Enterprise Architecture Evolution
Gap Analysis, a primitive of visualisation

Carolina Ferreirinha
Instituto Superior Técnico
Portugal
carolinaferreirinhamarques@tecnico.ulisboa.pt

ABSTRACT
Nowadays, enterprises are faced with the need to constantly adapt to the changing business needs that are either driven by economic, regulatory or technical reasons. The alignment of these changes in relation to the organisational strategy not only requires a deeply understanding of the necessary changes, but also the impact of these changes in the architecture, which are usually achieved through the description of the various architectural states over the years provided by the enterprise architecture tools. Thus, this thesis aims to improve the way of how the architectural changes are represented and then visualised within these descriptions, by introducing a primitive of visualisation that reflects the comparison between architectural states throughout time.

Keywords
Enterprise Architecture Evolution; Primitive of Visualization; Gap Analysis; EAMS

1. INTRODUCTION
Enterprises nowadays face challenges like changing markets, security threats, evolving technologies and new regulations that drive the need to adapt the enterprise [1]. The management of the evolution of an enterprise constitutes a complex challenge due to the involvement of different stakeholders in the planning of the different changes that occur, often simultaneously.

An enterprise architecture, defined by Zachman as the “the set of representations required to describe a system or enterprise regarding its construction, maintenance, and evolution”, supports these changes, through the representation of multiple architectural situations over time.

The challenge of transformation comprises different perspectives related temporally: the current state of the architecture, the future state of the architecture and the transition states of the architecture [2].

Some enterprise architecture tools allow you to visualize some of these perspectives, identifying, for example, which elements appear in the architecture and which cease to exist with time, identify a transformation plan or even compare, in a very simplified way, what is the architectural situation before and after the implementation of a particular project. However, these perspectives are never integrated nor take into account possible impacts of the represented changes, which results in a limitation regarding the visualisation of those changes in a systematic, uniform and completely understandable manner, throughout all organizational domains, available in any architectural view.

The lack of uniformity in the visualization of the evolution and the difficulty in comparing various architectural situations over time, makes a comprehensive analysis of the architectural changes and inherently its impact become an arduous task and complicates the alignment of these changes in relation to organizational strategy.

Since the architectural evolution is considered a process [3], it is assumed as a target for a methodological and structured vision that may be applicable to any architectural situation and, therefore, will be able to support the needs of all stakeholders, which reveals an added value to the process of decision-making.

1.1 OBJECTIVES
As aforementioned, the process of architectural evolution incorporates different perspectives that are temporally related: the actual state of the architecture, the future state of the architecture and the transition states of the architecture. Thus, the visualization of the transformation must reflect all relevant aspects for the architecture in these three states, thus allowing a full understanding of all evolutions and inherently of their impact.

The work presented has as main objective the improvement in the visualization of architectural evolution over time through the design of a primitive of visualization able to translate all the architectural changes (gap) in a particular architectural representation, as well as possible impact situations between any two architectural states that can refer to the past (as-was), present (as-is) and future (to be).

Thus, this work is intended to:
- From a theoretical perspective, formalize the calculation of the architectural comparison between two temporal moments $t_1$ e $t_2$ in any architectural representation;
- From a theoretical perspective, define a clear visualisation semantic to represent the output of that comparison in any architectural representation;
- From a practical perspective, implement the gap analysis primitive of visualization (composed by the comparison calculation and the respective representation of the result) in the EAMS tool blueprints.

The achievement of the objectives presented is directly related to the response to the following question:

*Is the calculation of the architectural gap between two distinct moments in time homogeneous for all enterprise architectural domains?*

The enterprise architecture, due to its complexity and multidisciplinary approach, can be described in various domains (sub-architectures), where each domain is developed by distinct stakeholders with their own concerns [2]. In order to develop the gap analysis primitive of visualization, it becomes important to understand whether the heterogeneity among the various domains
influences the calculation of the architectural gap, i.e., if the comparison between states of a particular architecture domain (e.g. Information Systems architecture) is implemented in a different way from the comparison of architectural states of another domain (e.g. Business architecture).

2. STATE OF THE ART

Some of the first approaches in the subject of enterprise architecture planning and transformation were introduced by Spewak[4], [5], Pulkkinen [6] and Niemann [7]. However, most of the research results were only focused on the unidirectional planning process that would improve the current architecture, establishing therefore the target architecture. The process of transforming the baseline architecture on target architecture was considered in a barely relevant way [8].

More recently [9], [10], Buckl presented a new perspective considered in a barely relevant way [8]. Transforming the baseline architecture on target architecture was establishing therefore the target architecture. The process of planning process that would improve the current architecture, most of the research results were only focused on the unidirectional

In [12], is discussed the role of transformation nature in enterprise change is a consequence of the execution of a project or regarding EA planning, emphasizing the role of projects as the main drivers, by introducing the principle that an enterprise change is a consequence of the execution of a project or a similar activity[8], [11].

