Automation Processes for Enterprise Architecture Management

Matthias Farwick, Berthold Agreiter, Ruth Breu
University of Innsbruck
Institute of Computer Science
Innsbruck, Austria
{matthias.farwick, berthold.agreiter, ruth.breu}@uibk.ac.at

Abstract—Creating and maintaining an enterprise architecture model that is both up-to-date and accurate is a difficult task due to the size and complexity of the models and the dispersed nature of EA information in organizations. In current EA maintenance processes, the models are maintained manually with only little automation, which is a time consuming task. Literature from research and practice has identified this challenge, however only few scientific publications actually address the issue of EA model maintenance and its automation. In our research effort on Living Models, we work towards solutions for a closer connection between EA models and what they represent in the real world. In this article we present (semi-)automated processes for maintaining enterprise architecture models by gathering information from both human input and technical interfaces and discuss implementation issues for realizing the processes in practice. This work is one of the first steps in the direction of minimizing manual work for EAM by automation and increasing EA data quality attributes such as consistency and actuality.

Keywords—enterprise architecture; maintenance; process; model; management; it-governance

I. INTRODUCTION

Enterprise Architecture Management (EAM) is a practice employed in mid-sized to large organizations that aims on capturing the relationship between the business and the supporting IT-landscape. By documenting the current status of this network of dependencies, the alignment of business and IT can be analyzed, and transformed to an architecture that optimally supports the business strategy. Other benefits of EA are the ability to assess risks, standardize the IT and check compliance with legal regulations. Several enterprise architecture (EA) frameworks have been proposed and are in use today, such as The Open Group Architecture Framework (TOGAF) [1], or the Zachman Framework [2]. Commonly, these frameworks define a metamodel used to describe the EA data, organizational best practices and governance mechanisms.

In the practice of enterprise architecture, the corresponding EA models can grow very large and expose complex relationships between EA model elements. Also, these large and complex models continuously need to be aligned with the real–world enterprise which they are supposed to represent, in order to be most effective. For example mergers & acquisitions, new business strategies, changing market requirements, or the exploration of new markets can immensely change the business and IT architecture of an enterprise. This combination of complex and evolving models implies that a large effort needs to be put into keeping EA models in–sync with what they represent in the real world. This is one of the most costly aspects of EA in general.

Enterprise Architecture tools are used to tackle the complexity of modeling and data collection. In 2004, ter Doest and Lankhorst stated in [3]: “We expect that in 7 years from now, enterprise architecture will be a real-time tool for management and redesign of the enterprise for better performance, flexibility and agility. (...) There will no longer (be) a boundary between the descriptive nature of architecture models and the operational side comprising a.o. business process management and IT management.”. Looking at the frameworks, processes and tool support of enterprise architecture today, one can state that this prediction is not fulfilled. According to a survey, EA tools are still mostly decoupled from the reality and need manual update mechanisms to stay in–sync with the real world [4].

Regardless, still only little research literature can be identified that tackles the maintenance of EA models (see Section VI). Or, as stated by Buckl et al. “Identifying, gathering, and maintaining knowledge about the EA is a challenge, which is only recently addressed by isolated approaches (...)” [5].

In our research on Living Models (cf. [6]), we are working towards solutions for a closer connection between EA models and what they represent in the real world. For this reason, we first conducted a survey among 29 international EA practitioners to find out the status quo in EA maintenance processes [7]. The survey confirmed our claim that more automation, less manual labor, and more current and trustworthy data is desired in EAM. Moreover, the survey served as a basis for deriving requirements for a tool which supports semi-automated EA maintenance. The publication at hand is part of this ongoing effort to enable automated maintenance of enterprise architecture models. In this contribution we take a subset of the requirements we have identified in [7], and focus on the processes for gathering and maintaining
EA data, supported by the envisioned EA tool. We present semi-automated processes for EA data collection and quality assurance. Similar to the requirements, also the processes described in this paper were developed together with EA practitioners.

