44] 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 [60].

Lastly, our process relies heavily on the quality of the EA model. Regarding 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” [34]. 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” [69, p. 30]. In general, research regarding EA quality agrees that it is defined by the ability to meet the EA users’ requirements [43,47]. Most of the related work divides quality aspects of EA into the quality of EA products, its related services, and EA processes [43,47].

There exist also approaches that try to measure the EA quality directly. The EAM KPI Catalog [45] proposes a set of 52 KPIs that help to measure and keep track on EA-related goals. Salentin and Hacks [55] present different measures to assess quality flaws of EA models and develop a prototype that can detect 14 of them. Timm et al. [63] sketch a framework that allows to assess the EA model quality in accordance with its purpose. All these works can serve as means to assess the quality of the input processed in our pipeline.

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. [6] define six principles that must be considered when assessing an enterprise model’s quality. Sandkuhl et al. [56] 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 [29]. 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. [48] since it transpired as effective in former research. It is split up into six single steps and two possible feedback loops:

- *Identify problem and motivate* Peffers et al. [48] differentiate four different entry points for design science-related research. Our entry point can be understood both problem-centered and design-centered. A problem-centered entry point is motivated by an existing problem that needs to be solved. Research has shown that reasons to change EA models are manifold [18] and raise many different challenges [28]. One of them is to handle different sources. One can differentiate two different

types of sources, technical sources like network scanners or databases and data created by human input like model instances in projects. Especially, the second type of sources is a challenge for many organizations as these sources are decentral in a location and time. Furthermore, the quality of the human-entered information can be of questionable state, leading to a pollution of the central model by unintended duplicates or typing errors. To solve this problem, we assume that the principle of continuous delivery offers efficient means to support the EA model maintenance process.

As the beforehand stated problem-centered facet has already been addressed in the software engineering domain by proposing continuous delivery, the entry point can be also seen as design-centered. Therefore, we reuse the established concept of continuous delivery and alter it that it suits the demands of an EA model maintenance as described in the problem-centered facet. Furthermore, EA is at the moment struggling to establish its goals in agile environments [64]. Reusing the means of the agile domain might ease the documentation of EA-related information, as the concerned are already familiar with similar tools.

- *Define objectives* Based on our research problem stated before, we conducted a literature study to support the definition of our objectives and requirements. By this, we identified three sources for objectives: First, we presented a roundtrip process for a distributed EA model evolution [24] which describes different tasks and their sequences focusing on a continuous evolution of the EA model. Our prototype will integrate the presented activities of a roundtrip process for EA model evolution in its concepts. Second, Farwick et al. [17] identified a set of 23 requirements on automated EA model maintenance grouped into categories like architectural, organizational, or data quality. We evaluate our prototype against these well-known requirements on automated EA model maintenance in research. Lastly, Fischer et al. [19] propose a meaningful roles and responsibilities concept for EA model maintenance. The concept foresees four roles related to process coordination (EA Coordinator, EA Repository Manager), data delivery (Data Owner), and quality checks (EA Coordinator, EA Stakeholder). Additionally, each role can have one of four responsibilities (Responsible, Accountable, Consult, Inform) per activity of an EA model maintenance process. Our prototype will reuse the proposed concepts incorporated in the design of our prototype as well. For this, as the concept proposes a notification process for stakeholders of the EA model maintenance with the according roles, we will consider a (semi-)automated notification process as an additional requirement on our prototype.

- *Design and development* In order to realize a prototype in accordance with the beforehand identified objectives, we start by aligning the input of the three objectives' sources. We gather the set of functionalities an automated process for EA model maintenance should provide. Then, we design a first draft of an abstract process model to describe the process of automated EA maintenance. As a next step, we refine this artifact by extending it by the concepts of continuous delivery. While doing so, we also focus on the specific concepts of the JARVIS framework, which we use as the framework to develop our CD pipeline. The final artifact of this step is a delivery model described as a formalized process model which serves as a model for a continuous delivery pipeline in JARVIS. To use the import service JARVIS provides, which translates delivery models to executable CD pipelines, we implement the required features as microservices and integrate them in JARVIS' infrastructure.
- *Demonstration* The demonstration is put into practice by applying the proposed means to a single fictitious case study of an airport departure system. Single case studies gain a first, in-depth reflection on means in real-life scenarios [68]. 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 [7, p. 623]. Further, we conducted a case study with five participants that used our pipeline to maintain collaboratively an EA model. Afterward, we interviewed them about their perception of the pipeline guided by identified quality criteria of EA model maintenance [25].
- *Communication* The communication is done with this paper itself, its previous version [26], and the presentation of the former version at a conference.

4 A pipeline for EA model evolution

Then, we will sketch our pipeline for an EA model maintenance process. Fischer et al. [19] contribute two main findings to our pipeline. First, they propose an EA model maintenance process, which we unite with our work from [24]. 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 the prototype JARVIS presented by Steffens et al. [60]. JARVIS is the implementation of a conceptual model for a new generation of software delivery

Fig. 1 The notations used in the process model of the EA model maintenance deployment pipeline

Fig. 2 EA model maintenance deployment pipeline

Fig. 3 1. Stage: prepare EA model data

Fig. 4 2. Stage: check EA model quality

systems. It focuses on two specific challenges for adopting CD. The first is the lack of flexibility and maintainability of software delivery systems. The second is the insufficient user support to model and manage delivery processes and pipelines. Further, it allows integrating the proposed processes into a deployment pipeline. Therefore, we create a formalized process model of the process as JARVIS is equipped to use formalized process models to describe pipeline models.

The process model describes the necessary activities of the automated EA model maintenance process we developed. In order to implement the CD pipeline with JARVIS, we express each activity in the process model by one of the three activity types JARVIS defines: Transformations, Assessments and Quality Gates [60]. Transformation activities take one or multiple artifacts as input and transform them to a new artifact, e.g., by merging or mutating them. Assessment activities measure certain criteria of an artifact. They take one artifact as input and create a report which contains possibly multiple

results of the measurements. Quality gates promote the input artifact according to the reports provided by the assessment activities and a given policy. We implement each activity in a microservice-based manner and integrate the resulting microservices in JARVIS' infrastructure.

Further, common CD pipelines are built up by multiple stages [32]. Steffens et al. define a stage as a sequence of activities which is usually completed by a quality gate [60]. Therefore, we structure the activities of the process model into multiple stages with a defined start and finish node. Figure 2 shows the resulting process model for EA model maintenance and Figs. 3, 4, 5, 6 and 7 the concrete realization of each stage. The notation of our process model is described in Fig. 1.

The upcoming part describes the concepts of each stage of the pipeline of Fig. 2 following the order of the depicted stages. Per stage we will briefly sketch how we implement our concepts with ArchiMate models. However, as the first process steps from Fischer et al. and Hacks et al. of initializing

Fig. 5 3. Stage: evolve EA model

Fig. 6 4. Stage: stakeholders approve EA model changes

Fig. 7 5. Stage: update global EA model

and collecting the necessary data of the EA model evolution can be omitted, we first describe the environmental setup to be used with our CD pipeline.

We store the global and project EA models in version control (VC) repositories, as following the principles of CD of Humble et al. [32] implies the use of VC such as subversion or git [16].

For an organization having n projects running, we assume that $n + 1$ VC repositories¹ exist which store EA models and related artifacts. One repository stores the organization-wide EA model artifact, which we name *global EA model*. Additionally, the global EA repository can store further artifacts related to the global EA model, such as constraints or schemes which must hold for all projects of the organization. Thus, we consider these artifacts to define the general superset of constraints, scheme definitions, etc., for all EA models in the organization.

In addition, we use one additional repository for each currently running project of the organization, leading to $n + 1$ repositories in total. Each project EA repository stores a copy of the global EA model artifact, which we name *project EA model*, and optionally additional artifacts. These artifacts can be considered as a more specialized subset of the organization-wide artifacts, i.e., they define more restrictive constraints, scheme definitions, etc., which only hold for the

according project. We can, for example, restrict the usage for certain elements or relationships in project *A* which may be used in project *B*.

After a project EA model has been changed, the according responsible of the project EA model (e.g., the solution architect) commits the applied changes to the project EA repository. Thus, in a given CICD environment each commit of the changes of a project EA model automatically triggers our CD pipeline. In the end of a successful pipeline run, the pipeline deploys the new global EA model in the global EA repository.

5 Concepts and realization

In our implementation, we develop several microservices and integrate them in JARVIS' infrastructure. We implement the transformations as JARVIS' transformation commands and assessments as JARVIS' assessment commands. For every last activity of a stage except the fifth stage, we reuse JARVIS' quality gate microservice as we consider each last activity as a quality gate according to Steffens et al. [60].

In the concrete implementation of the pipeline, we use ArchiMate 3.0.1 as the EA modeling language. As file format, we use the OpenGroup ArchiMate model exchange file format standard [62]. We implement our prototype for the "archimate3_model.xsd"² standard, i.e., our prototype does not transform any visual data of the models but only the model data itself.

¹ To ease the reading we only talk about different repositories. However, the terms *repository* and *branch* can be used interchangeably during the rest of the paper.

² http://www.opengroup.org/xsd/archimate/3.0/archimate3_Model.xsd.

5.1 Stage 1: prepare model data

The pipeline starts by first checking out the new model's versions from the repository with the according transformation activities. As JARVIS provides the git-service microservice [15], which encapsulates git transformation commands such as a git checkout, we reuse its service for these activities. In conclusion, our prototype requires git³ as VC system. Both artifacts are then provided to the "Align Model Data" transformation activity.

Align model data Hacks and Lichter [24] 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 comparable and mergeable to the global 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 global model.

The question remains what a level of detail concretely is. In our realization, we interpret one kind of detail level as the deepness of hierarchies of elements being connected by a chain of aggregation or composition relationships, since each aggregation or composition adds more granular information to the modeled concepts. We coin this term the *Aggregation-Composition Hierarchy Level* of an element in the ArchiMate model.

To preserve the semantics [53], we take advantage of the concepts of *derived relationships* [61]. In their research based on ArchiMate 1, Buuren et al. work out that a total order of the relationship types can be recognized in terms of a weighing function $W : R_T \rightarrow \mathbb{N}$, where R_T is the set of all relationship types. The function maps each relationship type to some weight [9]. A set of the ArchiMate 3 relationship types can be rerouted using this weighing function. For *Flow* and *Triggering* relationships, we can use a different approach based on a kind of transitivity for such relationship types. Thus, if we are not able to reroute a relationship based on the weighing function or the additional rules, we remove the relationship from the model as well. This is repeated recursively until no change in the model is detected.

The aligned model and the global model are then both provided to the "Compute Change Set" transformation activity.

Calculate change set This activity uses the provided input models to compute the existing deviations between both and provide these as a new artifact called "Change Set."