In [12], is discussed the role of transformation nature in corresponding management methods, introducing different types of architectural states, namely the AS-IS state (current) and the TO-BE state (future), which should be considered when planning the transformation.

However, the actual transformation may happen differently than planned, introducing the notion of WILL-BE state [8]. The WILL-BE state is then used as a basis for subsequent planning steps instead of TO-BE state.

According to [13], there are different kinds of changes that can be considered: extend, delete and modify. An entity is extended when it is changed, keeping the initial information, structure and behaviour. In contrast, it is said that an entity is modified when it is changed, not keeping the initial information, structure and behaviour. On the other hand, an entity is removed if it is no longer part of the architecture.

Later on, in [8], Aier defines a transformation model as a successor relation between two architectural models in different moments in time, which may assume the types, 0:1, 1:0, 1:1, N:1, 1:N and N:M, where the elements in the right are successors of the elements in the left.

By mapping the work of Boer [13] and Aier [8], it is easy to perceive that each successor relation type introduced in the latter work is reflected in the removal of an architectural element, and in some cases the introduction of one or more architectural elements that replace it. The situation of modification and/or extension can be translated into a special case of the successor relation type 1:1 in which the successor entity is the same as the predecessor entity, with differences in their structure, information and behaviour (caused by successor relations that occur in their relations). The work of Boer [13] does not include the introduction of architectural elements (i.e. the element becomes part of the architecture without replacing any other integral element of architecture) as its focus is directed to the analysis of change impact in existing architectural elements. If the element did not exist in the architecture and came into existence then, although the architecture has changed, the introduced element has not changed, merely joined the architecture.

However, it can be said that the two work converge as they depict architectural changes as a replacement of architectural elements at two different moments in time.

More recently [11], Buckl found out that many papers present a perspective that modelling EA transformation can be achieved through the combination of a period of validity of a given architectural element, leaving behind three crucial aspects for the representation of the transformation: EA elements do not change accidentally, EA elements may replace each other and EA elements have a lifecycle.

The second aspect discussed by Buckl (EA elements can replace each other) meets the work of Boer [13] and Aier [8] in the definition of the change of an architectural element as a successor relation. Still, Buckl introduces another type of architectural change reflecting not only the replacement of EA elements but also taking into account the internal state of the entities of the architecture (third aspect). Thus, the change of the internal state of an architectural element becomes another kind of architectural change that may occur simultaneously or not with other types of changes already discussed (introduction, removal, modification and extension).

In the subject of architectural elements lifecycles, Sousa [14] introduces four invariant states to classify all organizational artefacts:

- **Gestating**
  Represents the state where an artefact is after it is conceived, i.e. after it starts being planned, designed or produced. At this state, the artefact does not yet exist as an active element of the enterprise.

- **Alive**
  Represents the state where an artefact is after it is at birth. In this state the artefact is able to produce behaviour as part of the organizational processes and transactions.

- **Dead**
  Represents the state where a living or gestating artefact is in a state, the artefact does not yet exist as an active element of the enterprise.

- **Retired**
  Represents the post-death state where the artefact is unable to further interact with other artefacts.

In [11], Dieffenthaler describes how gaps can be derived from two EA models in different moments in time (t1 and t2). The comparison of these models results in the intersection of two sets (architectural elements in t1 and t2, respectively), allowing the identification of three subsets: onlyCurrentArchitecture (elements that only exist in the architecture in t1, i.e. where removed), onlyTargetArchitecture (elements that only exist in the architecture in t2, i.e. were introduced) and stable (elements that exist in the architecture in t1 and t2, but may have been extended, modified or unchanged).

Typically, the change in architectural element causes changes in neighbouring architectural elements (direct impact). However, these changes may propagate throughout the architecture and affect other elements (indirect effect). So every little change in architectural elements may cause multiple effects and have consequences throughout the architecture [15].

In [13], Boer states that the impact of an architectural element change in another is based on the relation type semantics between them and in the type of change under consideration, i.e., if two elements are related, the occurrence of an architectural change in one of them may not impact the other which means that the occurrence of impact depends on the semantics of the relation between them.
Furthermore, in [15], Langermeier identifies five architectural relation type classes (located at, provides, consumes, structurally dependent on and behaviourally dependent on) and for each one of them, defines a set of impact rules that take under account the worst and best case for each type of change considered in his work (removal, extension and modification).

Some enterprise architecture tools aim to support the visualisation of architectural evolution using the concepts already presented.

For example, the ABACUS Avolution tool, uses gap analysis concept to colour the manually generated views and also provides project planning information.

![Figure 1 – View showing changes with colour outlines](image)

The view presented in Figure 1, depicts the location of the architectural changes provoked by the execution of a particular project in the context of an enterprise. The identification of the components that undergo changes is accomplished using a different colour (purple) on its contour and the description of such changes is shown when placing the mouse in the component through a tooltip. This view allows a comparative analysis between two architectural states (baseline architecture and target architecture) making thus a gap analysis. However, being a manually generated view, it does not allow the immediate visualization of the impact on the architecture, since it does not include the analysis of the execution of multiple projects simultaneously.

