Atlas: the Enterprise Cartography Tool

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

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

Abstract. The need to maintain architectural representations of enterprises is an indescribable fact nowadays given their constant evolution. However, despite the large number of enterprise architecture tools available, organization 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. In Atlas, architectural viewpoints are generated automatically from the models and have a time bar allowing a seamless navigation in time, from past to future models.

Keywords: Enterprise Cartography, Enterprise Architecture, Architecture Visualization, Architecture tools

1 Introduction

The need to maintain explicit knowledge of the architecture of enterprises is an indisputable fact nowadays given their constant transformation and evolution. 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 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].

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*[3], that includes more than one hundred health organizations.

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].

[3] Serviço Nacional de Saúde: https://www.sns.gov.pt

Despite the differences, they both require enterprise architecture as input data and they both causes changes in enterprise, thus likely induces changes in the enterprise architecture. 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 enterprise architecture application scenarios and the corresponding literature sources, making clear importance of knowing the enterprise architecture for enterprise maintenance, planning and evolution. In it´s simpler and basic from, information about enterprise architecture boils down to the list of enterprise artifacts and their dependencies. Given the variability and quality of existing enterprise architecture tools and their representation capabilities[7], one could argue that enterprises could easily create and maintain an Enterprise Architecture 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 enterprise architecture, being office tools one of the most 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 enterprise architecture 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 enterprise architecture tool, effort is required to update it in conformity with this new design. The person that performs such update in the enterprise architecture 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 meeting, 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 enterprise architecture tools used. The fastest the enterprise changes, the more effort is required to keep 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 enterprise´s 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 enterprise´s 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 competences on some technology could be sustained on the plans of a change initiative to build new products using that technology, regardless the fact that of that change initiative has succeeded or not.

AS-IS models . These models represent the current enterprise. *AS-IS* models are required for current operations and for reacting to events. As in previous models, the plans of current change initiates must be part of *AS-IS* models.

TO-BE models . These models represent the enterprise future, and are critical for planning the next change initiatives, as recognized both by enterprise architecture[8, 9] and project management[10, 11] communities.

In fact, to plan a project that is going to start in 6 month´s time, one needs to know how the enterprise will be when the project starts, not as it is today. For example, to plan a training session for a department staff to occur in 4 months ahead, one needs to know the number of people the department will have 4 months ahead, not the ones it has today.

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 the 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 architectural 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.

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 architectural views . Our finding is that hand-draw architectural views will likely become obsolete much faster 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[4] was designed to explore the findings presented above. It is an enterprise architecture tool conceived to reduce the effort needed to keep architectural views updated in enterprises designed in a decentralized, distributed and asynchronous process using multiple design tools. In Atlas, architectural views are generated automatically and have a time bar allowing a seamless navigation in time, from past to future state of each viewpoint. 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 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 architectural maps.

[4] www.linkconsulting.com/atlas

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 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 projects.

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 enterprise architecture. 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 a key artifact type in the meta-model. Thus, its architectural statements can be observed in the enterprise reality assessment and pinned down to a given point in time. Assuming that change initiatives are enterprise artifacts is in line with most architecture guidelines such as ISO 42010[23], or with the Work Package concept in both TOGAF[2] and ArchiMate[20].

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 artifact (the change initiative). Notice that the assumption that change initiative 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, dead, retired.

In figure 1 we present these five states, their fundamental transitions and there relation with productiveness.