In our implementation, we build the change set as an ArchiMate model and enrich the model by the according VC properties, i.e., mark added elements to be added, deleted elements to be deleted, etc. The resulting artifact is by any means not a valid ArchiMate model, i.e., it may contain discon-

nected components. However, our intention is to reuse several assessment activities to be used on both the change set and the merged model in the third stage. Since the merged model must be a valid ArchiMate model as it shall be deployed as the new global EA model after a successful pipeline run, we find it is meaningful to generate the change set as an ArchiMate model as well.

Quality gate: validate artifact existence The following activity serves as the quality gate of the first stage. It checks the successful execution of the proceeded activities and the existence of the three artifacts "global EA model," "aligned model" and "change set." Afterward, the first stage of our deployment pipeline is finished. This stage corresponds to the checkout and compile stages in classic software delivery pipelines.

5.2 Stage 2: check model quality

Fischer et al. and Hacks and Lichter both incorporate steps to check the model validity and quality such as consistency or correctness of syntax. In our pipeline, we model these as assessment activities, which are performed on the change set and produce a report for each assessment. This stage corresponds to static analysis for software source code. Duval et al. [16] incorporate an inspection phase into their continuous integration model in which relevant metrics for software quality are measured and evaluated. We adopt this by measuring well-known EA KPIs [45] to models inside the pipeline. Additionally, we add an assessment to this stage which examines the change set for a set of bad smells. The last activity of this stage plays the role of the quality gate and promotes the change set if and only if it is syntactically and semantically valid and has reached the required degree of quality according to the metrics and bad smell assessments.

Thus, our prototype covers the assessment of the syntactic and semantic consistency of the model. Future implementations might add assessments for horizontal and vertical or other semantic or consistency check strategies as well.

Check syntax As we work with XML EA models and have access to the model scheme definition of the OpenGroup exchange file format standard, we implement the syntax check activity as a validation of the change set against the `archimate3_model.xsd`⁴ file provided by the OpenGroup. Since the ArchiMate schema file defines the syntactic structure of an ArchiMate model, we can easily state that the validation is sufficient to proof that an ArchiMate model is syntactically consistent.

Check semantics In order to meaningfully assess semantic consistency issues, measurement is needed. For this purpose,

³ <http://www.git-scm.com/>.

⁴ <http://www.opengroup.org/xsd/archimate/3.1/html-model/>.

Spanoudakis et al. [58] introduce the term of *consistency rules*. By defining such consistency rules, we can conclude that an EA model is inconsistent if the model does not satisfy the consistency rules [58]. Spanoudakis et al. further define several classes of consistency rules in their research. We will reuse their definition of the *well-formedness* rules.

The ArchiMate 3.0.1 specification provides a set of constraints which define valid relationship types between every pair of element type. Specifically, there are 58 different element types specified in the ArchiMate 3.0.1 specification and, thus, we extract a total amount of $58^2 = 3364$ consistency rules from the specification [61].

Calculate EA KPI metrics To assess EA KPI metrics, we follow a generic and descriptive approach. One of the functional requirements proposed by Farwick et al. “demands a language with which the calculation algorithm for Key Performance Indicators (KPI) can be defined” [17]. Thus, we implement a first version of a Domain Specific Language (DSL) as a description language with which the calculation rules for EA KPI metrics can be defined.

Detect EA bad smells Additionally to the EA KPI metrics, we measure EA smells as the last assessment of this stage. EA smells are a counterpart for Code Smells. The term Bad (Code) Smell was introduced by Martin Fowler [21] and stands for bad habits when writing and designing code and software resulting in a decrease of the overall code quality. Hitherto, Salentin et al. developed a catalog comprised by a total of 45 EA Smells, and a tool, which can detect 14 of the proposed 45 smells in an ArchiMate model [55]. We adapt the tool in our pipeline and embed it in this activity. The pipeline user can setup thresholds for each smell. For further read toward EA smells, we refer the reader to the corresponding paper [55].

Quality gate: validate change set validity and quality The last activity of this stage is again a quality gate. It takes the reports created by the syntax and semantics check to approve the EA model if and only if the model is consistent with respect to its syntax and the defined consistency rules. Then, it uses the reports created by the EA KPI metrics Calculation and EA Bad Smell detection activities. The quality gate validates if the amount of averagely and badly calculated metrics as well as of the found EA smells is below the critical thresholds and approves or rejects the change set accordingly.

5.3 Stage 3: evolve EA model

In the third 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 global 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. The change set may contain disconnected components. However, we require a valid global EA model to not have any. Therefore, with respect to the last stage we add an additional assessment to this stage, which checks if parts of the resulting EA model candidate have components, which are not connected to the rest of the model.

Evolve EA model Our implementation of the Evolve EA Model activity takes the global EA model and the change set as input. For its calculation, the activity uses the metadata the change set was enriched by to add to and update each activity in the global EA model, which is accordingly marked in the change set and removes the accordingly marked elements in the global EA model as well. Before persisting the new artifact, it cleans up the model by any metadata to generate a clean ArchiMate model with respect to the OpenGroup model exchange file format standard.

Check for disconnected components In order to find any disconnected components in the ArchiMate model, we handle the EA model as a graph. Since relationships in ArchiMate models have a source and a target element, the model can be induced to a directed graph. However, we want to take advantage of the characteristics of undirected graphs as it suffices to verify if the induced directed graph is weakly connected.

Definition 1 Let $G = (V, E)$ be an undirected graph. G is called connected if there is a path between any two vertices in G .

Definition 2 Let $G = (V, E)$ be a directed graph. Let $\tilde{G} = (V, E)$ be the undirected graph built up by replacing all edges of G by undirected edges. G is called weakly connected if $\tilde{G}$ is connected.

Accordingly, our implementation of this activity builds the undirected graph $\tilde{G}$ of the ArchiMate model. Then, it applies a depth first search on $\tilde{G}$. If an undirected graph is connected, depth first search will visit all vertices in the graph from any arbitrary start node in the graph.

Quality gate: validate merged model validity and quality Again the last activity of the stage is a quality gate and behaves like the quality gate of the second stage. As an additional report, the quality gate approves the merged EA model only if the disconnected components check validated. In the end, the merged model is only approved to enter the user acceptance stage of the pipeline, if it met the required quality level.

5.4 Stage 4: stakeholders approve EA model changes

Up to this point, the pipeline performs its tasks completely autonomously. This means that the stakeholders are only involved in the whole EA model maintenance process 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. [4] 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.

Prepare update report Our first activity of this stage takes all artifacts and reports of the pipeline run as input and creates an artifact holding all information about the respective pipeline run, i.e., metadata added by transformation activities and the metrics calculated by the assessment activities. The activity stores this report to be sent to the stakeholders by the next activity.

Notify stakeholders We implement this activity to require two inputs, which provide the activity with the list of responsibilities and a list of mail receivers. Using this input, the activity detects the list of users, to which it sends the update report created in the last activity, and the list of users which it marks to give the required approval. Finally, the activity sends an email to all stakeholders with the update report attached with the possibility to approve or reject the result.

Collect responses The pipeline user defines a timeout for this activity, which defines the time the approval users have to approve the new EA model candidate. When the timeout is reached, this activity ends its idle state and the next activity is invoked.

Quality gate: check approvals The quality gate of this stage approves the merged EA model to be deployed if all required approvals are given. It is configured by the required ratio of given approvals and approves or rejects the new model accordingly.

5.5 Stage 5: update EA model

If all stakeholders approved the new global EA model candidate, the EA model candidate is promoted to the final stage where it is deployed to the EA model VC repository. The next run of the pipeline will use this new version of the EA model as the global EA model and so the roundtrip is completed. This stage does not come up with a quality gate as it is not meaningful to assess the deployment of the model. If the deployment fails, the git-service will throw an exception.

The thrown exception will cause the pipeline to fail and the pipeline user can see the reason of the fail in the pipeline client.

6 Demonstration

To demonstrate our artifact, we conduct a fictitious case study. For this we use a fictitious EA model, which models 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 [51]. A subset of this model is later also used for the evaluation of the pipeline, which will be explained in more detail in Sect. 7.

6.1 Exemplary EA model

Then, we illustrate a scenario of an airport departure system, which depicts the functionality of the services offered by the airport departure system to the passengers before boarding to an aircraft. We opted for this scenario, as many people are common with the departure processes at airports and, consequently, can easily understand the reasoning behind the single elements. Thus, no special domain knowledge is needed. However, we never had access to a real-world airport EA.

The airport example is modeled as an EA model based on ArchiMate 3.0.1 [61]. The model incorporates all the ArchiMate layers beginning with business, application, and technology architectures. It consists of 171 different elements and 250 relations. According to the complexity of our example, we consider it as a scoped down example of a real-world airport departure system since an EA model of a real-world airport scenario would easily contain much more elements and relationships. We will further discuss in detail the core layers of the system. Figure 8 shows an excerpt of the example we used. It models the aspect of the services the airport offers when passengers wait in the lounge before their flight starts.

The business layer depicts the business services offered to the customers [22]. In this example, the active entities of the business layer are the waiting passengers and vendors selling items to the passengers at duty-free shops. The waiting in lounge concept is composed of several services at the lounge such as duty-free shops or restaurants.

The application layer includes the payment system and the airside services to passengers. The passengers can pay by cash and by card payment at the services at lounge, which are both based on different underlying technologies.

The technology layer offers several components to the application layer. In this example, both card payment and the airside services require certain technologies, such as Wi-Fi access or a wireless credit card terminal.

Fig. 8 Excerpt of the boarding and departure process

As we describe in Sect. 4, we store the example EA model and an organization-wide xsd file for syntax validation in a git repository that serves as the global EA model. For the organization-wide xsd file, we use the official schema file of an ArchiMate 3.0 diagram⁵ defined by the OpenGroup ArchiMate model exchange file format standard. Thus, we allow all elements to be used in the EA models of our fictitious organization. We also store a copy of the EA model and a project specific xsd file in a second repository to be used as the project EA model. In the project specific xsd file, we remove random elements from the organization-wide xsd file and restrict the set of allowed elements which can be used in our example project. Then, we apply random changes to the project EA model, commit the changes to the project EA repository and trigger the CD pipeline. After a successful run, the pipeline deploys the new global EA model in the global EA repository. Figure 9 shows the case study design exemplary in a sequence diagram.

⁵ http://www.opengroup.org/xsd/archimate/3.0/archimate3_Diagram.xsd.

Figure 10 shows exemplary changes made to the global model shown in Fig. 8. The additions we made to the model are marked by a stronger coloring of the color tone of the corresponding model level.

When applying our case study, we also add invalid data to our model to elaborate the expected fail in the pipeline. This is to get first insights of how our pipeline handles several scenarios. We also use relationships in Aggregation-Composition Hierarchies to elaborate the behavior of our Align Model Data activity.

