A bottom-up approach for the representation and continuous update of Technology Architectures

João Soares
Advisor: José Tribolet
Co-Advisor: André Vasconcelos
Instituto Superior Técnico
{joao.s.soares, jose.tribolet, andre.vasconcelos}@ist.utl.pt

***Abstract*—In organizations where there are not any concerns about guiding Information Systems' (IS) development using Information Systems Architecture (ISA) models updated in a strict and automatic fashion, business requirements and external pressures frequently lead to an increasing gap between current Information Systems and the latest version of their ISA and, in particular, their Technology Architecture (TA) models, when they exist.**

One of the causes for the lack of "investment" in TA models is the considerable effort required to "discover" infrastructural features that serve as input to the development of such Architectures. Such a task can only be achieved using *automatic* approaches.

This work is focused on devising a *bottom-up* approach for the representation and continuous update of TAs in organizations using a tool that automates the discovery of architectural evidences (through network events) and mappings between those architectural evidences to a modeling language, using logical deduction rules. This approach also features a process based on the annotation mechanism to enable interaction contexts that allow actors in an organization to make explicit their knowledge about their perception of the TA reality.

This approach was applied in a concrete case study, but it can be applied to any organization in any context.

I. Introduction

Nowadays, as result of a crescent globalization, competition among organizations is becoming more and more driven by its capacity and flexibility of reorganization and redesign of business processes and business strategies. Those "readjustments" are almost always triggered by stimuli resulting from the introduction of new business models, new regulations, *etc.* [1].

EA (and by association, ISA and TA) is considered an essential piece to the resolution of the challenges that emerge as a result from the organization's need to quickly readapt themselves and their ISs [2] [3].

An ISA must be, in organizations, the map that guides technological growth targeted to support the business. Being an ISA "the set of design artifacts or descriptive representations, relevant to describe an object (IS) so that one can produce it in accordance with requirements and maintain it during their lifetime" [4], it becomes clear that its importance depends on its compatibility with the real IS that it describes. It is essential to the usefulness of the ISA that its construction, maintenance and evolution is made in line with the evolution of IS.

However, it is very difficult to manually maintain this knowledge up to date, given the increasing complexity and rapid evolution of IS, whose technical and functional distribution is always increasing, integrating components from different backgrounds and levels of technical maturity.

As a result, often in large mature organizations, there is no evidence of a continually updated record to accurately describe, formalize and detail the TA for the IS in production, including the hundreds of networked applications and all their technical dependencies and interdependencies. We consider, therefore, that is very important to develop new ways to support automatic creation and maintenance of formal description of TAs and keep that description synchronized with the true state of IS (AS-IS).

At a first glance, this work aims to cover tree great issues: the *capture* of the technology reality, the *mapping* between the results of the capture and architectural models and finally, the continuous *update* of those models.

This paper is structured as follows. We will start by describing the problem which we propose to solve, concerning the identification and constant update of Technology Architectures (Section II). Next, we provide a brief overview of the related work highlighting the modeling of TAs (Section III). Subsequently, we describe the bottom-up approach where our solution is included (Section IV). We evaluate the obtained results (Section V) and finally we present our conclusions and Future Work (Section VI).

II. Problem Statement

Considering the previously identified issues related to the benefits of IT awareness in an organization, it is possible to conclude that, although there is a common agreement in the advantages and potential arising out of the specification of an Information Systems Architecture, it is still not possible to represent that architecture at different levels and their relationship with the business model in a standardized, universal, simple and simultaneously with the richness and rigor required for subsequent inspection and/or simulation of different business scenarios and technology [5].

More specifically, there is a lack of global knowledge of the current ISA, nor are there any *automatic methodologies* or processes defined to collect and identify architectural evidences. Those evidences would then be a valid input to the construction of a Technology Architecture.

Taking this context as a starting point, the problem that frames this investigation can then be decomposed into two modular questions:

- **Q1:** *"How to automatically collect and identify architectural evidences?"* It will allow an efficient, fast and ruled collection of data, which will be a major improvement regarding the creation of TA from scratch. On the other hand, in cases where architectures are already modeled, will allow them to be much more maintainable since the process of incorporating changes will be much more agile;
- **Q2:** *"How to map architectural evidences into updated, trustworthy and reliable Technology architectural models?"* There is a recognized difficulty in mapping technology evidences into architectural models that must be up-to-date and aligned with the reality. The aim of such mappings is due to the need to frame these evidences in a context of EA (where the first step to this end involves the construction of a TA).

III. Related Work

Since this research relates to the identification of TAs, in this section we start by giving a contextualization of this issue integrated in the field of Information Systems Architectures. We also introduce the modeling language of FCEO2007 and we describe *Netfacts* model, a generic, framework independent taxonomy for the representation of TA evidences. We also analyze an *autodiscovery* tool and explain a process to continuously update an enterprise model trough the annotation mechanism.

A. Information Systems Architecture and Technology Architecture

According to [6], the ISA is the representation of the components of the Information Systems, their relations, principles and guidelines with business support as goal. Many authors (including Vasconcelos in [1], FCEO2007) subdivide an ISA in Applications Architecture, Information Architecture and Technology Architecture.

