Journal of Object Technology
Published by AITO — Association Internationale pour les Technologies Objets
http://www.jot.fm/

Evaluating the capabilities of Enterprise Architecture modeling tools for Visual Analysis

David Naranjo[a] Mario Sánchez[a] Jorge Villalobos[a]

a. Department of Systems and Computing Engineering, Universidad de los Andes, Colombia
http://ticsw.uniandes.edu.co/

Abstract In model analysis activities, it is critical to make early statements and diagnosis from a high level of abstraction. Currently, these tasks are difficult to perform, and they require both the involvement of experts and the elaboration of specialized artifacts. Furthermore, the complexity of the tasks increases as models become bigger and more detailed. In other contexts, it has been noticed that total / holistic / unfiltered visualizations may give insight about the models, providing analysts a starting point for exploration and general pattern discovery.

In this paper, we evaluate the support that six different Enterprise Architecture (EA) modeling tools offer to EA analysis activities, and assess the strengths and weaknesses of six visualization frameworks, in order to extend the analysis of enterprise models by Visual Analysis. The evaluation is based on a set of 14 requirements which are either visualization-related or specific to EA analysis, and its results were harvested from a) observed characteristics of the diagrams of these tools, and b) visualizations from an enterprise model, generated with the aforementioned visualization frameworks. These results point to several actionable subjects and research opportunities for the field of EA Modeling and Analysis.

Keywords enterprise architecture, analysis, visualization, tool evaluation

1 Introduction

Evolving markets, increasing IT adoption and augmenting complexity have drawn a growing interest in disciplines such as Enterprise Architecture (EA), where extracting new information out of enterprise models is often a difficult but relevant task. From a Business Analyst perspective, this means exploring the models and formulating relevant questions.

David Naranjo, Mario Sánchez, Jorge Villalobos. *Evaluating the capabilities of Enterprise Architecture modeling tools for Visual Analysis.* Licensed under Attribution-NoDerivatives 4.0 International (CC BY-ND 4.0). In Journal of Object Technology, vol. 14, no. 1, 2015, pages 3:1–32.
doi:10.5381/jot.2015.14.1.a3

One critical issue is the lack of consensus on what should be the starting point of these analyses. Their real value lies in their outcomes, which are the basis for strategic decisions that affect the whole company, but they require the analyst to ask the right questions. Analyses over EA models thus need to be flexible and scalable [KAPS10], because they are usually formulated in an *ad-hoc* manner, and are often based on hypotheses and scenarios that involve 'what if...' questions.

This blind spot in the analysis process creates a great challenge for the analyst, who has to acquire specific knowledge and experience to know where to look. The alternative is missing key information, or reaching false statements about the architecture. For instance, a situation would arise where there isn't an *a priori* knowledge of which services are the most used, or even their relation to the business processes of the enterprise. Acquiring knowledge of this type sometimes needs the elaboration of specialized artifacts (e.g., service and process catalogs, matrices and views) or the introduction of complementary methodologies, such as SOA service discovery [Bel09], that despite their usefulness and pertinence, add unwanted complexity to the task of finding overall patterns or draw new conclusions of the model.

We can divide approaches to EA model analysis in three complementary categories: queries, views and visualizations. *Queries* are questions expressed in a formal language. They often require previous knowledge of the concepts involved, and their complexity is usually proportional to the value of the outcome. *Views* are a specialized form of queries (i.e., a selection of model elements), and offer a partial glimpse of the model. However, views lack a sense of continuum, missing 'the big picture' as they cover specific stakeholder concerns. While most EA frameworks offer abstraction mechanisms such as layers and viewpoints in order to reduce the number of artifacts per model [BFKW06], it is difficult to see all the concepts in a holistic way [BW05], as legibility of a view decreases when it covers a wider range of elements.

Visualizations, seen as the visual representation of views, augmented with analytical facts, focus on delivering key information through a visual language and visual metaphors. They can inspire new questions and further exploration, and they can help identifying sub-problems, trends and outliers [IS11]. These trends, seen as visual patterns or **motifs**, are typical configurations of visual constructs. They represent critical/important elements (or groups thereof), and structural anomalies from the whole model, which may correspond to information patterns in the head of the analyst. Thus, images are of significant cognitive importance: they accelerate pattern recognition while having an almost unlimited capacity of communication by making use of several visual attributes that encode information or emphasize certain message.

Despite the benefits that it brings, visualization of models does not scale well: comprehensibility and communicability of a model deteriorates rapidly as complexity and size increases [CHP96]. Additionally, the manual creation of visualizations is an error prone and time consuming task [BEL$^+$07]. Under the need of mechanisms to support the coordination of business and IT on various levels of abstraction [BW05], existing tools offer only a view-oriented perspective of EA modeling, and their analysis capabilities are often based on report generation. Interaction (e.g. changing the focus of interest) and navigability (exploration of the model), two key elements in cognitive integration, are often neglected by these tools. We argue that in order to link partial views of a model and perform model-wide analysis, **bird-eye views** or 'long-shot' diagrams that comprise all elements and their relations are needed. These views act as overall cognitive maps into which information from individual views can be assembled [Moo09].

The issues described above can be summarized in the following question: *Is it possible to offer high-level, effective and all-encompassing visual representations of enterprise models that experts can use to make analysis and discovery of patterns and anomalies?* With this in mind, the objectives of this paper are: 1) to introduce the concept of Overview Visualizations and describe their advantages in EA model Analysis; 2) to expose a possible limitation in the support that current EA tools offer for model analysis; 3) to introduce, describe, and justify a list of 14 requirements that may be essential to an enterprise architect in order to overcome this limitation ; and 4) to apply these requirements to six popular EA management tools and six visualization tools, offering a comparative evaluation that assesses the strengths of each tool and gives space for future research on the key areas defined by the requirements.

This study examines the focus that some EA tools give to visualization, providing some hints on the direction that these tools point out and their ability to eventually support visual analysis. Given the maturity that these tools have achieved in the aspect of offered features for modeling [ELSW06], we seek to depict the current maturity level that these tools altogether have achieved in terms of model exploration and analysis. In this aspect we propose a set of requirements that arose when we were exploring the 'big picture' concept. However, the objectives of this study do not include offering a judgement of which tool is better than other. Instead, we see this study as an exploratory analysis of EA visualization, and we think each architect should carefully consider which tool serves better his particular purposes. Such an assessment is not complete if other (non-visual) requirements are not taken into account. For instance, price is a feature that most architects consider carefully when selecting an EA tool.

The structure of this paper is as follows: Section 2 will describe similar approaches, also commenting on the main issues that we have encountered when trying to represent large EA models. Section 3 will introduce Overview Visualizations, and comment on how we can achieve effective information visualization through the use of the visual semantics of these models. In Section 4 we will describe the requirements of a visualization tool for model analysis, from the visual and EA perspectives. Section 5 is a summary of our case study, with a description of our evaluation methodology, followed by Section 6, where we offer the results of the evaluation, and under this light we will comment on EA and Visualization tool support. Finally, Section 7 will be a space for discussion of the results, and point to research opportunities on this subject.

2 Related Work

Large model visualization is a problem that has been tackled from different disciplines: Database schemas in Information Engineering [FM86], library dependencies in Software Engineering [FR07], or even in ontologies of big domains [GDDS06, KHL+07]. This issue can be summarized with the *Database Comprehension Problem:* "Usefulness of any diagram is inversely proportional to the size of the model depicted" [FM86].

How to obtain cognitively manageable representations of large models is a known and mostly unresolved issue. Nevertheless, we will introduce three kinds of approaches, the first two pointed by Tzitzikas [TH06]: Visual Approaches (Section 2.1) such as graph layout optimization, and Semantic Approaches, e.g., model filtering (Section 2.2). We also include another kind, Analytical Approaches (see Section 2.3).