After the deployment, we verify whether the Align Model Data activity rerouted the added specializations *Long-Drink* and *Soft-Drink* as expected. According to our rerouting rules, we never reroute specialization relationships. Rerouting specialization relationships will break the semantics in most cases.

6.2 Mapped metrics

As described in Sect. 5, we developed a DSL to describe metrics in EA models. To demonstrate our KPI assessment,

Fig. 9 Workflow and setup of the case study

Fig. 10 Excerpt changes to the project EA model. Strong hues show additions with respect to the global model

Fig. 11 Model of audit findings metric in Archi described by the EA metric DSL

Fig. 12 Model of backup key roles metric in Archi described by the EA metric DSL

we map five KPIs from the EAM KPI catalog [45] and check the EA model against them. As we do not want to describe all KPIs of the catalog with our metric description language, we randomly chose five of them as representatives. Those KPIs are only exemplary and can be replaced by any other calculable metric, which can be expressed by the current version of our description language. Nonetheless, we must keep in mind that it can be quite challenging to assess the necessary input parameters (e.g., if interviews need to be conducted).

In the following, we provide brief descriptions on how we mapped and described the metrics defined in the EAM KPI catalog to ArchiMate. We also demonstrate each mapping by a figure, which shows the model mapping in a model.

Audit findings This metric measures the extent to which the IT adhered to internal and external compliance [45, p. 41]. As the information model of the KPI catalog is not directly reflected in ArchiMate, we describe this metric as a business object with the property “isAudit” and value “true” and two relationships (see Fig. 11).

Backed up key roles The backed up key roles KPI measures how completely qualified personnel has been built up [45, p. 20]. We model this by two elements of type “Business Role” and “Business Actor” and a relationship between both (see Fig. 12). The relationship must be of type “Assignment” and have the name *qualifies for backup* with the Business Actor being the source and the Business Role being the target of the relationship. To identify which business role a key role is, we query for Business Roles having the property “isKeyRole” and value “true.” We define the metric to calculate the ratio between the key roles having backup personnel and such key roles without a backup personnel.

Fig. 13 Model of employee qualification in Archi described by the EA metric DSL

Fig. 14 Model of IT process standard adherence metric in Archi described by the EA metric DSL

Employee qualification The performance of the training and HR process is measured by the employee qualification metric [45, p. 44]. We can copy the information model given in the catalog one to one to ArchiMate models and describe the metric definition accordingly (see Fig. 13). As before, we calculate the ratio between the elements of type “Business Actor” identified as qualified and those, which are not.

IT process standard adherence (service) This metric measures the extent to which IT applications adhered to the standardized IT processes [45, p. 51]. The information model defined by the catalog requires a business application, an IT process with a property defining that it is standardized and some relationship connecting both elements (see Fig. 14). We identify both elements by their type as well as the property “is standardized” with value “true.” For the relationship, we require an association relationship which is named either “complies to” or “is compiled by.”

Action plans for critical IT risks The last metric we describe is a metric which measures the completeness of defined action plans for the prevention of critical IT risks [45, p. 59]. The information model does not give a clear definition of what type of element an IT risk and the corresponding action plan shall be modeled by. It distinguishes between an IT risk and critical IT risks by a Boolean property of the element and defines some relationship between the IT risk and the action plan, which models a “prevents” relationship between both. In our mapping, we use the element type “Constraint” to model the risk (see Fig. 15). To query for these elements, we require two properties for these constraints. First, the element

Fig. 15 Model of action plans for critical IT risks in Archi described by the EA metric DSL

must have the property “is risk” with value “true.” Second, to distinguish between critical and non-critical risks, a critical risk shall be identified by the property “is critical” with value “true.” For the action plan we take use of the “Course of Action” element type of ArchiMate. Our mapping requires an association relationship between both elements which must be named “prevents” or “prevented by.”

For the demonstration purpose, we intentionally break the modeling rules of the defined KPI’s in the case study EA model to elaborate the behavior of the corresponding pipeline activity. For this, we add critical risk to the model, i.e., a constraint with both properties “is risk” and “is critical” and both set to true and connect it to some component in the model such that the disconnected components check does not invalidate the model. We also prepare the model in such a way that only one KPI is considered per time. Then, we define the according quality gate to fail if any KPI was measured as “bad.”

Lastly, we define the metric description of the “action plans for critical IT risks” KPI to interpret the value as bad if the ratio is $< 100\%$. We now run the pipeline once on the ArchiMate model, which has a critical risk defined without an action plan it prevents as defined by the metric description. We observe that the according quality gate does not approve the model as a KPI was measured with a bad outcome. We now modify the model such that the critical risk is prevented by an action plan and run the pipeline again. This time, the quality gate approves the model. By observing the developed report of the KPI metric assessment, we see that it indeed calculates a good outcome for this metric. We conclude that by adding an action plan to the critical risk, we increased the quality model with respect to the desired quality level.

To demonstrate the definition of critical KPI’s, we extend the setup described above by a second metric from our mapped metric descriptions. This time, we opt for the “IT process standard adherence” and define it as a critical metric in its pipeline description. We modify the model from the case study as described in Fig. 14 such that there is at least one application service which is not compliant to an application process, i.e., the ratio of all compliant application services will be less than 100% . Lastly, we define the

interpretation of the KPI as bad if the ratio of all compliant application services is less than 100% .

Having defined two metrics based on which we can prepare the model such that the KPI’s are assessed with a bad outcome, we can now show that the quality gate disapproves the model only because of the critical KPI. We first configure the according quality gate such that it disapproves the model if two or more KPI’s were measured poorly. Then we modify the ArchiMate model so that the metric of the “action plans for critical IT risks” is measured with a bad outcome. When we run the pipeline, we see that the quality gate did not approve the model.

Afterward, we modify the EA model once again such that the “action plans for critical IT risks” KPI is measured well. When running the pipeline with this setting, we perceive that the quality gate disapproves the model even though only one of two KPI’s has a bad value. This is because the critical “IT process standard adherence” metric is measured poorly. To verify this expected disapproval, we change the EA model once again. This time, the critical KPI is measured well, while the other one is measured poorly. We see that the quality gate approves the model, because only one non-critical metric is measured poorly.

6.3 Disconnected components

To demonstrate the check for disconnected components, we take our ArchiMate model from the case study and randomly remove certain relationships such that the model contains disconnected components. Whenever the removal of a relationship induced a disconnected component in the model, the pipeline fails due to the disconnected components policy. After reconnecting all disconnected components again such that the model does not contain any disconnected components anymore, the pipeline approves the model again.

6.4 Stakeholder approval

To demonstrate the stakeholder approval stage, we prepare several roles-responsibilities mappings as well as required approval scenarios. To demonstrate the sending of different emails based on the role of the user, we create the following setup:

1. User A is a solution architect
2. User B is an enterprise architect
3. User C is an enterprise architect
3. User D is a software engineer

For these user roles, we defined the following rules for receiving an email:

1. Roles which receive an email: solution architect, enterprise architect
2. Roles which need to give approval: enterprise architect
3. Additional mail receivers: User *E*, User *F*
4. Additional required approval; User *E*

Based on this setup, users A, B, C, E, and F should each receive an email, since both solution architects and enterprise architects receive emails and D and E are additional notification receivers. User D should not receive any notification, as software engineers are not expected to receive an email from our notification service. However, emails for users B, C, and E should contain a link to approve or reject the updated model, since both users are configured to give their approval. The email for users A and F should have the update report attached to the mail but should not provide a link with which the users can approve the model.

We prepare the pipeline such that it requires 2 out of 3 approvals in order to approve the new central model to be deployed. Additionally, we set it up to wait for 5 min to collect all approvals such that we can demonstrate and test the timeout of the activity. Besides the 66% approval ratio, we add an additional user *E* to be required to give his approval in order to test the required approvals by user functionality as well. With this setup, the quality gate should only approve the model for deployment, if within the defined threshold

User *B* and User *E*,
 User *C* and User *E* or
 User *B*, User *C* and User *E*

gave their approval. In any other case, the quality gate should not approve the model. If only user *E* gave his approval, the constraint that at least 2 approvals must be given will not be fulfilled. If both users *B* and *C* gave the approval, the constraint that user *E* must give his approval, will not be fulfilled. Our demonstration shows that the pipeline fulfills these requirements.

7 Evaluation

To evaluate our approach, we asked CD and EA practitioners to participate in our case study environment. Five persons participated for the evaluation of our CD pipeline. Each participant works as a research assistant in the IT field. We chose them such that the participants represent a heterogeneous set of IT experts. Four of them show little to no experience with EA modeling but have much experience in the practice of CD. The fifth participant showed more experience in EA modeling but less experience in the practice of CD. We will see that the experience and knowledge in the respective fields

seem to have an impact in the participant's evaluation of our approach.

7.1 Case environment

The participants were asked to work on an incomplete version of the case study ArchiMate model we presented in Sect. 6 using the Archi tool. In order for the participants to work easily on the provided ArchiMate model, we removed all details except one from the demonstrated model. The participants received the task to complete the EA model by modeling the missing aspects of the airport departure system. To collaborate with each other, the participants were asked to work on the model in the project branch and whenever they want to integrate their changes, to commit their changes to the project branch and trigger the provided CD pipeline. Additionally, we asked the participants to take the required KPI metrics in their modeling into account.

For the pipeline configuration, we chose a maximum Aggregation-Composition Hierarchy level of 3, such that relationships and elements from a hierarchy of level 4 onward should be removed from the model. To measure the EAM KPIs, we decided to define just two of the five metrics we presented in Sect. 6.2: IT process standard adherence (service) and Action plans for critical risks. We took this decision as we considered it difficult for the participants, who were not used to ArchiMate or EA modeling, to get used to more than two metrics when apply changes to the model. Additionally, we assess both metrics to be plausible choices when being used in an EA model of an airport departure system.

7.2 Evaluation method

We gave the participants three days to work together on the model. After the deadline expired, we conducted an interview with each participant. Each interview lasted 30–60 min. In the interview, we first asked specific questions in order to assess the knowledge of the participant in the areas of EA modeling as well as CD. Doing so, we could include the different backgrounds in the discussion of the interview results.

After we assessed the background of the participant, we asked to evaluate several quality criteria of the concepts and approaches of our CD pipeline. The participant graded the utility of each assessment activity in the pipeline as well as the Align Model Data activity approach on a scale from 1, very useful, to 5, not useful at all. In the last part, the participant was asked to give a grading for five quality factors of the process model presented in Sect. 5. For this, the participant had to grade the process model against five well-defined quality criteria for EA model maintenance processes proposed by Hacks and Lichter [25] on a scale from 1, quality criterion fully met, to 5, quality criterion not met at all.