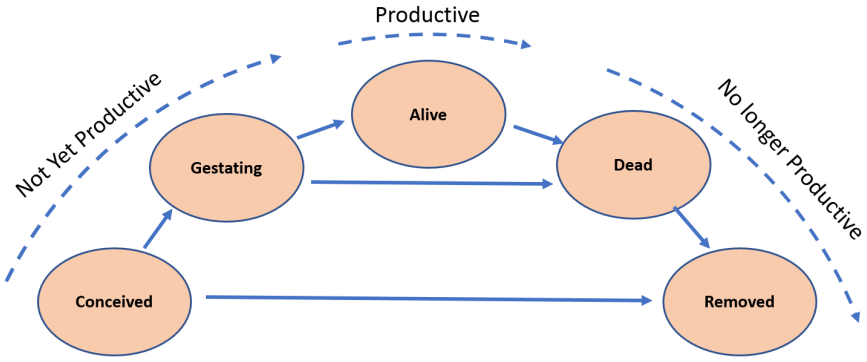

Fig. 1. Fundamental Artefact Lifecycle

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. For example, the idea of a change initiative that puts an artifact into production at some point means that both artifacts are 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 stared. An ongoing change initiative is necessarily a productive artifact, since it must have an impact on other productive artifacts and is expected to produce value after its completion.

Alive . An alive artifact is a productive one. As defined before, alive artifacts can play purposeful roles in the enterprise to create value. Conceived and Gestation artifacts have impact on life objects, namely in the ones that are conceiving or creating them. But their impact is passive, and 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 or gestating before being alive. One can consider other productive states, as for example *Deprecated*, but it is not relevant for the purpose of this paper.

Dead 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 dead 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-dead 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 artifacts that are productive in the enterprise *AS-IS* model did exist before in the enterprise *TO-BE* models. From principle 3, one knows that *Gestating* stage precedes the *Alive* stage, and from principle 2, one knows that *alive* artifacts did exist before in some change initiative plans, or *TO-BE* models. Therefore, *alive* artifacts in the *AS-IS* enterprise state did already existed as *Conceived* or *Gestating* in the *TO-BE* models of the some change initiative which means, from principle 1, that they existed as the *TO-BE* of the enterprise state.

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 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].

3 Atlas Overview

Altas is a web based enterprise architecture tool providing all functional elements 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 will be 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 also 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 architectural views that we called blueprints.

Atlas support configuration of behavior associated with architectural 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.

We now focus the description of how Atlas supports enterprise cartography. A typical scenario is presented in figure2 where information from several sources, including office tools, is processed and feeds the generation of AS-WAS, AS-IS and TO-BE architectural views and analysis. In this scenario, enterprise artifacts and their relationships exist in multiples sources, and each source will likely have it´s own metamodel. Furthermore, information in each source may be related with the past, present or future of enterprise architecture.

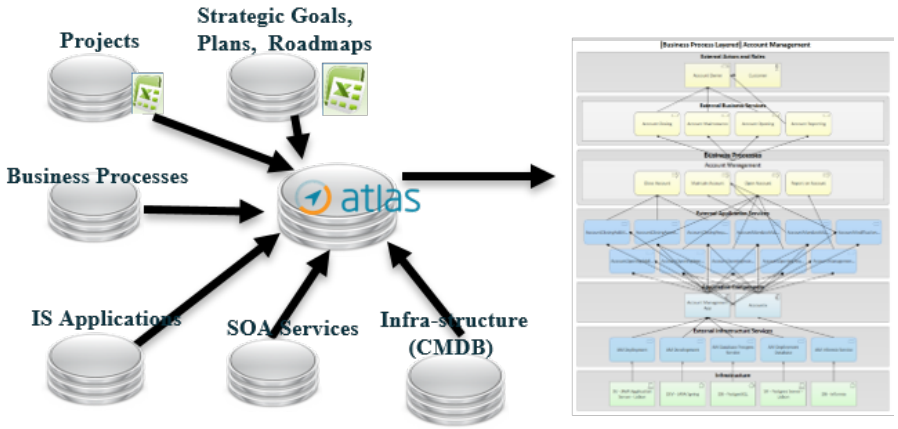

Fig. 2. Atlas collecting AS-IS and TO-BE models and produce AS-WAS, AS-IS and TO-BE architectural views

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[24]. Among the different technologies to handle model transformation[24], 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[25] which is very common and convenient for processing XML files. The second deals with the actual model transformation, and uses an high level type-based rules that operates on types, instances and relationships.