The contribution of this paper is twofold. First, we focus on how EA maintenance processes described in EA literature can be extended to meet the requirements for automated EA maintenance. Thereby, we focus on how the data quality attributes such as consistency and actuality can be ensured by the process. The aim of this process description is to provide the basis for a future technical implementation. As a second contribution, we discuss implementation aspects for such a process. We believe that the work in this direction lays the basis for future automation of EA maintenance. The remainder of this paper is structured as follows. In the next section we give a high level overview of the involved processes, the involved participants, and the required data structures. The following section, Section III, then explains the details of the proposed processes. How these processes support our requirements for automated EAM is described in Section IV. After that we discuss some implementation issues in Section V. In the following Section VI we highlight related work in the area of enterprise architecture maintenance. We then conclude the article in Section VII by pointing out the future direction of our research.

II. AUTOMATED EA MODEL MAINTENANCE PROCESSES

As already stated above, we have elaborated a list of requirements for automated EA model maintenance in [7], based on our literature review, a survey among EA practitioners, and our own experience. Several of these requirements demand an according EA maintenance process. A summary of the requirements can be seen in Table I.

In this section, we propose a computer-aided process that implements these requirements. For its description we use the Business Process Modeling Notation 2.0 (BPMN)¹ [8].

The foundation of our processes is an EA repository that not only stores, collects, and consolidates EA data, but is also capable of supporting the data maintenance process with task lists, and a corresponding role system. It is our general aim to increase automation in EA maintenance. Therefore, data sources can be integrated into the data collection process, so that information is retrieved from where it is available, instead of entering the same data multiple times - in the corresponding information system and the EA tool. However, it is unrealistic to assume that data coming from an external source always offers appropriate quality, and does not introduce inconsistencies. For this reason, each data source needs to have an owner assigned to it who can be contacted in order to resolve such issues. In case manual intervention by a data owner is needed, she is presented with a task list in the EA tool which lists the consolidation steps that need to be executed. In addition, data that is stored in the EA model has meta data attached to it, such as an expiry date and the data origin. These attributes are used to drive the maintenance process (see Section II-B), and hence to keep the data quality high. Our overall goal is to maximize the automation of the maintenance process, however, we do not assume that full automation is possible. The reason for this is the realization that some tasks will never be possible to automate such as defining the correct level of abstraction of EA data for making decisions.

In our process definition, we focus on increasing the data quality and actuality, and especially on data consistency. The result is, that in our approach there always exist two EA model instances. One approved as-is model, and one model we call the working copy model. The approved model is the model that is used by EA stakeholders, such as the CIO, to analyze the current EA. No changes can be directly applied to the approved model. Each change first has to be applied to the working copy, which goes through a maintenance cycle and is locked from changes, to become the new approved model. If the focus is laid more on data actuality, a non-blocking process needs to be created which we do not discuss in this article.

An overview of the overall maintenance process can be seen in Figure 1. First, the process participants need to be defined (step 1). This includes that roles are assigned to users in the EA tool. Depending on these roles, users have different views and are notified on different events. The required roles are presented in Section II-A. After that, the attached data sources need to be set up in step 2 (see Section III-A for specifics of the setup process). This task includes a subprocess to define data mappings from the incoming data onto the internal data model, and also defines the responsibilities for each data source. Here, we call a data source Management Data Repository (MDR). Examples for such MDRs are project portfolio management tools, business process management tools or even sensors in application servers which report the deployed applications. With step 3, the maintenance cycle starts. In this cycle, the proposed changes to the EA model and the automatically collected data are integrated into a new consolidated as-is model. Optionally, the release subprocess (step 4) is executed, that allows the complete set of stakeholders to cast vetoes against changes. After a maintenance cycle, new MDRs can be added or a new maintenance cycle can be started after a specific time has passed. The maintenance cycle and the setup process are further explained in the following subsection.