Before going into detail, we want to briefly discuss about the limitations of our evaluation. As we presented in Sect. 6, we used a fictitious case study model for the evaluation of our prototype. Even though the model is not a simple one, as it contains 171 nodes and 250 edges, it cannot be considered as a replacement for a real EA model, as real EA models contain thousands of nodes and edges and, therefore, have a higher degree of complexity. On the other hand, the modeling possibilities in a general airport departure system are limited. For example, enterprises model explicit employees instead of general ones in their EAs. By modeling explicit employees of a company, architects are more flexible in defining KPI metrics. In conclusion, we note that we were limited in the design of our case environment for the evaluation of our prototype.

However, we argue that the EA model is still sufficient to be used for an evaluation. It contains all necessary data to test and evaluate all activities of our pipeline. Even though we were limited in the choice of KPI's to be measured by our pipeline, we were still able to find well suited KPI's which serve as representatives for other KPI's and other domains of EA models. We keep a further evaluation with real EA models open for future work.

7.3 Insights from the application

Since the participants applied changes to the model with the Archi tool, they were not able to incorporate inconsistencies in their changes, which would invalidate our semantic or syntactic consistency checks. For that reason, to evaluate both consistency checks, we applied sample tests.

For the syntactic consistency check, we randomly broke the ArchiMate xml model file in Sect. 6. We ended up with nine unit tests, in which we broke the xml format, added invalid xml tags based on the xsd definition and removed required references within the model. In the unit test in which we broke the xml format, the syntax check activity failed, i.e., did not produce a report to be used by the quality gate. This behavior is expected since we implement the syntax check using an xsd validation. If the xml format is broken, we cannot serialize the xml file properly. Thus, the xsd validation throws the expected exception. In the cases in which we did not break the xml format but the validity with respect to the underlying xml schema file, in all tests the syntax check produced the expected report.

For the semantic consistency check, we randomly chose ten valid and ten invalid relationships between random element types as defined by the ArchiMate 3.0 specification, to conduct both true-positive and true-negative unit tests. First, we verified that the case-study model validated the semantic consistency check, i.e., that it was semantic consistent with respect to our defined semantic consistency rules. Then, we modified the model test-wise such that it included exactly

one of the invalid relationships and ran the unit test. In all cases, the semantic inconsistency was detected as expected and the quality gate disapproved the model. We repeated the process with the valid relationships to conduct true-positive test cases. Again, all tests showed the expected result.

It was difficult to elaborate the stakeholder approval stage in the evaluation by the participants. They worked in our case study in short breaks of their work. By this, it was not possible to meaningfully add this stage to the case study setup. It would not have helped the participants assessing the value of our pipeline if they would have to wait hours to see their deployed changes. We did not have control over the participants to quickly approve the model changes whenever they received an email. However, a small timeout would not have been meaningful for the evaluation as well, as it might have caused multiple disapproved deployments in case that the required amount of participants would not have been able to give their approval in time. Due to this, we must rely on the evaluation through our demonstration for this stage. Additionally, we will take the difficulties in the stakeholder approval setup into account in the discussion.

All participants were able to successfully deploy valid EA models to the release repository multiple times. Based on their experiences, it was not the case that the pipeline behaved unexpectedly. Valid EA models were deployed in the release repository, and invalid EA models were rejected. One participant unintentionally committed changes which included disconnected components. The pipeline validly disapproved his changes in stage 3 of the pipeline. He then used the feedback provided by the violated policy in order to fix the quality issue. Then, he successfully deployed the model using our CD pipeline. Another user violated the "action plans for critical IT risks" KPI as he added a risk to the model but did not link a course of action element with it, which lead to a disapproval of his changes by the quality gate in the second stage. After he added the missing action plan to the risk, the pipeline successfully deployed his changes to the release repository.

7.4 Interview results

The pipeline quality factors were graded with a big spread between the participant opinions. Figure 16 shows the grading as a line diagram. The straight line visualizes the mean value of all grades. The triangles show the best grade of a factor, the crosses, respectively, the worst grading.

Interestingly, the huge spread in the grades correlates with the different backgrounds of the test users. All worst grades were given by the single participant, who had experience in the field of EA but few experiences in the practice of CD. The other participants showed few experiences in the field of EA and similar much experience in the practice of CD. Their grading spread was much smaller.

Fig. 16 Mean grade per pipeline quality factor assessed by the test users

We argue that the huge differences stem from different perspectives the test users had toward a CD pipeline for EA model maintenance. The users who are experienced in CD tend to give rationales based on the CD concepts we implemented. The user with more knowledge in EA modeling argued stronger from the perspective of an EA architect. Mostly, he was missing more feedback in terms of the reasons for a failed pipeline run. He also criticized the user experience of the pipeline, by not deploying his changes when the quality standard was not met. He would have liked to still deploy the artifact and to receive a respective warning such that he could clean the lack quality up in his next changes. Without wanting to belittle his opinion, we found out that his opinion does not reflect the general idea of CD. This also coincides with his less available experience in the field. Nevertheless, we consider his opinion as very valuable. It indicates that our concepts might cover CD concepts rather than requirements toward EA modeling.

We will discuss briefly the grades given per quality factor in the following. Doing so, we will identify the single user with more experience in EA but less experience in CD as *Tester 1* to make the text more readable.

Value of approach to detect inconsistencies The first question was to assess the value the participant sees in automatically checking an EA model for inconsistencies. Since the testers were not able to invalidate the consistency checks by using Archi, they were given a scenario in order to give a meaningful assessment for this concept. In the scenario, the participant was asked to assume that s/he ran a script after s/he applied changes to the model, which just applies some layouting optimization. The script, however, was buggy

which lead to broken syntax and semantics. Based on this, the participant should reason for the value of the automatic consistency checks.

All testers evaluated this concept as useful to very useful. The main reason is that this is expected to be a standard quality assurance concept for any kind of model.

Value of our implementation to detect inconsistencies Then, we wanted to know the tester's opinion for our approach to detect inconsistencies, i.e., to use xsd validation for syntactic and consistency rules using semantically invalid relationships for semantic consistency check.

All participants, except Tester 1, graded our approach with grade 1 or 2. Tester 1 graded our approach with a 5, i.e., the worst grade possible. He argued that the feedback misses which element or relationship is the reason of the inconsistency, making the outcome of this assessment not valuable.

Value of approach to calculate KPI's In the next question, we wanted to know how testers assess the usefulness of automatically calculated KPI's. As in the first question, the participants had the same opinion and accordingly graded this concept with 1 and 2 only.

Value of our implementation to calculate KPI's In the assessment on our approach to calculate KPI's automatically, the participants showed a wide spread. Even though the general idea was considered as very valuable, some participants had concerns regarding the usage of a DSL. The common opinion is that the usage of DSL's has advantages and disadvantages and it depends on the complexity of the language

and the acceptance of the end user for how useful such an approach will be in practice.

Tester 1 criticized like in the second question the lack of feedback given by the pipeline in the case of a failure.

Value of detecting disconnected components The usefulness of detecting disconnected components was difficult for the test users. One user assessed it as very valuable because the assessment found her disconnected component validly. One tester abstained as he had not enough knowledge to assess the importance of not having disconnected components in EA models. Two participants graded it with 3 and 4, but due to the same reason of why one tester abstained the assessment.

Tester 1 again criticized the lack of feedback given by our activity.

Value of our aligning model data implementation The assessment of our approach to remove too detailed elements from the model resulted in mixed feelings by the participants. Two of them found it very useful. However, the usefulness rises and falls with the implementation and the adaptability. Others considered the concrete deletion as a too hard intervention in the model. They would have found it better to mark the elements instead of removing them.

Value as a CD/CI approach for automated EA model maintenance The question whether CD is a suitable approach for automated EA model maintenance was graded without a big spread. All test users had the opinion that a well thought and implemented CD pipeline can be a valuable approach to automate the maintenance of EA models.

7.5 Process quality assessment

Hacks and Lichter define five quality factors for automated EA model maintenance processes, specifically comprehensibility, effectiveness, completeness, minimality, and efficiency [25]. We assessed four of these quality factors in the conducted interviews. However, we were not able to give a meaningful assessment toward the efficiency of the process. The authors define the efficiency as the “degree the process is perceived—in terms of time—to maintain changes into an enterprise architecture model” [25]. We were not able to include the stakeholder approval stage to our conducted evaluation with the test users. It is expected that this stage will be a huge bottleneck in terms of efficiency, but since the test users could not give meaningful assessments for the efficiency of the process, we ignored its assessment in our interviews.

Figure 17 shows the evaluation for the EA model maintenance process in a line diagram. The notation is the same as in Fig. 16. We see that the worst average grade is given to effectiveness with a grade of 2. The minimum grad given was

a 4 for the completeness by Tester 1. We will briefly discuss the evaluation for each quality factor in the following.

Comprehensibility All test users could easily understand our process model. One participant would have liked to have a better naming of the activities and stages.

Effectiveness The effectiveness was overall assessed to have good quality. One tester sees a problem in the practicability due to the stakeholder approval stage. There might be changes which should not cause an email and required approval by all stakeholders, e.g., when fixing a typo. Other testers criticize that the changes must be committed before the feedback can be given. It would be more effective if feedback toward the quality of the model could be given upfront before committing the changes.

Completeness The completeness was graded with 1 by 4 of the 5 test users. Tester 1 was missing activities to measure the EA debt of the model. The other test users could not come up with any missing step in the pipeline.

Minimality The minimality of the process model was measured with a high degree. No tester was able to come up with any unnecessary step in the pipeline. However, two testers did not want to give the best grade as they do not consider their knowledge as deep enough in order to be able to come up with additional process steps.

7.6 Fulfillment of requirements

After having evaluated our prototype by our case study and the conducted interviews, we now want to assess to which degree we fulfilled the requirements defined by Farwick et al. [17].

AR1: “The collection of EA data must be federated from the repositories of the data owners (departments etc.)” [17].

This requirement is met. The data owners can be mapped to our solution architects. They change their project EA model and integrate their changes with the project repository. The pipeline then collects and merges the EA model states and integrates the changes into the central EA repository.

OR1: “An organizational process must be in place that regulates the maintenance of EA models” [17].

We consider this requirement as fulfilled. Our prototype does not provide such an organizational process. But assuming the existence of an organizational process, which defines the responsibilities as required, our pipeline can be used as the technical process supporting the respective roles. See for this requirement OR1.1.

OR1.1: “The organizational maintenance process must be supported by a technical process” [17].

Fig. 17 Mean grade per process quality factor assessed by the test users

We see this requirement as fulfilled by our pipeline, which supports the maintenance process by the automated data integration and quality assessment in certain time intervals.

OR1.2: “The system must be able to adapt the maintenance process to the existing processes in a company” [17].

The pipeline can be integrated to the existing processes in a company. It requires a simple git repository setup. The pipeline itself is packaged as a standalone software via docker⁶ images and docker-compose configuration for easy container installation and orchestration. Thus, we see this requirement as fulfilled.