Consider the case where one wants to import a BPMN[26] model and that *Data Stores* must be imported as *Applications* in the Atlas repository. To configure such rule, one assign 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 file format into Atlas XML integration 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 simple imports the transformed objects into the Atlas repository. In the example given, the transformation rules script is a simple "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 batch's 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 units may use the same notation differently and different rules might be required.

Another rather important aspect is to be able to perform different behaviour in case of the imported artifacts match existing objects or if new objects were created in the Atlas repository. The rules allows a behaviour 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 figure2 one needs to find a mechanism to match objects existing in different systems that correspond to the same enterprise artifact. In other words, one needs a 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 figure2, 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 been done simply by change the application name, from *CRM* to *ERP*. However, in this case, the name has change but the *OID* in the a 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 artifact, breaking the biding between OIDs and an 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[27] and System generated 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 two hospitals 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 selected the 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.

Atlas full object names are unique URLs composed as follows:

serverURL/RepositoryName/ClassName/instanceNameField_N. .. .instanceNameField_0.

The concatenation of the fields *instanceNameField_N. ... .instanceNameField_0* must be unique within the objets of the same *ClassName*. All fiels but the last are optional (i.e can be *null*), except the last one (*instanceNameField_0*) that cannot be null.

3.3 Object State

In Atlas, object state is defined as a the set of values of their property, 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 are available as Numeric, Text, Rich Text, Hyperlink, Boolean and for reference properties, users can define their own reference types.

Object properties do not change type (structure) over time, this means that, for example, an integer property cannot not become a reference property. However object sate, as 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. 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 demoted object. However, in some situations it is very convenient to add properties to references. This is done by binding a reference property with a class, whose structure becomes the structure of that reference. Any class

defined in the meta model can be assigned, and the same class can be assigned to different reference properties.

3.4 Metamodel 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 do 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 objects inspection after that time, however, as soon as the time bar goes back that time the deleted 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* operation promotes a textual property to a reference property to a class created from the original 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[28–30] 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 if the necessary actions can be performed automatically of if they requires human intervention[31–33].

3.5 Object Life cycles

Object life cycle is defined over a set of object date properties. Therefore, 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 architectural view.

Despite the arbitrary number of states a life-cycle may have, life-cycle states are marked as being:

Not Yet Productive The artifact if not yet productive. These includes all defined states prior the object productive state (alive).

Productive The artifact is alive. This include all user defined states that the object may go through while it´s live. For example, one may consider in production, deprecated, or others.

No Longer Productive . These includes all states after the object is no longer productive.

Users define life-cycle states in order and classifying each state as *Productive* or not *Productive*. The not *Productive* states prior to the first *Productive* state are considered *Not Yet Productive*, and states after the last *Productive* state are considered *No longer Productive*. This classification is used in several Atlas built in functions as for example to hide or display *Not Yet Productive* and *No longer Productive* in architectural views.

3.6 Change Initiatives

According to the Cartography principles, enterprise artifact evolves in its life-cycle as a result of some change initiative. In Atlas, any used defined class may be a change initiative class, and more than one change initiative class may exist. For each class in the meta model, one may associate another class as it´s the change initiative class, as well as the relationships types between change initiatives with related enterprise artifacts.

In Atlas, change initiatives are optional concepts, since they are not directly related with object life-cycle. However, in practice they are fundamental to establish a process to manage the evolution of objects life-cycle. For example, if project aims at implementing application *newApp* and decommissioning application *oldApp*, then changes in the project´s execution dates should be reflected in the dates of applications *newApp* and *oldApp*.

In practice, this is supported by add two lists of objects in change initiatives[12]:

- the *productive list*, in which referred object will become productive when the transformation initiatives becomes no longer productive.
- the *non-productive list*, in which referred will be no longer productive when the transformation initiatives becomes no longer productive.