![Figure 2 – Projects interdependences matrix](image)

Figure 2 represents another ABACUS view focused in architectural evolution analysis, which lists the interdependencies between projects that are responsible for the transformation. The matrix has all the projects listed horizontally and vertically, and each cells contains the architectural elements changed by the execution of the referencing projects. Therefore, the diagonal of the matrix shows the architectural elements affected by each project and the other cells can help identifying potential conflicts.

In turn, BiZZdesign tool also features a set of views which aims to support the visualisation of transformation based on the concepts of gap analysis and roadmap.

![Figure 3 – Roadmapping browser (gap analysis)](image)

Figure 3 depicts a comparative analysis between a baseline architecture and a target architecture and, therefore, a gap analysis. In this view, the architectural elements are coloured according to the architectural states they belong: only baseline architecture (orange), only target architecture (green) or both (blue).

![Figure 4 - Application Roadmap](image)

In Figure 4 is represented an architectural transformation plan, with a particular periodicity (yearly). For each year, are specified which elements are part of the architecture, allowing the identification of which architectural elements were removed or introduced. The relation between adjacent columns allows both the identification of the origin of new elements and the destination of removed elements.

EAMS is an Enterprise Architecture tool in which all architectural views have a time slider associated, allowing its content to vary from the past (AS-WAS) to the present (AS-IS) or any point in time (TO-BE). The time-slider is marked with the moments in time, in which there were projects that produced a change in that architectural view. When the handle moves along the slider, and crosses a mark, the name of the project that led to change appears on the left and the content of the architectural view changes.
The concept “depth of the graph” is used to refer to the minimum number of relations that links two artefacts. Throughout this work, an artefact is considered indirectly related with another if there is a path between them and the depth of that path is higher than 1. Thus, two artefacts that are related through a path whose depth is one, are directly related.

It is also assumed that each artefact has a $T_{BeginDate}$ property that defines the date in which the artefact becomes “alive”, a $T_{EndDate}$ property that defines the date in which the artefact becomes “dead” and a relations property that corresponds to the set of direct relations of the artefact (relations in which the artefact is the origin or destination). Each artefact also has a lifecycle associated [14], [17], constituted by different states, where each state is represented by a colour.

Additionally, it is assumed that each relation has a $T_{BeginDate}$ property that defines the date in which it becomes “alive”, a $T_{EndDate}$ property that defines the date in which it becomes “dead” and yet a property that reflects its relationType.

### 3.3 ARCHITECTURAL CHANGES

The analysis of the works of Boer [13], Aier [8], Buckl [11], Sousa [14] and Tribolet [17] lead to a final set of architectural changes that may occur between two distinct moments in time ($t_1$ and $t_2$):

- **An architectural artefact can be introduced in the architecture**
  An artefact is introduced in the architecture if in $t_i$ it isn’t part of the architecture, but is in $t_j$, i.e., in $t_i$ the artefact is not alive but it is in $t_j$.

- **An architectural artefact can be removed from the architecture**
  An artefact is removed from the architecture if in $t_i$ it is part of the architecture, but isn’t in $t_j$, i.e., in $t_i$ the artefact is alive but it isn’t in $t_j$.

- **The lifecycle state of an architectural artefact can be changed**
  The lifecycle state of an architectural artefact has changed if in $t_i$ it is in a different state than in $t_j$. Obviously, this state change may constitute an architectural introduction or removal if the artefact becomes or ceases to be alive between $t_i$ and $t_j$.

- **A relation between two architectural artefacts can be introduced**
  A relation between two architectural artefacts is introduced in the architecture if in $t_i$ it isn’t part of the architecture, but is in $t_j$, i.e., in $t_i$ the relation is not alive but it is in $t_j$.

- **A relation between two architectural artefacts can be removed**
  A relation between two architectural artefacts is removed from the architecture if in $t_i$ it is part of the architecture, but isn’t in $t_j$, i.e., in $t_i$ the relation is alive but it isn’t in $t_j$.

As discussed, the architectural changes may occur both in architectural artefacts and relations. This work aims to improve the representation and visualization of all these changes, through the gap analysis primitive.

### 4. PROPOSAL

As aforesaid, in this work we intend to introduce a primitive of visualisation that reflects the calculation of the architectural comparison between two temporal moments $t_1$ and $t_2$. 

![Figure 5 - EAMS time-slider](image)

EAMS is an Enterprise Architecture tool in which all architectural views have a time slider associated, allowing its content to vary from the past (AS-WAS) to the present (AS-IS) or any point in time (TO-BE). The time-slider is marked with the moments in time, in which there were projects that produced a change in that architectural view. When the handle moves along the slider, and crosses a mark, the name of the project that led to change appears in the context of an architectural view. The elements represented in blue are “alive” in the interval and the elements represented in red in the context of an architectural view. The elements represented in Figure 5 depicts the architectural situation in an interval of time, in the context of an architectural view. The elements represented in blue are “alive” in the interval and the elements represented in red are “dead” in the interval.