OR2: “Each data source must have an owner/responsible” [17].

We see this requirement as being fulfilled. Even though the pipeline does not explicitly define data owners, the usage of git implicitly does. The solution architects of the project repositories are the owners of the respective data sources.

OR2.1: “The technical maintenance process must allow for delegation of rights to ensure that the QA process is always executable” [17].

A project branch can be used by several solution architects at the same time. Thus, we consider this requirement as fulfilled.

IR1–IR1.5: These requirements can be ignored in our work as our approach does not tackle them. The requirements are all related to change detection in the real-world EA [17]. We argue that change detection tools, which trigger change events and the delivery system, would be fulfilled by our delivery system. However, these requirements aim toward an active change detection, which is none of the objectives

we followed with this approach. Therefore, we do not have the claim to fulfill these requirements.

IR2: “The system must have a machine understandable internal data structure” [17].

This requirement is not fulfilled by our prototype. Even though we implement all commands extendable with respect to the modeling language used, i.e., additional implementations can be added in the future, the current implementation only supports the Open Group ArchiMate model exchange file format.

IR2.1: “The system must be able to be configured to transform incoming data, to the internal machine understandable data format” [17].

The requirement is not further clarified by Farwick et al. According to the brief description and the fact that our prototype does not fulfill IR2, we do not argue that it fulfills this requirement neither. Thus, we consider this requirement as not met.

DQR1: “The system must provide mechanisms that help the QA team to ensure data consistency” [17].

Our pipeline implements two types of consistency checks which ensure data consistency to a certain extent. However, the implementation of the syntax command has proven to be incompatible to be used on the Change Set, making it only valuable to detect syntax inconsistencies in the merged EA model. The design and architecture of the pipeline offers the extension of more implementations of consistency checks. Thus, we consider this requirement to be met by our design and implementation of the pipeline.

DQR2: “The system must provide mechanisms to ensure data actuality that is sufficient for the EA goals” [17].

The implementation of the pipeline currently does not include a data actuality check activity except the check of

⁶ <https://www.docker.com/>.

the ArchiMate model version. However, due to the extendable design of the delivery system, an according activity can easily be implemented and added to the Check EA Model Quality stage in the pipeline. Thus, we consider this requirement as partially met.

DQR2.1: “Each element in the systems data structure must have a creation time stamp and an expiration date (volatility)” [17].

As we did not completely fulfill DQR2, we did not fulfill this requirement neither. However, like for DQR2 our design allows for extension of the delivery model by the implementation of according activity commands. Thus, we consider this requirement as partially met.

DQR3: “The system must provide mechanisms to adjust the granularity of data” [17].

This requirement is fulfilled by our Align Model Data activity. By removing too deep Aggregation-Composition-Hierarchy Levels, the activity brings both the global and the project EA model to the same level of detail.

DQR4: “The system must provide mechanisms that allow for the automated propagation of changes” [17].

This requirement is met by our pipeline. The automated consistency checks lead to a pipeline fail with a message of the failure report, if the model was not consistent with respect to the checked consistency type. Thus, the pipeline identifies model inconsistencies and reports them to the solution architects.

DQR5: “The system must be able to identify and resolve data identity conflicts from different sources via identity reconciliation” [17].

This requirement is not met, as no according activity is modeled in the pipeline. However, it is again the case that the pipeline design offers the opportunity to extend the model by an according quality check. Thus, we consider this requirement as partially met.

FR1: “The system must allow for the definition of KPIs calculations” [17].

The metric description DSL fulfills this requirement. The tests we ran showed that the detection of the queried elements and relationships work as they suppose.

FR2: “The system must be able to calculate the defined KPIs from runtime information” [17].

As for FR1, this requirement is met by the implementation of our EA KPI metrics activity as we have seen in the outcome of our tests.

NFR1 “The system must scale for large data input” [17].

Due to the lack of usable data sources, we could not stress test the whole pipeline with large data sets. We tested the Align Model Data command with a large data set, as it is the implementation involving by far the heaviest calculations with a positive outcome. However, as we could not stress test the whole pipeline with this data set, we cannot verify that we met this requirement and thus consider it as unclear.

Table 1 Fulfilled requirements by Farwick et al. [17]

ID	Fulfilled
AR1	+
OR1	+
OR1.1	+
OR1.2	+
OR2	+
OR2.1	+
IR2	–
IR2.1	–
DQR1	o
DQR2	o
DQR2.1	o
DQR3	+
DQR4	+
DQR5	o
FR1	+
FR2	+
NFR1	?

+ shows fulfilled; o partially; – not fulfilled requirements

For an overview, we listed all requirement fulfillment in Table 1. We used + for fulfillment, o for partially met and – for not met.

8 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 [20]. 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 their 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. [19] lacks the roundtrip approach. As we count on short feedback cycles as being 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 can only approve EA model candidates, which reached a certain degree of quality.

In the evaluation, we showed that our prototype meets 14 out of 23 requirements for EA model maintenance processes in the literature. It is difficult to compare this outcome with related work that we presented. Up to our knowledge, the related concepts did not use those requirements for their evaluation. We argue that the presented process model is an extension of the model presented by Hacks et al. [24]. We do not find any requirement which is met by this process model but not met by ours.

The EA maintenance process by Fischer et al. [19] meets several requirements of the set. For example, it fulfills *DQR1* as it involves a manual consistency check and *OR2.1* by defining several roles and responsibilities in the EA maintenance process. However, it is difficult to compare both processes in general. The process model defined by Fischer et al. defines many manual activities, while our model defines mostly automated processes. The usage of version control repositories is an additional feature which makes a comparison between both processes difficult.

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.

9 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 [67]. In recent years, the field of EA 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.

All in all, we give an answer to our research question based on our observations and evaluation of our pipeline. We argue that one means of answering this question is whether CD supports the quality assessment of EA models. Matthes et al. define one objective of EA to measure the degree of achievement of EA goals [45]. Thus, we see the automatic assessment of EA KPI metrics as an option of measuring the conformance of the EA model to the requirements of the EA stakeholders. Thus, it is safe to say that CD shows some promising practices and applicability toward the automated quality assessment of EA models.

This does also conform to the statements of the participants in our evaluation. All participants agreed that our CD pipeline is a valuable approach for EA model maintenance. However, a short caveat is given in the usage of version control. Even though the practice of CD requires the usage of version control repositories, the participants criticized the process flow in the evaluation. According to them, it would be much handier if the quality feedback would be given before a commit of the changes. We conclude that CD shows some valuable concepts in the automated maintenance of EA models but that the practice might need some adaptations toward the management of the EA model artifacts.

Assessing the quality criteria of an EA model by using automated metric calculations at runtime can surely improve the value of EA modeling by retrieving continuous value to the EA stakeholders about the degree of achievement of the

EA goals. A metric description language is even vertically extendable in a sense that it may define abstractions for metrics, which are not directly related to the measurement of EA KPIs but give other insights of the EA model's quality, which specific EA stakeholders might be interested in.

Additional to the automatic quality assessment, we see great value in the automated validation of EA models by our consistency checks. Even though the syntax check for the change set must be refined, we are able to detect multiple semantically invalid relationships. It would have cost much time to detect such errors by manual reviews.

Another aspect for the feasibility of CD for automated EA maintenance is the degree of CD supporting the merging of asynchronously changed project EA models. Our concept foresees a distributed artifact storage. The global EA model can always be received from the global EA repository by multiple solution architects of the projects in an organization. Therefore, the CD approach forces a VC repository setup, which allows for steady and up-to-date information gathered from the global EA repository. The asynchronously changed models are automatically merged by the pipeline and deployed as the new global EA model, keeping the data up to date. Even though our concepts do yet not include conflict detection (the text-based conflict detection implemented by git is not suitable for XML documents [2]) and resolution algorithms—problems which usually occur in asynchronous collaboration environments [13]—we argue that the automatic merging of EA models shows great value for EA model maintenance. However, it should be supported in the future by more suitable conflict detection algorithms, such as graph-based algorithms [2].

In the end, we argue that CD is in fact a feasible approach for automated EA model maintenance.

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 providers might cause issues, and 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 global 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.

Apart from limitations, our approach should be discussed in future in relation to other existing research and how these different techniques can be integrated to each other. For example, our research, the research on federated EA model maintenance [37], and collaborative model merging [38] share common elements and can easily benefit from each other.

Another aspect is related to the fact that we focused on a single organization. However, in some domains the EAs of different organizations are tightly coupled to each other. Our approach might ease the synchronization between such organizations. Nonetheless, in such configurations new issues will arise like inter-organizational communication that should be researched.

Future research could also elaborate on technical aspects. Hitherto, we have used our own tool to create the pipeline. Therefore, different tools can be compared to each other and which advantages they offer in the context of EA model maintenance automation. Additionally, we assumed that short feedback cycles improve the EA model maintenance as experienced in software engineering. However, a lot of small tasks might bother the affected stakeholders. Therefore, research could be conducted to determine the “perfect” size of changes that should be processed with our pipeline.

Funding Open access funding provided by Royal Institute of Technology.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix: Interview questions and answers

EA background

1. Did you ever hear of Enterprise Architecture (EA) and EA modeling before participating in this evaluation?

Tester 1: Yes, not really much with the modeling but familiar with the notations.

Tester 2: Nein

Tester 3: Kontakt schon, aber selbst noch nicht benutzt

Tester 4: Ja, ich habe den Namen gehört und eine grobe Vorstellung davon, was es tut.

Tester 5: Ja

2. Have you ever been involved in EA modeling?

Tester 1: Not really

Tester 2: Nein

Tester 3: Nein

Tester 4: Nein

Tester 5: Nein

3. Have you ever used ArchiMate as a modeling tool for EA?

Tester 1: I modeled some very simple models in ArchiMate, but nothing industrial

Tester 2: Nein

Tester 3: Nein

Tester 4: Nein

Tester 5: Nein

4. Have you been aware of the difficulties of collaborative EA modeling?

Tester 1: Not really

Tester 2: Nein

Tester 3: im Prinzip schon, nicht auf EA bezogen, aber bei allen Varianten des kollaborativen Modellierens

Tester 4: Erst durch deine und Simons Arbeit

Tester 5: Nein

5. Are you aware of it now?

Tester 1: Yes, I think when two people are working on the same model, the process is difficult to decide which elements shall be inserted in the model. I am not sure if all these errors shall be alternated.

Tester 2: Ja, schon etwas mehr, durch den Tool support, der ja schon eher dürftig ist

Tester 3: Ist gleich geblieben

Tester 4: Alles Wissen, was ich davon hab, ist aus deiner Masterarbeit. Durch diese Anwendung habe

ich jetzt nicht mehr oder weniger Wissen darüber gewonnen.

Tester 5: Ja

6. Have you been aware of how to assess the quality of an EA model?

Tester 1: Yes, I understand quality of EA models from my knowledge of EA smells and EA debt. E.g. in terms of complexity, like how many elements shall be included in one model or also someone models the collection of business processes, what are the do's and do-nots?

