Analytic Pattern and Tool for Analysis of a Gap of Changes in Enterprise Architectures

Richard Dijkstra[1] and Ella Roubtsova[2]
[1]*Sligro Food Group, The Netherlands*
[2]*Open University of the Netherlands, The Netherlands*

Keywords: ArchiMate Enterprise Model, Analytic Pattern, Gap of Changes, Tool for Analysis of a Gap of Changes, Case of Replacing Legacy System with ERP with Best of Breed.

Abstract: Change analysis of enterprise architectures is usually done by observing differences between two enterprise architectures, As-Is and To-Be. If the As-Is and To-Be have a lot of differences, it is problematic to manually create a correct view on changes. This paper proposes a revision of a definition of the Gap of Changes and defines it as a language independent analytic pattern suitable for using in tools. The paper describes a tool built on the basis of this definition. The change analysis without the tool and with the tool output has been tested in a workshop. The added value of the tool was acknowledged by the workshop participants.

1 INTRODUCTION

Change analysis of enterprise architectures is one of the repeated activities fulfilled when an enterprise is changed. This analysis is usually done manually, by observing visualizations of As-Is and To-Be enterprise architectures.

One of the notations used for modelling of enterprises is the ArchiMate language, an Open Group Standard (The Open Group, 2018). For visualization of changes, ArchiMate specifies a concept *Gap* as "a statement of difference between two plateaus", where "a plateau represents a relatively stable state of the architecture that exists during a limited period of time" (The Open Group, 2018). This concept does not define what architecture elements and relations belong to a *Gap*.

A precise definition of a "Gap of Changes" has been proposed by Bakelaar et al. (Bakelaar et al., 2016) as a view derived from the As-Is (current) and To-Be (desired) architecture and comprising obsolete, changed, unchanged, new elements and their relations in the context of a change. The definition includes the new relationships "replaced by" and "extended by". The "Gap of Changes" has been illustrated in ArchiMate with one industrial case (Bakelaar et al., 2016).

The initial goal of this work was to evaluate the use of the "Gap of Changes" on another industrial case of transforming a Legacy system into an Enterprise Resource Planning (ERP) with Best of Breed (BOB) components. However, the manual derivation of changes for this case appeared impossible without a tool support. The case initiated the idea of automatic creation and analysis of the "Gap of Changes" and added the followup research question on how such a tool can be implemented.

In order to realise this idea, the definition of a "Gap of Changes" has been revised and built into a tool for creating and analysis of a Gap. The tool has been tested within the case where the initial legacy architecture is replaced by a desired architecture that comprises an ERP (SAP) system combined with a Best-of-Breed (BOB) solution (Stibo Master Data Management (StiboSystems, 2018)). The usefulness of the tool has been tested in a workshop with several participants.

The structure of the paper is the following:

- Section 2 presents the related work.
- Section 3 contains the revised definition of a "Gap of Changes".
- Section 4 reports the tool implementation, its input and output.
- Section 5 describes the workshop for testing of the usefulness of the tool on a case of replacement of a legacy system with an ERP/BOB.
- Section 6 discusses the usage of the revised definition and the tool for analysis of changes.
- Section 7 concludes the paper.

Dijkstra, R. and Roubtsova, E.
Analytic Pattern and Tool for Analysis of a Gap of Changes in Enterprise Architectures.
DOI: 10.5220/0007655502260233
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 226-233
ISBN: 978-989-758-375-9

2 RELATED WORK

2.1 Enterprise Modelling in ArchiMate

Enterprise modelling is supported by many enterprise modelling languages, for example, 4EM (Sandkuhl et al., 2014)), ArchiMate (The Open Group, 2018) etc. These languages present an enterprise in different dimensions. In this paper, we visualise enterprise architectures using the ArchiMate language, so we use the ArchiMate visualization conventions. However, the research results of this paper are also applicable for other enterprise modelling languages.

The ArchiMate diagrams have a vertical and a horizontal directions. The vertical direction from bottom to top expresses the structure of a business system: technology, application, business. The horizontal direction from left to right expresses the aspects of the system: passive, behaviour and active structure.

