Atlas: the Enterprise Cartography Tool

Pedro Sousa[1][2], Ricardo Leal[2], and André Sampaio[2]

[1] Instituto Superior Técnico, Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal,
pedro.manuel.sousa@tecnico.ulisboa.pt,
[2] Link Consulting SA, Av. Duque de Ávila 23, 1000 Lisboa, Portugal
ricardo.leal@linkconsulting.com
andresampaio@linkconsulting.com

Abstract. The need to maintain architectural representations of enterprises is an indescribable fact nowadays given their constant need evolve. However, despite the large number of Enterprise Architecture tools available, enterprises are unable to maintain architectural models updated, given the effort it entails. Atlas is an Enterprise Architecture tool conceived to reduce the effort needed to keep architectural models updated, in enterprises where changes are constant and design occurs in a decentralized, distributed and asynchronous process using multiple design tools. Atlas uses Enterprise Cartography as the approach paradigm to be able to generate architectural views automatically from the models and information gathered from multiple sources, with a time bar allowing a seamless navigation in time, from past to future models.

Key words: Enterprise Cartography, EA, Enterprise Architecture, Architecture Visualization, Architecture tools

CEUR-WS.org/Vol-2229/paper2.pdf

1 Introduction

The need to maintain explicit knowledge of the architecture of enterprises is an indisputable fact nowadays given their constant need to evolve. Either for the purpose of enterprise governance, engineering, compliance or maintenance, architectural representations are an enterprise asset that must be governed[1]. By explicit knowledge we consider both the existence of models of the Enterprise Architecture, hereafter EA, kept in some repository as well as the capability to present graphical representations of the enterprise artifacts and their dependencies, also referred as architectural views[2, 3], here after referred as EA views for space reasons.

We refer to *enterprises* and not to *organizations* since the first term has a wider scope according to the the Open Group definition were "*enterprise is any set of organizations that have a common set of goals*"[2], and the approach presented here applies to both. An example of an enterprise is the *Portuguese National Health System*[1], that includes more than one hundred health organizations.

[1] Servio Nacional de Sade: https://www.sns.gov.pt

We also consider a distinction between enterprise transformation and evolution, the difference being that the former is mostly related with top level restructure and the later is related with the optimization of current state of affairs[4]. Despite the differences, they both require EA as input data and they both causes changes in enterprise, thus likely induces changes in the EA. Here after we use the term *change initiatives* to refer to both transformation and evolution initiatives. Since change initiatives are temporary, unique and a purposeful activity, they correspond to the concept of projects as defined in the project community[5].

Stefan[6] presents a list of 28 EA application scenarios and the corresponding literature sources, making clear importance of knowing the EA for enterprise maintenance, planning and evolution. In its simpler and basic from, information about EA boils down to the list of enterprise artifacts and their dependencies. Given the variability and quality of existing EA tools and their representation capabilities[7], one could argue that enterprises could easily create and maintain an EA repository where all models are kept updated so that core enterprise activities could benefit from such information[6].

However, based on our experience of nearly two decades in organizations from various countries and industry sectors, we found that this is not the case. On the contrary, enterprises are far from being able to create and maintain an EA repository updated, given the effort that such endeavor would entail. We believe that a significant part of such extra effort results from the following organizational aspects:

– Enterprise evolution uses multiple "Enterprise Architecture" tools. Enterprise evolution uses multiple tools for EA, being office tools commonly used ones. Most of the times such tools are not integrated and do not provide coherent information. For example, when the board of director of a company, acting as an architect designing the company, decides to create a new department they will not go to an EA tool to model the new department. Most likely, the decision will appear the board meeting minutes (probably an office document) a few weeks or months before the intended date. If the organization structure is modeled in some EA tool, effort is required to update it in conformity with this new design. The person that performs such update in the EA tool is not designing but merely updating the model and its architectural representations.
– Enterprise evolution is mostly a distributed and asynchronous process. Enterprise evolution is planned and designed by different units in an asynchronous and distributed process, involving many actors and many dimensions of concerns without formal mechanisms of communication between the different designers. For example, when a director decides to make changes in their department, it probably will conduct meetings with other departments to check for possible dependencies and impacts. Directors of other departments can do the same for their departments. Despite the number of face-to-face meetings, there is no design process established so that the knowledge gathered in such meetings is consolidated and shared across the enterprise, so that design actually uses coherent and updated information.

In such context, one can envisage the huge effort required to update the models in the EA tools used. The fastest the enterprise changes, the more effort is required to keep such models updated. When enterprises are already faced with lack of resources for the day to day projects, they are not willing to allocate effort to keep enterprise models updated.

But the problem is actually more complex because enterprises models are also a moving target. In fact enterprises need models that refer to different points in time, namely:

AS-WAS models. These models represent the enterprises past state of affairs, including not only the past architecture but also the plans of change initiatives of the past. They are most useful for auditing and accountability purposes since they can hold justifications for the decisions taken in the past. For example, the decision took in the past to acquire development capacities on some technology could be sustained on the plans of a change initiative to build new products using that technology, regardless the fact of that change was actually put forward or dismissed.

AS-IS models. These models represent the current enterprise. *AS-IS* models are required for current operations and for reacting to events.

TO-BE models. These models represent the future enterprise, and are critical for planning the next change initiatives, as recognized both by EA[8, 9] and project management[10, 11] communities. In fact, to plan a project that will start in 6 months, you need to know how the enterprise will be when the project starts, not what it is today, as many changes can come in the meantime.

Naturally, *AS-WAS* models are just the old AS-IS and do not pose any challenge. But *AS-IS* and *TO-BE* models are real challenges, since they must be take into consideration the multiple ongoing change initiatives. This difficulty has a substantial impact on the ability to plan change initiatives and, consequently, has an impact on the costs, time and risks of achieving their expected outcomes[8, 11].