¹In BPMN 2.0 cogwheels denote automated tasks, person icons denote human tasks, three parallel lines in a task denote possible multiple instances, a plus sign in a task denotes a subprocess contained in a task, arrows starting with a diamond denote conditional flows and arrows starting with a short crossing line denote the default flow.
<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR 1</td>
<td>The collection of EA data must be federated from the repositories of the data owners (departments etc.)</td>
</tr>
<tr>
<td>OR 1</td>
<td>An organizational process must be in place that regulates the maintenance of EA model</td>
</tr>
<tr>
<td>OR 1.1</td>
<td>The organizational maintenance process must be supported by a technical process</td>
</tr>
<tr>
<td>OR 2</td>
<td>Each data source must have an owner/responsible</td>
</tr>
<tr>
<td>IR 1</td>
<td>The system must be able to detect changes in the real-world enterprise architecture</td>
</tr>
<tr>
<td>IR 1.1</td>
<td>The system must provide a mechanism to define the mapping from incoming data to the internal data structure</td>
</tr>
<tr>
<td>IR 2</td>
<td>The system must have a machine understandable internal data structure</td>
</tr>
<tr>
<td>DQR 1</td>
<td>The system must provide mechanisms that help the QA team to ensure data consistency</td>
</tr>
<tr>
<td>DQR 2</td>
<td>The system must provide mechanisms to ensure data actuality that is sufficient for the EA goals</td>
</tr>
<tr>
<td>DQR 2.1</td>
<td>Each element in the systems data structure must have a creation time stamp and an expiration date (volatility)</td>
</tr>
<tr>
<td>DQR 4</td>
<td>The system must provide mechanisms that allow for the automated propagation of changes</td>
</tr>
<tr>
<td>DQR 5</td>
<td>The system must be able to identify and resolve data identity conflicts from different sources via identity reconciliation</td>
</tr>
</tbody>
</table>

Table I

A subset of our requirements for automated EA model maintenance defined in [7].

![Figure 1. Overview of the maintenance process modelled in BPMN 2.0.](image)

A. Process Participants

The participants of the processes consist of persons with specific roles and automated process participants such as the EA repository and the MDRs. The roles defined here roughly correspond to the roles defined in [9], with additional responsibilities, that are described in the following.

**EA Coordinator:** The EA coordinator is the overall responsible for the EA data quality, and initiates each maintenance cycle (see Section III-B). In addition, the EA Coordinator has the responsibility to decide which MDRs should be integrated into the maintenance process (see Section III-A).

**EA Repository Manager:** The EA Repository Manager is the umbrella term for the framework that both provides the means to store, edit, and retrieve EA information, but also drives the data collection process.

**Management Data Repository (MDR):** An MDR is a data source such as a project portfolio management tool with an external interface that can be called to automatically retrieve data that is relevant to the EA. Each MDR needs to have a Data Owner assigned to it, that can be contacted in the case of data conflicts. External data sources are specifically important if parts of the architecture have been outsourced. In this case, the organization which is responsible for the outsourced architecture can provide an MDR that exposes this architecture via an interface.

**Data Owner:** Data Owners are the persons who manually insert data into the EA repository. They could also be responsible for the data provided by an MDR data source, to which they have been assigned responsibility. In particular, this group of users is in charge of resolving conflicts resulting from data that was manually entered by them, or originates from a MDR they are the responsible person for.

**EA Repository:** The EA repository is the umbrella term for the repository that runs the process and the EA repository. This role is also (partly) responsible for the technical integration of the EA repository with remote EA data sources.

**EA Stakeholders:** The EA Stakeholders are the group of people who are actually using the EA data to make decisions, e.g. regarding future projects. Typical examples are the CIO and IT-managers.

B. Meta Data

We argued that, in order to increase EA data quality, a specific maintenance process should be followed. This process, in turn, needs certain meta data so that the according process activities are triggered at the right time. Default
values for specific data types may already be set by the meta-
model level, or may be added at runtime. In the following,
we list each of these meta-data items, mentioning their
purpose and collection strategies.