An enterprise architecture view is always an abstraction showing the elements of structure, vision, and evolution of the organization, business processes, information systems and infrastructure. It is not a model of all IT functionality. For visualization of the IT functionality, one should use the UML notations, such as class diagrams, sequence diagrams and state diagrams.

Changeability is an immanent property of any enterprise architecture. A special enterprise architecture view can be devoted to expressing changes. For such a view, a textual presentation of changes is always needed to support discussion between enterprise architects and different business roles. This will increase their ability to understand and accept changes as the effect of a design conversation (Hoffman and Maier, 1967).

The "Gap" concept within ArchiMate (The Open Group, 2018) is a candidate for such a view on changes as it is defined to present differences between two plateaus in enterprise architectures. The specification of the ArchiMate (The Open Group, 2018) does not define what to include into a "Gap".

2.2 Need for Analysis of Enterprise Models and their Changes

It has been noticed by several authors, that the Enterprise Modelling community has a strong focus on enterprise (Dietz and Hoogervorst, 2008), "been restricted in their use to non-analysis purposes" (Sunkle et al., 2013). In other words, there is a need in support of the architecture analysis.

On the other hand, there is a possibility for analysis. The architectural languages, such as ArchiMate (The Open Group, 2018), or 4EM (Sandkuhl et al., 2014), present sets of elements and their relations and suitable for analysis. The matter is only to find the useful repeated analysis questions and built them as formal requests into tools.

Such commercial tools as Sparx EA (Sparxsystems, 2019) and BizzDesign Enterpise Studio (Bizzdesign, 2019) have analysis functionality allowing to define patterns. However, they have not recognized and defined a Gap of Changes as a pattern, repeatedly used used by different business roles for planning and implementation activities.

(Diefenthaler and Bauer, 2013) have recognised a Gap of Changes and proposed a solution for "gap analysis using semantic web technologies on a high-level current and target state of an enterprise architecture". The authors identify only the replacement relations between elements of two architectures, whereas also other relations exist between the elements of As-Is and To-Be architectures.

2.3 Existing Definition of a Gap of Changes

In (Bakelaar et al., 2016), it has been noticed, that As-Is and To-Be architectures can be seen as the largest plateaus and the elements from an As-Is and a To-Be architectures can be associated with a "Gap".

A formal definition of a *Gap of Changes* has been given in (Bakelaar et al., 2016) for visualization of changes.

A *Gap of Changes* is defined as a tuple:

$$Gch = (O_{obsolete}, O_{new}, O_{unchanged}, O_{changed},$$

$$R_{obsolete}, R_{new}, R_{replaced-by}, R_{extended-by}, R_{border}).$$

The sets of objects are defined as follows:
$O_{obsolete} = \{o |\ o \in O_{AsIs}\ and\ o \notin O_{ToBe}\}$ - obsolete objects, visualised as grey boxes.
$O_{new} = \{o |\ o \notin O_{AsIs}\ and\ o \in O_{ToBe}\}$- new objects, visualised as green boxes.
$O_{unchanged} = \{o |\ o \in O_{AsIs}\ and\ o \in O_{ToBe}\ and$
$\forall x : (o,x) \in R_{ToBe} \Leftrightarrow (o,x) \in R_{AsIs}\}$ - unchanged objects, visualized in the original ArchiMate colour.
$O_{changed} = ((O_{AsIs} \cap O_{ToBe}) \setminus O_{unchanged})$ - changed objects, visualized as orange boxes.

The relationships are defined as follows:
$R_{obsolete} = \{(a,b) | (a,b) \in R_{AsIs}\ and\ (a,b) \notin R_{ToBe}\}$ - depicted as grey arrows.
$R_{new} = \{(a,b) | (a,b) \notin R_{AsIs}\ and\ (a,b) \in R_{ToBe}\}$ - depicted as green arrows.
$R_{<replaced-by>} \subseteq O_{obsolete} \times O_{new}$- black arrows. Figure 1.

$R_{<extended-by>} \subseteq O_{changed} \times O_{new}$- black arrows. Figure 2.
$R_{border} \subseteq (O_{unchanged} \times O_{changed}) \cup (O_{changed} \times O_{unchanged})$ - green arrows.