Tester 2: Nein

Tester 3: Ich kenne das von normalen Modellen und Geschäftsprozessmodellen, von EA Modellen im Spezifischen aber nicht. Allerdings ist ein gewisses Wissen durch eine Ableitung von Beckers Geschäftsprozessen bekannt

Tester 4: ? War mir nicht bewusst, aber hätte ich mir vorstellen können, denn wo ist das denn nicht so? Bei jedem Business Artifact will man das ja haben.

Tester 5: Nein

7. Are you aware of it now?

Tester 1: Same as before

Tester 2: Schon mehr, also die grundlegende Idee habe ich verstanden

Tester 3: Ist gleich geblieben

Tester 4: Ist mir jetzt durchaus bewusster, weil wir ja auch Beispiele zu KPI's, also was interessiert einen an so einem Modell?

Tester 5: Ja

8. Have you been aware of how to assess the consistency of an EA model, i.e. in terms of syntactic or semantic consistency?

Tester 1: I cant give a precise answer, but syntactic if you use the same notation for the same purpose to model the same information and semantic consistency means you use the same word, so that you do not use the same words for the same meaning. Inconsistencies between model description and model representation.

Tester 2: Nein

Tester 3: Ja, nicht spezifisch auf EA Modelle, aber auf allgemeine Geschäftsprozessmodelle, aber ich glaube, es sind nicht genau die gleichen Inkonsistenzen wie in der EA Domäne

Tester 4: Syntaktisch und semantisch inconsistent wusste ich nicht, vermutet aber schon. Ich könnte nicht sagen, was typische semantische und syntaktische Inkonsistenzen sind, aber wie bei jedem Modell

hätte es mich gewundert, wenn es keine Inkonsistenzen gäbe.

Tester 5: Nein

9. Are you aware of it now?

Tester 1: Same as before

Tester 2: Kann man schon sagen, ja

Tester 3: Im Prinzip durch die definierten KPIs und durch die definierten Regeln, dass man eine einheitliche Detailtiefe in den Modellen hat, dass man bestimmte Regeln der eigentlichen syntaktischen und semantischen Regeln, die über die Sprache hinausgehen, einhalten muss und dadurch konsistenter bleibt.

Tester 4: Bin mir durchaus bewusster, weil ich es jetzt einmal angewandt habe, ich habe ein Modell selbst gesehen und bearbeitet. Allein dadurch hat man schon ein besseres Verständnis.

Tester 5: Nicht wirklich

CD background

1. Did you ever use a CD pipeline in the context of software engineering?

Tester 1: Yes, not complex one, e.g. last year in the lab and we created a pipeline setup. But never was involved in building pipelines in the industry.

Tester 2: Ja

Tester 3: Ja

Tester 4: Ja

Tester 5: Ja

2. Have you ever created a CD pipeline by yourself, e.g. using Jenkins or GitLab CD/CI?

Tester 1: Yes, GitLab CD/CI.

Tester 2: Ja

Tester 3: CI habe ich benutzt und selbst erstellt, CD habe ich benutzt aber nicht selbst erstellt

Tester 4: Ja

Tester 5: Ja

3. How experienced do you consider yourself in the practice of Continuous Delivery on a scale from 1 (very good) to 6 (completely new to it)?

Tester 1: 5

Tester 2: 2

Tester 3: 2

Tester 4: 2

Tester 5: 2

CD pipeline evaluation

1. How was your experience with using our CD pipeline for integrating the changes you applied to the EA model in the enterprise wide EA model?

Tester 1: I think it was quite easy, like I committed my changes and triggered only one button, so overall it was really easy, the steps are very short. Also that there is information whether my pipeline is still running or already completed, so right on the webpage, so I think it was really nice. I was just confused about the names of the repository.

Tester 2: Die Idee ist gut, was noch besser sein könnte, ist dass wenn in einer der Stages nicht funktioniert, also wenn ich einen Fehler gemacht habe, der durch die Assessments entdeckt wird, dann ist das immer noch ein sehr inkonsistenter Workflow. Also die Gegebenheiten wurden geschaffen, das ist gut, aber die Feedback loop müsste angepasst werden, also dass man nicht erst eine Fehler sieht, wenn man bereits committed hat.

Tester 3: Aus Sicht des Users ist es schön, dass man damit eigentlich nichts zu tun hat und das alles im Hintergrund geschieht. Es reicht zu wissen, dass es eine gibt, das Wissen wie die Pipeline arbeitet, ist für mich als Endnutzer nicht wichtig. Der Output für fehlerhafte Pipeline runs ist hilfreich. Allerdings muss ich als Anwender nicht wissen, welche Stage genau gefailed ist. Es reicht eigentlich eine Fehlermeldung, die mir sagt, QualityGate hat das Modell nicht approved aus Grund x.

Tester 4: War gut, konnte ich als Blackbox nutzen und dann hat es seinen Job gemacht. Das ist genau das, was ich als Nutzer von so einer CD pipeline erwarte, dass wenn ich Änderungen commite, dass die Pipeline dann auch am Ende ein Modell deployed, wenn es die Qualität erfüllt.

Tester 5: + Easy and very similar to collaborative programming and therefore an intuitive approach + Automatic checks ensures that certain quality is maintained

- Automatic triggering and showing who triggered it would be nice (although I know that JARVIS doesn't provide this currently).

2. Were you able to deploy a new central EA model with our CD pipeline for EA model evolution?

Tester 1: Yes

Tester 2: Ja

Tester 3: Ja

Tester 4: Ja

Tester 5: Ja

3. If not, what was the reason for it?

Tester 1: -
 Tester 2: -
 Tester 3: -
 Tester 4: -
 Tester 5: -

4. Even though you were not able to break syntax and semantics of the model since you used Archi for modeling, which in some way prevents this: Assume after you saved the changes you applied to the model and afterwards you ran a script which you developed several months ago for optimizing the formatting of the model. Also consider that there was a bug in your script, which broke the syntactic and semantic consistency of your model. How do you assess the value of an automated approach to detect such consistency errors?

Tester 1: 2, The checking itself is valuable but without getting the feedback of which elements are broken in terms of consistency, it is not valuable.

Tester 2: 1, wichtige QS Maßnahme wie immer

Tester 3: 1, Sehr wichtig, denn wenn ich da einen Fehler im XML habe, werde ich nicht mehr in der Lage sein, das Modell in Archi zu importieren. Im Falle der Syntax. Im Falle der Semantik ist es ja dann noch wichtiger, denn, wenn wir davon ausgehen, dass semantisch inkorrekte Modelle im Archi Tool importiert werden können, ist es mir ohne einen automatisierten Test ja nicht einmal möglich zu erkennen, dass das Modell fehlerbehaftet ist. Bei verschiedenen Tools ist das ja noch schlimmer, da nicht deckungsgleiche Modelle gleichermaßen verwendet werden können.

Tester 4: 1, Das ist ja quasi mein Quality Gate, also sehr wertvoll.

Tester 5: 1, Man sollte nie ein falsches Modell hochladen, deshalb ist es immer wertvoll, wenn das vernünftig gechecked wird.

5. Do you consider our approach to detect such inconsistencies as valuable?

Tester 1: 5, due to the missing feedback

Tester 2: 1, sehe ich keine Probleme mit

Tester 3: 2, Das ist natürlich sehr nützlich, wir haben damit das abgedeckt, was man allgemeingültig in Unternehmen überprüfen kann. Wenn die Checks darüber hinaus geht, wird das unternehmensspezifisch. Dafür bräuchte man dann eine externe Instanz, die die Regeln definieren. Z.B. dass man definiert, dass alles auf Englisch beschrieben werden muss.

Tester 4: 1, Von meinem Wissen, das ich über EA Modells habe, klingen beide Approaches sinnvoll.

Tester 5: 1

6. Do you consider to automatically calculate KPIs for EA models as valuable?

Tester 1: This is very valuable

Tester 2: 1, aber konkrete Instanzen hängen von den KPIs ab. Das ist die Grundmetrikproblematik.

Tester 3: 1, Ja, sehr sinnvoll, den genau so wie beim SE hat man gewisse Qualitätsansprüche, die man an das Modell stellt, und die müssen ja auch gemessen und überprüft werden.

Tester 4: 1, sehr sinnvoll, weil das auch eine QS Maßnahme ist.

Tester 5: 2, Was genau ich aus den einzelnen KPIs interpretieren soll, weiß ich jetzt noch nicht so genau, aber in Kombination mit den Quality Gates finde ich das sehr sinnvoll, dass die Qualität dadurch weiter hochgehalten wird und gar nicht weiter promoted wird.

7. Do you consider our approach to automatically calculate KPIs for EA models as valuable?

Tester 1: 5, The checking itself is very valuable but again if the feedback is missing the reason for the invalidation, it is not valuable

Tester 2: Muss ich mich enthalten. Das hängt sehr von der DSL ab, wie gut sie passt. Kann die Vorteile einer DSL haben, dass gewisse Nutzergruppen die Implementierung ermöglicht, die sonst diesen Nutzergruppen vorenthalten werden. Nachteil ist, es muss natürlich eine passende DSL sein, wenn sie zu limitiert oder zu generisch ist, dann kann ich wieder eine allgemeine Programmiersprache nehmen und da steht und fällt dann der Nutzen einer DSL. Das ist auch ein zusätzlicher Aufwand in der Wartung des Gesamtsystems.

Tester 3: 2, An und für sich auch sehr nützlich, muss ich aber von der Mächtigkeit der Sprache abhängig machen. Davon ausgehend, dass die Sprache dahingehend erweitert werden kann, dass wir den gesamten EAM KPI Katalog abbilden können, würde ich dem Ansatz eine 2 geben.

Tester 4: 2, Du musst erstmal die DSL erlernen und kennen und es ist immer die Frage bei so DSLs, ob sie wirklich alles abbilden können, als was ich programmatisch abdecken könnte.

Tester 5: 1, Ich finde das sehr gut, weil man sehr flexibel KPIs definieren kann und das passt auch zum JARVIS Framework, weil man dadurch wieder sehr flexibel die QGates definieren kann.

8. Do you consider our check for disconnected components as valuable for EA model maintenance?

Tester 1: 2, 5, in general this is valuable, but due to the missing feedback why the model is disconnected

Tester 2: 4, Ich sehe das als Geschmacksfrage, ob man disconnected components erlauben möchte.

Tester 3: 3, Also disconnected Components sollte es eigentlich nicht geben in einem EA Modell. Wenn es sowas aber doch in der Realität gibt, dann hätte ich lieber eine Warnung, dass es disconnected Components gibt, als den direkten Fail der Pipeline.