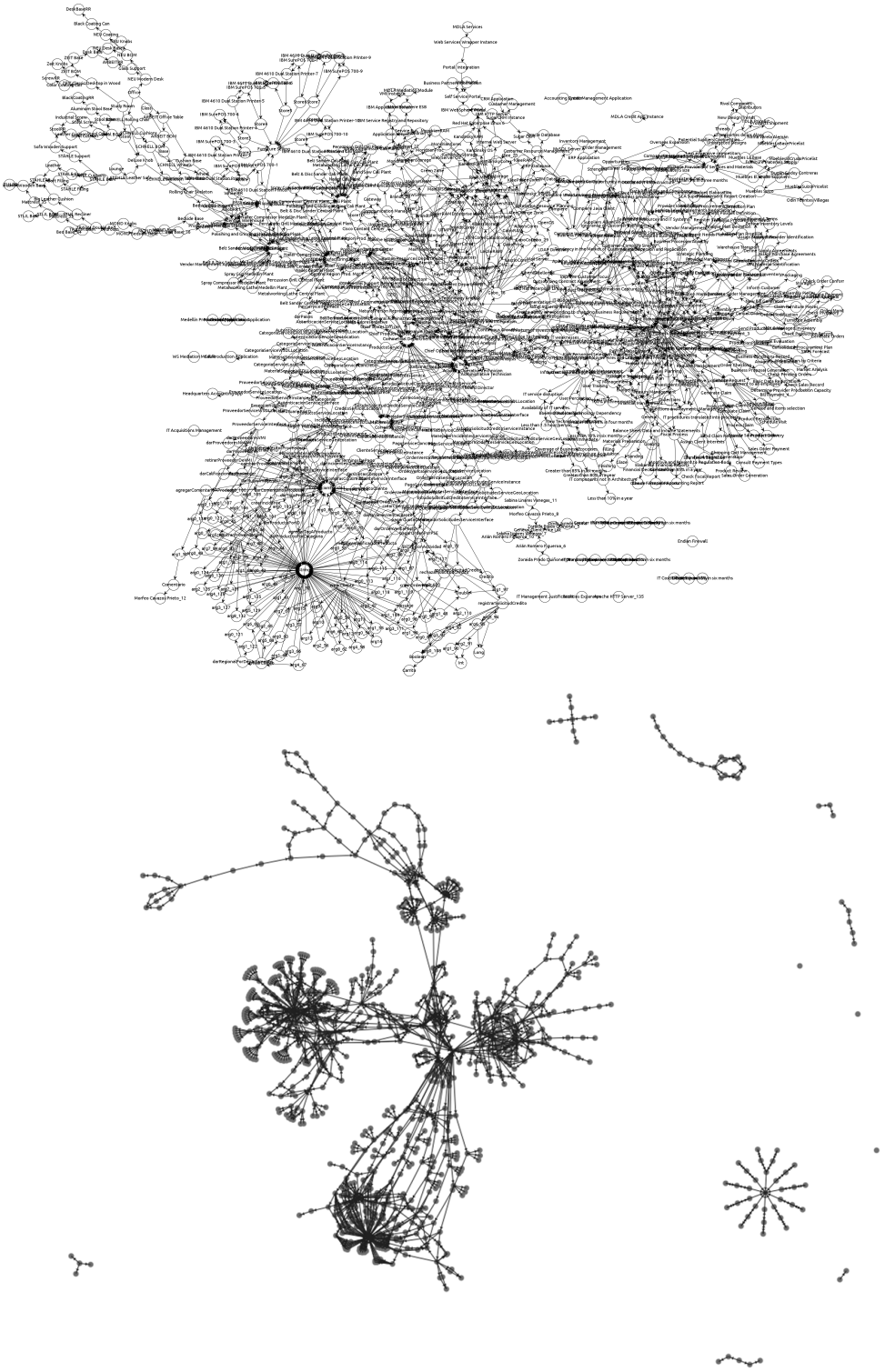

Figure 1 – Force-Directed layout applied to the model of our case study using GraphViz (top) and GraphStream (bottom). The display of all labels hinders readability, but showing just elements and relations alone implies a loss of semantic correspondence with the model.

2.1 Visual Approaches

Typical visual representations of models fall short as these models get bigger; it is common to find diagrams that extend themselves through very large sheets of paper, in some cases covering entire walls. Visual representation of these models is not an easy task, as an increase of visual elements is detrimental to the human's cognitive load as well as to the system's response time [KY93]. For instance, most CASE (Computer Aided Software Engineering) tools require arrangements 'by hand' [TH05], which means that a great amount of time is spent dragging elements to form a decipherable diagram.

Graph Layouts

Graph layouts are a method for dealing with situations with high information density by structurally arranging elements based on graph properties. There are several techniques for optimizing node placement, such as Force-Directed Layouts (FDLs), based on a spring model of attraction and repulsion, with several implementations such as the Fruchterman-Reingold algorithm [FR91], and enhancements made by several authors [LY05, Hu05, EHH10]. Most visualization tools use FDLs (see Fig. 1) because of their flexibility and tendency to be aesthetically pleasing, exhibiting symmetries and producing crossing-free layouts for planar graphs [Tam10]. Some modeling environments, such as KIELER[SSvH14] and Enterprise Architect[Spa] take advantage of the different FDL algorithms to manage the complexity of large diagrams.

Another common technique is using Circular Layouts. These algorithms place nodes onto a circumference, generally with their edges across the embedding circle, with the benefit of encoding information in different layers of the same representation. The tricky part is to minimize edge crossings, which is a hard problem [Tam10]. However, if we organize elements under some domain specific criteria, we can obtain a more clear representation (see Fig. 2). For instance, we can assert that Enterprise Models have groups of common elements/clusters. If we order elements in the circumference by groups, the inner part of the circle will display inter-group relations, thus reducing edge crossings.

Other interesting approaches that have been given little attention involve projections into non-Euclidean coordinates, such as Hyperbolic ($\mathbb{H}^2$) and Spherical ($\mathbb{S}^2$) spaces. However, implementation of these algorithms have some limitations, such as their restriction to quasi-hierarchical graphs[Mun00], or even their handling of very large graphs [KW05].

2.2 Semantic Approaches

Model Filtering and Clustering

Visual arrangement of model elements is not the only way at hand for dealing with model complexity. Another recurring problem with large models is that all object types are considered to be of equal importance. That means that visually, these models are seen as flat conceptual schemas [CHP96]. This is not always the desired result, as on a high level of abstraction an analyst would like to see that key structural concepts are given more visual importance than secondary or supportive concepts.

We usually decompose a complex problem space in layers (or levels) of abstraction, in order to depict different views of the same problem. From a semantic point of view, it is possible to create a hierarchy of views that cover the whole model. Approaches like [FM86][CHP96][Moo97][Egy02] propose mechanisms that allow designers to 'zoom out'

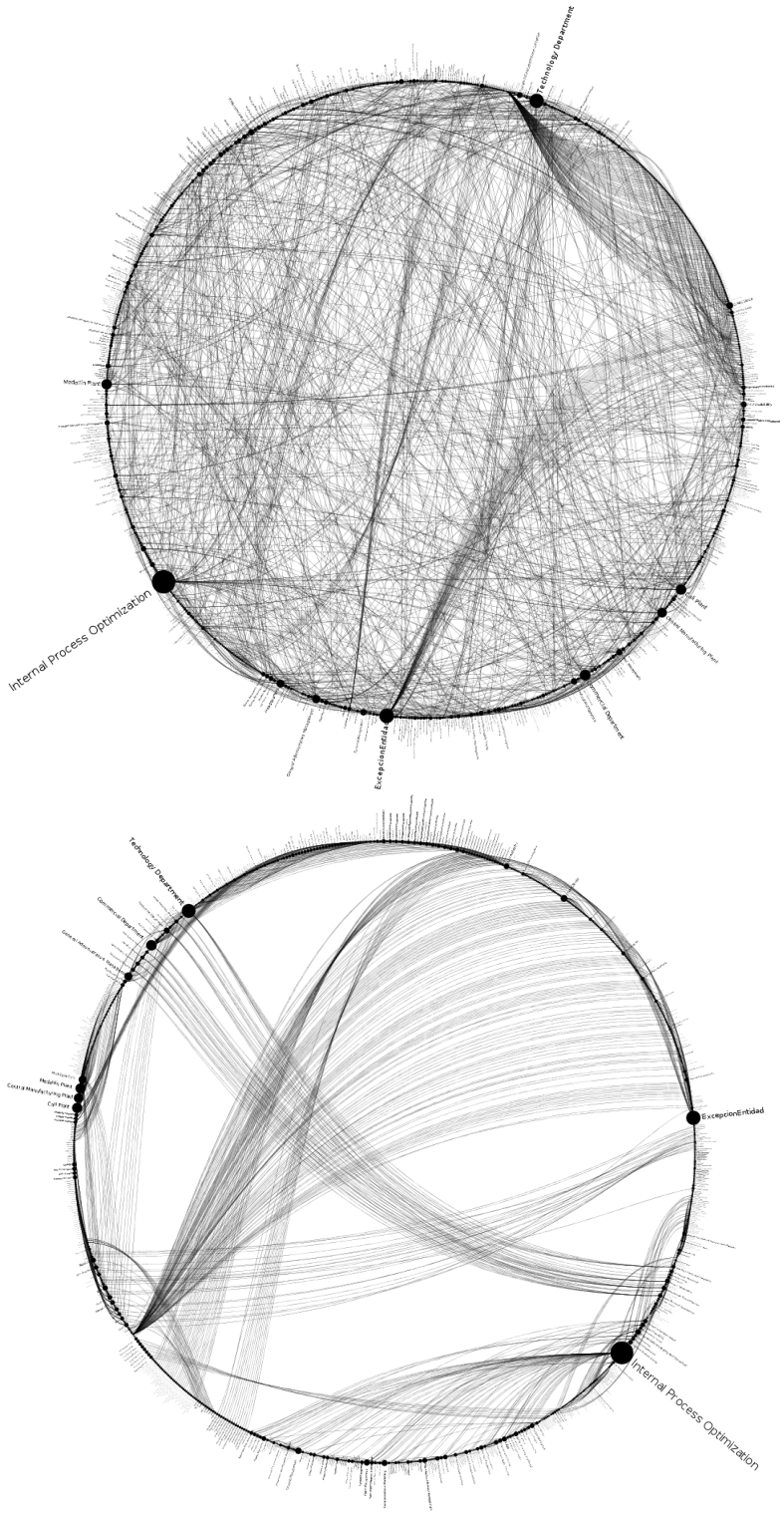

Figure 2 – Circular layout made with gephi[BHJ09] and prefuse[HCL05], two visualization tools. Using different layouts and arranging the relative position of elements can reduce visual complexity and improve understanding.