Figure 1 and Figure 2 illustrate the relations between As-Is and To-Be objects in the view "Gap of changes":

- $< replaced-by >$: $(Previous\ app., New\ app)$ and
- $< extended-by >$: $(ApplicationComponent, ApplicationFunction)$.

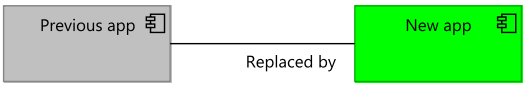

Figure 1: "Previous app" is" Replaced By" "New app".

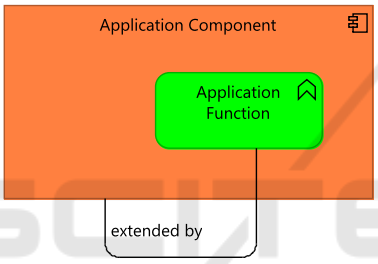

Figure 2: "Application Component" is "Extended By" "Application Function".

3 REVISED GAP OF CHANGES FOR TOOLING

3.1 Observations

Experiments with changes have shown that the initial definition of a *Gap of Changes* defined (Bakelaar et al., 2016) leaves room for overlooking changes.

Figure 3: R_{new} : ("Application Component1","Application Component 2").

Let us show examples:
Figure 3 shows that the relation R_{new} (green) may be defined between new elements that both do not exist in the As-Is architecture. This case is not part of the R_{new} definition in (Bakelaar et al., 2016).

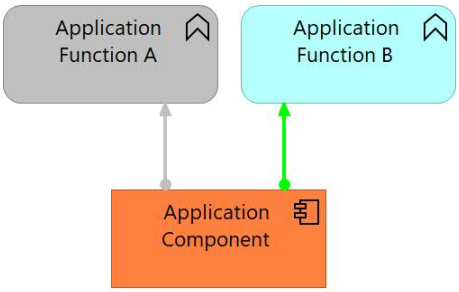

Figure 4: Obsolete and Border.

Figure 4 shows that the "Application Function A" (in grey) is obsolete. The "Application Function B" in this example is unchanged and qualified as "unchanged object", although it is assigned to a changed Application Component (Figure 4).
This unchanged object "Application Function B" is a border object (visualised in original blue colour) needs a border relation (green). The definition of *Gap of Changes* by (Bakelaar et al., 2016) does not require such a relation to the unchanged elements included into the Gap.
The Gap of Changes combined with the standard modelling ArchiMate methods, is of great benefit because the architect needs to be more precise during the architectural conversations to make an exact replacement and extension of design.
We propose a revised definition of a Gap of Changes with some practical improvements based on the described observations. The definition is designed as a structure or a pattern for automatic analysis of changes.

3.2 Revised Gap of Changes (GOC')

GIVEN: As-Is architecture: (O_{AsIs}, R_{AsIs}).

To-Be architecture: (O_{ToBe}, R_{ToBe}).

$$O_{obsolete} = \{o |\ o \in O_{AsIs}\ and\ o \notin O_{ToBe}\},$$

$$O_{new} = \{o |\ o \notin O_{AsIs}\ and\ o \in O_{ToBe}\}.$$

LET us introduce new constructions.

- A changed object is a part of both As-Is and To-Be and it has at least one relation in both As-Is and To-Be and has at least one relation that is part of To-Be and not of As-Is or part of As-Is and not of To-Be.

$$O_{changed} = \{o | o \in O_{AsIs} \wedge$$

$$\exists x | (x \in O_{AsIs} \wedge (o,x) \in R_{ToBe} \Leftrightarrow (o,x) \in R_{AsIs}) \wedge$$

$$\exists y | (y \in O_{AsIs} \wedge (o,y) \in R_{ToBe} \not\Leftrightarrow (o,y) \in R_{AsIs})\}.$$

We assume that a changed object cannot be in isolation (with empty set of relations).