So, the focus of our research has been in finding a low effort method that allows enterprise to have updated AS-WAS, AS-IS and TO-BE models and EA views. To keep architectural models and views updated, one needs to address two main issues: (i) how to gather information about the changes in the enterprise models and views and, (ii) how to update the architectural models and views based on the gathered information. Again, our experience in actual companies has shown us directions for such questions.

– Gather information about the changes. Our finding is was that, in general, it is easier to gather information about what people are currently doing than about what they did in the past. Since what people are doing today is what most likely will be enterprise TO-BE, this finding means that it is easier to know the expected TO-BE than the *AS-IS* of the enterprise. So, since TO-BE models will become *AS-IS* and *AS-WAS*, this finding tell us that the focus should be target to TO-BE models.

– Update EA views. Our finding is that hand-draw EA views will likely remain obsolete much longer than generated views. Hand-draw models requires placing symbols in a drawing canvas, where aesthetic aspects are one key concern and time consuming factor. Therefore, to keep such views updated, one needs both to update the contents as well as the aesthetic aspects. So, this finding tell us to avoid hand-draw view and support only automatic generated views from gathered models (information).

Atlas[2] was designed to explore the findings presented above. It is an EA tool conceived to reduce the effort needed to keep EA views updated in enterprises designed in a decentralized, distributed and asynchronous process using multiple design tools. In Atlas, EA views are generated automatically and have a time bar allowing a seamless navigation in time, from past to future state of each EA view. Information about changes is gathered by the observation of the plans of ongoing change initiatives.

This approach was first presented in[12], initial implementation was described in [13, 14], and results from first projects were presented in[15, 16]. The term Enterprise Cartography was coined in[17]. In the next section we present the concept of Enterprise Cartography, and then we present relevant aspects of the Atlas tool in supporting the above ideas.

2 Enterprise Cartography

Simply put, *EC* is the process of abstracting, collecting, structuring and representing architectural artifacts and their relations from the observation of enterprise reality. The expression "enterprise reality" refers to the present state of the enterprise. Traditionally, the observation of this reality is a subjective perception of an observer, and consequently it is probable that there are different actors that perceive different realities of the same enterprise. However, as we propose and will become clear along the text, the perception of the present state of the enterprise (the reality) is sustained on relevant facts captured in logs and models, and represented through artifacts based on previously defined and agreed upon models.

We refer to "architectural artifacts" as the enterprise's observable elements whose inter-dependencies and intra-dependencies express the architecture of the enterprise. Naturally, different institutions may consider various sets of artifacts as part of their architecture. To abstract, structure and represent the architecture related artifacts, one needs the whole set of knowledge and concepts implied in architecture visualization and representation. For example, the concepts of semiotic triangle[18, 19], the model of architecture description presented in ISO IEEE 1471[3] and a symbolic notation, such as the one used in ArchiMate[20], are concepts necessary to the production of EA views.

Ongoing change initiatives and their plans are an essential part of enterprise reality because they define the near future *TO-BE* of the enterprise if no further

[2] www.linkconsulting.com/atlas

decisions are taken that might have an impact on them. This is a fundamental concept that we call *emerging AS-IS*, which we define as the state of the enterprise after successful completion of ongoing change initiatives. This corresponds to the inertia of a body; if no contrary action is taken, inertia determines the body future position and speed. Similarly, if no opposite actions are done, ongoing initiatives determine the future of the institution and thus the TO-BE of models and architecture.

Given that in today enterprises, change initiatives are omnipresent, the concept of *Emerging AS-IS* is not only an essential capacity for the planning of new change initiatives, but also for the monitoring of the enactments of ongoing change initiatives. When driving a car the faster the car is moving, the farther ahead one should focus our eyes to match the longer time and distance context within which we need to steer it. Likewise, when steering an enterprise, the faster the enterprise is changing, the more important is to know the *emerging AS-IS*, as time flows.

Enterprise cartography is a purely descriptive perspective, since it does not explicitly incorporate the purposeful design of the new enterprise artifacts, as one expects in EA. Such a difference is also evident in the definitions of the *architect* and *cartographer* roles. According to the IEEE Standard Glossary of Software Engineering Terminology[21], the term *architect* is defined as *"The person, team, or organization responsible for designing systems architecture"*, and in the Merriam-Webster dictionary[22], the term *cartographer* is defined as the *person who makes maps*. So, an *architect* is essentially a person that designs and shapes intended changes to the architecture of the present enterprise reality, while a *cartographer* is essentially a person that aims at representing reality as it happens, including such changes as they are occurring.

2.1 Enterprise Cartography Principles

Principle 1: **Change initiatives are enterprise artifacts.**
A change initiative is an enterprise artifact. This statement is in line with most architecture guidelines such as the Work Package concept in both TOGAF[2] and ArchiMate[20]. However, we go further and also state that their plans can be observed as part of the enterprise *AS-IS* models. We assume that change initiatives have plans because they purposefully aim at some desire goals and therefore should have plans to achieve them. Furthermore, these plans and goals includes *TO-BE* models of the enterprise .

Principle 2: **Changes in the set of productive artifacts are planned ones**.
This principle states that artifacts do not become productive or non-productive randomly, but only as a result of change initiative. Since change initiatives are also enterprise artifacts (principle 1), an artifact becomes productive or non-productive only by the action of another productive artifact (the change initiative). Notice that the assumption that ongoing change initiatives are productive artifacts, is sustained by the purposefulness aspect of the change initiative, by which the enterprise becomes more valuable after the purposeful changes than before.