Zachman [7] state that an ISA represents a set of views over data, functions, networks, people, time and even motivations, but in this investigation we have a more restrictive understanding about this issue. This investigation's scope is situated in the Technology Architecture *of* Information Systems. The intention is to frame it in the general context of an Information Systems Architecture that in its turn is part of a general Enterprise Architecture.

A Technology Architecture corresponds to the formalization of the choices related to the types of technology used to support each of the information systems that an organization has. Additionally, it aims to foresee the technology independently of functional components and information of an organization.

B. Framework CEO 2007 - TA

Proposed by Vasconcelos in [1], this framework features a taxonomy of concepts used to define an ISA (architectural primitives). This taxonomy is defined as a UML profile using OCL language [8].

At the TA level, CEO 2007 [1] describes the technologies that implement the applications (defined in the Application Architecture) and describes the technological environment provided to those applications. The main concept here is called *IT Block*, and it corresponds to a high level (abstract) representation of a technological component.

IT Blocks are responsible for the implementation of applications and *Technology Operations*, that are available to other technological components through *Technology Services*.

CEO 2007 distinguishes the following technological components types:

- *Technological Infrastructure:* physical and infrastructural components that support technological platforms and provide computing, data storage or communication capability:
  - Components without capability of computing: networking component (*e.g.*, router), peripheral (*e.g.*, printer, keyboard), or other specific device;
  - Components with capability of computing: personal computer (PC), server, mobile (*e.g.*, PDA), or other specific component.
- *Technological Platform:* responsible for providing services that provide execution environment to other technology platforms or software applications, *e.g.*, operating systems or database management systems;
- *Technological Application:* software component that represents the implementation of an application and operates on a technological platform, *e.g.*, software modules.

In order to enable correlations of architectural components with low level evidences obtained from network traffic, Alegria in [9] proposed some extensions to FCEO2007 metamodel. From this set of extensions we highlight the following primitives:

- Network Connection: an association between an infrastructure component and a network (IP Area Network)
- IP Area Network: a communications network that connects infrastructure components via a network connection (Network Connection);
- Network Service Port: an interface to a technology service.

After these extensions, although being quite comprehensive and flexible in terms of infrastructure, software applications, runtime environments and technological services, FCEO2007 still shows limitations regarding the support to cluster and Virtual Machine Infrastructures (VIM).

C. Application Discovery and Dependency Mapping

Application discovery is the process of automatically analyzing artifacts of a software application and physical "nodes" that constitute a network (*e.g.*, servers, firewalls, *etc.*). *Dependency Mapping* creates visibility between discovered applications and infrastructure dependencies.

Automated application discovery and subsequent dependency mapping, can capture, connect and unveil intricate relationships including the way in which applications behave and relate to the TA on which they rely.

Automated application discovery normally leverages an effective non-agent-based discovery method - *passive discovery* - to automatically discover the structural and behavioral aspects of applications. Passive discovery operates by watching the network traffic, examining all the applications as they execute, and then identifying relationships based on real business usage. Passive discovery also means that the management burden of maintaining software agents is null. Passive discovery is just a part of the equation regarding these systems. Some products often use a hybrid approach, where *active discovery* is also used to enable the collection of deep and fundamental data about configuration details.

1) Solar Winds APM: Application Performance Monitor (APM), by Solar Winds[1] delivers "agentless application performance monitoring for applications and servers" [10]. The core function of this tool is in fact its ability to monitor physical and virtual applications and server infrastructure, but our interest in this tool regards its capacity to "scan" "nodes" in a network and discover applications that run on them.

Solar Winds APM leverages out-of-the-box and user-generated "server and application *monitors*", that enable any application or server to be discovered and monitored in a network or in a data center. These *monitors* function also as mere "templates" or discovery patterns to the identification technology components. Capture capabilities range also from high-level applications and their components, database components, servers, network devices, storage elements, virtual environments, to J2EE or .NET components, *etc.*.

Application Performance Monitor stores all gathered information in a Microsoft SQL Server[2] database and offers database utilities to manage it. *Database Manager* allows queries perform, database values edits, data export, database repairs and compaction on the APM database. Trough this tool it is also possible to specify where discovered information is stored (in terms of relational tables). This investigation made use of this functionality to approximate the *schema* of the stored relational data to the conceptual model that will be described in Section III-D.

D. Netfacts Model

In [9], Alegria proposes a conceptual model that provides context and relates the various TA kinds of evidences that can be inferred from network traffic analysis. This model is defined with the intention to be generic and independent from any modeling framework and any technique or traffic analysis tool.

The generality and independence of the model aims to provide a reference point from which the manifestations of Information Systems on the network can be coded and treated in a uniform manner, in order to facilitate the relationship between these low-level manifestations and any modeling language, independently of their Framework of origin.

The central entity and mediator of this model is the flow of communication across the network (*Network Flow*) which represents a line of communication with the beginning and end between two TCP/IP extremes. Different *Network Flows* can relate to each other through a dependency relationship.

Communication between network assets (*Network Host*) via *Network Flows*, is done through a set of protocols (*Application Layer Protocol*).

A *Network Flow* can be associated with the software components (*Software Component*) used in each end of that communication, as well as services (*Service*) and operations (*Operation*) including its parameters (*Operation Parameter*).