- Unchanged objects are part of both As-Is and To-Be and are not part of the set changed objects. The set of unchanged objects $O_{unchanged}$ may be large compared to the set of changed $O_{changed}$. Using the set of $O_{changed}$ simplifies the analysis of changes.
- The new concept of a border object is defined.
  Let us remind that a border relation relates a changed object and an unchanged object and depicted as a green arrow:

$$R_{border} \subseteq (O_{unchanged} \times O_{changed}) \cup (O_{changed} \times O_{unchanged}).$$

  The consequence of this definition is a set of border objects, being unchanged objects that need to be included into a Gap of Changes. A border object is an unchanged object that has at least one border relation.

$$O_{border} = \{o | (o \in O_{unchanged} \wedge \exists x : x \in O_{AsIs} : (o,x) \in R_{border})\}$$

  The concept of a "border object" becomes very useful if two or more changes have a shared border object. This might not be noticed visually.

The revised Gap of Changes (GOC') looks as follows:

$$GOC' = (O_{obsolete}, O_{new}, O_{border}, O_{changed}, R_{obsolete}, R_{new}, R_{replaced-by}, R_{extended-by}, R_{border}).$$

This definition can be used as a structure from which the sets of new, obsolete, changed and unchanged objects and relations can be queried for an estimation of the Gap and for work planning.

4 TOOL FOR ANALYSIS A GOC'

Even for small changes, the recognizing of the changed elements is error prone. The number of elements in a GOC' can be large, for example, in complete system replacements, like our case, shown in Figure 5, Figure 6, Figure 7. The analysis of changes has to be supported with a tool.

The tool for analysis a GOC' called EAGOC Notebook is written as part of this research in Python 3.0 (Python, 2018) using a Jupyter Notebook (Jupyter, 2018). The tool analyzes the repository using CSV exports from the modelling tool Archi (Archi, 2018).

The EAGOC Notebook uses simple data structures like dictionary and dataframe (Pandas.DataFrame, 2018). The initial repositories of As-Is and To-Be architectures are used as an input for the GOC' structure. An enterprise architect can make selections among elements of the GOC' and produce lists of new, obsolete, changed and border objects and relations.

The selection of subsets for analysis is based on the definition of GOC'. The queries are independent from the analysed case, written once and can be reused.

For example, the set of new objects:

$$O_{new} = \{o | \, o \notin O_{AsIs} \; and \; o \in O_{ToBe}\}.$$

In the EAGOC Notebook this is checked with the following assertion using sets, where the symbol == is used in Python for testing equality:

```
set_new_elements==
set_tobe_elements-set_asis_elements
```

Another example is a set of the "replaced by" relations:

$$R_{<replaced-by>} \subseteq O_{obsolete} \times O_{new}.$$

In the EAGOC Notebook the following set assertion:

```
{(relations_dict[r].source,
relations_dict[r].target)
  for r in
   set_replaced_by_relations}<=
    {*product(set_obsolete_elements,
     set_new_elements)}
```

is used to select the list of pairs of $R_{<replaced-by>}$. This list is used to support visualization of the pairs.

The complete set of relationships of the Gap of Changes can be listed by the following request:

```
goc_elements_df
[goc_elements_df.state==
    'changed object']
      [['object_seq',
      'object_type',
      'name',
      'element_id']]
```

When an element in the To-Be needs extra attention, one can view this in Archi, but also obsolete objects may appear. In the EAGOC Notebook, the obsolete elements can be filtered with the function:

```
<element>.get_related_tobe_elements()
```

The tool is available on GitHub (Dijkstra, 2018)

5 TESTING THE USEFULNESS OF GOC' AND TOOL

5.1 Industrial Case Study

Our industrial case study is the Product Data management at the Sligro Food Group (Sligro, 2018). Food distribution uses customer data, vendor data and food product data with attributes like product name and storage conditions. These data safeguards unambiguous communication between sales, procurement and supply chain operations. This case focuses on a system that enables data management as a process of getting products ready to be bought, shipped and sold. For this, Sligro has two legacy systems PIM and SIAM which will be replaced by one component called Stibo, because having a single component for managing product data is less complex than using two components.