Principle 3: **All enterprise artifacts have a five-state life-cycle: conceived, gestating, alive, retired and removed.**
Despite the fact that common EA notations (e.g.[20] do not formalize artifacts states, it is many convenient to have models that mix existing and not existing artifacts, to express model changes. In this principle we say that existing artifacts with an enterprise do not "fell from the sky", but go through an evolution process instead. Let us first consider as *productive* the artifacts as the ones that are somehow involved in the enterprise business processes that produces values. In figure 1 we present these five states, their fundamental transitions and there classification with productiveness: *Non Productive Yet* , *Productive* and *No Longer Productive* states.

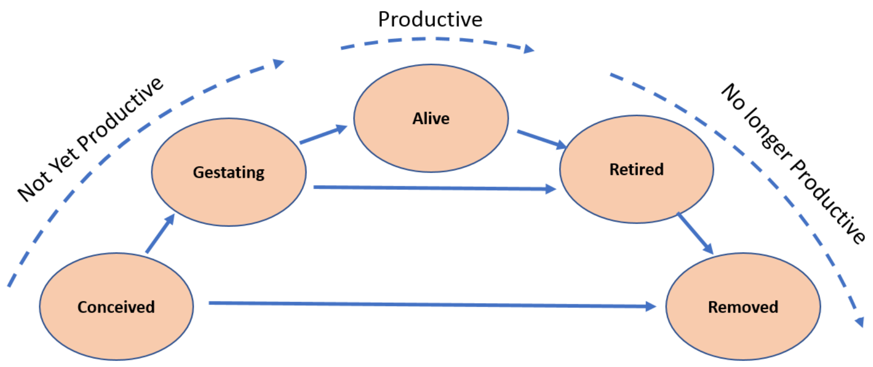

Fig. 1. Existence Artifact life-cycle

- *Conceived* state, as the first state of existence. It corresponds to a state where the artifact is planned but its materialization into a productive artifact did not started yet. A conceived artifact is an idea that exists *TO-BE* models (or plans) of some change initiative, even if itself is still in the conceived state.
- *Gestating* is the state when an artifact is being constructed or acquired to become productive. As conceived artifacts, gestating ones do not play a role in the enterprise transactions and processes. This state only differentiates from the conceived state by the fact that the change initiative that aims at putting the artifact into production has already started. An ongoing change initiative is necessarily a productive artifact, since it can have an impact on other productive artifacts and is expected to produce value after its completion.
- *Alive*. An alive artifact plays purposeful roles in the enterprise to create value. Conceived and Gestation artifacts might have an relationship on life objects, namely in the ones that are conceiving or creating them. But their relation is a passive one, not a purposeful role as alive objects must have to create value. From Principle 2, it is clear that an artifact cannot

be brought into existence as alive; it always exists first as conceived before being alive.

Notice that in this paper we only consider Alive as the only productive state. However, one can consider other productive states, such as for example the *Deprecated* state. However, it is not relevant for the purpose of this paper and we continue with the case of alive as the only productive state.

- *Retired*[3]. is when an alive artifact no longer plays a role in the enterprise transactions and processes to create value. As in the conceived and gestation states, a Retired artifact may still have an impact on alive artifacts. In fact, even if it does not create behavior or value to the enterprise, it may be the target of several housekeeping activities that are necessary after being dead.

  The dead state can be achieved directly after gestating state, without becoming alive. An artifact planned to become alive by a given change initiative might never be alive either because the initiative was canceled or because it simply changed plans and decided to no longer put that artifact into production.
- *Removed.* Represents the post-Retired state where the artifact has no impact in the remaining artifacts. A removed artifact is unable to interact with alive enterprise artifacts. An artifact can move from conception directly to removal when it never materialized in a gestation, meaning that it never went beyond an idea.

Principle 4: **The *TO-BE* state precedes the *AS-IS* state.**

This principle states that productive artifacts in the *AS-IS* model existed before in the *TO-BE* models of some changing initiative. Furthermore, such models were part of the enterprise observed *AS-IS* at some point in time prior to the current time.

Principle 5: **The emerging AS-IS can be inferred by observing the AS-IS of an enterprise**.

The *emerging AS-IS* state differs from the *AS-IS* state by the artifacts planned to be brought into production/retirement by ongoing change initiatives. Since ongoing initiatives are alive artifacts, their plans (*TO-BE* models) are in the scope of the cartographer observations of the enterprise reality.

Therefore, one can foresee the set of productive artifacts at some point in time in the future by consolidating the *AS-IS* with the *TO-BE* models of the ongoing change initiative whose completion date precedes the desired moment in time[12]. This can be stated as:

$$P(t_m) = P(t_0) \cup NYP(CI_{t_n}) \setminus NLP(CI_{t_n}) \forall t_0 \leq t_n \leq t_m \qquad (1)$$

Where $P(t)$ represents *Productive* artifacts at time t, $NYP(CI_t)$ and $NLP(CI_t)$ represents respectively *Not Yet Productive* and *Not Longer Productive* artifacts in plans of Change initiatives that produces results at time t.

[3] In previous publications the *Retired* stated was named as *Dead*.

3 Atlas Overview

Altas is a web based EA tool providing all functionality one could expect from such a tool. It allows the full configuration of the meta model, i.e. the classes and references types whose instances are used to express models of the organization. Users can also configure in and out data both in format as well in contents for integration with other tools and systems.

Atlas supports custom configured user interfaces. Given the wide scope of usage of EA tools within organizations, it will likely be used by employees without basic modeling concepts. So, besides allowing the configuration of elements in Atlas default interfaces (e.g, tabs in object edition properties), it allows users to configure specific forms, where users only see the desired properties, even if they are from different objects. Atlas also provides analytic elements such as charts, dashboards and EA views that we called blueprints.