The analysed tools, despite being of the most advanced in this field, only allows the visualisation of architectural evolution in particular situations, justifying the need to define a primitive of visualisation that captures all the gaps in any architectural state, between any two moments in time, in any architectural view across all organizational domains.

### 3. FOUNDATIONS

#### 3.1 PRIMITIVE OF VISUALISATION

In the scope of the proposed solution, a primitive of visualisation defines a way to add information to any architectural view by introducing specific semantics of that kind of information. The primitive of visualisation presented in this work aims to translate the direct comparison in any architectural view between two distinct moments in time.

![Figure 6 - Primitive of Visualisation](image)

#### 3.2 ENTERPRISE MODELLLED AS A GRAPH

An enterprise $E$ can be modelled as a directed graph $G_E$ of artefacts and their relations, respectively the nodes and edges of the graph. A graph is composed by a pair $(A_E, R_E)$, where $A_E$ represents the artefacts of the enterprise and $R_E$ represents the relations between them [16]. Throughout this work, a relation between two artefacts $a_1$ and $a_2$ will be represented as $r_{a_1,a_2}$, being $a_1$ and $a_2$ the origin and the destination, respectively.

The lifecycle state of an architectural artefact has changed if in $t_i$ it is in a different state than in $t_j$. Obviously, this state change may constitute an architectural introduction or removal if the artefact becomes or ceases to be alive between $t_i$ and $t_j$.

- **A relation between two architectural artefacts can be introduced**
  A relation between two architectural artefacts is introduced in the architecture if in $t_i$ it isn’t part of the architecture, but is in $t_j$, i.e., in $t_i$ the relation is not alive but it is in $t_j$.

- **A relation between two architectural artefacts can be removed**
  A relation between two architectural artefacts is removed from the architecture if in $t_i$ it is part of the architecture, but isn’t in $t_j$, i.e., in $t_i$ the relation is alive but it isn’t in $t_j$.
4.1 ARTEFACTS CLASSIFICATION
The gap analysis concept reflects a direct comparison between two architectural states in different moments in time and its main objective is the achievement and the subsequent detailed visualization of this comparison. This proposal follows the approach introduced in [1], in that it similarly implements the result of the comparison by classifying architectural artefacts into distinct subsets. Recalling the architectural changes discussed in section 3.3, it is easily concluded that the sets proposed by Diefenthaler (onlyCurrentArchitecture, onlyTargetArchitecture and stable) only cover two types of architectural changes between two distinct moments in time (\(t_1\) and \(t_2\)): the introduction and the removal of architectural artefacts.

In addition, the primitive presented also suggests a visual mapping for each set, allowing the identification of the architectural changes that occur between any two moments in time, in any architectural view as introduced in section 3.1.

Thus, this work concretizes the architectural comparison between two distinct moments in time (\(t_1\) and \(t_2\)) completely, through the classification of the artefacts that constitute the architecture in five particular sets:

- **Introduced architectural artefacts**
  Set of architectural artefacts that in \(t_1\) are not part of the architecture but in \(t_2\) do, i.e., an artefact is classified in this set if it isn’t alive in \(t_1\) but is in \(t_2\).

- **Removed architectural artefacts**
  Set of architectural artefacts that in \(t_1\) are part of the architecture but in \(t_2\) don’t, i.e., an artefact is classified in this set if it is alive in \(t_1\) but isn’t in \(t_2\).

- **Changed architectural artefacts**
  Set of architectural artefacts that in \(t_1\) are in a different lifecycle state than the one they are in \(t_2\), without this transition constituting an introduction or removal of the architectural artefact.

- **Potentially changed architectural artefacts**
  Set of artefacts that are part of the architecture both in \(t_1\) and \(t_2\), keep their state lifecycle state unchanged, but have, a certain level of depth, an alive relation with artefacts that in \(t_1\) have a different lifecycle state than in \(t_2\) (changed architectural artefacts) or have alterations in their set of relations (in \(t_1\) are related with artefacts with which they are not related in \(t_2\) and/or vice versa). For a given artefact, the introduction or removal of a related artefact always leads to a change in its relations set, at certain level of depth. However, this change may occur without the introduction or removal of artefacts. The artefacts of this group did not suffer any direct architectural change, nevertheless, the changes occurring in their context place them into a potentially changed situation.

- **Stable architectural artefacts**
  Set of artefacts that are part of the architecture both in \(t_1\) and \(t_2\), keep their state lifecycle state unchanged, and do not integrate, at a particular level of depth, none of the above sets.