In the As-Is architecture Figure 5, the product data as data object "Product" is being used in the application function "Product selection". This application function is realised through the application component PIM. PIM also realises the application function "Assortment management", where products are selected for availability on the web-shop. In the SIAM component, the logistic reference data is added to the product data, e.g. the shops where that product is being sold.

In the To-Be architecture (Figure 6), the application components PIM and SIAM are replaced by Stibo. In Figure 6 the reader can also see the application functions "Product selection" and "Assortment management", but now they are realised by the application component Stibo. The replacement is driven by the need for consistency of data used by different application functions and uniformity of interfaces.

The Gap of Changes is shown in Figure 7. The replacement of PIM and SIAM by Stibo is modelled according the GOC' with obsolete (grey) PIM and SIAM and a new (green) application component Stibo. Stibo has a *replaced-by* relation with PIM and SIAM.

The integration platform has the same interface "Online updates" to the web-shop in both As-Is (Figure 5) and To-Be (Figure 6). The element "Online updates" is related to a changed element "Integration platform". Therefore, it has the original colour indicating that it is part of GOC' (Figure 7) and it is a border element according the improved GOC' definition.

All changes have been counted using the EAGOC Notebook and listed in table 1. Most of the changes took place in the application layer.

Table 1: All Changes counted by the tool EAGOC Notebook and changes found by the workshop participants P1,...,P5 observing the As-Is and To-Be models without the tool.

	All	P1	P2	P3	P4	P5
Obsolete object	25	6	7	6	7	11
New object	13	2	4	2	4	6
Changed object	13	0	0	0	1	0
Obsolete relation	41	1	0	1	0	0
New relation	26	0	0	0	0	0
Border object	6	0	0	0	1	0
Replaced by relation	12	3	0	3	0	0
Extended by relation	4	0	1	0	1	0

5.2 Workshop to Test the Need of the Tool

In order to test the usefulness of the EAGOC Notebook, a workshop was conducted. Five participants $P1, ..., P5$ took part in the workshop, two enterprise architects and three managers of the Sligro company.

First, the As-Is model (Figure 5)) and the To-Be model (Figure 6) were given to participants. The legend of the models and the colours in ArchiMate were explained. The participants were asked to count obsolete objects, new objects, changed objects, obsolete relations, new relations, border objects, "replaced-by" relations, "extended-by" relations.

Table 1 shows the numbers of the changes found by participants using the As-Is and To-Be architectures. The zero means that some comments were written, but no elements were counted. The participants were able to identify a subset of obsolete objects and new objects. All participants commented that they can easier recognise changes in objects, than in relations.

Second, the As-Is, to-Be and GOC' were given to participants. Most participants were confused and counters got even lower values.

Third, the lists of changed elements generated by the EAGOC Notebook and the actual numbers of changes were given to the participants to recognise the changes. The lists were accepted as useful application of the tool.

6 DISCUSSION

Analytic Pattern. The Gap of Changes (GOC'), presented in this work, is an analytic pattern both for the way of its creation and for the way of its application.

The way of its creation is the analysis of two repositories describing two enterprise architectures As-Is and To-Be.