Atlas also supports the configuration of behavior associated with EA views, validation rules, state propagation between objects in the repository, among others. Such behavior can be stated both in queries and rules that run directly on the selected repository as well as in jobs and batch's that run on a dedicated sandboxes.

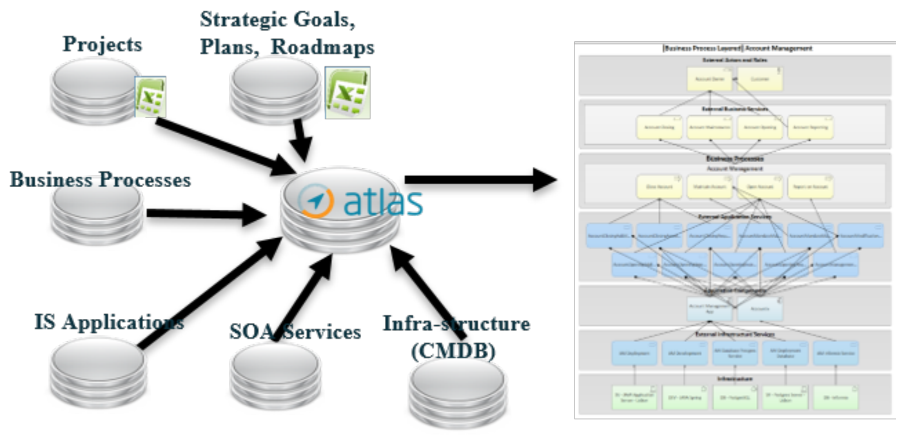

Fig. 2. Atlas collecting AS-IS and TO-BE models and produce AS-WAS, AS-IS and TO-BE EA views

We now focus the description of how Atlas supports enterprise cartography. A typical scenario is presented in figure 2 where information from several sources, including office tools, is processed and feeds the generation of AS-WAS, AS-IS and TO-BE EA views and analysis. In this scenario, enterprise artifacts and their relationships exist in multiples sources, and each source will likely have its own meta model. Furthermore, information in each source may be related with the past, present or future of EA.

3.1 Model transformation

Model transformation occurs whenever data flows between sources with different meta models, as is the case when Atlas gathers information from other sources, and has been a relevant topic in software development and integration[23]. Among the different technologies to handle model transformation[23], in Atlas we adopted a two step approach, each with its own technology. The first step deals with the format transformation and is based on XSLT[24] which is very common and convenient for processing XML files. The second step deals with the actual model transformation, and uses a high level type-based rules that operates on types, instances and relationships.

Consider the case where one wants to import a BPMN[25] model and that *Data Stores* must be imported as *Applications* in the Atlas repository. To configure such rule, one assigns a batch to the desired source or file extension and then defines three jobs. The first job is configured with the XSLT script that converts the BPMN tool format into Atlas XML format. The second job is configured with text file script with the transformation rules, that transforms the source model and objects into Atlas model and objects. Finally the third job imports the transformed objects into the Atlas repository. In the example given, the transformation rules script is a "rename data type Data Store to Application". The names of the *Data Stores* in the BPMN model are matched against *Applications* in the Atlas repository, and unmatched *Data Stores* will yield the creation of new *Applications*, as discussed in the next section.

If several batchs are assigned to the same source or file extension, the user is requested to select one to upload the file. This is a very useful feature since within the same enterprise, different teams or departments may use the same notation differently and different rules might be required.

Another rather important aspect is to be able to perform different a behaviour in case of the imported artifacts match with objects that already exist in the Atlas repository or if new objects were created during the importation. The rules allows the importation behaviour to be dependent of any state in the Atlas repository, but the match for existing objects is done solely on the object names, as discussed next.

3.2 Object Names

In the scenario presented in figure 2 one needs to establish a mechanism to match objects existing in different sources that correspond to the same enterprise artifact. In other words, one needs an identification mechanism valid throughout the different sources to match imported objects against the ones that already exist in the Atlas repository.

The use of 128 bits Object Identifiers (OIDs, also known as UUIDs) ensures uniqueness for most practical purposes even when generated by different and independent tools. However, these system-generated OIDs are not useful in the scenario presented in figure 2, since the same enterprise artifact would have objects with a different OID in each source.

Change is also a factor against the use of OIDs. Consider that a process modeling tool holds a model of the *sales process* in which the *CRM* application is used to inquire the status of the client payments. Later, the process changes and instead of using the application *CRM*, it now uses the application *ERP* for the same purpose. This change can be done simply by change the application name, from *CRM* to *ERP*. However, in this case, the name has change but the OID in the process modeling tool is the same, despite the enterprise artifact has changed. This leads to a situation where the same object now referrers to a different enterprise artifacts, breaking the existing biding between OIDs and enterprise artifacts even in the context a single tool.

User defined object names can easy the task of matching objects between sources, if object names correspond to the names of the artifacts have in the enterprise. However, in most cases, object names are used defined and normally are not unique, since uniqueness is assured by OIDs. In Atlas, we took a different approach where identification and denotation is established by user defined object names[26] and OIDs are not disclosed to user, except for audit traces. This approach implies that, from the user perspective, objects of the same class cannot have the same name.

In practice this can be a strong restriction in large enterprises where one could expect to have several organizations with different business processes or actors with the same name. For example, in the case of *Portuguese National Health System*, one could expect to have *Actors* with the same name in different hospitals. To support this scenario, users can define composite object names, by selecting object properties to be used for its identification, as primary keys in relational databases. In this case, the name of the object could be defined by both *hospital* and *Actor* names.Notice that users can change the name of objects, given that they are object editable properties.

Despite the fact that in most situations users deal with a flat object name space, full object names are unique URLs composed as follows:

serverURL/RepositoryName/ClassName/ArtifactNameField_N. ..
.ArtifactNameField_0.