Tester 4: Enthaltung, Unter der Voraussetzung, dass EA Modelle immer geschlossen sein sollten, sehe ich das sinnvoll an. Wenn die Prämisse nicht erfüllt ist, ist das eher schädigend, weil ich im Prozess unterbrochen werde, obwohl er weiterlaufen sollte. Daher enthalte ich mich hier.

Tester 5: 1, Das ist bei mir der Fall gewesen, v.a. das kann man einfach überprüfen. Ich hab das einfach vergessen bzw. das ist mir nicht aufgefallen und da dann das No-Go des QGates aufgrund dieses Tests zu bekommen, war da sehr hilfreich.

9. Do you consider our implementation for removing too detailed elements and relationships from the EA model as valuable?

Tester 1: 5, Warning would be better than removing them.

Tester 2: 1, Kann sehr wertvoll sein, hängt aber natürlich davon ab, wie gut es funktioniert und wie sehr das im konkreten Kontext gewünscht ist. Die Detailtiefe liegt immer im Auge des Betrachters. Wenn es anpassbar ist und technisch problemlos funktioniert, ist das ein wertvolles Feature.

Tester 3: 3, Vom Gefühl her halte ich es für sinnvoller, wenn man das Modell nicht akzeptiert und der Modellierer das von sich aus löschen kann. Also lieber eine Markierung als dieser harte Eingriff in das Modell.

Tester 4: 4, Erachte ich organisationsweit als schwierig, da ich glaube, dass man je nach Organisation mehr oder weniger Details erlauben möchte. Ich finde es schwierig zu beurteilen, wenn man noch nicht so mit EA Modellen gearbeitet hat und ich nicht weiß, wie weit man hier typischerweise verzweigt oder aggregiert.

Tester 5: 2, Generell finde ich das eigentlich ganz gut, aber eventuell kann es ja auch Prozesse geben kann, die man durchaus in der Detailtiefe haben möchte. Da fände ich es besser, wenn man die Detailtiefe selbst einstellen oder abstellen könnte.

10. If you consider it in between somewhat valuable and not valuable, why do you consider it that way? (Was answered in question 9).

11. Do you consider a CD/CI approach as a suitable approach to automate the maintenance of EA models?

Tester 1: 2, I think the basic idea for automating several steps of the quality assurance is really nice. However, I think many steps cannot really be automated in practice.

Tester 2: 1, Als Basis auf jeden Fall, wäre auch schön weitere Mechanismen zu haben, aber als Basis QS ist das sehr plausibel.

Tester 3: 2, CD/CI ist grundsätzlich ein sinnvoller Ansatz, um die verschiedenen Projekte mit einander zu verknüpfen und die Änderungen am Modell zu verteilen.

Tester 4: 1, Weil es im Grund auch einfach nur eines Prozess automatisiert, der sonst manuell durchgeführt werden würde. Wenn ich also anstelle dessen einfach einen Menschen drüber schauen lassen würde, habe ich die selben Schwierigkeiten, wie wenn ich das bei Code mache. Ob ich Code anpasse oder die Architektur des Unternehmens anpasse, ist für mich ein ähnlicher Use Case.

Tester 5: 1, Das ist einfach ein automatischer Weg ist, wie die beiden Modelle gemerged werden, und stellt währenddessen auch sicher, dass die Qualität hoch gehalten wird und nimmt einem auch eine Menge an Arbeit ab. Für Software Engineers ist das dazu noch sehr intuitiv, da man die Arbeit mit VC und CD Pipelines bereits von Codeebene kennt.

12. Which improvements do you see for the future for our CD pipeline for EA model maintenance?

Tester 1: warning instead of failing for many checks, integrate the tool with some kind of EA smell registry and all the factors which are registered in a registry. So an automation process for keeping the registry up-to-date, more relation between the result of the checks and the models, which explain why the result of the check is this way. What are the elements which contribute to this result?, some kind of hints how I should change the model to improve the certain KPI or quality factor, before your commit you are hinted towards that the quality is not met

Tester 2: Also die Gegebenheiten wurden geschaffen, das ist gut, aber die Feedback loop müsste angepasst werden, also dass man nicht erst eine Fehler sieht, wenn man bereits committed hat. Ein Verständnis dieser Pipeline zu gewinnen ist am Ende auch wichtig. Dadurch spielt die Benennung der Stages und Aktivitäten eine wichtige Rolle. Dadurch ist es aus Domänenperspektive des Endnutzers nicht hilfreich -> bessere Benennung und Beschreibung

Tester 3: Richtung Merge conflicts. Im Moment ist das ja kollaborativ im Sinne von hintereinander. Kollaborativ im Sinne von gleichzeitig arbeiten wäre natürlich noch deutlich spannender.

Tester 4: Aktuell können wir nicht parallel arbeiten. Sowas wie feature branches und unterschiedliche Strategien, die man vllt unternehmerisch verfolgen könnte, darzustellen. Also ggf. habe ich eine gewisse Strategie auf operativer Ebene, die mein Modell beeinflusst, und eine auf marketing Ebene, die das Modell völlig anders beeinflussen können. Also denkbar wäre eine Auswahl von n Branches, auf denen im Projekt gearbeitet werden kann.

- Tester 5: a. Automatic triggering of pipeline
b. More detailed feedback on what went wrong
c. Show who triggered the pipeline

Quality of the maintenance process

1. Comprehensibility: The process model can easily be understood

Tester 1: 1

Tester 2: 2

Tester 3: 1

Tester 4: 1, So wie du es erklärt hast, sehr verständlich, von der Grafik her, mit der Erklärung der JARVIS Syntax, ist es auch sehr verständlich.

Tester 5: 1, Die Namen der Aktivitäten und Stages sind selbstaussagend und gut nachzuvollziehen.

2. Effectiveness: The observed process is well suited for keeping an enterprise architecture model up-to-date. It clearly implements the functionality it is intended to offer

Tester 1: 1

Tester 2: 2, Wenn die QG keine Probleme bereiten im Sinne von, dass sie fälschlicherweise Fehler erkennen, dann würde ich das als effektiv betrachten.

Tester 3: 3, Der ist grundsätzlich dafür nützlich, aber der vermischt 2 Sachen. Beim SE wenn ich dem Git-Flow folge habe ich den Feature Branch und wenn ich entscheide, das Feature ist abgeschlossen, dann möchte ich das Feature commiten. Hier soll ja nach jedem Commit die Pipeline getriggert werden und jeder Stakeholder sein Approval abgeben. Das kann natürlich sein, dass man das macht, weil man keine Merge Conflicts bekommen möchte, aber ich würde stärker unterscheiden zwischen, ich habe etwas modelliert und möchte das speichern und zwischen etwas, was die Qualitätsmerkmale und KPIs erfüllt und auf der anderen Seite dass ich etwas fertig modelliert habe und ich das mit dem zentralen Modell integrieren möchte.

Tester 4: 2, Ich würde eine Stufe runter gehen, weil ich die Branches noch nicht sehe, und jedes Unternehmen, das EA Modelle anwenden würde, würde auch verschiedene Projekte haben und bräuhete dementsprechend auch verschiedene Branches.

Tester 5: 2, An und für sich sehr gut, nur das Stakeholder Approval erschwert die Praktikabilität des Prozess Modells. Oft vergisst man ja noch ein kleines Teil im Modell hinzuzufügen und dann hat man das schon committed und die Pipeline getriggert und dann muss man das alles nochmal machen und auf die Approvals aller Stakeholder warten.

3. Completeness: The observed process contains all necessary process steps to maintain an enterprise architecture, keep it up-to-date and to assure a high degree of quality.

Tester 1: 4, a warning and some checks are possibly missing

Tester 2: Ich muss mich enthalten, weil mein Verständnis nicht ausreicht, um mir weitere Schritte einfallen lassen zu können.

Tester 3: 1, Man könnte höchstens etwas haben wie andere KPIs oder andere Überprüfungen

Tester 4: 1, Klingt für mich alles gut. Ich vermisste gerade eigentlich gar nichts.

Tester 5: 1, Da fällt mir einfach nichts weiter ein, was man da machen könnte.

4. Minimality: The observed process contains only those process steps that are necessary to maintain an enterprise architecture model.

Tester 1: 1

Tester 2: 1, Abseits des disconnected components check, der Geschmackssache ist, totally agree

Tester 3: 1

Tester 4: 1, Ich wüsste nicht, was ich hier als zu viel betrachten würde.

Tester 5: 2, Ich habe jetzt keine Schritte erkannt, die unnötig sind. Eventuell könnte man den Schritt mit den Stakeholder approval weglassen, also dass man nur die Stakeholder über die Änderungen informiert.

5. Efficiency: In terms of time the observed process maintains changes into an enterprise architecture model efficiently. The new central EA model is deployed quickly after triggering the maintenance process.

Tester 1: Enthaltung

Tester 2: Enthaltung, Hier könnte man die Email technisch austauschen mit einer anderen Technik. Vielleicht ist das Versenden an Emails und das Collecten an Approvals organisatorisch auch nicht das Gelbe vom Ei. Mehr Anpassbarkeit wäre hier sicher sinnvoll.

Tester 3: Enthaltung, Das wird durch die Approvals wahrscheinlich sehr langsam werden, daher kann man hier aktuell keine Aussage zu treffen.