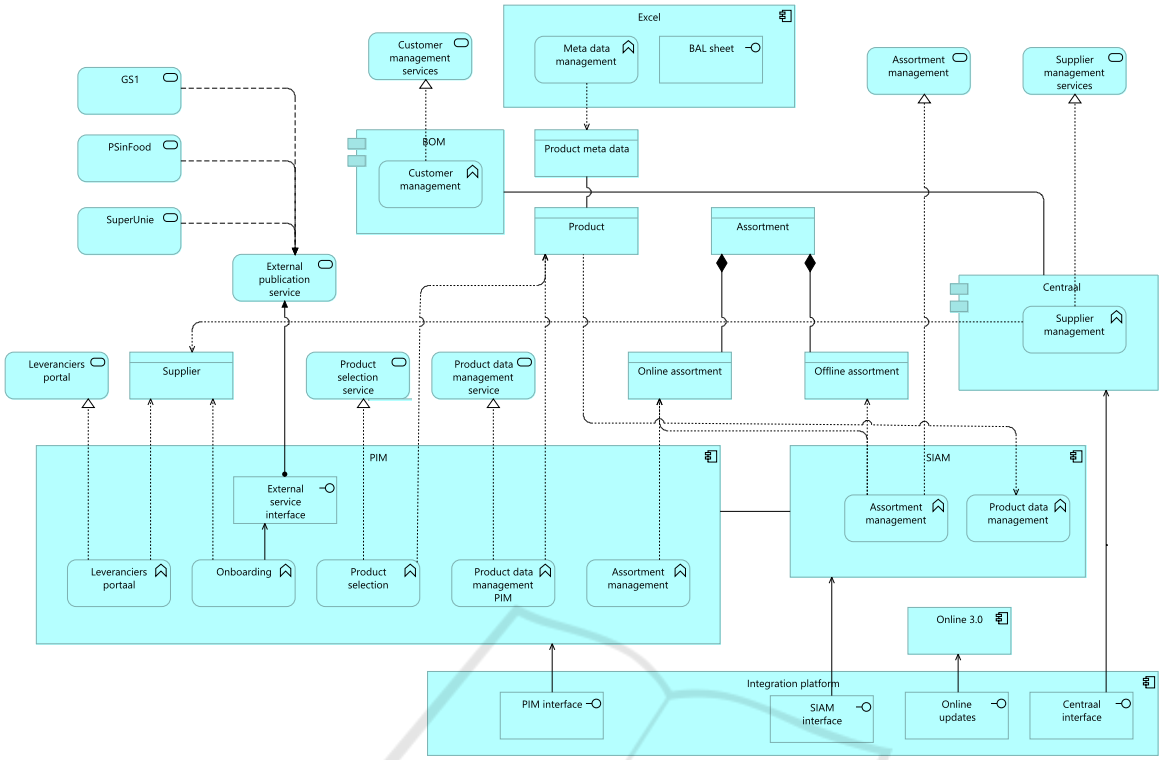

Figure 5: As-Is architecture.

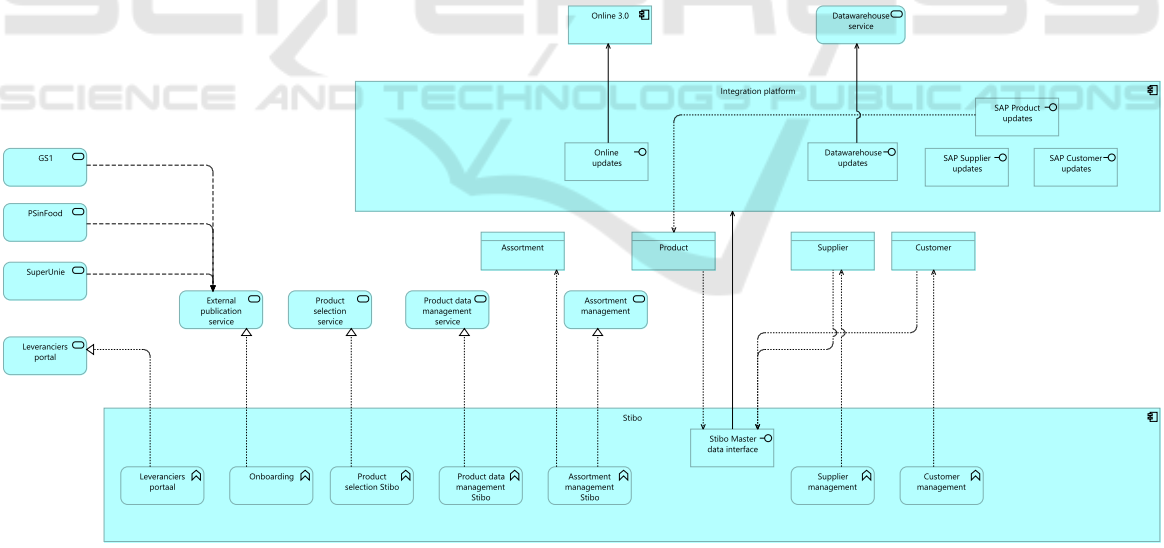

Figure 6: To-Be architecture.