For network assets (*Network Host*) represented by one or more IP addresses it is possible to infer the names by which these addresses are known in the network.

As a major limitation, we can say that this model, despite being transversal regarding the representation of TA evidences, does not offer mechanisms to distinguish between native support components and virtually supported components.

E. PROASIS

PROASIS is a process to continuously update an enterprise model trough the annotation mechanism [11], where organizational actors act as active *updaters* of an AS-IS business process model, through the comparison between the modeled activities and the ongoing real executed activities, becoming key agents in reducing the gap between the organization and its representation.

In this context, annotations are a mechanism that enable interaction contexts, allowing actors to make explicit their knowledge about their activities through graphic representations, establishing a conversation and negotiation channel among the individual actors (the parts) and between the individual actors and the organization (the whole). The organization here refers to all its members that previously shared and agreed upon model representation.

The *AS-IS business process model dynamic updating process* (PROASIS) is based on the analysis of misalignments between the shared model and ongoing executed processes within an organization. Annotations are the mechanism used to collect the updates proposed by organizational actors, constituting an input to the misalignment analysis.

PROASIS supports several granularity levels within business process models (modeled using BPMN) and depending on the modeling element in each level of detail, several actors may play different roles (annotators of the model and reviewers or evaluators of the annotations made).

Regarding annotations itself, they allow making proposals for correcting the model (corrective maintenance), capture changes in action or interaction contexts (adaptive maintenance), make free comments that could anticipate problems (preventive maintenance) and promote process continuous improvement (perfective maintenance).

[1]http://www.solarwinds.com

[2]http://www.microsoft.com/sqlserver/

The following text expresses the essence of PROASIS: *"The "client" of PROASIS (corresponding role of the operational model that detects misalignment between the model and "reality" - role: annotator) wants to update the model, so it makes an annotation (update request). This update request is received by the modeler (which is who actually update the model if the annotation is approved) and by the reviewers. When reviewers receive the annotation, they can begin the revision of the annotation (which is optional). The approval of the annotation (role: approver) is made based on the analysis of the annotation (update request) and reviews. If the annotation (request update) is approved, the model will be updated and delivered to the "client""*.

IV. Information Systems Architecture Representation and Update

Since our investigation addresses specifically the issue of ISA automatic discovery and constant update we propose ourselves to address that trough two strands: initial discovery and subsequent automatic update using *autodiscovery* mechanisms and update promoted by actors sharing a TA model in an organization. Figure 1 shows the process that implements our bottom-up approach. Each subprocess is a step of our approach, therefore giving a view of all the work that has been developed in this investigation that will be described with more detail in the following sections. The last step, «Update & Completion» (in red), will just be presented in this document as a mere proposal, and therefore it will lack of validation.

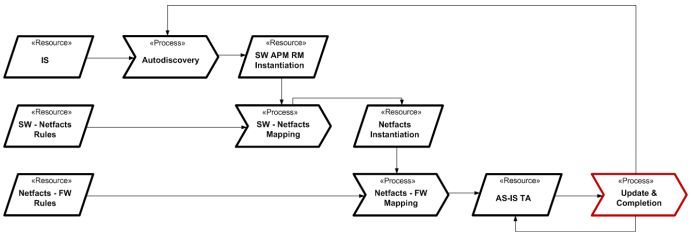

Fig. 1. AS-IS ISA representation/update subprocesses - A bottom-up approach for the representation and continuous update of Technology Architectures

The *inputs* of this process are the real Information Systems of an organization and the set of mapping rules that enable the translation of the *autodiscovery* results to a modeling language. The *output* of this process consists in the architectural models that reflect the organization's TA. Although this approach is framework/modeling language independent, this work was done using FCEO2007 with some proposed extensions to its metamodel.

A. Solar Winds APM Autodiscovery

This section describes the steps of the processes called «Autodiscovery» with special focus on the changes that were made regarding the way of storing captured information in the relational database managed by the tool used («SW APM RM Instantiation» – Solar Winds APM Relational Model Instantiation). This process implements the techniques that Application Discovery and Dependency tools use (see Section III-C).

Solar Winds APM was the tool chosen to perform *autodiscovery* in this work, mainly due to its ease and speed in deployment. The aim of this work, as explained before, does not include the definition of *autodiscovery* mechanisms and for sake of brevity we will not describe that process in this document.

Solar Winds APM stores, among other items, its discovery results in a SQL Server database. In order to approximate this representation to the *Netfacts* model (a generic, *autodiscovery* tool independent and framework independent model to represent the manifestations of Information Systems and its TA) some changes were made regarding the way that information is stored in the database. To accomplish that objective we used a functionality provided by Solar Winds APM called *Database Manager*. Figure 2 depicts the most important tables and their attributes considering our approach.