**Expiry Time:** In order to keep the actuality of data in
the EA repository high, it is important that data is updated
on a regular basis. By specifying the time after which a data
item can be considered outdated, update cycles for expired
data items can be triggered. This expiry time is set at the
meta-model level for each data type, but can also be changed
at runtime for specific data sources. The expiry time should
be chosen depending on the estimated change frequency of
that data type. For example, it is likely that business units
change less frequently than the IT-infrastructure, such as
servers. These expiry times should be chosen according to
the experiences in each individual enterprise.

**Date of last change:** By storing the date when a
data item was last changed it can be inferred whether the
information of a data item is expired. This indicates that
measures need to be taken to check whether the data has
actually changed in reality. How this information is actually
used in the maintenance process to take action is described
in Section III-B.

**Data Responsible:** Each logic data cluster must contain
the information about a person or group that is responsible
for this data. A data responsible is needed in cases where
the quality of the data is in doubt and needs to be manually
verified.

**Data Sources:** If data is fed into the repository automatic-
ically from a certain data source, the information about its
origin needs to be kept to always be able to trace its source,
or its data owner. For example, if the information from a
specific data source contains errors, it can be identified
which data items are affected by this.

**Data Source Trustworthiness Attributes:** In some cases
it is necessary to also define the estimated data quality of
the data that is provided by a specific automated data source.
This is important in order to be able to estimate how much
trust can be put on conclusions drawn from information
from a specific data source. Also, if it is indicated that the
data coming from a specific repository has low quality, a
mandatory manual check could be added to the integration
process.

### III. THE AUTOMATED PROCESSES

In the following we describe the required processes in
more detail. These have to be seen in the context of a process
engine running on top of the EA repository. This means,
that the EA repository drives the process by providing EA
data and its meta-data. Furthermore, the repository assigns
tasks to technical services and to humans via task lists. The
processes are a basic guideline for achieving high EA data
quality with appropriate actuality.

#### A. Setup Process

First we describe the setup process which is executed
when a new data source is added to the maintenance process.
The decision to integrate a new MDR for data collection is
made by the EA Coordinator. This decision should be made
based on several factors:

1. How important is the provided data for the questions
   that the EA effort is supposed to answer? Guidelines
to these types of EA questions can be found in [10],
2. How much effort needs to be put into creating a
   suitable interface for exporting from the data source?
3. How much effort does it currently take to gather
   the same information manually, and enter it into the
   repository?
4. How important is the actuality of the data for EAM?
5. Can a sufficient level of security been reached for the
data exchange?

These questions are supposed to help the decision making
for starting the integration effort for a specific data source.
The weighting of the answers to the questions has to be
made on a case by case basis.

In the beginning of the automation effort, data sources
which are easy to integrate and have high importance to
EA questions, should be integrated first. In general, the
automation should be used sparingly at the beginning of
the integration effort, and only increased where the above
questions indicate an easy and effective integration, where
security standards can be met. Figure 2 shows the setup
process definition in BPMN. At the start of the process the
EA coordinator decides on the integration of a new data
source (step 1). The owner of this MDR then has to provide a
technical interface that can be called by the EA maintenance
process (step 2). In some cases these interfaces already exist,
in other cases they need to be implemented specifically for
the export to the EA tool. An example for how such an
interface can be provided is described in Section V. If the
interface is machine understandable, e.g., if it is semantically
annotated the EA repository utilizes a semantic mapping
between the provided data, and the internal data structure
(step 3). In step 4 the EA repository manager creates or
refines this mapping. This step is necessary to have a human
review of the automatically created mapping.