The way of its application is the analysis of elements of changes, selection of subsets of changes based on different criteria. The selection of obsolete objects and relations, changed objects and relations and other is needed for implementation of changes, estimation of work distribution and management. So, the Gap of Changes structure makes the Enterprise models analysable on changes.

Consistency of As-Is and To-Be. The experiments show that the creation of a Gap of Changes from two enterprise architectures contributes to consistency analysis.

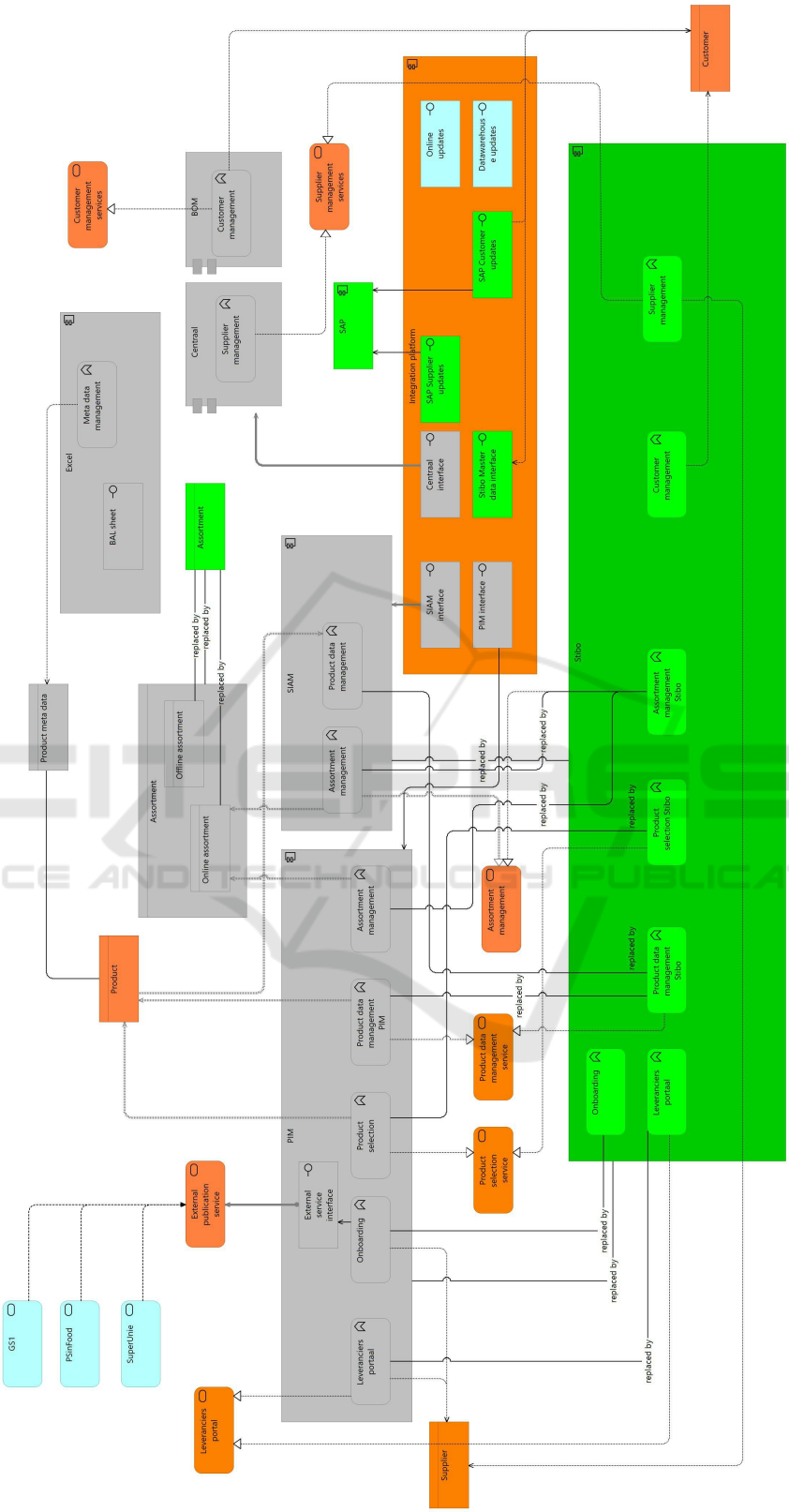

Figure 7: Cap of Changes view.