The concatenation of the fields *ArtifactNameField_N* to *ArtifactNameField_0* must be unique within the objects of the same *ClassName*. All fiels but the last are optional (i.e can be *null*), except the last one (*ArtifactNameField_0*) that cannot be null.

3.3 Object State

In Atlas, object state is defined as a set of properties values, whose types are user defined. A time bar in the object edition interface allows users to visit properties past states, up to the time of its creation. Basic types as Numeric, Text, Rich Text, Hyperlink, Boolean are available. Users must define their own reference types for reference properties

Object properties do not change type (structure) over time, this means that, for example, a numeric property cannot not become a reference property. However, object sate, as a collection of properties, can change structure over time since properties may be added or removed from its class over time as discussed in the next section. Therefore, objects of the same class can have an entirely different set of properties.

References properties are assured to be either null or to refer to one or more objects. There are no dangling references, since reference value assignment triggers object creation and object deletions nullify references. In Atlas, object are created when their names are registered thus, by assigning a new name (value) to a reference properties a new object is created with that name. Such created object has no state yet, only a name.

By default reference properties have no state associated besides besides the name of the denoted object. However, in some situations it is very convenient to add properties to references. This is done by binding a class with a reference property, whose structure becomes the structure of the class. Any class defined in the meta model can be bound to a reference property, and the same class can be bounded to different reference properties.

3.4 Meta model Co-Evolution

As described earlier, classes also change over time whenever users remove or add properties. By moving back the time bar in the class edition interface, one can see the class properties at any point in time, up to time the class was created. Property creation and removal does not affect the state of the objects in the repository. If a property is deleted from a class at a given time, it will no longer be visible in the object's structure after that time, however, as soon as the time bar goes back in time the deleted property will appear in object state history with the corresponding values.

Besides adding and removing properties, Atlas also provides support for complex operations in meta-model and is able to propagate the necessary changes to objects in the repository. For example, the *Property to Class* meta model operation promotes a textual property to a reference property to a class created from the original text property. An example could be promoting the *Home Address* text property of a *Client* class into a reference property to a new class *Home Address*, so that clients with the same address could share the same object. A meta-model evolution primitive catalog, with primitives such as *Flatten Hierarchy*, *Merge Class*, *Split Class*, *Property to Class*, *Class to Property*, among others[27, 28] is available.

These type of changes in the meta model requires the changes on existing objects in the repository according with the changes made to the corresponding classes. To support such operations, Atlas is able to inform the impact that each change has in the objects existing in the repository and execute them if the necessary actions can be performed automatically [29, 30, 31].

3.5 Object life-cycles

Object life-cycle is defined over a set of object user defined properties of type, one for each life-cycle state. Users can define several life-cycles for the same class.

For example, for the application artifact, one may define the existence life-cycle and technical quality life-cycle. For each life-cycle, users can define any number of states, and assign a color to each one, so that objects are shaded with the color corresponding to the state of the object at the selected date in the EA view.

As explained in the next section, Atlas has specific built-in behavior associated with the visualization of object life-cycles that requires uses to classify them as *Not Yet Productive*, *Productive* or *No Longer Productive*. Users define life-cycle states in order and some sates may be declared as *Productive*. The states prior to the first *Productive* state are considered *Not Yet Productive* and states after the last *Productive* state are considered *No longer Productive*.

3.6 Visualizing *AS-WAS*, *AS-IS* and *TO-BE* EA views

In atlas, EA graphical views are generated on-the-fly given its unique name, in the form of an URL as *serverURL/RepositoryName/EAViewName?arguments*. This means that, by placing hyperlinks to EA views, users that access and interact with them outside the Atlas tool in documents or in other web interfaces.

EA views are defined in a simple XML language, where one can define containers that can hold others containers or a *content query* whose result set determines the objects that will be displayed inside that container. Containers can be set in a hierarchy and grow and shrink within user defined limits according to the number of objects of the query result set. Objects can also be containers, whenever one wants other results sets to be presented inside.

Once generated, EA views are interactive interfaces where users can navigate in time, run predefined analyses, jump to another EA views and drag objects between containers, triggering containers *out-query* and *in-query* to do the necessary state update associated with the object movement between containers. For example, dragging an object of actor *John* from a container representing *Sales Unit* to a container representing *Legal Unit* will trigger the corresponding *out-query* on the *Sales Unit* and *in-query* on the *Legal Unit* containers with the actor *John* as argument.

In what concerns time navigation capability of EA views, in Atlas each and every EA view is as a movie, where one can see its contents from the minimum to the maximum date found among all objects that encompass the EA view. As presented in figure 3, the time bar has two handlers, one on the left for the lower date and another in the right for the upper date.

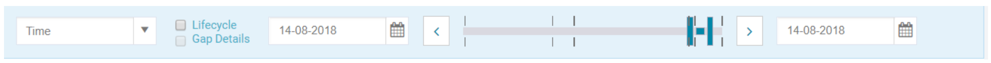

Fig. 3. EA view Time bar

By moving the time bar handlers back and forward one can see the architecture view in the selected time. The visualization can occur either in Absolute

Mode or Gap Mode. In the Absolute Mode, the view presents the contents that corresponds to the time selected. Two options can be selected:

– Default: Only productive objects at selected time are presented. Symbols are presented in their default color.
– life-cycle: All objects are presented, but their symbols is shaded with the life-cycle colors. For example, a typical life-cycle configuration is to shade in grey the artifacts that are *Not Yet Productive* at the selected time, and to shade in red the objects that are *No Longer Productive* objects at the selected time.