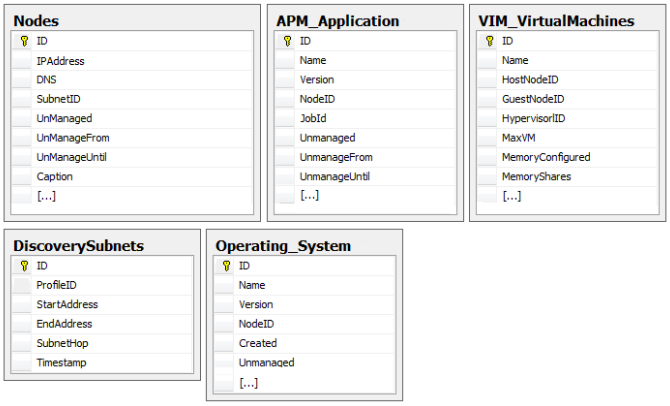

Fig. 2. Solar Winds APM SQL tables

B. Netfacts *model Extension*

Netfacts model, proposed by Alegria in [9] is a generic model to incorporate, contextualize and correlate various types of evidences exhibited by Information Systems and their TA that can be inferred through the capture and analysis of network traffic. This model was defined with the objective of being generic and independent of any TA modeling framework and any technique or tool to traffic analysis.

In this investigation, we tried to adapt at the most the way that information inferred through *autodiscovery* is stored in order to be the most approximate possible to the taxonomy that *Netfacts* model represents. On the other hand, based on the analysis done to the types of information that can be captured by *autodiscovery* tools and on the identified Netfacts model limitations, we proposed extensions to *Netfacts* metamodel in order to fulfill the objectives set out earlier.

We defined a new attribute to be added to «Network Host» entity defined in the model proposed by Alegria: it indicates whether a «Network Host» is supported or not by a Virtualized environment («isVirtualized»).

With this investigation, *Netfacts* model faces the reality of Virtual Machines. In this context, Virtual Machines Managers [12] can be understood as a mean to emulate the operation of a «Network Host». In order to represent those dynamics we propose an extension to the *Netfacts* metamodel that involves a new entity called «VirtualMachineManager»,

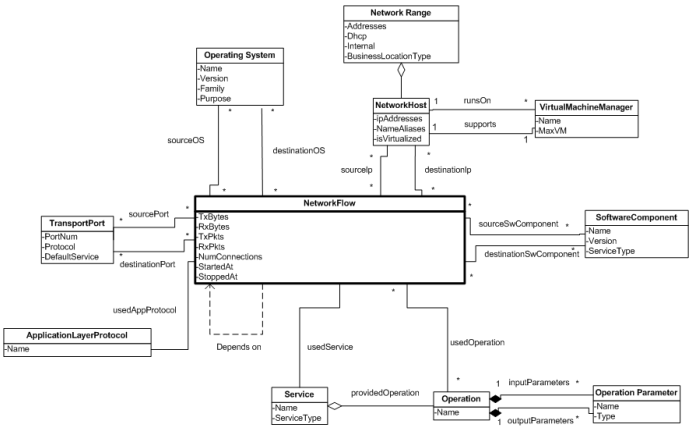

Fig. 3. Netfacts extension metamodel

C. Mapping between Netfacts *model and Solar Winds APM Relational Model*

This Section describes process that implements the mapping rules («SW - Netfacts Rules») that enable the translation of the relational data stored in the automatic discovery phase to the *Netfacts* model («SW - Netfacts Mapping»).

The specification of the rules follows a subset of the formalism of First-order logic (Horn clauses, used in Prolog [13]), assuming our *Netfacts* model extension (Section IV-B and Solar Winds APM Relational model (Section IV-A) as the domain of discourse).

The following lines are the example of a mapping rule:

Name:

$MapFCEOVITIBtoNetHost$

Description:

Mapping of a FCEO2007 «Virtualized IT Infrastructure Computational Block» and a *Netfacts* «Network Host ».

Formal Definition:

$MapFCEOVITIBtoNetHost(x, y) \equiv$
$\forall x \exists a \exists b \exists c \exists d(FCEOVITIB(x) \wedge PAddress(x, a)$
$\wedge\, ConcreteName(x, b) \wedge IndirectlySupports(x, d)$
$\wedge\, IndirectlySupports(c, x) \Rightarrow$
$\exists y(NetHost(y) \wedge Name(y, b) \wedge IPAddress(y, a) \wedge$
$Virtualizes(c, y)$
$\wedge\, IpOS((a, d))))$

Foundation:

The mapping between these two elements is done through their analogous attributes and support relations.

D. FCEO2007 Extension

Framework CEO serves as a conceptual basis for specification and discussion of Information Systems Architectures in this dissertation. After the extensive research and the extension regarding the original FCEO2007 [14], made in [9], one can say that this model allows TA description at a technological, application and informational level, with a broad frame onto Enterprise Architectures. The use of FCEO2007 enables mappings from the physical infrastructure up to the informational entities and business processes. Furthermore, FCEO2007 enables different *views* over a TA.

The advent of the use of an *autodiscovery* tool opened up possibilities for the identification of other relevant architectural components like, *e.g.*, Virtual Machines. Traditional TA metamodels do not reflect this tendency and do not make any distinction between virtually enabled assets and physical assets. Based on the analysis of the various ISA modeling frameworks and taking again into account the *Netfacts* model, we propose some extensions to FCEO2007 meta-model in order to cover the virtualization field.