The work Boer [13], as already discussed, offers some relevance to relation semantics with regard to the impact of a change in the neighbouring artefacts, i.e., if two artefacts are related, the occurrence of an architectural change in one of them, may or may not impact the other, depending on the type of the relation that links them. This idea of the architectural change in a particular artefact causing possible architectural changes in another artefact is present in this work when the completion of the gap analysis proposal comes the set of potentially changed artefacts. When assumed, for example, that an artefact is potentially changed due to a change in the lifecycle state of a related artefact, between two different moments in time, it is inherently assumed a possible impact of that change in an artefact that did not actually suffered it. What is added now is that an architectural artefact is potentially changed due to a change of the lifecycle state of a related artefact between two different moments in time, depending on the type of relation that unites them (at a certain level of depth, obviously), or also the classification of an artefact as potentially changed due to a change in its relations set between two different moments in time, depends on the semantics of that relation type.

The artefacts classification as potentially changed in the proposed gap analysis should then take into account the semantics of the relations between artefacts.

4.2 RELATION TYPES SEMANTICS
Remembering and summarizing the information already presented, an architectural artefact can be considered potentially changed between two moments in time (\(t_1\) and \(t_2\)) if:

- Its set of relations is changed at a certain level of depth.
- The lifecycle state of a related artefact at a certain level of depth, changes.

However, as aforementioned, the occurrence of these events is not enough to classify a particular artefact as potentially changed. Exemplifying even more abstractly, starting from the first presented event, changing the set of relations of a given artefact caused by the removal of a relation of type A, at a certain depth and between two different moments in time, can place it in a situation of potentially changed. However, the change in the set of relations caused by the removal of a relation of type B, under the same conditions, may not be enough to place it in that situation and the artefact, despite the change that occurred, is considered stable. Reinforcing the idea, it all depends on the semantic of the relation type that actually caused the change in the set of artefact relations. For instance, in a more concrete way, assuming that there is an artefact A using an artefact B (relation of type uses), the transition of the artefact B to the deprecated state will potentially lead to an effect on artefact A, and therefore it makes sense that the Application A integrates the set of potentially changed artefacts. In the opposite case, assuming that the same change occurs in the artefact A, it is not reflected any change in the artefact B (relation of the type used by), and therefore classify it as possibly changed if artefact A changes its lifecycle state is illogical.

Thus, to enable the classification of an artefact as potentially changed it is then necessary to classify each relation type that binds two artefacts and perceive the existence or not of a potential effect from the occurrence of one of the two events aforementioned, taking into account the semantics of the same.

In a concrete manner, for calculating the set of potentially changed artefacts, each type of relation will be classified as has effect or has no effect, depending on its semantics. If a type of relation assumes the class has effect and occurs an architectural change in a particular artefact that constitutes a change of the set of relations of that type and/or a state change in a related artefact, through relations of this type, at a given level depth, between two different moments in time, after the completion of the gap analysis calculation, the artefact is classified as potentially changed. Otherwise, if the relation in question is of the class has no effect, the artefact will not be considered potentially changed, remaining stable. Notice that it is the class of each relation type that dictates
the navigation in depth throughout the enterprise graph looking for architectural changes.

4.3 FORMALIZATION

The gap analysis primitive defines a way to automatically visualise a direct comparison between two architectural states in distinct moments in time \((T_1\) and \(T_2\)) applicable to any architectural view, independently of the domain or modelling language, adding a colour mapping to translate the semantic of this comparison.

In addition to the graph representing the enterprise and obviously the two moments in time for which it is intended to carry out the architectural comparison, the primitive also takes into account the depth (which dictates the granularity of the analysis with respect to the set of potentially changed artefacts), and also the set of interesting relation types \(S(\text{RelationTypes})\). The latter can only contain the relation types classified as has effect and acts as a filter which influences the paths (relation) to cover. Thus, the gap analysis primitive is defined by the function:

\[
\text{GapAnalysis}(T_1, T_2, G_E, S(\text{RelationTypes}), \text{depth})
\]

The application of this primitive allows quick identification of which artefacts and relations kept (stable, changed and potentially changed), introduced or removed between these two architectural states, as shown in section 4.1.

Thus, during the calculation of the gap analysis, each artefact of the enterprise, represented in a \(Gr = (A, R)\) graph, is classified in one of the following sets:

- **Architectural artefacts introduced** between \(T_1\) and \(T_2\), i.e., the set of artefacts \(A_{\text{Introduces}}^{T_1 \& T_2}\) that only exist in the architecture in \(T_2\). Formally, this set can be described as:

\[
A_{\text{Introduces}}^{T_1 \& T_2} = \{ a \in A_E | a.\text{BeginState} > T_1 \land (a.\text{BeginState} \leq T_2 \lor a.\text{EndState} > T_2) \land a.\text{EndState} = \text{null} \}
\] (1)

- **Architectural artefacts removed** between \(T_1\) and \(T_2\), i.e., the set of artefacts \(A_{\text{Removed}}^{T_1 \& T_2}\) that only exist in the architecture in \(T_1\). Formally, this set can be described as:

\[
A_{\text{Removed}}^{T_1 \& T_2} = \{ a \in A_E | a.\text{BeginState} \leq T_1 \land a.\text{EndState} \leq T_2 \land a.\text{EndState} > T_2 \}
\] (2)