In Gap Mode, the Visualize the view presents the gap between two points in time T_i and T_f, by shade the objects symbols according to the difference of life-cycle state of each object on the selected $T_i$mes T_i and T_f[32]. In this mode, artifacts are marked as:

– **Introduced**. An artifact is considered to be introduced between T_i and T_f if it is *Not Yet Productive* at T_i but it is *Productive* at T_f.
– **Withdraw**. An artifact is considered to be withdraw between T_i and T_f if it is *productive* in T_i and is *No Longer Productive* in T_f.
– **Changed**. An artifact is considered to be changed between T_i and T_f if it is *Productive* in both T_i and T_f and if at least one artifact with which it has a reference to as been introduced or removed between T_i and T_f. The users can select the depth of the graph traversal analysis and for each step, the references to be used.

In the next three figures (4 to 6) we present the same EA view with different options and dates of visualization. The first figure presents the architecture at 10/05/2018 in absolute mode with life-cycle option on. The container in the middle show the components of application *Account Management App*, that consume services in the container on the middle left, that in turn and provided by applications in the leftmost container. Likewise, the *Account Management App* components provide the services presented in the middle right container that are consumed by the applications in the rightmost container. Bellow, one presents the data objects used in each service. Objects shaded in grey or red are, respectively, *Not Yet Productive* or *No Longer Productive* at 10/05/2018.

By switching the life-cycle mode off, the shaded objects (both introduced and Withdraw) disappear from the EA view, leaving only the objects that were, are or will be[4] productive at 10/05/2018. The corresponding figure is not shown since it can be easily perceived.

By moving the left part of the time handle back in time to 05/04/2014, the visualization switches to Gap mode between 05/04/2014 and 10/05/2018, as presented in figure 6, where *introduced* objects appear in green and *withdraw* ones appear in red.

Next, we set the life-cycle mode on again, mixing the Gap and life-cycle modes, as presented in figure 5, where the evolving objects are shaded in two

[4] Both the dates used in this example (05/04/2014 and 10/05/2018) could refer to the past, present of future.

halves: On the left side, with the color associated with the life-cycle state at the first date (05/04/2014) and, on the right side, with the color associated with the life-cycle state at the second date (10/05/2018).

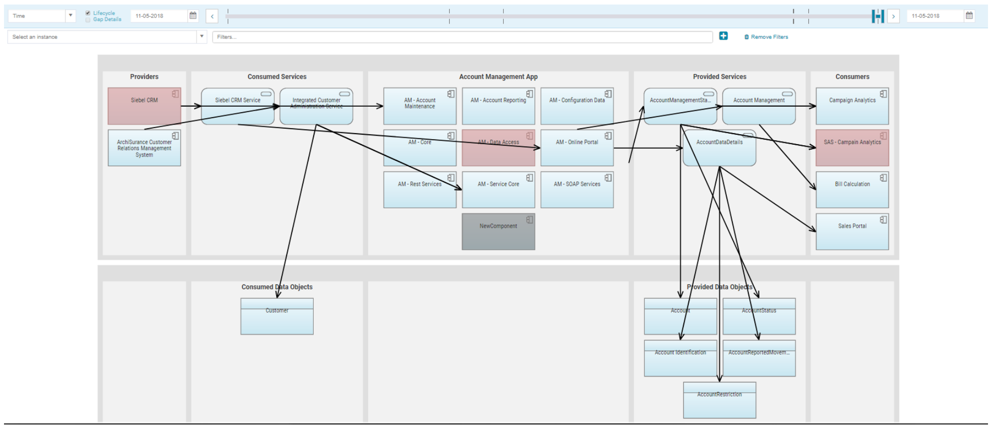

Fig. 4. EA view at 10/05/2018 with object life-cycle

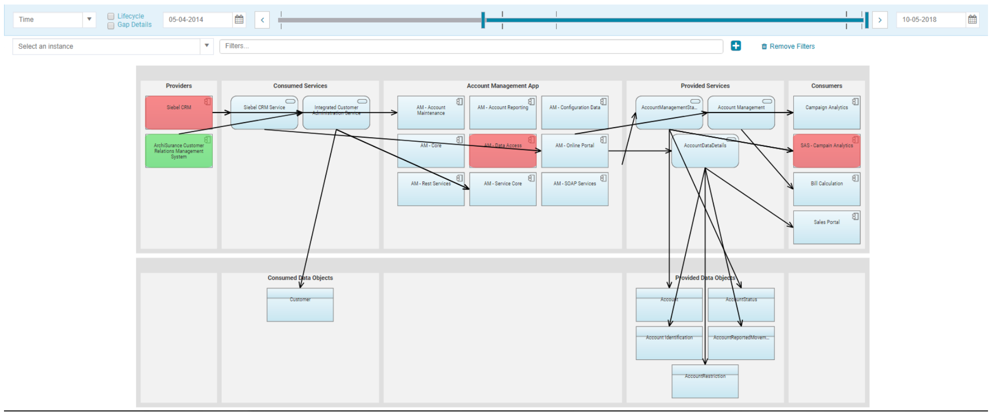

Fig. 5. EA view Gap between 05/04/2014 and 10/05/2018

The time bar presented in these three figures show four markers, excluding the extremes, revealing the dates of change in that EA view. These dates correspond to object life-cycle evolution but can also be traced back to the change initiatives that have produced such evaluations, as explained next.