In line with the extensions proposed to *Netfacts* model, we propose an extension to the FCEO2007 that involves two new primitives called «Virtual Machine Manager» and «Virtualized IT Infrastructure Computational Block»

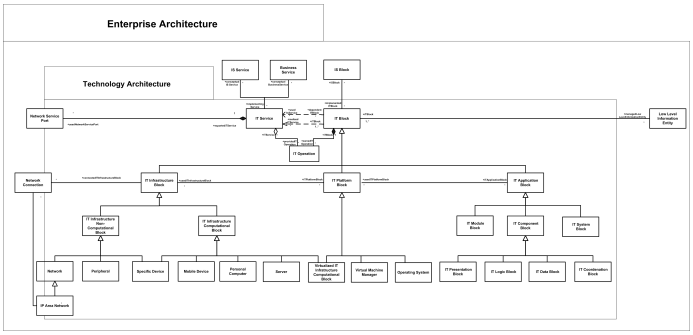

Fig. 4. FCEO2007 extension TA metamodel

E. Specific Framework Mappings

This phase of our approach is marked by the transcription of the terms early defined at a conceptual level to the level of description of a TA using a modeling language for this purpose. As it was said before, our approach was build independently of any technology or modeling language, but we decided to adopt FCEO2007 as basis for the description of our solution. Due to this fact, this process was described using mappings to our extension of FCEO2007 («Netfacts - FW Mapping»).

In [9], Alegria presents a lot of work on the mapping of *Netfacts* model entities to FCEO2007 and we will use those contributions because they fit perfectly in our approach. Due to this fact, we redefined or defined from scratch new mapping rules that became necessary due to our extensions to *Netfacts* model and FCEO2007. The specification of the rules follows a subset of the formalism of First-order logic [13] assuming

our FCEO2007 extension and our *Netfacts* model extension as the domain of discourse.

F. Continuous update and Completion

Section III-E describes a process to continuously update an enterprise model trough the annotation mechanism [11]. In this process, organizational actors act as active updaters of a business process AS-IS model through the comparison between the modeled activities and the ongoing real executed activities in order to reduce the gap between the organization and its representation.

Throughout this document, and based on bibliographic references, we emphasized the importance of a constantly updated ISA as "guide" for the construction, maintenance and planning of SIs that aim to support business requirements proactively. To fulfill this requirement, we decided to incorporate the process referred in the above paragraph with the necessary adaptations in order to make it suitable for the case of Technology Architectures and to integrate it with the useful capabilities offered by *autodiscovery* mechanisms.

Recalling the reflection we did when describing the initial phase of our approach, we identified insufficiencies in the information provided by the application of *autodiscovery* that reflected themselves in *Netfacts* conceptualization. In this sense, we can consider that this phase of our approach can also be interpreted as a "completion" phase where an organizational actor can fill in the "gaps" that may arise in the obtained TA models arising from the limitations displayed by the *autodiscovery phase*, which are generally regarding to:

- Dependencies between «IT Application Block»s or between «IT Application Block»s and «IT Platform Block»s (specified using, *e.g.*, «IT Service»s);
- Dependencies between «IT Infrastructure Block»s (specified using «Network Connections»s).

The adjustments made in PROASIS were focused above all in the relations of the operational roles with actors and the the kind of annotations they can do, since the underlying process of PROASIS is independent of any enterprise model on which it operates.

V. Case Study and Results Evaluation

This section presents a case study and the correspondent results evaluation in order to validate part of our bottom-up approach. The case study we present here includes all steps of our approach, except the «Update & Completion» phase for the reasons already mentioned in the previous section.

The concepts and processes defined in Section IV were put into practice and validated in a case study corresponding to the TA of an ecosystem of Information Systems.

Our case study corresponds to the application of our approach since the «Autodiscovery» process, until obtaining the *resource* «AS-IS TA». As our solution describes, we used Solar Winds APM to collect architectural evidences. After that process we performed the mappings manually in order to obtain TA models.

The research on which this document is drawn up was done in collaboration with AMA - *"Agência para a Modernização Administrativa*, and this case study was also encompassed within this company, namely its *development/test subnetwork*.

A. Case Study

As it was said before, the case study presented here was performed in the context of AMA and its *development subnetwork*. Before applying the approach developed in this research work, we proceeded to representation of the TA of the targeted IS using the practices of the *Current Systems and Technology* step of the Spewak Method [15], which we applied informally. We also relied on existing architecture documents that resulted from previous development and planning processes of IS and its ISA. In order to clarify occasional doubts, some independent traffic analysis tools were also used.

The ecosystem of the Information Systems present in the *development subnetwork* of AMA, covers a complex cloud of systems and services, resulting from several years of evolution, sometimes not always guided by the principles of ISA planning and Enterprise Architecture. The lack of a stable informational-level architecture of the organization (to guide the development of the IS and TA) contributes to the number of systems involved in the execution of a process and increases the needs of integration and reconciliation between the components of Technology Architecture. The growth of unstructured IS, in this case, also results from the fact that we are in presence of a network in which tests are performed.

In this work, we have not applied the developed solution to the entire ecosystem of Information Systems present in the *development subnetwork* of AMA, we focused ourselves only in one set of meaningful and important Systems to assess the work produced. In the following subsection we will describe one of these IS, since the remaining substantially present the same characteristics.

1) Document Management Information System (SharePoint): This IS hosts a *Sharepoint*[3] platform. SharePoint is a web application platform developed by Microsoft. SharePoint is designed as a centralized replacement for multiple web applications and supports various combinations of enterprise website requirements. It is typically associated with web content management and document management systems.