To be able to avoid identity conflicts, rules for identity
reconciliation of incoming data are defined in step 5. After
that, the trustworthiness of the data source is provided
(step 6). In parallel, the data owner can optionally specify
expiry dates for data coming from his/her data sources and
thereby override the default expiry dates defined on the
meta-model level. To ensure that the security requirements
for data, coming form the newly integrated source, are met
security policies for the data exchange are then defined by
the data owner (step 8). These could be requirements such
as confidentiality or integrity of the exchanged data as well as access control mechanisms for the interface. As the final step, the data owner needs to appoint a responsible person that is contacted in case problems, such as data conflicts, occur at runtime (step 9). This information will be stored as meta-data with the respective elements.

B. Maintenance Process

One of the findings from our survey [7] that we mentioned in the introduction, is that the data actuality that is demanded by the majority of our survey participants lies between one week and one month. Therefore, we concluded that EA data is mostly not needed in real time. Hence, in our maintenance process we focus on the data quality attributes such as, consistency, and then, as a second aspect, on actuality. This is reflected by the fact that we envision two types of views for EA participants. One is a view for EA stakeholders that need to make decisions based on the current architecture. In this view, the architecture can not be modified. The second view is provided for EA data providers. This view lets data providers modify the model of the current architecture. The changes, however, are not directly reflected in the as-is architecture view of EA stakeholders. Before the information is integrated, the quality of the resulting model has to be assured and data from MDRs have to be integrated. This is done by a maintenance cycle that consolidates the newly entered data. The state machine that is depicted in Figure 3 explains this fact. Initially, there is a working copy which is identical to the as-is architecture. Data providers can manually change this working copy. These changes are only made on the working copy and not directly pushed to the as-is model. When a new maintenance cycle starts (described below), the working copy is locked, i.e. no changes are allowed. The reason for this is to avoid conflicts that can occur when new data is added during the maintenance cycle. When the maintenance cycle finishes, the consolidated working copy is committed and adopted as the new as-is model. This version is then also the new working copy.

The maintenance cycle can be seen in Figure 4. As the basis for this process we used the process definition from the work of Fischer et al. [9] and modified it in the direction of automation. Please note that exception handling and resulting loops are not considered in the process definitions. This process is again initiated by the EA Coordinator. Reasons for the initiation could be a routine cycle, e.g. every two weeks, or the expiry events from specific data items. In addition, automated QA reports could be the reason why a maintenance cycle is started (not shown in diagram). The EA coordinator can decide on the part of the data in the as-is architecture that should be updated. This data is selected based on specific EA questions to be answered, or QA requirements. The selected range of data (optionally including specific data owners or MDRs to be contacted) is passed to a service of the EA repository (step 2). This service then selects the corresponding data owners and MDRs that are needed to update the requested data set. Thereby, the data owners are notified that their action is needed. The EA coordinator thus can concentrate on what data should be updated, rather than knowing whom to contact to update the required data. In the next two steps of the process, both the MDRs and the required data owners contribute new information in parallel. In step 3, the interfaces of the MDRs are called by the EA repository. In step 4, the data owners are provided with a task list that reminds them to enter the data they are responsible for. The provided data is then passed to the integration subprocess (step 5).

The subprocess is depicted in Figure 5. Its purpose is the automated conflict resolution and identity reconciliation of data which is composed from multiple sources. A more detailed description follows in Section III-C. As soon as the integration subprocess has finished, the EA repository manager is responsible for reviewing the conflicts that could not be solved by the integration process. If conflicts are present, the responsible data owners get according tasks assigned in their respective task lists to resolve them. Their changes are then fed back into the integration subprocess until all conflicts are resolved. When no more conflicts are found, either the release subprocess is executed, or the modified model is directly committed as the new as-is model. The release subprocess is supposed to integrate even more human stakeholders to assess the quality of the new model. The stakeholders can pass vetoes against the current version of the model to the EA repository manager, who is then again responsible for passing these vetoes to data owners to resolve possible conflicts. More specifics on the release subprocess including the rationale for when to execute it is given in Section III-D.