The Gap of Changes often reveals unexpected differences of enterprise models. These can be caused by undocumented changes inserted by the enterprise architects or hidden design decisions taken on the fly.

So, the EAGOC Notebook, presented in this work, serves not only to find and count changed elements, but also to analyse inconsistencies of As-Is and To-Be enterprise architectures.

In our experiment, the application of the EAGOC Notebook to the repository with As-Is and To-Be has helped to find the isolated objects with no relations and multiple objects presented the same real life object.

7 CONCLUSION

Nowadays, the formal methods experience revival of their application. They become more practical and try to close "the gap between research and practice, and the gap between software development and traditional engineering disciplines" (Parnas, 2010).

The results presented in this work is an application of formal methods. A repeated analytical request about changes in enterprise architecture, has been formalised as a semantically defined structure and named "Gap of Changes" (GOC'). A formal, language independent definition has made it useful for building a tool EAGOC Notebook for analysis of changed objects and relations: obsolete, extended, replaced, etc.

The experimental testing of the tool and the "Gap of Changes" as an analytic pattern has shown its added value in the understanding of changes, forming the common terminology of change expression and countable estimation the volume of changes.

The currently implemented EAGOC Notebook (Dijkstra, 2018) presents the changes as lists of the GOC' elements: changed, obsolete and border objects and different sorts of relations, found useful for the case of large changes.

These lists of GOC elements form the complete repository for automatic visualization of a Gap of Changes. Such a visualization may become a direction for future work.

An additional application of the EAGOC Notebook for revealing hidden design decisions and inconsistencies in the As-Is and To-Be enterprise architectures has been found. Such an additional application of the EAGOC Notebook can be tested with other case studies.

REFERENCES

Archi (2018). Archimate tool. *https://www.archimatetool.com/*.

Bakelaar, R., Roubtsova, E., and Joosten, S. (2016). A framework for visualization of changes of enterprise architecture. In *International Symposium on Business Modeling and Software Design*, pages 140–160. Springer Berlin Heidelberg.

Bizzdesign (2019). Enterprise-architecture-software-tools. *https://bizzdesign.com/products/enterprise-studio*.

Diefenthaler, P. and Bauer, B. (2013). Gap analysis in enterprise architecture using semantic web technologies. In *ICEIS (3)*, pages 211–220.

Dietz, J. L. and Hoogervorst, J. A. (2008). Enterprise ontology in enterprise engineering. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 572–579. ACM.

Dijkstra, R. (2018). Enterprise architecture. gap of changes. *https://github.com/RichDijk/EAGOC*.

Hoffman, L. R. and Maier, N. R. (1967). Valence in the adoption of solutions by problem-solving groups: Ii. quality and acceptance as goals of leaders and members. *Journal of Personality and Social Psychology*, 6(2):175.

Jupyter (2018). Jupyter Notebook. *http://jupyter.org/*.

Pandas.DataFrame (2018). Pandas DataFrame. *https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html*.

Parnas, D. L. (2010). Really rethinking "formal methods". *Computer*, 43(1):28–34.

Python (2018). Python Software Foundation (US). *https://www.python.org/*.

Sandkuhl, K., Stirna, J., Persson, A., and Wißotzki, M. (2014). Enterprise modeling. *Tackling Business Challenges with the 4EM Method. Springer*, 309.

Sligro (2018). *Sligro food group, https://www.sligrofoodgroup.nl*.

Sparxsystems (2019). Sparx systems enterprise architect. *https://www.sparxsystems.eu/enterprisearchitect*.

StiboSystems (2018). Stibosystems. *https://www.stibosystems.com/*.

Sunkle, S., Kulkarni, V., and Roychoudhury, S. (2013). Analyzable enterprise models using ontology. In *CAiSE Forum*, volume 998, pages 33–40.

The Open Group (2018). *ArchiMate 3.0 Specification. http:/pubs.opengroup.org/architecture/archimate3-doc/*.