The Technological Architecture of this portal (Figure 5) follows a classic *2-tier* pattern. The web *front-end* component consists of a virtualized server, «Virtualized IT Infratructure Computational Block» (*PORTAL.AMADEV*). The *back-end* data component consists of a another virtualized server (*SQL.AMADEV*) that performs a data service to support Sharepoint through a relational database.

2) Application of the bottom-up approach: The discoveries, using a deployment of Solar Winds APM in a virtual machine running in a server present in the *development subnetwork* that was set up just to this purpose, were made during the day of August 18, 2011 and were composed of two distinct moments.

[3]http://sharepoint.microsoft.com

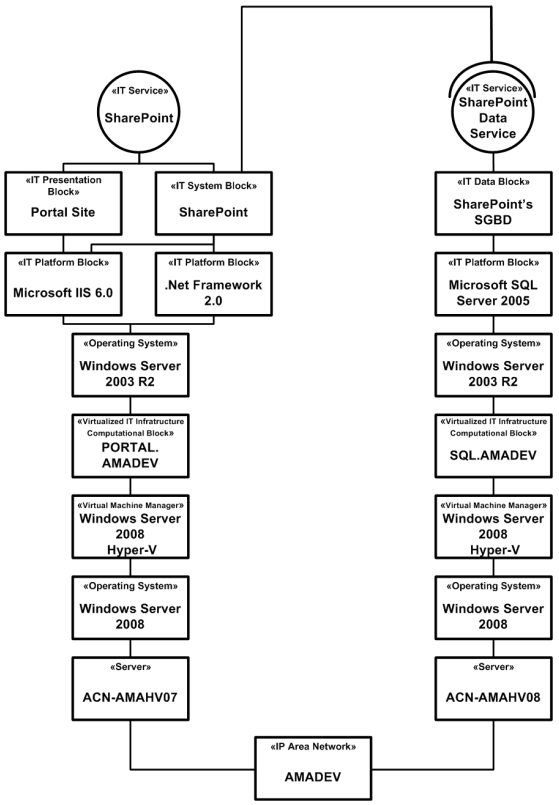

Fig. 5. Document Managerment Information System (AMA's subnet)

The first stage (from 14h00 to 17h00) was characterized by discoveries of the "nodes" present in the network, and for this purpose we selected a start and a finish IP Address in order to restrict the "scan". After the node discovery, we performed the "application discovery and dependency mapping" stage (from 17h00 to 19h00), which consists in assigning automatically a *monitor* to each "application" in each node. In some cases we needed to introduce "credentials" like, *e.g.*, Windows Administrator on target server, *etc.*

Listing 1. Example of the resulting log of the first *autodiscovery* phase, corresponding to the IS presented above

```
27 "ICMP" "10.10.201.152" 0 0 30-12-1899
    00:00:00 30-12-1899 00:00:00 "PORTAL"
    "PORTAL" "" "" "" " (...) 18-8-2011

28 "ICMP" "10.10.201.153" 0 0 30-12-1899
    00:00:00 30-12-1899 00:00:00 "SQL" "
    SQL" "" "" "" " (...) 18-8-2011
```

Listing 2. Example of the resulting log of the second *autodiscovery* phase, corresponding to the IS presented above

```
14 "Windows Server 2003 R2" 0 0
    "10.10.201.152" (...) 18-8-2011
```

Solar Winds APM stores the results of the discoveries in a SQL database. After that process finished its execution, we proceeded to the mappings described earlier. Finally, we just schematized the resulting FCEO2007 extension entities to obtain TA diagrams (as in Figure 5).

3) Results Analysis: In this section we analyze the results obtained by applying the approach developed in this work, as described in the previous chapter. This analysis is organized by FCEO2007 extension architectural component type and at the end we will point out the major problems that were observed.

IT Infrastructure Block: All servers («Server») were positively identified by at least one of your network connections. Regarding the names of the servers previously known, all of them were positively identified.

Virtualized IT Infrastructure Computational Block: All "virtualized servers" were positively identified and correctly linked to the «Virtual Machine Manager» that supports them. Attributes such as *processing capacity*, *volatile storage capacity* and *permanent storage capacity* (inherited from «IT Infrastructure Block») were also correctly identified.

IT Platform Block: All operating systems («OperatingSystem ») were positively identified.

All technology platforms («IT Platform Block ») were positively identified except *.NET Framework 2.0*, which is used in the front-end of the presented IS.

Virtual Machine Manager: All "hypervisors" were correctly identified. All the IS in AMA's *development subnetwork* exhibited *Microsoft Hyper-V* Virtual Machine Managers; to test out if our approach is able to identify other vendors we set up a *VMware* machine that was correctly identified.

IT Application Block: All databases («IT Data Block») have been positively identified.

Among the group of applications, only the databases («IT Data Block») are verifiable in relation to its attributes, because they are the only ones where identifiers specified that could be used to correlate with the facts inferred from *autodFcovery*.

Major Problems: On the occasions that some architectural component was not found (apart from "nodes" that have always been discovered), it is always due to lack of a specific *monitor* loaded in the *autodiscovery* application. In the SI shown above, there was one of those situations, particularly for *.NET Framework 2.0*. The more coverage in *monitors*, the greater is the usefulness of the approach developed. We believe, however, that we achieved a significant and relevant coverage that failed only in this case and that the production of a user-made *monitor* update for this application would imply a very small effort.