Tester 4: Enthaltung

Tester 5: Enthaltung

References

- Alegria, A., Vasconcelos, A.: It architecture automatic verification: a network evidence-based approach. In: 2010 Fourth International Conference on Research Challenges in Information Science (RCIS), pp. 1–12 (2010)
- Altmanninger, K., Kotsis, G.: Towards accurate conflict detection in a VCS for model artifacts: a comparison of two semantically enhanced approaches. In: Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling, vol. 96, Citeseer, pp. 139–146 (2009)
- Barjis, J., Kolfshoten, G.L., Verbraeck, A.: Collaborative enterprise modeling. In: Proper, E., Harmsen, F., Dietz, J.L.G. (eds.) *Advances in Enterprise Engineering II*, pp. 50–62. Springer, Berlin (2009)
- Bass, L., Weber, I., Zhu, L.: *DevOps: A Software Architect's Perspective*, 1st edn. Addison-Wesley Professional, Boston (2015)
- Bebensee, B., Hacks, S.: Applying dynamic Bayesian networks for automated modeling in archimate: a realization study. In: 2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW), pp. 17–24 (2019). <https://doi.org/10.1109/EDOCW.2019.00017>
- Becker, J., Probandt, W., Vering, O.: *Grundsätze ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement. BPM kompetent*. Springer, Berlin (2012)
- Burnstein, I.: *Practical Software Testing: A Process-Oriented Approach*. Springer, Berlin (2006)
- Buschle, M., Ekstedt, M., Grunow, S., Hauder, M., Matthes, F., Roth, S.: Automating enterprise architecture documentation using an enterprise service bus. In: 18th Americas Conference on Information Systems (2012)
- van Buuren, R., Jonkers, H., Iacob, M.E., Strating, P.: Composition of relations in enterprise architecture models. In: International Conference on Graph Transformation, Springer, pp. 39–53 (2004)
- Chen, L.: Continuous delivery: overcoming adoption challenges. *J. Syst. Softw.* **128**, 72–86 (2017)
- De Lucia, A., Fasano, F., Scanniello, G., Tortora, G.: Enhancing collaborative synchronous uml modelling with fine-grained versioning of software artefacts. *J. Vis. Lang. Comput.* **18**(5), 492–503 (2007)
- Debois, P.: Agile infrastructure and operations: how infra-gile are you? In: Agile 2008 Conference, pp. 202–207 (2008)
- Dewan, P., Hegde, R.: Semi-synchronous conflict detection and resolution in asynchronous software development. In: ECSCW 2007, Springer, pp. 159–178 (2007)
- Dollmann, T., Houy, C., Fettke, P., Loos, P.: Collaborative business process modeling with comomod: a toolkit for model integration in distributed cooperation environments. In: 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 217–222 (2011). <https://doi.org/10.1109/WETICE.2011.36>
- Döring, J.S.: *An architecture for self-organizing continuous delivery pipelines*. Master's Thesis, RWTH Aachen (2018)
- Duvall, P., Matyas, S.M., Glover, A.: *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, Boston (2007)
- 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)
- 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 (2012)
- Fischer, R., Aier, S., Winter, R.: A federated approach to enterprise architecture model maintenance. In: EMISA (2007)
- Fowler, M.: Continuous integration (2006). <http://martinfowler.com/articles/continuousIntegration.html>
- Fowler, M.: *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, Boston (2018)
- Guild, B.A.: *A Guide to the Business Architecture Body of Knowledge (BIZBOK Guide)*, vol. V04 (2014)
- Hacks, S., Lichter, H.: A probabilistic enterprise architecture model evolution. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC), pp. 51–57 (2018). <https://doi.org/10.1109/EDOC.2018.00017>
- 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)
- Hacks, S., Lichter, H.: Qualitative comparison of enterprise architecture model maintenance processes. In: 40 Years EMISA 2019 (2020)
- Hacks, S., Steffens, A., Hansen, P., Rajashekar, N.: A continuous delivery pipeline for EA model evolution. In: Reinhartz-Berger, I., Zdravkovic, J., Gulden, J., Schmidt, R. (eds.), *Enterprise, Business-Process and Information Systems Modeling. BPMDS 2019, EMMSAD 2019*, Springer International Publishing, pp. 141–155 (2019)
- 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 (2011)
- Hauder, M., Matthes, F., Roth, S.: Challenges for automated enterprise architecture documentation. In: TEAR/PRET (2012)
- Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *MIS Q.* **28**(1), 75–105 (2004)
- Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. *IEEE Comput.* **34**, 120–122 (2001)
- Holm, H., Buschle, M., Lagerström, R., Ekstedt, M.: Automatic data collection for enterprise architecture models. *Softw. Syst. Model.* **13**(2), 825–841 (2014)
- Humble, J., Farley, D.: *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, 1st edn. Addison-Wesley Professional, Boston (2010)
- ISO, IEC, IEEE: *Systems and Software Engineering: Architecture Description* (2011)
- 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)
- 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)
- Jugel, D., Schweda, C.M., Zimmermann, A.: Modeling decisions for collaborative enterprise architecture engineering. In: Persson, A., Stirna, J. (eds.) *Advanced Information Systems Engineering Workshops*, pp. 351–362. Springer, Cham (2015)
- Khosroshahi, P.A., Aier, S., Hauder, M., Roth, S., Matthes, F., Winter, R.: Success Factors for Federated Enterprise Architec-

- ture Model Management. In: A. Persson, J. Stirna (eds.) *Advanced Information Systems Engineering Workshops, Lecture Notes in Business Information Processing*, Springer International Publishing, pp. 413–425 (2015)
38. 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)
 39. Kleehaus, M., Hauder, M., Uludag, Ö., Matthes, F., Villasana, N.C.: It landscape discovery via runtime instrumentation for automating enterprise architecture model maintenance. In: *AMCIS* (2019)
 40. Kotusev, S.: The history of enterprise architecture: an evidence-based review. *J. Enterp. Archit.* **12**(1), 31–37 (2016)
 41. Landthaler, J., Uludag, Ö., Bondel, G., Elnaggar, A., Nair, S., Matthes, F.: A machine learning based approach to application landscape documentation. In: Buchmann, R.A., Karagiannis, D., Kirikova, M. (eds.) *The Practice of Enterprise Modeling*, pp. 71–85. Springer, Cham (2018)
 42. Laukkanen, E., Itkonen, J., Lassenius, C.: Problems, causes and solutions when adopting continuous delivery: a systematic literature review. *Inf. Softw. Technol.* **82**, 55–79 (2017)
 43. Lim, N., Lee, T.g., Park, S.g.: A comparative analysis of enterprise architecture frameworks based on EA quality attributes. In: 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, pp. 283–288 (2009)
 44. 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. *J. Syst. Softw.* **123**, 263–291 (2017)
 45. Matthes, F., Monahov, I., Schneider, A.W., Schulz, C.: *EAM KPI Catalog v1.0*. Garching (2011)
 46. Nakakawa, A., Bommel, P., Proper, H.: Towards a theory on collaborative decision making in enterprise architecture. pp. 538–541 (2010). https://doi.org/10.1007/978-3-642-13335-0_40
 47. Niemi, E., Pekkola, S.: Enterprise architecture quality attributes: a case study. In: 2013 46th Hawaii International Conference on System Sciences, IEEE, pp. 3878–3887 (2013)
 48. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. *J. Manag. Inf. Syst.* **24**(3), 45–77 (2007)
 49. Plataniotis, G., De Kinderen, S., Proper, H.A.: Ea anamnesis: an approach for decision making analysis in enterprise architecture. *Int. J. Inf. Syst. Model. Des. (IJISMD)* **5**(3), 75–95 (2014)
 50. Prilla, M., Nolte, A., Herrmann, T., Kolfschoten, G., Lukosch, S.: Collaborative usage and development of models: state of the art, challenges and opportunities. *Int. J. e-Collab. (IJeC)* **9**(4), 1–16 (2013)
 51. Rajashekar, N., Hacks, S., Silva, N.: A performance comparison of graph analytic methods for supporting enterprise architecture model maintenance. In: Feltus, C., Johannesson, P., Proper, E. (eds.), *Practice of Enterprise Modelling 2019 Conference Forum*, CEUR-WS.org, pp. 54–63 (2019)
 52. Rittgen, P.: Coma: a tool for collaborative modeling. *CAiSE Forum* **344**, 61–64 (2008)
 53. Rumpe, B.: *Agile Modeling with UML: Code Generation, Testing Refactoring*. Springer, Berlin (2017)
 54. 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)
 55. Salentin, J., Hacks, S.: Towards a catalog of enterprise architecture smells. In: Gronau, N., Heine, M., Poustchi, K., Krasnova, H. (eds.), *WI2020 Community Tracks*, GITO Verlag, pp. 276–290 (2020)
 56. Sandkuhl, K., Stirna, J., Persson, A., Wißotzki, M.: Enterprise modeling. In: *Tackling Business Challenges with the 4EM Method*, vol. 309, Springer (2014)
 57. Simon, D., Fischbach, K., Schoder, D.: An exploration of enterprise architecture research. *Commun. Assoc. Inf. Syst.* **32**(1), 1–72 (2013)
 58. Spanoudakis, G., Zisman, A.: Inconsistency management in software engineering: survey and open research issues. In: *Handbook of Software Engineering and Knowledge Engineering: Fundamentals*, vol. II, World Scientific, pp. 329–380 (2001)
 59. Ssebuggwawo, D., Hoppenbrouwers, S., Proper, E.: Assessing collaborative modeling quality based on modeling artifacts. In: van Bommel, P., Hoppenbrouwers, S., Overbeek, S., Proper, E., Barjis, J. (eds.) *The Practice of Enterprise Modeling*, pp. 76–90. Springer, Berlin (2010)
 60. 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 (2018)
 61. The Open Group: ArchiMate 3.0.1 Specification (2017)
 62. The Open Group: The open group archimate model exchange file format. <https://www.opengroup.org/open-group-archimate-model-exchange-file-format> (2019)
 63. Timm, F., Hacks, S., Thiede, F., Hintzpeter, D.: Towards a quality framework for enterprise architecture models. In: Lichter, H., Anwar, T., Sunetnanta, T. (eds.), *Proceedings of the 5th International Workshop on Quantitative Approaches to Software Quality (QuASoQ 2017) Co-located with APSEC 2017, CEUR-WS.org*, pp. 10–17 (2017)
 64. Uludag, , Kleehaus, M., Caprano, C., Matthes, F.: Identifying and structuring challenges in large-scale agile development based on a structured literature review. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC), pp. 191–197 (2018)
 65. Välja, M., Korman, M., Lagerström, R., Franke, U., Ekstedt, M.: Automated architecture modeling for enterprise technology management using principles from data fusion: a security analysis case. In: *Portland International Conference on Management of Engineering and Technology (PICMET)* (2016)
 66. 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)
 67. 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)
 68. Yin, R.K.: *Case Study Research: Design and Methods*, 5th edn. Sage Publications, Thousand Oaks (2013)
 69. Ylimäki, T.: Potential critical success factors for enterprise architecture. *J. Enterp. Archit.* **2**(4), 29–40 (2006)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Alex Richard Sabau born 1989 in Braşov, Romania, graduated as M.Sc. in Computer Sciences at the RWTH Aachen in 2020. Since his graduation he works as a research assistant as the software construction chair at the RWTH Aachen under the direction of Prof. Dr. Horst Lichter with the research topic “DevOps in heterogeneous organizations”. His scientific interests include the creation of agile cultures, knowledge organization and management in distributed teams and the improvement of Enterprise Architecture Management.

Enterprise Architecture Management.

Andreas Steffens is a member of the research group on software construction at RWTH Aachen University. His research interests focus on DevOps, Continuous Delivery and Cloud and their impact on fields like architecture, quality, and processes. Currently he is vice-speaker of the working group on DevOps and Microservices in German Informatic Society. In the emerging field of DevOps he organized several national and international scientific workshops.

Simon Hacks is a postdoc in the Software Systems Architecture and Security group, Computer Science at KTH Royal Institute of Technology, Sweden. He received his Ph.D. from the RWTH Aachen University, Germany in the field of Enterprise Architecture Models. He supervised several theses related to Enterprise Architecture and supported the teaching of lectures in Software Engineering, Software Quality Assurance, Object Oriented Software Construction, and Software Project

Management. He is Co-Chair of the VEnMo workshop, PC member and reviewer for several national and international journals and conferences/workshops.