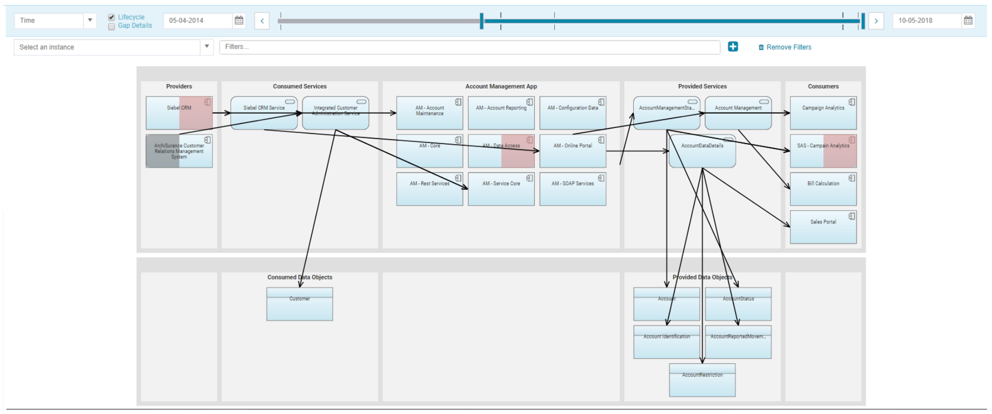

Fig. 6. EA view Gap between 05/04/2014 and 10/05/2018 with life-cycle

3.7 Change Initiatives

According to cartography principles, enterprise artifacts evolve in its life-cycle as a result of some planned change initiatives. In Atlas, used defined class may be a change initiative class, and more than one change initiative class may exist. For example, one may define the *Project* class as change initiative of Technology artifacts, and the *Work-Package* class for business changing initiatives.

For simplicity, Atlas assume that change initiatives produce the expected results at the very end, meaning on the date they become *non-productive*. To model a situation where a change initiative produces results before its *non productive* date, one must use a hierarchy of change initiatives, where the top one has a child for each result it has to produce with the *non productive* date matching the desired date.

The relationship between change initiatives and object life-cycle states is a key factor to simplify the update and management of enterprise artifacts life-cycle dates. For example, if a project, that is replacing one application by a new one delays one month, this delay should be reflected in the life-cycle of both applications, specifically in the old application retirement date and in the new application alive date.

In Atlas this relationship is established by a user defined reference property declared at the class level, either at the change initiative class or at the object class. For example, in a typical scenario, one may define the *Alive List* and the *Retired List* reference properties in a project class to identify the objects that will become alive and retired. Such relationships allow the update of objects life-cycle according to the life-cycle of the corresponding change initiative. In this scenario is, the kickoff and completion of change initiatives yields the following behavior:

- When the change initiative starts: propagate the current date to both change the initiative alive date and gestation date of objects in the *Alive List*.

– When the conclusion date of a change initiative has a new forecast: propagate the forecast date to the retired date of the change initiative, to the alive date of objects in the *Alive List* and to the retired date of objects in the *Retired List*.

Such state propagation is not a built-in behaviour in Atlas and must be configured in state propagation rules associated to object life-cycle. This provides more flexibility and independence regarding to the actual life-cycle states the user can define for each class. A more advanced usage of these dependencies and rules is to perform repository coherency validation analysis, in particular based on the time dependent relationships[33].

4 Conclusion

The goal we pursuit with Atlas and Enterprise Cartography is to be minimize the effort required to provide updated and trustful EA views, in enterprises where their architecture is the result of many local architectures, each designed with different drivers and optimization factors. So far, we can say that achieving this goal is fundamentally dependent on the enterprise's ability to provide good enough plans and descriptions of their change initiatives. Using the scenario presented in previous section, this boils down to having the *Alive List* and the *Retired List* of change initiatives completed. In such case the effort required to keep *AS-IS* and *TO-BE* EA views updated is non-existent in practical terms.

Since the *Alive List* and the *Retired List* of a given change initiative represent, for all practical matters, what needs to be done in that change initiative and what impact it has in the enterprise, they are also related with the cost, time and risk of that change initiative. In other words, the can conclude that the more the enterprise is willing to manage cost, time and risk of their change initiatives, the more likely will be the success of our proposal.

However, we also conclude that we still miss a clear and easy mapping between project management practices and standards with the EA artifacts and their life-cycle evolution to easy the completeness of change initiatives. For example, if a project intends to create and deploy a new application service on a given date on a given infrastructure, then these facts should be stated somehow in the project plan and deliverables, so that they can be subsequently imported to complete the *Alive List* and the *Retired List* of the corresponding change initiative.

The Project Management Body of Knowledge[5] defines the work-breakdown structure as a "A hierarchical decomposition of the total scope of work to be carried out by the project team to accomplish the project objectives and create the required deliverable." Since enterprise artifacts life-cycle is by definition a result (deliverable) of a project, they are likely to appear in project work-breakdown structure. We are currently working in the establishment of a more formal mapping between enterprise artifact life-cycle and project work-breakdown structure[11]. The idea is to integrate Atlas with the project management tools used in the enterprise and capture the information needed for completion of change initiatives and make enterprise cartography a more simple and accessible capability to enterprises.

Acknowledgements

This research was supported by the Link Consultings project IT-Atlas (n 11419), under the IAPMEI, 2020 Portuguese PO CI Operational Program.