Regarding the "application discovery" moment, we were sometimes asked to enter credentials (see Section V-A2). Though this is not an actual problem, it is a concern that limits the *automatic* nature of our approach in the sense that it requires some human interaction.

B. Conclusions/Summary

For this case study, we presented in this report only one IS that is, however fairly representative for the other ones that are present in the ecosystem of IS in the *development subnetwork* of the AMA. Our approach allowed us to represent mostly, the expected TA (identified previously using Spewak method

practices), in particular the primitives related to physical and virtual infrastructures, platforms and applications. In fact, during this case study, it was not possible to identify technological services, a situation that is due to the limitations of the tool used for *autodiscovery* and had already been identified earlier in this document. However, one the objectives of the «Update & Completion» process of our approach is just to cover those gaps using human interaction. For the component of our approach that was tested in this case study, we identified some areas for improvement that relate to the generation of custom *monitors* for certain architectural components.

VI. Conclusions and Future Work

Information Systems Architectures and by association Technology Architectures must be, in organizations, the map that guides technological growth targeted to support the business and its importance depends on its compatibility with the real IS that describes. It is essential to the usefulness of the ISA that its construction, maintenance and evolution is made in line with the evolution of IS.

ISAs are regularly relegated to a secondary role in the context of organizations. It is very difficult to maintain such knowledge given the increasing complexity and rapid evolution of IS, whose technical and functional distribution is always increasing, integrating components from different backgrounds and levels of technical maturity.

This investigation was conducted under the reasoning that is very important to develop new ways to support automatic creation and maintenance of formal description of TA and keep that description synchronized with the true state of IS.

A. Main Contributions

The main contribution of this investigation was to develop a **bottom-up approach** that comprises the **automatic capture of architectural evidences** that reflect the Technology Architecture of Information Systems in an organization, the **mapping of the capture results into TA models in a ISA modeling language** and the **update** of those models.

This contribution is underpinned by a number of other contributions that collectively enabled the achievement of the objective we set ourselves:

1) Autodiscovery *as a source of information about the reality of IS and TA:* This contribution answers directly to out first research question (**Q1**). We carried out a a systematization of various *autodiscovery* tools as a mean to infer relevant information about TA. We extended the used *autodiscovery* tool so that the schema of the evidences stored is as close as possible to *Netfacts* model [9], who in its turn has also been extended to contemplate new possibilities arising from *autodiscovery*. In this way it is possible to infer and store information about IS and its TA automatically and non-intrusively.

The use of such a tool opened the door to the detection of *virtualized* components. According to the bibliographic review made this type of infrastructures has never been contemplated in the research works about TA, a sub-area of EA. To fill that gap, we extended the EA Framework that was used as conceptual basis in this work (see Section VI-A4).

2) Mapping between autodiscovery *results and an ISA modeling language:* We established a mapping between the *autodiscovery* results, therefore, answering partially to our second research question (**Q2**), by mapping the structure of concepts referred to in Section III-D and ISA modeling language,the CEO Framework [1]. This mapping defines a link between the architectural evidences (in a low level of abstraction) and TA primitives for a ISA modeling language (at a high level of abstraction). Its definition was performed using the formalism of First-order logic.

3) Our approach as part of a process of Planning, Construction, Monitoring and Verification of an ISA: We integrated our approach as part of a process of planning, construction, monitoring and verification of an ISA facing the reality of the IS in production [9]. Our approach to the *representation and continuous update of Technology Architectures* is defined as a process that takes advantage of the contributions mentioned above: automatic capture of the reality; mapping of the capture results to TA models; promote update upon TA models.

4) Extension and Application of FCEO2007: This research was conducted in collaboration with CODE group having been chosen the CEO Framework (FCEO2007) to be used as ISA modeling language in this work. FCEO2007 was analyzed in the context of Virtualized Environments, and after identifying some weaknesses in this area we proposed some metamodel extensions at Technology Architecture level, so it endows these capabilities. With our extension, one is able to model and differentiate between virtually supported components and natively supported components.

B. Other contributions

1) Continuous update of Technology Architecture models: We adapted a dynamic update process originally suited to business processes [11] to enable its use regarding Technology Architectures. We considered that there are two main groups of actor roles that can update a TA model: *organizational roles*, and *autodiscovery agent*. The first group of roles represents traditional *stakeholders* in an organization and the second represents an "agent" of *autodiscovery*, that in the case of our approach is *Solar Winds APM*. We included this last role because new discoveries are a vehicle for changes to the model. In the adaptation of this process we summarized the role of the various actors depending on the model element to be annotated and we created new annotation categories based on [11]. This contribution, if properly validated, would be complete the answer to our second research question (**Q2**).

C. Limitations

Despite the contributions made by this research work, we identify some limitations that were not addressed:

1) Process of Planning, Construction, Monitoring and Verification of the IS and its TA: As it was said before, our approach was integrated in a process of planning, construction,

monitoring and verification of the IS and its TA initially proposed by Vasconcelos [1] and extended by Alegria [9]. However, this integration was not validated or applied in a case study featuring the whole process, therefore lacking practical proof of the theoretical hypothesis of its usefulness and validity.