C. Integration Subprocess

The integration subprocess is a core part of the maintenance process. During this process, manual changes to the repository are locked. In the first step, it integrates all information into one single preliminary model. This information contains the working copy of the EA model with the manual changes made before the execution of the
maintenance cycle, the changes collected from the MDRs, and the manual changes requested during the maintenance cycle from data owners. With all information collected into a single model, rules are applied to consolidate the model. For this, we assume that the data is stored in a machine understandable format. In the second step, identity reconciliation is performed. This step includes the merging of information coming from different data sources, that refer to the same real object, e.g., an information system. In many cases identity reconciliation mechanisms can produce results that are almost certainly correct, e.g., if the IDs of data objects coming from different data sources have been pre-mapped. However, in some cases, when automated resolution fails, the decisions of the reconciliation mechanism need to be double checked by a human (see step 3). After the reconciliation step additional changes can be inferred using change propagation techniques (cf. [12], [13]).

In step 5, the model is then populated again with this inferred knowledge. The final step of the subprocess then attempts to resolve conflicts. An example for a conflict is information coming from different data sources that overlaps each other, i.e., relates to the same object and has differing values for specific object attributes. Conflicts which can not be automatically resolved are then escalated to the main maintenance process, to be checked by a person (step 7 in Figure 4).

D. Release Subprocess

The release subprocess (step 8 in Figure 4) is an important mechanism to increase the quality of data in the EA repository. In a release workflow, all EA stakeholders are asked to check the EA repository for redundancy and inconsistency, before it is committed as the new as-is model. The stakeholders can pass a veto on the working copy to indicate that they disagree with the content of the model. We recommend that a release workflow is not executed in every maintenance cycle because each release workflow means additional work for EA stakeholders. Depending on the EA culture of an organization, such a release subprocess could be executed in cycles of several months to create major consolidated versions of the as-is model. Another reason for the execution of the release workflow could be mergers or acquisitions which often entail major enterprise architecture redesign. Minor changes should be made without the release workflow. For the sake of brevity we do not describe a possible release workflow here. Examples can be found in [9] and [14].

IV. SUPPORT OF REQUIREMENTS

In the previous sections we have described the processes and meta information for semi-automated EA maintenance. Since these processes are the result of our requirements analysis for automated EA maintenance, we discuss how
our processes relate to the defined requirements. We distinguish between architectural requirements (AR), organizational requirements (OR), integration requirements (IR) and data quality requirements (DQR). See Table I for a brief description of the requirements which contains the subset of the requirements, we defined in [7], that are concerned with the maintenance process.

**AR 1:** This requirement demands a federation of data from the data sources of the data owners to a central repository. We cover this requirement by automatically integrating information from MDRs via interfaces during the maintenance process and also allowing data owners from specific departments to manually add data.

**OR 1:** This requirement demands an organizational maintenance process that governs the maintenance process. Clearly, this requirement is covered by our process definition in Section III.

**OR 1.1:** This requirement demands the technical implementation of the organizational process. Our process definition was built with technical process support in mind, hence it could be assisted by a business process engine.

**OR 2:** This requirement demands a responsible persons for each data source. In Section II-B we define meta-data for information in the EA repository. Here we also require that each EA data item has information about its origin attached to it. In addition, the setup process described in Section III-A includes a step for setting up the responsible person for each data source.

**IR 1:** This requirement demands that the processes must support the detection of changes in the real world architecture. This is supported by our processes by being able to set up MDRs that can monitor the actual status of the architecture. One example is an MDR that is part of an application server, which monitors the applications deployed on the server.

**IR 1.1:** This requirement demands the possibility of mapping incoming data to the internal data structure of the EA repository. While we leave the specifics of this mapping to future work, we address it as part of the setup process in Section III-A.

**IR 2:** This requirement relates to the internal data structure of the EA repository, that is needed for automation. In our opinion, a logic-based machine understandable data structure is mandatory for any EA automation. However, the specifics of this data structure is also left to future work.
**DQR 1:** This requirement demands that quality assurance is technically assisted. Our envisioned process assists in keeping the EA data consistent in various ways. For example, the integration subprocess includes automated steps for identity reconciliation and consistency checking. In addition, at various points in the processes, humans are integrated to re-check the data in the EA repository. Finally, the model locking mechanism ensures that during the maintenance process no changes with potential conflicts can be entered into the repository.