on class diagrams, allowing them to view a model on a higher level of abstraction. This is achieved by finding key concepts of semantic importance that keep the underlying network connected, in combination with traditional abstraction techniques, such as generalization, aggregation, and association. Recent approaches like [VO10] rely on capturing the information needs from an user by the definition of an interest function that filters model elements in relation to their relative closeness and global importance.

However, it is difficult to achieve *precise* automatic abstraction. This means that we have to deal with information loss (e.g., unwanted filtering of relations or elements) and make heavy use of abstraction rules that certainly are not easily created and are specific to the metamodel (e.g., an abstraction for a UML model is different from an abstraction in BPMN or ArchiMate models). For this reason, generic filtering and clustering techniques are used instead, but it has been shown that they require additional user input and manual intervention or correction [Egy02].

2.3 Analytical approaches

Automatic layout algorithms alone are not sufficient for the analysis of large models, and current filtering techniques are not satisfactory as well. In addition, classical hierarchical decomposition techniques that are used for visualizing large plain graphs are rarely applied or tested on conceptual diagrams [TH06].

Taking into account the benefits and limitations of both approaches, Tzizikas and Hainaut [TH05] propose a way to 'tame' large conceptual diagrams using a ranking algorithm that filters elements, finally displaying them with a FDL algorithm. Chan et al. [CKM10] offer a bottom-up approach for interactive visual analysis, where the user starts with a localized view of the model, which is constantly filtered through navigation of the graph, keeping a window of interest through the relations of the visible elements, which are displayed also by a FDL. An important restriction of their approach is that the model must be hierarchical.

In other application fields, such as Software Architecture, we can find approaches that make use of more sophisticated visual metaphors, such as Voronoi Treemaps [BDL05] or 3D city maps [WL07, Soa07, AD07], and offer flexible analysis methods and multiple visual metaphors (e.g. SourceMiner [NdFCSJM12]). For instance, see the work of Hipp and Reichert [HMR11, Rei12, HMMR13] in Business Process Visualization, and the tool ..cantor.dust.. [Dom12] for the analysis of program binaries.

Concerning Enterprise Architecture, Ernst et al. [ELSW06] offer an extensive evaluation of tool support, recognizing the importance of visualization in analysis and EA management tasks. Buckl et al. [BEL+07] build visualizations using model transformations, which they use to map semantic elements to their graphical representation on a layered cartography metaphor, a methodological aid which greatly facilitates analysis tasks.

3 Towards effective EA overview visualizations

Our approach for exploring and visualizing EA models, which will be described in this section, can be considered an analytic approach, and is based on the hypothesis that Enterprise Models, seen as complex networks, have some topological properties on their own that differentiate them from simple conceptual models or random graphs (see [NSV13]):

- These models grow in a complex fashion, i.e. new elements and relations are not introduced randomly.
- Enterprise Models are structured and have first-order clusters that represent layers/domains.
- Based on the previous property, their structure is semi-hierarchical, and each domain/layer can be connected to another by inter-domain relations.
- Relations are often given more importance that elements themselves.
- Finally, there are elements that are structurally more important, as they keep the network connected.

While there can be models in other application domains that may share these properties, we consider that the problem of visually analyzing this kind of models is worth the attention, so we will explore this problem from our experience with Enterprise Architecture.

3.1 Overview visualizations

EA models are usually layered (e.g. business layer, information layer, applications layer, technology layer). Guaranteeing that separation of concerns, while providing traceability among those layers, seems to be one of the biggest challenges in modeling enterprise architectures. That challenge becomes a critical issue for large, complex models.

For this reason, when we are modeling and analyzing Enterprise Models, we usually cover one or more stakeholder concerns by selecting suitable model elements and bundling this selection as a viewpoint [KAPS10]. Views that conform to these viewpoints describe parts of the model within a given scope. This makes sense when we need to develop representations that are understandable by both business and technical experts [Moo09], or in general, when we need to provide a medium for communication between people with different professional backgrounds [Fra02]. This often requires using a strategy for the design of these views. For instance, the ArchiMate [Obj14a] language suggests a set of Viewpoints that have different levels of detail (Content dimension).

These levels of detail ascertain three different needs: 1) To provide an *Overview* that covers multiple layers and aspects, allowing the use of all concepts and relations of the language, e.g. with Layered and Landscape Map Viewpoints. 2) To give *Detail* with Viewpoints that focus on a concrete layer and one aspect of the model that the architect needs to point out, e.g. with Actor Co-operation or Application Structure Viewpoints, and 3) To provide *Coherence* by focusing on relations between the different layers, e.g. with Organization or Product Viewpoints.

While localized/filtered views (e.g. Detail and Coherence in the case of ArchiMate) are certainly useful as they allow detailed examination of an Enterprise Model, we will focus on *overview visualizations*, which we consider are decisive for overall pattern discovery. These total views bring useful information that is drawn from the overall relationships of the entire [data] set [Ber81], uncovering emergent patterns that are visible only when all elements/relations are displayed.

These patterns, or *Motifs* [PH13], are typical configurations of visual constructs, and are part of the topology of the model. They are useful for structural analysis of a model,

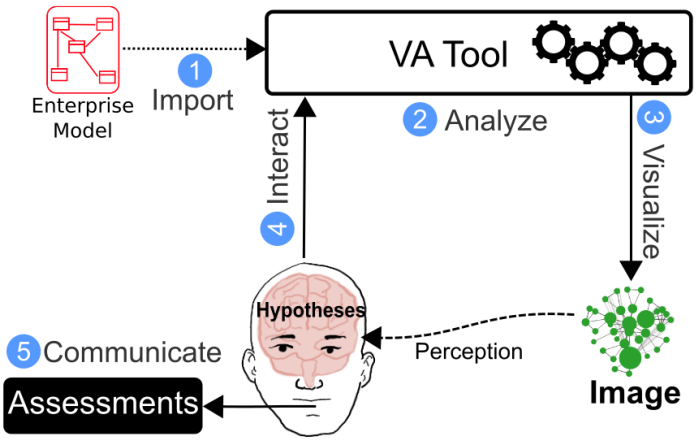

Figure 3 – The EA Visual Analysis process.

as they are easily distinguishable by an expert, and may have a correspondence with semantic patterns in his experience (see Section 4.2.4). Also, Overview visualizations can be a valuable tool for inspecting the interconnection between the domains/layers of the model (see Section 4.2.1.

3.2 A Process for EA Visual Analysis

In Section 2 we identified two key concerns that surface in the visualization of large models: a) the use of algorithms for the automatic placement of elements of the model to minimize visual complexity, and b) the need of abstraction mechanisms that reduce information overload. However, a generally overlooked issue is to maximize the effectiveness of these visualizations, that means, to provide an overview of the model that is expressive enough to support the tasks of an analyst.

This implies using the whole spectrum of the visual language to communicate properties of the model. Bertin [Ber83] described a set of seven visual variables (position, shape, color, size, texture, value [opacity] and orientation), which are modifications to 'marks' on a visual language. A 'mark' can be seen as an atomic unit of notation -a sign- that represents (or encodes) information. Visual mapping is the process of translating semantic information into visual constructs that encode this information [NSV13].

In order to increase the expressiveness of EA model visualizations, our strategy is to take advantage of this mapping. Inspired by the Visual Analysis process delineated by van Wijk [vW05], we illustrate the EA Visual Analysis process (see Fig. 3) as an iterative process between an Analyst and a Visual Analysis System, where hypotheses are generated and refined by interacting with visualizations.