References

1. Hoogervorst J. A. (2009) ”Enterprise governance and enterprise engineering”. Springer.
2. The Open Group. (2011). ”TOGAF Version 9.1”. (10th ed.). Zaltbommel, the Netherlands, Van Haren Publishing.
3. 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive System. Replaced by ISO/IEC 42010.
4. Proper, H.A., Winter, R., Aier, S., de Kinderen, S. (Editors)(2017) ”Architectural Coordination of Enterprise Transformation”, Springer.
5. Project Management Institute (2017). ”A Guide to the Project Management Body of Knowledge” (PMBOK Guide)Sixth Edition Newtown Square, Pa, September 22.
6. Bischoff, S., (2017). ”The Need for a Use Perspective on Architectural Coordination” pp 87-98, in Proper, H.A., Winter, R., Aier, S., de Kinderen, S. (Editors)(2017) ”Architectural Coordination of Enterprise Transformation”, Springer.
7. Roth, S., Zec, M., Matthes, F., (2014): Enterprise Architecture Visualization Tool Survey 2014. Technical Report. Sebis, Technische Universit at Munchen.
8. Ugwu, K. (2017). ”Understanding the complementary relationship between enterprise architecture and project management”. in Architecture and Governance Magazine (online version, accessed in May 2018).
9. Labusch, N., (2017). ”Information Requirements for Enterprise Transformation” pp 111-117, in Proper, H.A., Winter, R., Aier, S., de Kinderen, S. (Editors)(2017) ”Architectural Coordination of Enterprise Transformation”, Springer.
10. Schomburg, K., Barker, T. (2011). ”Integrating the IT PMO with enterprise architecture for better government”. In proceedings of PMI Global Congress 2011North America, Dallas, TX. Newtown Square, PA: Project Management Institute.
11. Bernardo, M., Sousa, P., (2018) ”Portfolio Management. Enabling a dynamic Organization IS representation”. 22nd International Congress on Project Management and Engineering (ICPME 2018), Madrid, Spain.
12. Sousa, P., Lima, J., Sampaio, A., Pereira, C., (2009). ”An Approach for Creating and Managing Enterprise Blueprints: A case for IT Blueprints”. The 21st International Conference on Advanced Information Systems, Lecture Notes in Business Information Processing, vol. 34, pp. 70–84, Springer-Verlag, The Netherlands .
13. Sampaio, A. (2010) An Approach for Creating and Managing Enterprise Blueprints. Master Thesis in University of Lisbon - fenix.tecnico.ulisboa.pt
14. Leal, R., (2010) Navigation model between Architectural Views: ”An approach for a new paradigm: Navigation in Enterprise Architecture”, Master Thesis in University of Lisbon - fenix.tecnico.ulisboa.pt

15. Sousa P., Gabriel R., Tadao G., Carvalho R., Sousa P.M., Sampaio A. (2011). "Enterprise Transformation: The Serasa Experian Case." In: Harmsen F., Grahlmann K., Proper E. (eds) Practice-Driven Research on Enterprise Transformation. PRET 2011. Lecture Notes in Business Information Processing, vol 89. Springer, Berlin.
16. Sousa P., Sampaio, A. Leal, R. (2014). "A case for a Living Enterprise Architecture in a Private Bank", In proceedings of the 8th Workshop on Transformation & Engineering of Enterprises (TEE 2014), July, Geneva.
17. Tribolet, J; Sousa, P.; and Caetano, A. (2014). The Role of Enterprise Governance and Cartography in Enterprise Engineering. Enterprise Modelling and Information Systems Architectures, 9 (1): 38-49.
18. Morris, C.W., (1938). "Foundation of the Theory of Signs", International Encyclopedia of Unified Science, Vol. 1, No. 2.
19. Dietz, J. (2006). "Enterprise Ontology - Theory and Methodology", Springer.
20. The Open Group. (2015). "ArchiMate 2.1 Specification". Van Haren Publishing, Zaltbommel, www.vanharen.net.
21. IEEE. (2010). "Systems and software engineering – Vocabulary", in ISO/IEC/IEEE 24765:2010(E) , pp.1-418, Dec. 15
22. https://www.merriam-webster.com, accessed in September 2017.
23. Tratt, L. (2205) "Model transformations and tool integration", Software and Systems Modeling, May 2005, Volume 4, Issue 2, pp 112122.
24. W3C (1999) XSL Transformations (XSLT).
25. Grosskopf, Decker and Weske (2009). "The Process: Business Process Modeling using BPMN". Meghan Kiffer Press.
26. Sousa, P., Rito, A., Alves Marques, J. (1995). "Object Identifiers and Identity : A Naming Issue", In IEEE Proceedings of the 4th International Workshop on Object Orientation in Operating Systems, Lund, Sweden.
27. Wachsmuth, G. (2007). Metamodel adaptation and model co-adaptation. In The European Conference on Object-Oriented Programming 2007 (ECOOP) (Vol. 4609, pp. 600624).
28. Cicchetti, A., Di Ruscio, D., Eramo, R., and Pierantonio, A. (2008). Automating co-evolution in modeldriven engineering, in Proceedings of the 12th IEEE International Enterprise Distributed Object Computing Conference, IEEE Computer Society, pp. 222231.
29. Silva, N., Ferreira, F., Sousa, P., Mira da Silva, M. (2016). "Automating the Migration of Enterprise Architecture Models". In International Journal of Information System Modeling and Design (IJISMD) 7.2 pp 72-90.
30. Silva, N., Mira da Silva, M., Sousa, P., (2018). "A Tool for Supporting the Co-Evolution of Enterprise Architecture Meta-models and Models", In 27th International Conference on Information Systems Development, August 22-14, Lund, Sweden.
31. Silva, N., Sousa, P., Mira da Silva, M.(2018)."CO-EVOC: An Enterprise Architecture Model Co-Evolution Operations Catalog". In 24th Americas Conference on Information Systems, August 16-18, New Orleans, USA.
32. F.,Carolina, P. Sousa, A. Sampaio. (2016) "Visualiazação da Evolução da Arquiteturas Empresariais", 16 Conferênçia Associação Portuguesa de Sistemas de Informação (CAPSI), 22 September, Porto.
33. Xavier, A., Vasconcelos, A., Sousa, P. (2017) "Rules for Validation of Models of Enterprise Architecture-Rules of Checking and Correction of Temporal Inconsistencies among Elements of the Enterprise Architecture". International Conference on Enterprise Information Systems ,2, 337-344.