- **Architectural artefacts changed** between \(T_1\) and \(T_2\), i.e., the set of artefacts \(A_{\text{Changed}}^{T_1 \& T_2}\) that exist in the architecture both in \(T_1\) and \(T_2\) but in \(T_1\) and \(T_2\) have a different lifecycle state than in \(T_2\). Formally, this set can be described as:

\[
A_{\text{Changed}}^{T_1 \& T_2} = \{ a \in A_E | (a.\text{BeginState} \leq T_1 \land a.\text{EndState} > T_2) \lor ((a.\text{BeginState} > T_2 \land a.\text{EndState} = \text{null}) \land (a.\text{BeginState} > T_2) \land (a.\text{BeginState} \neq \text{state}(a, T_1)) \}
\] (3)

- **Architectural artefacts potentially changed** between \(T_1\) and \(T_2\), i.e., the set of artefacts \(A_{\text{PotentiallyChanged}}^{T_1 \& T_2}\) that exist in the architecture both in \(T_1\) and \(T_2\) and keep their state lifecycle state unchanged, but are related with at least one artefact (throughout an has effect relation type) that changed its lifecycle state between \(T_1\) and \(T_2\) and/or is the origin or destination of relations classified as has effect that were introduced and/or removed at a certain level of depth. Formally, this set can be described as:

\[
A_{\text{PotentiallyChanged}}^{T_1 \& T_2} = \{ a \in A_E | a.\text{BeginState} \leq T_1 \land (a.\text{EndState} > T_2 \lor a.\text{EndState} = \text{null}) \land \text{state}(a, T_1) \neq \text{state}(a, T_2) \land \exists a_{adj}, a_{adj} \in \text{adjacent}(G_E, a, S(\text{RelationTypes}), \text{depth}) \land a_{adj}.\text{BeginState} \leq T_1 \land a_{adj}.\text{EndState} > T_2 \lor a_{adj}.\text{EndState} = \text{null} \land \text{state}(a_{adj}, T_1) \neq \text{state}(a_{adj}, T_2) \}
\]
Notice that the adjacent function, given a particular artefact, returns all the related artefacts up to a given level of depth without any filter. At a level of depth = 0, the adjacent function returns the element itself. The function state returns the lifecycle state of a given artefact in a particular temporal moment.

The gap analysis primitive introduces a visual semantic that will be achieved through a colour mapping for each of the artefacts depending on the set they belong to (Table 1).

<table>
<thead>
<tr>
<th>Classification Sets</th>
<th>Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_{\text{Introduced}} )</td>
<td>Green</td>
</tr>
<tr>
<td>( A_{\text{Removed}} )</td>
<td>Red</td>
</tr>
<tr>
<td>( A_{\text{Changed}} )</td>
<td>Combination of two colours representing the lifecycle states in ( T_1 ) and ( T_2 ), respectively.</td>
</tr>
<tr>
<td>( A_{\text{PotentiallyChanged}} )</td>
<td>Yellow</td>
</tr>
<tr>
<td>( A_{\text{Stable}} )</td>
<td>No colour</td>
</tr>
</tbody>
</table>

Table 1 - Gap analysis colour mapping

5. GAP ANALYSIS IN EAMS BLUEPRINTS

As presented in section 3.1, the proposed primitive of visualisation aims to include semantics of the concept gap analysis in any architectural view, in the case of EAMS tool, to any blueprint. Therefore, it becomes possible to compare two architectural states between two different moments in time in any blueprint of the tool, colouring each represented artefact according to the set to which it belongs: introduced, removed, changed, potentially changed or stable (4.1 and 4.3).

5.1.1 Primitive Mapping

For classifying each of the artefacts represented in a particular blueprint in each of the presented sets (introduced, removed, changed, potentially changed or stable) between two distinct moments in time \( t_1 \) and \( t_2 \), it becomes imperative to be able to identify in each moment in time \( t \):

a. If an artefact is alive or dead
b. The state of the artefact lifecycle
c. If each relation belonging to the set of direct relations of an artefact is alive or dead

Additionally, it is also necessary to identify:

d. The classification of each relation type (has effect or has no effect)

Table 2 depicts the match between the stated tasks and how they are implemented in EAMS.