This process begins with an initial **Import** of the Enterprise Model. This model can be **Analyzed**, i.e. processed under a series of functions that operate in terms of its structure, adding new information based on existing knowledge.

After this processing, the analyst will be able to **Visualize** the model structure with several Visualization Techniques. We use these visual representations as a memory aid to amplify cognition - that means, we transform data into images to derive insights, using pattern recognition from the human visual system to process visual information.

ID	Visual Requirement
MVE	Maximize Visual Economy
EVE	Enhance Visual Expressiveness
MN	Minimize the Noise
NI	Navigate and Interact
KC	Keep the Context
DNI	Derive New Insights
GSC	Guarantee Semantic Correspondence

ID	EA Analysis Requirement
IRD	Identify and Relate Domains
EKE	Emphasize Key Elements
OFI	Offer a Focus of Interest
FSD	Facilitate Structural Diagnosis
DSC	Display Semantic Characteristics
UAQ	Uncover Architectural Qualities
PFM	Provide a Flexible Metamodel

Table 1 – Visual and Enterprise Architecture Requirements

As the analyst starts to **Interact** with visualizations of the model, Hypotheses (which start as expectations, i.e. weak formulations) get refined over time. This interaction modifies the parameters of a visualization.

Within each iteration, these formulations are confirmed or denied, as the analyst starts to associate visual patterns with EA patterns that are present from knowledge and experience. Finally, when the Analyst has acquired sound insights on the model, he is able to **Communicate** results from the analysis in terms of Assessments of the architecture.

Achieving effective visualizations is rarely a straightforward process. We can find a large body of work on taxonomies such as the Data State Reference Model [Chi00], or visualization frameworks and patterns for choosing and using the appropriate data visualization techniques –see [Shn96, ZF98, MLO00, BE02, AS05, Epp06, Mun09, HS12]–. However, from our experience, we consider that the structural analysis of EA (or similar) models has a set of requirements on its own (see Section 2.3).

4 Requirements for an unfiltered view of large EA models

With the issues mentioned above, we can say that there is no single recipe for dealing with high information density and complexity in EA models. A combination of different techniques is required, in addition to a broader use of the visual language, in order to facilitate understanding and allowing the user to explore the 'big picture' of the architecture.

With the assistance of experts and fellow architects, we defined 14 criteria that surfaced in the process of exploring the concept of this 'bird-eye' or 'big picture' overview in the context of EA. These criteria were separated in two categories. *Visual Requirements* and *EA Analysis Requirements*. The format is the same for all requirements; they have an identifier (see Table 1), a name and a description.

These requirements are not an exhaustive list of the elements to take into account both in Data Visualization and the Enterprise Architecture domains. Instead, we offer an extensible evaluation that hopefully will bring additional criteria in the design or selection of an EA management and/or a visualization tool, in order to fill the current analysis gap.

We take various elements from the work of Moody [Moo09] on Visual Notations and the work of Gallagher et al. [GHM08] on Software Architecture Visualization, supporting ourselves with some other authors when necessary.

4.1 Visual Requirements

This first set of requirements deal with general visual principles that revolve around large model representation. As such, they are not described in terms of the context they are used, the tool employed, or the visual metaphor selected, provided that we can transform the Enterprise Model and serve it as input to any visualization toolkit, modeling environment, or any other specialized application.

Instead, these principles are common strategies that can be applied to *model visualization*, seeing these models as structured information that have some common properties.

4.1.1 **Maximize Visual Economy (MVE)**

Visual modeling languages, like UML or BPMN, have a concrete syntax that encodes specific concepts of their metamodel into symbols. This mapping gives *semiotic clarity* to the notation [Moo09]. However, this one to one (or even one to many) correspondence also results in complex visualizations that are difficult to decode due to the large amount of symbols involved.

Applying visual economy to a total view is essential, and can be done by using a minimal (but coherent) amount of visual constructs to represent the models. As the enterprise metamodel may have a large number of concepts, an unfiltered view of the model requires symbol overload -multiple concepts represented by the same symbol- in order to avoid additional decoding by the user. This brainpower can be used instead for extracting other kind of insights from the model.

4.1.2 **Enhance Visual Expressiveness (EVE)**

We can relate expressiveness of a visualization to the number of channels that it uses to communicate. This requirement refers to a proper use of the visual vocabulary supported by the mapping of model elements, relations and their properties to values of available visual variables (see Section 3.1).

However, in order to emphasize certain messages, reduce errors and counteract noise, a visual notation should also use redundant coding [Moo09], which means that more than one variable should be used to express some feature. It is important to maintain a balance in the use of these variables, as their indiscriminate use may be also detrimental to the expressiveness of a visualization.

4.1.3 **Minimize the Noise (MN)**

The designer of a visualization should avoid including information that is not relevant to the message he seeks to transmit [IS11]. Noise can be inserted in the encoding and decoding processes of a message, and it represents unwanted information that accidentally appears, causing a random variation in the visual variables and distorting the intended message [Moo09].

Multiple edge crossings (see Fig. 2) and overlapped symbols are a form of noise, as they make difficult the process of decoding visual information. Other forms of noise can be additional elements such as watermarks, unrelated labels, and comments that are not relevant to the level of abstraction at hand. For instance, several tools allow the introduction of arbitrary visual elements without defined semantics in the context of the visualization [BEL+07].

4.1.4 Navigate and Interact (NI)

Inspecting certain properties of a model and traveling through different levels of abstraction is essential to analysis, as it is seldom a static process. It is desirable that the user could make use of common user navigation techniques such as panning, zooming, bookmarking, and rotating in both 2D and 3D environments [GHM08], and when necessary, modify the arrangement of certain elements easily. Also, animation of layouts and the use of projections into different coordinate systems can be helpful in the exploration of the model.

4.1.5 Keep the Context (KC)

The user will lose context when there is too much distance between elements. Scrolling endlessly through a diagram is of no practical use [KY93], so we need mechanisms for encoding visually large models without losing 'the big picture'. Focus-plus-Context [Mun00] views, as well as other techniques that involve self-organizing layouts can be used to keep the context.

4.1.6 Derive New Insights (DNI)

The power of models lies in their capacity of offering new insights, based on defined facts. This can be done by two complementary approaches: The first one is to directly ask questions in terms of elements and characteristics of the model, by describing dynamic -*ad hoc*- queries about elements and their attributes, and finally displaying the results of these queries in a visual form.

On the other hand, the pre-processing of models also allow new insight derivation, by the means of general analysis functions that calculate graph properties (e.g. centrality, eccentricity, paths, breadth trees, weights, depths), as well as domain-specific functions that summarize and rank elements of the model in terms of its metamodel.

4.1.7 Guarantee Semantic Correspondence (GSC)

As model visualizations get more complex, the user may need tools that offer a mechanism for remembering the correspondence between the graphic and conceptual domains. For instance, cartographers often include legends that help readers on decoding visual variables and symbols into meaning [Ber83].

Moreover, the visual metaphors and tools that we use in the field mediate with this correspondence, making more difficult (or less) to travel all the way back from visual meaning to the semantic model. In the case of editors for Visual Modeling Languages, they offer a good semantic correspondence, as they also offer a set of symbols that can help the user in this visual-semantic mapping. However, these kind of model views are rarely economic or expressive enough for overview analysis of large models (see Sections 4.1.1 and 4.1.2).

Semantic Correspondence can be achieved through familiarity, i.e. when the structure of the visualization resembles artifacts/diagrams of the domain of the stakeholder. However, as this is not always possible, visualizations should include legends, or any other mechanism that relates visual variables to concepts.

4.2 Enterprise Architecture Analysis Requirements

These requirements make use of Visual Requirements (see Fig. 4), and represent a (wish)list of features that an architect/analyst would like to have in his toolbox,

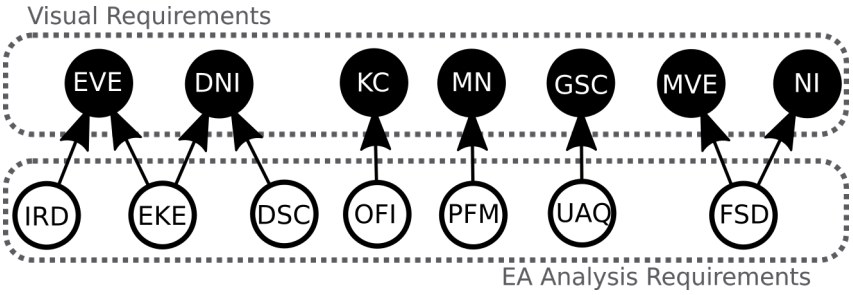

Figure 4 – Dependencies between requirements.

allowing him to diagnose, explore, and provide assessments of the EA model in terms of interesting elements and groups, outliers, and visual patterns.