2) Detection of some Technological Platforms: An important part of the utility and capability of the proposed approach comes from the rigor and range of detection of signatures protocols and software components. In the context of practical application of the extended *autodiscovery* tool in the case study and other situations, we noted that some technological platforms were recurrently not identified (*e.g.*, *.NET Framework 2.0*). This happens due to the fact of the nonexistence of default *monitors* to enable applications or servers to be discovered. In those cases one must create a custom *monitor*. However, we did not contemplate that possibility because we consider that it would step outside out approach's automatic character.

3) Continuous execution: The need for agility, real-time management and immediate response that modern organizations face considering internal and external environment dynamics, makes application of concepts and methods developed in this work more important if done in real time and continuously.

Despite the fact that the conceptual and theoretical work done in this paper provides a process to capture TA evidences and map them to TA models - continuously and in real time - the use of our approach was not in this mode since these processes were performed at distinct occasions and not on an ongoing basis.

D. Future Work

This section describes avenues of research for the evolution of this work.

1) Expert System Features: We consider that our approach would benefit from the inclusion of features such as:

- Questioning the user interactively: which would incorporate inputs from the user, only when needed, during the mapping process, to resolve ambiguities or conflicts. These inputs would then be incorporated in the associated *knowledge base* [16].
- Explanation of the conclusions drawn by an inference engine: system's ability to answer the questions "Why?" and "How?" in relation to the conclusions regarding the resulted mappings between architectural evidences and TA elements. The first question relates to the facts that validate a mapping. The second has to do with the description of every step of the mapping process.

2) Complex relationships between Information Systems: The growing trend of using SOA and the decoupling of IS and its services using middleware systems results in the fact that many modern IS do not interact directly with technology service providers. In these cases, in terms of network traffic observed by *autodiscovery* tool, the network flows detected may not reflect rigorously the real TA relationships. A next research step, which we consider very important is to deal with this type of high-level, complex relationships between IS.

3) Representation of a complete ISA: The motivation for this work is not restricted to the Technology Architecture, encompassing a Information Systems Architecture and, ultimately, an overall Enterprise Architecture. Hence, the extension of the architecture of the methodology proposed here to other levels of ISA and EA is a path of investigation still open.

4) Other ISA Frameworks: Despite the use of FCEO2007 as ISA modeling language, we believe that the essence of this work can be applied to other languages and frameworks (*e.g.*, ArchiMate [17]), since they incorporate Technology Architecture components or provide means to extensions to that effect. The methodology of using mapping rules between a domain of architectural evidences and a TA language modeling can be applied to other modeling languages.

REFERENCES

[1] A. Vasconcelos, *Arquitecturas dos Sistemas de Informação: Representação e Avaliação*. PhD thesis, Instituto Superior Técnico, 2007.

[2] Land, M., *et al.*, *Enterprise Architecture (The Enterprise Engineering Series): Creating Value by Informed Governance*, p. 5. Heidelberg: Springer, 2008.

[3] S. Townson, "Why does Enterprise Architecture Matter?." White Papper, The Open Group, 2008.

[4] Zachman, J., "Enterprise architecture: The issue of the century," *Database Programming and Design magazine*, pp. 1 – 13, 1997.

[5] Boar, B., *Constructing Blueprints for Enterprise IT Architecture*. New York: John Wiley & Sons, 1999.

[6] Maes, R., *et al.*, "Redefining business - it alignment through a unified framework," in *Proceedings of the Landelijk Architectuur Congre*, 2000.

[7] Sowa, J., Zachman, J., "Extending and formalizing the framework for information system architecture," *IBM Systems Journal*, vol. 31, pp. 590 – 616, 1992.

[8] OMG Standard, *Object Constraint Language - OMG Available Specification, Version 2.0*. Multiple, 2006.

[9] A. Alegria, "Verificação automática de modelos de arquitectura tecnológica de sistemas de informação em rede," Master's thesis, Instituto Superior Técnico, 2009.

[10] Solar Winds, "Application performance monitor," tech. rep., Solar Winds, Austin, 2001.

[11] N. Castela, M. Zacarias, and J. Tribolet, "Business Process Model Dynamic Updating," *CENTERIS 2010*, 2010.

[12] Smith, Nair, *Virtual Machines: Versatile Platforms for Systems and Processes*. Massachusetts: Morgan Kaufmann, 2005.

[13] I. Bratko, *PROLOG Programming for Artificial Intelligence*. Boston: Addison-Wesley, third ed., 2001.

[14] A. Vasconcelos, A. Caetano, J. Neves, P. Sinogas, R. J. Mendes, and J. Tribolet, "A framework for modeling strategy, business processes and information systems," in *5th International Enterprise Distributed Object Computing Conference (EDOC 2001)*, (Seattle), pp. 69–78, IEEE Computer Society 2001, 2001.

[15] S. H. Spewak and S. C. Hill, *Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology*. New York: John Wiley & Sons, 1992.

[16] D. Merrit, *Building Expert System in Prolog*. Heidelberg: Springer, 1989.

[17] Lankhorst, M., *et al.*, *Enterprise Architecture at Work*. Heidelberg: Springer, second ed., 2009.