**DQR 2:** This requirement demands that the actuality of the data in the EA repository is as good enough for answering the specific EA question of an enterprise. As we have stated above, the EA maintenance process ensures that the data in the EA repository is as up-to-date as the frequency of the maintenance process. The frequency of the process can be adjusted to the demands of the questions that should be answered by the EA data. However, if actuality is the most important quality characteristic, the maintenance process should be adjusted to allow on-demand changes to the repository. This would, in turn, very likely reduce the consistency of the data in the repository.

**DQR 4:** This requirement demands the detection of propagating changes when changes are applied to the EA repository. This is important for keeping the data quality high if specific changes affect other elements. In our process, change propagation is part of the integration subprocess. Concrete development of such a mechanism is part of our future work.

**DQR 5:** This requirements regards to the reconciliation of data items that define the same physical object. The integration subprocess contains a task for this purpose. In addition, if this automated task fails, the data owner(s) are contacted to finish the reconciliation.

V. IMPLEMENTATION ASPECTS

In the process descriptions of Section III we have made several assumptions regarding implementation aspects. First, we assume that the EA repository is equipped with a process engine that is capable of executing the maintenance process by providing task lists for human process participants and by calling technical interfaces, such as Web services. These processes could, for example, be implemented with a BPEL [15] engine. This would allow for a flexible process description, that could be customized according to the needs in a company. Task lists are supported by various business process engines such as the open-source business process engine jBPM\(^2\).

Second, we assume that the internal data structure of the EA repository is machine understandable to allow for consistency checks and identity reconciliation mechanisms. In practice, this could be achieved by the use of Semantic Web technologies, such as OWL [16], which is based on description logics. Applying Semantic Web technologies also allows for the use of rules to infer new information, which is essential for the calculation of propagating changes.

Third, we state that EA data should be able to be retrieved from various information sources, such as project portfolio management tools. One way to implement this integration is via the Configuration Management Database federation (CMDBf) [17] standard by the Distributed Management Taskforce (DMTF). This standard describes a reference architecture and communication interfaces based on Web services for the standards-based exchange of configuration items. This standard could be re-used for the exchange of EA information. However, it is not clear yet whether the standard will find wide spread adoption, despite the support from major tool vendors such as HP, IBM and Microsoft.

VI. RELATED WORK

The problem of EA evolution and EA model maintenance has been identified by literature both from practice [1], [11], [18]–[20] and research [5], [9], [10], [21], [22]. However, most of the work in this area falls short of giving concrete advice on how to tackle EA maintenance in practice and does not cover strategies for automated EA maintenance.

For example, the Architecture Development Method (ADM) embedded in The Open Group Architecture Framework (TOGAF) [1] only states that some kind of “monitoring” should be applied to keep the models up-to-date, but does not state what kind of process and tool support this implies in practice. The Department of Defense Architecture Framework DoDAF version 2.02 is similarly short in concrete advice for data collection. It states that the “architectural data should be stored in a recognized commercial or government architecture tool.” [18]. A concrete process for collecting data is not given. In the same line, the documents of the Federal Enterprise Architecture Framework (FEAF) [20] mention the importance of data collection and quality assurance processes, but do not give concrete advice. We identified the most precise data collection process definition among the EA frameworks in the Pragmatic EA Framework (PEAF) specification [19]. This framework describes a detailed process for collecting information of the current architecture and tasks for data cleansing. However, it also does not discuss the implications for tool support or automation, since the framework is supposed to be tool independent.