4.2.1 Identify and Relate Domains (IRD)

Dependencies: EVE.

To facilitate analysis, we are accustomed to separate and classify a collection of elements into smaller groups. Classical EA frameworks are usually divided into three or four groups, namely Business, Applications, Technology and Information layers. Nevertheless, these layers often are incomplete, inconsistent or not quite rigorous [BW05].

Similar to these layers, we can identify smaller groups (**domains**) in our model, which are logical abstraction elements that group common concepts, allowing to separate the universe of discourse into more manageable partitions, which are interconnected and together form the EA metamodel.

These domains can be a part of the metamodel, or they can be discovered after we generate the visualization of the model. For instance, we can perform a segmentation of the elements of the metamodel, in order to identify these domains, and then apply a force-directed layout to each cluster (See Fig. 5).

Visual differentiation between these domains is necessary, as the real value of most EA metamodels lies in the relations between these domains, thus providing a bridge between the different layers and provide alignment to the architecture. Also, this division is the first level of abstraction that we can take advantage of.

4.2.2 Emphasize Key Elements (EKE)

Dependencies: EVE, DNI.

In Visual Analysis, we can also discover critical elements that have high structural or semantic importance. It would be valuable that these *Key Elements* had a visual emphasis or distinction in size, brightness, or any ordinal visual variable. Two complementary approaches can serve us on this task:

Metamodel Approach: Architects define concepts that have the highest semantic importance in the metamodel (e.g. Service, Process, Strategy, or Organization Unit). These Key Concepts may appear in different EA frameworks and metamodels, and may have more significance than supportive concepts such as Activity, Primitive Data Type, or Sales Order. In general, these pivotal elements can be: 1) extracted automatically from the metamodel, as they extend, aggregate, or have several relations with smaller (children) concepts, or 2) have a given weight assigned by an expert.

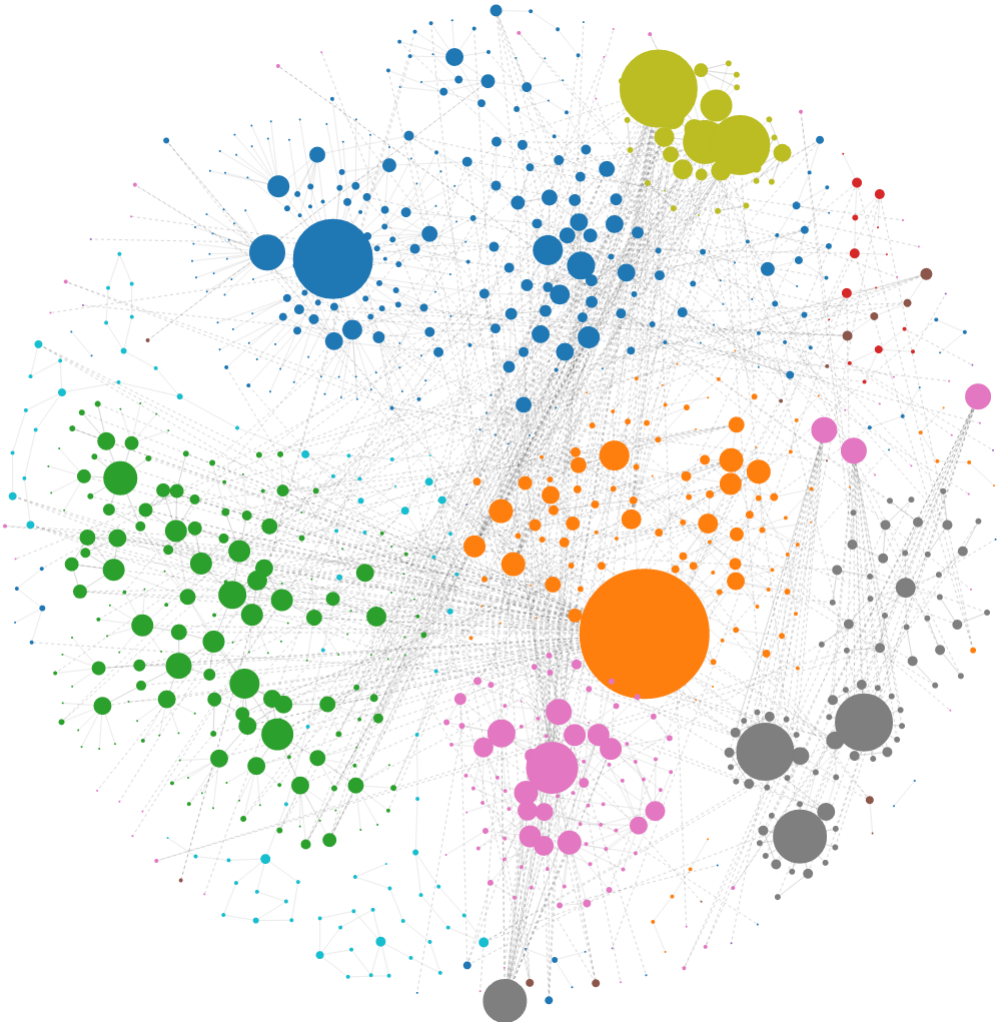

Figure 5 – Architectural Domains - Each cluster represents a domain, organized with a Force Directed Layout and with dashed lines for inter-domain relations. This customized FDL visualization was made with d3.js [BOH11].

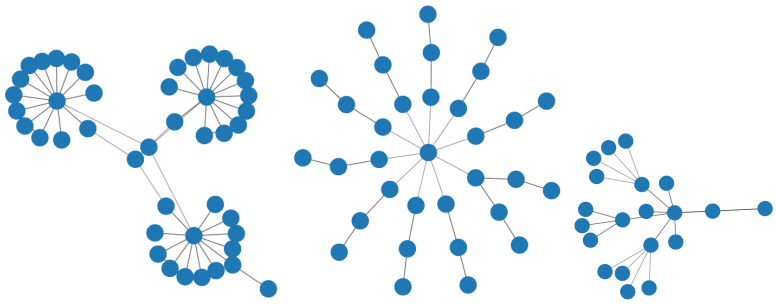

Figure 6 – Structural Diagnosis - Visual patterns found in the visualization can lead to structural assessments of the model.

Model Approach: We can view the model as a network and identify the nodes that preserve its structure, i.e. that keep the network connected. These nodes can be discovered by calculating the criticality of an element by using graph ranking functions, which assign a score to each node based on its relations with other nodes.

4.2.3 Offer a Focus of Interest (OFI)

Dependencies: KC.

In order to define a focus/scope for the analysis, the analyst should be able to create groups of elements for further examination. As it happens with the *depth of field* in photography, this person would want to give more detail to interesting elements, while less important elements are also present but unfocused.

Analyses of an architecture can handle different levels of detail, which are dependent of their scope, and may involve concepts found in different viewpoints (see Section 3). Thus, it is important to navigate these levels of abstraction, and move between different groups of elements in order to get more insights.

Taking into account that we are interested in unfiltered overview visualizations, we can provide a focus of interest by using visual constructs that denote similarity to represent semantically similar elements. For instance, we could use spatial proximity (i.e., position) to represent common elements, or use color shades (opacity) to highlight relevant groups of elements.

4.2.4 Facilitate Structural Diagnosis (FSD)

Dependencies: MVE, NI.

Diagnostics are assessments that can be made by identifying recurring patterns in the model. Experts in a domain can identify them easily, as these patterns usually come from experience. For instance, physicians are able to diagnose a disease or ailment with a quick glimpse to a specialized artifact, e.g. a MRI scan or a clinical report of a patient.

In Visual Analysis, these information patterns are translated into visual patterns, typical configurations of visual constructs (see Section 3.1). For instance, the patterns of Fig. 6 are sub-structures found in the model, and represent topologies of the Organization Structure of a company. This is useful when the analyst wants to identify critical elements, structural anomalies and outliers from the whole model.

4.2.5 Display Semantic Characteristics (DSC)

Dependencies: DNI.

As described in Section 4.1.6, an analyst would appreciate the inclusion of mechanisms where he can extract information that is not directly available from the model. We can enrich visualizations with additional information (such as query results), and metadata (e.g. annotations), in order to display semantic properties of the model. In the context of EA, we can find several application scenarios:

Impact Analysis: Discovery of potential changes in the architecture can be made with a visualization of the whole model, supported by functions that calculate weights, find paths and compare elements.