These lists are used to update the life-cycle of objects whenever the life-cycle of the change initiative changes. A typical definition is to propagate the dead date of the change initiative to the alive date of objects in the *productive list* and to the date of in the *non-productive list*. Notice that this behaviour must be defined in state propagation batches since it is not pre-defined in in Atlas.

The relations between change initiatives and the changed objects allows different validation analysis that can be used for repository coherency purposes, in particular based on the time dependent relationships[34] .

3.7 Visualizing *AS-WAS*, *AS-IS* and *TO-BE* Architectural Views

In atlas, architectural views are graphical representation of AS-WAS, AS-IS and TO-BE enterprise models, generated on-the-fly by resolving it´s URL. Architectural 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 objects to be presented inside another objects.

Once generated, architectural views are interactive interfaces where users can navigate in time, run predefined analyses, jump to another architectural views and drag objects between containers, triggering containers *out-query* and *in-query* to do the necessary changes in the model 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*and *in-query* on the *Legal Unit* containers with the actor *John* as argument.

We will focus on the navigation in time capability of architectural views. By moving the time bar back and forward one can see the architecture view in two visualization modes:

Absolute Mode: Visualize a point in Time . The viewpoint is presented with the contents that corresponds to the time given. Two filters can be applied
- Only productive objects at selected time are presented. Symbols are presented in their natural color.
- 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.

Gap Mode - Visualize the Gap between two points in time Ti and Tf . The viewpoint shades the objects symbols according to the difference of life cycle state of each object on the selected times Ti and Tf[35, **?**]. In this mode, artifacts are classified as:
- Introduced . An artifact is considered to be introduced between Ti and Tf if it is not in the living state at Ti but it is in Tf is already in this state.
- Removed. An artifact is considered to be removed between Ti and Tf if Ti is in the living state and Tf is in the dead state.
- Changed. An artifact is considered to be changed between Ti and Tf if both Ti and Tf are in the alive and if at least one artifact with which it has a relation undergoes a substitution relation, that is, be introduced or removed between Ti and Tf. The users can select the depth of the graph traversal analysis and for each step, the references to be used.

4 Conclusion

Acknowledgements

This research was supported by the Link Consulting's project IT-Atlas (nº 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 IEEE Std 1471-2000 First edition 2007-07-15
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)[U+0096]Sixth Edition – 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 project management". in Architecture 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 2011—North 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, Heidelberg
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, Switzerland.
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) , vol., no., pp.1-418, Dec. 15
22. https://www.merriam-webster.com, accessed in September 2017.
23. IEEE. ISO/IEC/IEEE 42010:2011. (2011). "Systems and software engineering - Architecture description". ISO.org. November 24.
24. Tratt, L. (2205) "Model transformations and tool integration", Software Systems Modeling, May 2005, Volume 4, Issue 2, pp 112–122
25. W3C (1999) XSL Transformations (XSLT).
26. Grosskopf, Decker and Weske (2009). "The Process: Business Process Modeling using BPMN". Meghan Kiffer Press.
27. 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.
28. Wachsmuth, G. (2007). Metamodel adaptation and model co-adaptation. In The European Conference on Object-Oriented Programming 2007 (ECOOP) (Vol. 4609, pp. 600–624).
29. Cicchetti, A., Di Ruscio, D., Eramo, R., and Pierantonio, A. (2008). "Meta-model differences for supporting model co-evolution," in Proceedings of the 2nd Workshop on ModelDriven Software Evolution.
30. 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. 222–231.
31. 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.
32. 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.
33. 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.
34. 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, SCITEPRESS.
35. F.,Carolina, P. Sousa, A. Sampaio. (2016) "Visualiazação da Evolução da Arquiteturas Empresariais", 16º Conferência Associação Portuguesa de Sistemas de Informação (CAPSI), Porto 22 September.