On the side of research literature we describe a process for how information from infrastructure cloud environments can be integrated into enterprise architecture models in [23]. In addition, we identified three other research papers which are closely related to the work in this publication. The first one is the work by Fischer et al. [9] on the federated approach to enterprise architecture management. In this work the authors describe a process for collecting EA data from different stakeholders into a central repository. Opposed to the work

\(^2\)http://www.jboss.org/jbpm
in this publication the authors do not elaborate on where automation techniques can come into play. In addition the process integrates only one data source per maintenance cycle, which takes at least two days. This hinders the scalability of their process in case of many data sources. The work also does not discuss if an as-is model is blocked from changes during the maintenance cycle. Another publication that is closely related, is the EA pattern description by Moser et al. [14]. In this work, the authors describe a process pattern for automated EA data collection. However, they only describe high level patterns and leave out implementation issues. In contrast, our work has the aim of ultimately implementing the automated data collection processes. Therefore, this work provides a more detailed process and discusses implementation issues of automated EA maintenance. The work of Hafner and Winter [24] proposes a process for application architecture management. In their work, they take the application architecture management processes from three different companies and compare them with their requirements for managing application architecture. From those processes they derive a consolidated process definition, that should form the basis for implementing application architecture management in an organization. Opposed to our work, they do not focus on automation and data collection aspects, but rather on high-level management tasks. All three works above have in common that they do not distinguish between tasks that have to be executed by persons, and tasks that can be (semi-)automated.

On the technical side existing mechanisms from the data warehouse [25] and extract-transform-load (ETL) [26] communities relate to the data integration goals presented in this paper. However, the type and the granularity of typical data collected in data warehouses differs from the data collected in EA repositories. Nevertheless, further investigation is needed to evaluate possible reuse of data warehouse and ETL technologies for the purpose of EA repository maintenance.

Several EA tools provide import mechanisms for EA data. For example planningIT\(^3\) allows for the definition of manual data collection workflows. The EA tool Troux\(^4\) provides an integration platform that addresses several of the quality and data integration requirements presented in this paper. However, at the time of writing it was not possible to assess the actual capabilities apart from reading marketing material.

**VII. Conclusion & Future Work**

Building and maintaining an enterprise architecture model that reflects the current situation of an enterprise and does not contain errors, is a challenge that has been identified by both researchers and practitioners. Still, little literature can be found that addresses the problem of maintaining an EA model of sufficient quality at all times. In addition, current EA model maintenance process descriptions only cover the topic of automation on a high-level of abstraction. In our research effort on Living Models, we aim at bringing models closer to what they represent in the real world. This contribution is part of this effort in which we target the automation of EA model maintenance. As a step in this direction we have proposed several processes for semi-automated enterprise architecture model maintenance. These processes are supposed to reduce manual work for EA model maintenance and to increase data quality attributes such as consistency and actuality. The basis for the processes is formed by a set of requirements we gathered from a survey, interviews with EA practitioners and our own experience. The definition of these maintenance processes lays the foundation for our future work in this area.

As the next step in this direction, we will elaborate the concrete architecture for our envisioned EA method and tool. This will include the selection of technologies for the underlying data model. In addition, we will implement the processes defined in this article within the prototypical implementation of an EA tool, based on our requirements. Here we will first focus on identity reconciliation and conflict resolution techniques in order to keep the quality of data high, that originates from automated repositories. In addition

\(^3\)http://www.alfabet.com  
\(^4\)http://www.troux.com
we will implement exemplary data collection interfaces for selected information system. However, a complete evaluation of possible integration sources is out of the scope of this project. It is our long term goal to integrate our findings into the open-source enterprise architecture management tool iteraplan5 and evaluate our approach in a real-world setting.

ACKNOWLEDGMENT

We would like to thank Bernd Imhof and Markus Rink from E.ON AG, and Filiz Zevri from Munich RE for giving us insight on how EA is performed in their organizations. This work was partially supported by the Austrian Federal Ministry of Economy as part of the Laura-Bassi - Living Models for Open Systems - project FFG 822740/QE LaB.

REFERENCES