<table>
<thead>
<tr>
<th>Task</th>
<th>EAMS</th>
</tr>
</thead>
</table>
| Identify if an artefact is alive or dead in a particular moment in time \( t \) | It is possible by reusing the functionality getActiveState (artefact, domain, \( t_1, t_2 \)), with \( t_1=t_2=t \), that returns the artefact state in the interval from \( t \) to \( t \):
| Identify the lifecycle state of an artefact in a particular moment in time \( t \) | Each lifecycle state indicates if an artefact in that state is alive or dead:
| Identify if each relation belonging to the set of direct relations of an artefact is alive or dead in a particular moment in time \( t \) | It is possible through the identification of the situation of the artefact that assumes the other end (origin or destination) of each relation in that \( t \). It presents limitations in that it does not identify if a relation is alive or dead independently of the artefacts that constitute it. Classification of each relation type as has effect or has no effect. It is possible through the evolution of the table that holds information about the relation types, which now has a Boolean field has effect. |

Table 2 - Matching between tasks and their implementation in EAMS

5.1.2 Visualization Modes

In EAMS, all architectural views have a time slider associated that allows to visualise, for each time interval, which artefacts are alive or dead in that interval if the Highlight option is disabled and otherwise, allow to visualise the lifecycle state of each artefact in that interval. The selection or not of the Highlight option in the blueprints intrinsically represents two modes of visualizing the transformations through time: a more detailed mode, which shows the lifecycle states of each artefact and a more general mode that allows only to identify alive or dead artefacts. The idea of propagating these two visualisation modes into the gap analysis primitive, proved to be interesting because it would allow the comparison between two architectural states at different granularity levels. Thus the gap analysis primitive is now composed by two visualisation modes that follow the existing logic in the blueprints of EAMS:

- **No Highlight Mode**

  The calculation of the architectural comparison between two temporal moments, achieved by the primitive presented only takes into account the isAlive state of the artefacts, which generally means that without the Highlight option enabled, there is no changed artefacts set, and the potentially changed artefacts set only takes into account the introduction or removal of related artefacts (through a relation type classified as has effect).

- **Highlight Mode**

  The calculation of the architectural comparison between two temporal moments, achieved by the primitive presented, takes into account the various lifecycle states of the artefacts, which generally means that with the Highlight option enabled, it is possible to classify the artefacts in each of the aforesaid sets (section 4.1). Yet, the sets of introduced and removed artefacts now constitute the set of changed artefacts, given that the calculation of the primitive ceases to take into account the property isAlive. Thus, the introduction and removal of artefacts between two moments in time are regarded as a change in the artefact lifecycle state and not as a change in the property isAlive.
6. DEMONSTRATION

In order to demonstrate the application of the gap analysis primitive in the blueprints of EAMS, in the real context of the enterprise architecture of a German credit agency, we performed an architectural comparison between 2012/05/21 and 2016/08/26. The definition of the artefacts lifecycle states is depicted in Figure 7.

Figure 7 - Artefact lifecycle states

As mentioned before, EAMS blueprints allow the visualisation of the architecture in any temporal moment. Figure 8 and Figure 9 represent the Application Organic blueprint in 2012/05/21 and 2016/08/26, respectively. Each artefact is coloured according to their lifecycle state at that moment.

Figure 8 - Blueprint Application Organic in 2012/05/21 with highlight enabled

Figure 9 - Blueprint Application Organic in 2016/08/26 with highlight enabled

The analysis of both figures, allows to conclude that between 2012/05/21 and 2016/08/26, the following architectural changes occurred:

- **under implementation (isAlive=false)** → **productive (isAlive=true)**
  The applications Portoberechnung, Portoberechnung, App-Category Firmenkunden Portal and Web became productive.

- **productive (isAlive=true)** → **deprecated (isAlive=false)**
  The applications LexiCan, HiScout, DeviceSecure Manager and CRM-UK became deprecated.

- **productive (isAlive=true)** → **decommissioned (isAlive=false)**
  The applications SyncMe, Loga and Unternehmensdatenbank were deactivated.

Figure 10 and Figure 11 show the visualization resultant of the application of the gap analysis primitive between 2012/05/21 and 2016/08/26, at a level of depth = 0, with and without highlight respectively. Both views allow the direct identification of the changes that occurred between the two moments of analysis.

Figure 10 - Gap Analysis in the blueprint Application Organic between 2012/05/21 and 2016/08/26 with highlight enabled at a level of depth = 0

Figure 11 - Gap Analysis in the blueprint Application Organic between 2012/05/21 and 2016/08/26 with highlight disabled at a level of depth = 0

Figure 10 reflects the result of applying the gap analysis primitive with the highlight mode enabled, allowing a clear identification of which applications suffered changes and what has changed. Analysing a single visualization, it is concluded that:

- The applications Portoberechnung, Portoberechnung, App-Category Firmenkunden Portal and Web belong to the set of changed applications given that they moved from the under implementation state to the productive state.

- The applications LexiCan, HiScout, DeviceSecure Manager and CRM-UK belong to the set of changed applications given that they moved from the productive state to the deprecated state.

- The applications SyncMe, Loga and Unternehmensdatenbank belong to the set of changed applications given that they moved from the productive state to the decommissioned state.

On the other hand, Figure 11 reflects the result of applying the gap analysis primitive with the highlight mode disabled where it is possible to clearly identify architectural artefacts that were introduced or removed between the selected dates. Analysing a single visualization, it is concluded that:

- The applications Portoberechnung, Portoberechnung, App-Category Firmenkunden Portal and Web belong to the set of introduced applications given that they moved from the under implementation state, where isAlive=false, to the productive state, where isAlive=true.