Model Validation: Models are created under a set of rules (that can be represented by constraints such as OCL) and a common language (their metamodel). Analysts and modelers require flexible graphical validation and visual detection of structural anomalies, e.g. metamodel conformity, attributes without values, or duplicate objects.

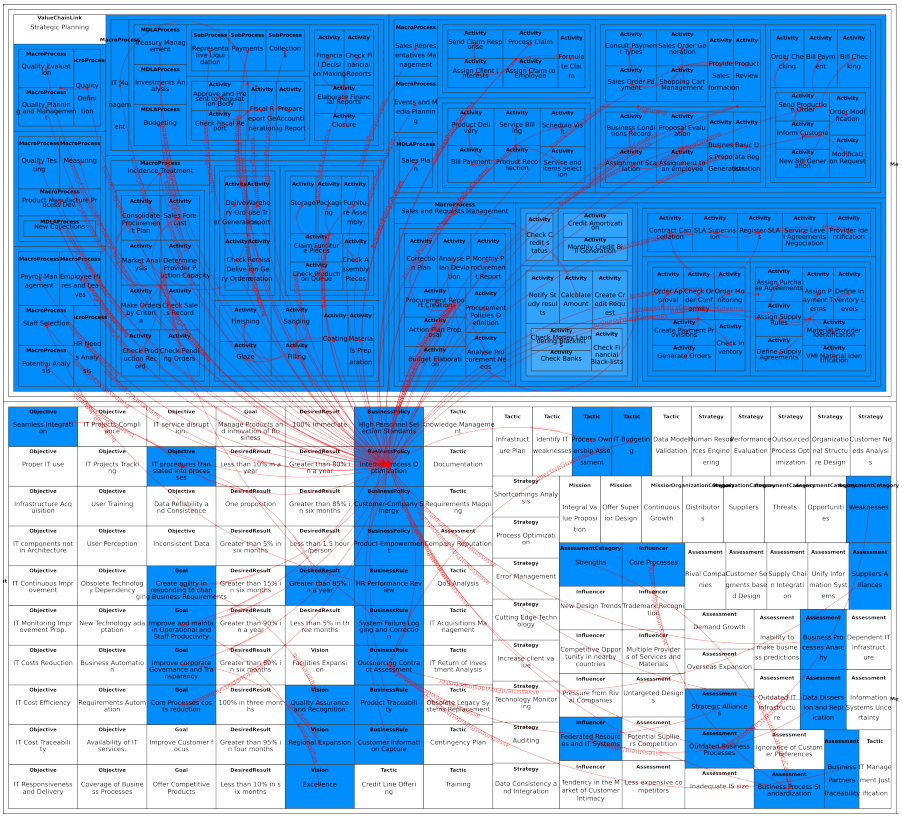

Figure 7 – Semantic Characteristics - Combining queries and visualizations for obtaining better insights. Treemap visualization with d3.js [BOH11], highlighting model elements that are impacted by the removal of a given node.

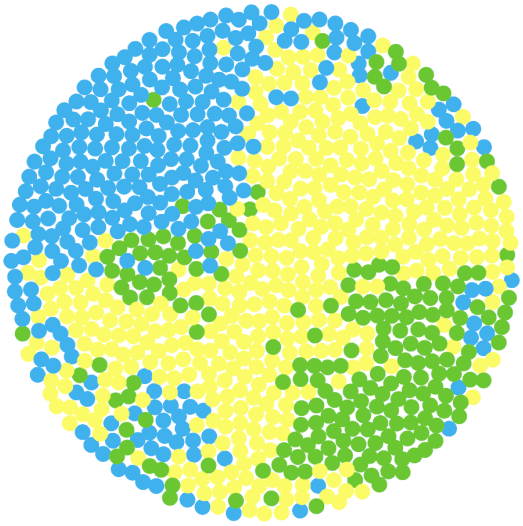

Figure 8 – Example of the Architectural Qualities principle, displaying the division of the model in three abstraction layers, from light to dark color intensities: Business (Yellow), Applications (Blue) and Infrastructure (Green).

Traceability of Analysis: An analysis tool should provide mechanisms that record decisions and their rationale (see Anamnesis, [PdKP12]), and in general, of the whole analysis process, including annotations on the elements and groups of the model.

User-defined analysis functions: *Ad hoc* analyses can be enhanced using custom functions that insert attributes to elements and relations in a user-defined fashion.

4.2.6 Uncover Architectural Qualities (UAQ)

Dependencies: GSC.

As we can make assessments in terms of certain elements or groups of the model, architectural concepts can be given in terms of the whole architecture. These assessments involve the collection of evidence and short descriptions or statements about these architectures.

Even the most simple statements require the elaboration of artifacts (such as matrices, catalogs or diagrams) that support these assessments. For instance, an analyst would question the orientation of an enterprise architecture; e.g., Which domain/layer is predominant in the architecture? If each domain has a color, which color describes better or dominates the architecture? (See Fig. 8).

4.2.7 Provide a Flexible Metamodel (PFM)

Conceptual models should not come with visualization information within them; this information needs to be decoupled [KAPS10, ELSW06] from their metamodels, as representational data adds additional noise to the model when it is defined in an implicit manner (as annotations or as new concepts). Thus, visual models should decorate conceptual models, describing *how* to visualize data, instead of *what* to visualize.

However, information loss can occur between transformations from semantic to visual models, which can result in incomplete or inaccurate visualizations. This gap between the models leads to uncertainty that affects analysis tasks, as the user has to deal with missing or misleading data. At the same time, too detailed visualizations minimize readability. As suggested by Shneiderman [Shn96], additional attributes should be displayed selectively (details on demand).

5 Evaluation

It is desirable at this point to evaluate the capabilities of the different EA modeling tools for various reasons. First, EA Frameworks such as TOGAF emphasize on the selection of an appropriate set of tools for architecture development, as the EA model needs to be "managed and controlled, particularly with a view to re-use" [The09]. Given the breadth of the market for such tools, it becomes critical to evaluate which tool serves an architect better for his specific needs, as these tools require in general a significant investment in cost and learning.

Second, it is clear that most of these tools are focused on modeling, and analysis of EA models is performed by both queries and views, complemented with static reports that answer specific questions. Given that neither of the tools offer support for dedicated visual analysis, it is necessary to assess to what extent those tools meet the aforementioned requirements.

Finally, it is desirable to assess the capabilities of visualization tools that focus on the analysis of large complex networks, and measure the difference between what is offered by EA tools and what can be done with specialized visualization toolkits.

5.1 Case Study

Muebles de los Alpes[1] is a fictional furniture manufacturing company that is part of our Enterprise Architectures Laboratory. This company has a complete technological infrastructure, as well as a thorough Enterprise Architecture description. In order to explore different concepts and applications of Model-driven Enterprise Architecture we built an enterprise metamodel divided in 11 domains that describe aspects of the architecture from different layers.

These domains incorporate elements from other metamodels (i.e., Business Motivation Model[Obj14b], ArchiMate[Obj14a], BPMN[Obj14c]) , or are customized representations of standards, frameworks such as Service Oriented Architecture, or even a formalization of common concepts such as Organizational Structure and Infrastructure. This left us with a metamodel of 112 elements, with domain-wide and inter-domain relations. The corresponding model has 904 elements and 1596 relations. This complexity allowed us to explore different mechanisms for representing visually this model under several visualization tools.

5.2 Methodology

This evaluation attacks two fronts. First, we evaluated EA modeling tools by inviting an user, with experience both in modeling and EA, to model part of the architecture of our Case Study, which was presented in a document. Due to the variety of modeling languages and expressiveness of their metamodels in each tool, we associated fragments of the architecture that could be best described using the constructs at hand. For instance, Business Processes were modeled in more detail with tools that supported the BPMN language, while other tools required a less granular approach, that nevertheless covered more domains (consider ArchiMate, where we can model Business, Application, Technology and Motivation domains).

The user had to interact with the tools, and inspect their modeling environments, as well as the reports and diagrams that they generated. The goal of this first step is to familiarize with the modeling tools by using them, so we did not evaluate the accuracy or correctness of the produced models. While this empirical approach can lead to a partial assessment of a tool, the user was encouraged to navigate the toolbars and explore the different options available, in order to assess the analytic and visual capabilities of each EA modeling tool. The modeling process consisted of modeling a domain of the architecture. This process was accompanied by sample diagrams, guidance on offered features, as well as providing the manuals for each tool.

As a second step, in order to evaluate Visualization toolkits, we took the complete Enterprise model from our Case Study and developed several Overview Visualizations, taking into account the requirements described in Section 4. Selected visualizations were shown to the person, and he evaluated its score in each requirement.

Our approach for generating visualizations is based in [Bul08, BEL+07, KAPS10], and involved using ATL model to model (M2M) transformations and model to text (M2T) transformations, as each tool accepted data under tool specific languages (i.e.