- The applications SyncMe, Loga and Unternehmensdatenbank belong to the set of changed applications given that they moved from the productive state, where isAlive=true, to the decommissioned state, where isAlive=false.

Note that at a level of depth = 0, it isn’t possible to identify potentially changed applications.

Figure 12 shows the visualization resultant of the application of the gap analysis primitive between 2012/05/21 and 2016/08/26, at a
Figure 12 - Gap Analysis in the blueprint Application Organic between 2012/05/21 and 2016/08/26 with highlight enabled at a level of depth $= 1$

As it is possible to verify, the application of the gap analysis primitive at a higher level of depth enables the identification of potentially changed artefacts (coloured in yellow).

In contrast to the other sets resultant from the application of the primitive, the set of potentially changed artefacts semantics does not enable, in a concrete way, what changes occurred so that a particular artefact integrates this set, making it necessary to analyse the gap analysis detail window in the context of these artefacts.

The gap analysis detail window depicts the reason why a given artefact is classified in a certain set. In the particular case of the potentially changed artefacts, the detail window allows the identification of the architectural changes that introduced it in that set, i.e., the architectural changes that occur in the related artefacts.

If the primitive is applied at a level of depth higher than 1, the detail window shows the path between the potentially changed artefact and the architecturally changed artefact.

Thus, Figure 13 depicts the gap analysis detail window for the potentially changed application AMVBatchClient.

Figure 13 - Gap analysis detail window in the context of the application AMVBatchClient

The analysis of the detail window allowed us to conclude that AMVBatchClient is considered potentially changed given that the service it uses (VPBatchFA) becomes deprecated.

Complementarily, a tabular representation of the result of the application of the primitive has also been implemented. This representation allows to quickly overviewing all the classification sets, offering a quantitative perspective. The gap analysis overview window is presented in Figure 14.

Figure 14 - Gap analysis overview window

7. CONCLUSION

The perception of a change in the architecture implicitly suggests the comparison of the architectural states before and after it occurs. Thus, the representation and visualization of this comparison presents itself as an important element regarding the understanding of the evolution of the architecture over time, that is, the set of architectural changes that happened, are happening or will happen.

Accordingly, the first step in the development of the solution was based on the identification of a finite set constituted by the architectural changes that may occur either on the artefacts or the relations between artefacts, in two different moments in time. The set found [Introduction of an artefact, Removal of an artefact, change of an artefact lifecycle state, introduction of a relation between artefacts, removal of a relation between artefacts] proved to be independent of the artefact type in question, i.e., the architectural changes that may occur between two moments in time are analogous to constituent artefacts of different architectural domains (e.g., information systems, business or technology architectures). If the set of architectural changes is homogeneous throughout all architectural domains, then it is possible to infer that the calculation of the proposed architectural comparison (classification of artefacts taking into account the finite set of architectural changes found) is also homogenous throughout all domains, which leads directly to the answer to the question presented in section 1.1.

The proposed approach presents a primitive of visualization for architectural comparison between any two points in time, based on a holistic perspective of the gap analysis concept.

7.1 CONTRIBUTIONS

The development of this work allowed the accomplishment of a few contributions, namely:

- Concept of primitive of visualization in an architectural view
- Definition of the possibility of adding information to any architectural view, introducing specific semantics of this type of information through visualization primitives.

- Systematization of architectural changes
- Analysis and systematization of all the architectural changes that may occur in artefacts and relations between two different points in time.

- Formalization of the architectural gap analysis
- Formalization of architectural comparison calculation between two different points in time applicable to any architectural view, based on the occurrence of architectural changes as well as the spread of these changes.

  - Gap analysis holistic perspective
    - Introduction of new classification sets (changed artefacts set and potentially changed artefacts) regarding the typical gap analysis (only takes into account artefacts introduction or removal) in order to capture all the architectural changes.

  - Analysis of relation types semantics
    - Identification of the necessity of taking into account the semantic of each relation type in gap analysis calculation.
    - Identification of two relation types classification classes: has effect and has no effect.
    - Analysis and classification of Archimate meta-model relation types.
Visualization semantics of gap analysis
Definition of a visual semantic capable of showing the result of the architectural gap analysis between two different points in time.

Additionally, the article “Visualizing Enterprise Architecture Evolution” was published in the conference CAPSI [18].

7.2 FUTURE WORK
The proposed solution allows to identify all the architectural changes that occurred between two different moments in time and infer the possibility of certain artefacts suffer the impact of these changes taking into account the semantics of relation types.
However, a deeper analysis not only regarding the semantics of relation types, but also the type of artefact that undergoes changes and the nature of that changes, may allow the proposed primitive to evolve towards the capability to identify more accurately which artefacts effectively suffered the impact of a change that took place in another artefact, and more relevant, in what way does that impact it. That is, there is a high potential to evolve the gap analysis primitive to a primitive of impact analysis which could manifests itself as an important aid in planning and even detection of conflicts between architectural projects.

8. REFERENCES