[1] If the reader wants to know more about this and other case studies, more information is available in: `http://www.losalpes.com.co`

Tool	Vendor	Version
Architect [BiZ13]	BizzDesign	3.2.1 Professional
Rational System Architect [IBM]	IBM	11.4
Enterprise Architect [Spa]	Sparx Systems	9.3 Ultimate
Iteraplan [Gmb14]	Iteratec	3.0 M1
SAMU Repository [Ato13]	Atoll Technologies	5.31
Corporate Modeler Suite [Cas13]	Casewise	2011.4

Table 2 – Selected EA modeling tools

GraphML[2], Dot[3], CAIDA[4], or even Java). This often required the design of the metamodels of these languages.

Figures 1, 2, 5, 6, 7 and 8 are tailored visualizations that resulted from the exploration of these visualization tools.

In order to allow visual comparison of the different tools, we make use of Kiviat diagrams [ELSW06] (also known as starplots [GHM08]). Each angular spoke represents a requirement, which we separated in two sections. Right hand side describes EA-specific Requirements, and should be read clockwise. Left hand side represents Visual Requirements, and should be read counter-clockwise (see Fig. 11).

We would like to highlight that this evaluation does not have a representative sample, as it involves the experience of one user. Thus, the content of this section serves as an illustration of how can we evaluate the visual capabilities of a set of modeling and visualization tools. In order to replicate it, we suggest using a substantial sample of subjects, as well as considering other critical aspects that influence the evaluation, such as previous experience with some of the tools and their degree of usability, as well as the time employed, background and visual literacy of the test subjects.

In our case, the test subject was a last-semester CS student, with considerable experience in BizzDesign Architect and Sparx Systems Enterprise Architect, with little or no familiarity with the rest of the tools. The time employed on each tool was not constrained, but the recommendation was to employ at least 2 hours with known tools, 3 hours for unknown tools.

5.2.1 EA Modeling tools

In order to evaluate the visualization capabilities of modeling tools, we selected 6 common EA management suites (see Table 2). Each tool seeks differentiation by offering diverse sets of features, thus proposing a different paradigm of EA modeling and visualization.

5.2.2 Visualization toolkits

We selected six visualization toolkits or frameworks for the evaluation of the requirements from a visual perspective (see Table 3). These tools support visualization of large sets of multivariate data, focusing on graph representations under different layout algorithms and offering a wide range of capabilities both for processing and displaying the data. For this reason, most of the tools are used in other fields, such as biology

[2] http://graphml.graphdrawing.org/specification.html
[3] http://www.graphviz.org/doc/info/lang.html
[4] http://www.caida.org/tools/visualization/libsea/

Tool	Version	Licence
GraphViz [EGK+04]	2.28	EPL v1.0
Prefuse [HCL05]	beta release 2007.10.21	BSD
GraphStream [DGOP07]	1.1	LGPL v3
Gephi [BHJ09]	0.8.1	GPL v3
CIRCOS [KSB+09]	0.64	GPL v3
d3.js [BOH11]	3.3.3	BSD

Table 3 – Selected Visualization toolkits

and social sciences, where large volumes of information are the rule rather than the exception.

Criteria for selection included:

- **Ability to process the model:** The toolkit should be able to process thousands of elements and relations.
- **Expressiveness of the visualizations:** Supported visualization techniques should provide a clear overview of the model. Thus, toolkits that only offered too detailed techniques (e.g. bar or pie charts) were discarded, as well as unrelated (e.g. maps) toolkits.
- **Licencing:** We selected only Open Source toolkits, preferring the ones that their licencing allowed to be modified and integrated into modeling tools.

6 Results

As we commented in Section 1, we are not offering a ranking of the different EA modeling tools in the market. We are conscious that the tools and frameworks evaluated here serve multiple purposes, and vary greatly in scope and target customer segments, price, and maturity, among other factors. In this study, we are evaluating the visual capabilities of these tools that enable Visual Analysis, seen as a vehicle for understanding the whole model and obtaining additional knowledge from it, as well as collecting evidence to confirm/deny hypotheses about this model.

6.1 EA Modeling Tools

As neither of the tools offered an unfiltered view of the whole enterprise model, criteria for evaluation were based in how they envisioned EA visualization under each requirement. Evaluation range was an integer score from zero to five. For example, a score of zero implied that their paradigm gave no clues about a requirement, or their approach did the opposite of the requirement. A score of three was given if there were signs of an effort to resolve issues that a requirement described, but improvements were needed, or their vision was not totally clear. A score of five was given if there was a conscious and successful effort to address a requirement.

6.2 Visualization Toolkits

The other part of this evaluation is the assessment of Visualization toolkits and frameworks. However, as these are general purpose visualization tools, we relaxed the

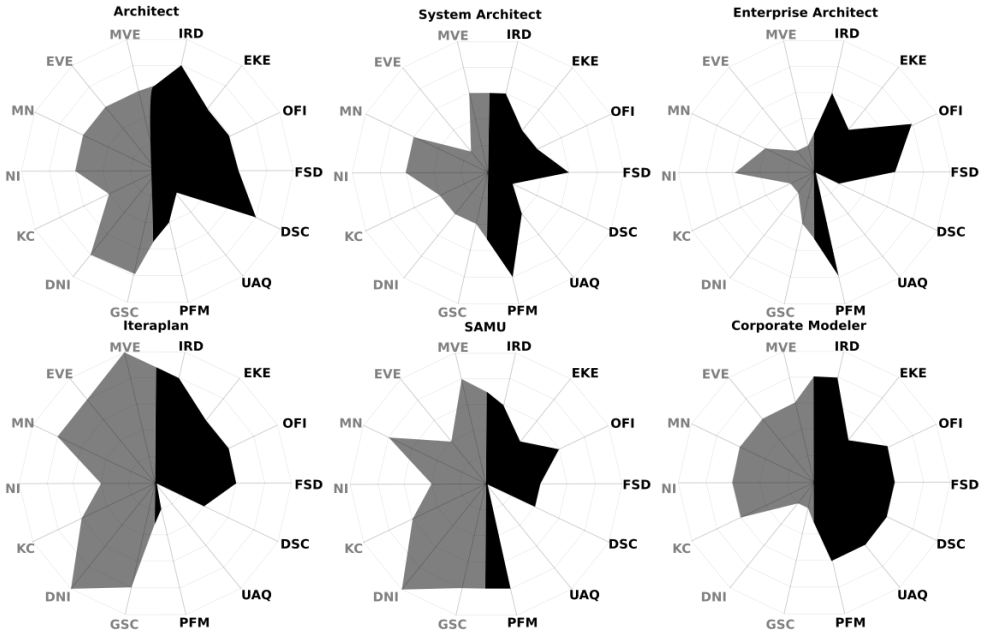

Figure 9 – Evaluation results for EA Modeling tools.

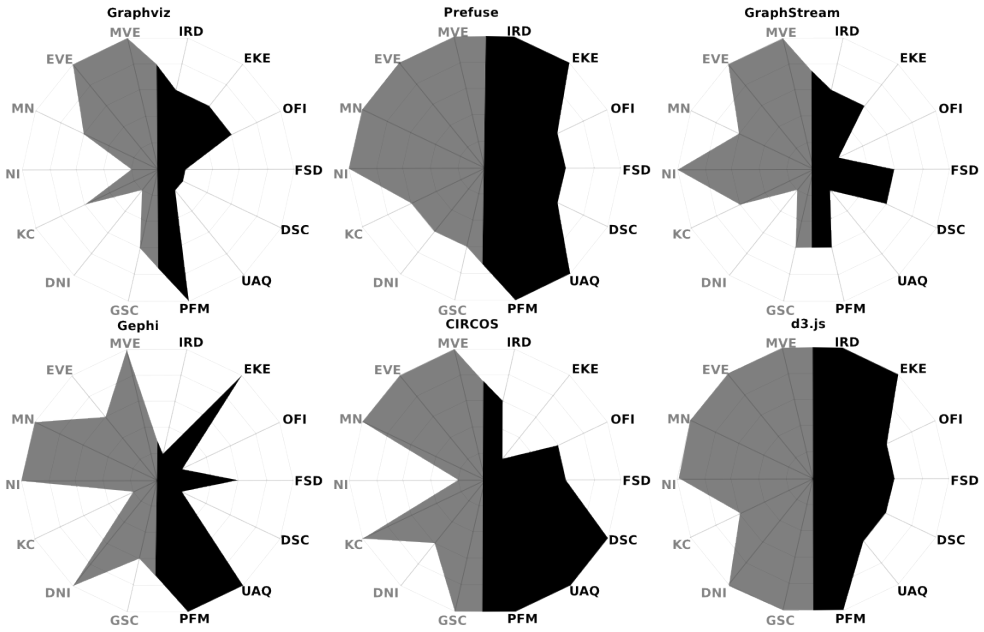

Figure 10 – Evaluation results for Visualization tools.

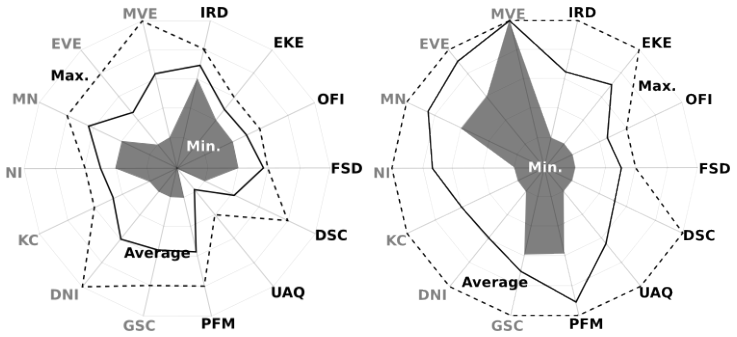

Figure 11 – Summary of the EA (left) and Visualization (right) results.

precision of our measurements to three possible values, based on the implementation of the visualizations: 5 if it was manageable to fulfill a requirement; 3 if it seemed achievable (at least partially) to fulfill, often with a certain degree of difficulty; or 1 if it was very difficult or not possible to represent this requirement (see Fig. 10).

6.3 Outlook

An overview of EA and Visualization tool evaluations is shown in Fig. 11, showing minimum, average and maximum values for each requirement. We provide a summary of the findings of the evaluation, highlighting the strengths and weaknesses of the EA modeling tools under each requirement. Due to space constraints, we only include the set of EA Modeling tools in this outlook[5].

Maximize Visual Economy

Some of the tools manage several visual languages (e.g. UML, BPMN, ArchiMate) that are known to have a large number of symbols. Also, these symbols are commonly large and of squared form, which makes it difficult to handle them visually, and usually a lot of decoding is necessary.

On the other hand, other tools manage a predefined set of symbols that represent concepts. These symbols are intuitive and offer a good semantic correspondence in most of cases. However, this also has its drawbacks. Most of the symbols represent technical concepts, so the user may need to associate more abstract (or non-technical) concepts with a symbol.

Enhance Visual Expressiveness

Concerning Visual Expressiveness, the interface and the views of some tools are aesthetically pleasing. It is also interesting that they make use of visual elements to offer a view of AS-IS -> TO-BE transitions. However, in some of these tools there is not a conscious effort to map visual variables to semantic meaning. In most cases, they only use shape, color and position for this mapping, thus offering **limited expressiveness**.

Other tools offer the user ways to customize different visual variables like size, line width, font, transparency, color and shape, and make use of several tools to highlight elements. However, the user is responsible of making these adjustments, so

[5]For a more detailed report of this evaluation, please visit `http://prim.com.co/docs/jotTR.pdf`.

the chance is high that these variables have no semantic correspondence. These tools are **manually expressive**.

Minimize the Noise

There are some efforts to manage complexity, but they are often truncated by technological barriers. Some tools offer a selection of layout algorithms that can aid understanding and reduce clutter, but usually cannot handle diagrams with a considerable (>50) number of elements.

Reports are usually concise, so there is no irrelevant information in them. General layout is well designed and organized. However, there is no explicit mechanism for complexity management, as most tools don't offer different layout algorithms depending on size of the view.

Some tools generate clean diagrams that offer just the necessary information. However, edges are difficult to follow and need additional decoding, as a lot of elements are present.

Navigate and Interact

The tools have various controls for navigating, zooming and selecting elements, and other generate a model explorer in HTML for easy textual navigation.

Even tough they don't have specialized interaction controls, they offer **simple interaction** and their canvas is large and navigation is easy and quick. In some tools there is a general landscape map that organizes building blocks and offers navigability throughout their metamodel.

Keep the Context

The user is **maintained in context** with diagrams that are visually manageable, as they are generated in predefined formats on most of the cases. Other diagrams (in special information flow diagrams) can be too wide, and there is no explicit mechanism for dealing with model complexity as the number of elements increases.

Derive New Insights

The support for deriving new insights in every tool is based on queries of the model. They provide a **flexible query language**, directly inspecting the model database through SQL statements, or by specialized query languages. Filters are easy to formulate, and include all attributes from the concepts involved, with no information loss at glance.

Guarantee Semantic Correspondence

In general, there is no legend in diagrams, as they **require specialized knowledge**, i.e. the user must be familiar with the the visual language description, and there are no clues to give semantic meaning in the diagrams. There are mechanisms that associates visual constructs with semantic meaning, but is separated from all the views. Generated reports offer some additional information (usually text).

Identify and Relate Domains

Most tools support different viewpoints, and offer a **summarized view of EA layers** of the architecture, in most cases displaying the four classical architectural layers in addition to a projects domain. They have a navigational view, and seem to be

conscious of relations between layers. However, there is no way to discover which concepts belong to which layer, or layer/domain interdependency in detail.

They also show the hierarchy of architectural layers by using a color code, usually with dependencies and borderline elements. Such a view is well structured but static, and more detail is needed on inter-layer relations.

Emphasize Key Elements

Some tools define a **minimal set of key concepts** in their metamodel, so they all seem to have the same semantic importance. A side effect is that there is no mechanism for inspecting key elements of the architecture as the metamodel evolves and the model gets more complex.

Other tools depend on the emphasis given by the metamodel to certain concepts through **predefined viewpoints**. However, these viewpoints are only definitions of allowed constructs in the view, so there is no mechanism (other than symbol containment) for giving more importance to key elements.

Offer a Focus of Interest

In general, there is no explicit way to establish a focus of interest. Some context is present in views- however, they are not flexible and are more useful in modeling than in analysis activities.

Reports offered by some tools give a fixed focus of interest that brings a context to analysis. However, there is usually no mechanism for creating new diagram types or extend existing ones.

Facilitate Structural Diagnosis

In general, it was difficult to provide a clear structural diagnosis from the different diagrams. Overall patterns are difficult to recognize, given the localized nature of the diagrams. However, given its dependencies (see Fig. 4), the score of this requirement is influenced by the visual representation of elements (MVE) and the interaction capabilities the tools have (NI).

Display Semantic Characteristics

Some tools have a faint notion of Semantic Characteristics by displaying **non-visual conformance reports**, i.e. a list of semantic validation errors, or **impact analyses** by exploring the model and deriving relations. Consistency checks offer a way for validating correctness and completeness of the model. As support for this requirement is not always an easy task, they have a set of predefined diagnostics that list non-conformant elements. However, there is no visual display of the results of these diagnostics, nor a mechanism for their extension.

A small set of tools have mechanisms that enrich existing views by adding semantic elements by relation derivation and navigation through the whole model, e.g. comparisons, customization of element colors, relation types, and semantic highlighting.

Uncover Architectural Qualities

There is only a faint notion of this requirement in a tool that generates heat maps that complement analysis tasks. Thus, this requirement offers great opportunities for further exploration by tool vendors.

Metamodel Flexibility and Visual Model

Some of the tools handle several metamodels, making possible to have diagrams under different visual languages. They are metamodel agnostic in a way, with all the benefits and drawbacks that this brings. For instance, it provides flexibility in modeling, but hinders analysis, as there is no unified metamodel.

Other tools can perform minor changes to the metamodel by the concept of profiles, adding attributes to concepts. With some tools it is possible to add/modify/remove concepts easily.

7 Discussion

Views and queries are valuable for communicating Enterprise Architectures, as well as for performing scoped (i.e. detailed) analyses. However, as organizations gain complexity both in business functions and IT services, models that make an abstraction of these aspects also get larger and more difficult to analyze. We propose the use of Overview Visualizations to get valuable insights and ease the burden on EA analysts. With this in mind, we examine similar approaches, propose a set of requirements for the Visual Analysis of these models, and examine six EA modeling tools and six visualization toolkits, evaluating their visual capabilities through the artifacts (i.e. visualizations, diagrams and reports) they generate.

In general, the reader can reflect on the fact that there is much road ahead in the domain of EA tooling. Even tough there are tools that give more focus to visualization, offer more consistent (if not effective) means to analyze, explore and communicate architectures, and clearly help analysts in the process of extracting new information, the average suggests that visual analysis, (and in general, visualization) need more exploration both from EA tool vendors and researchers.

Some of the strong points of the evaluated EA tools, and which we consider are the starting point for further exploration by tool vendors are:

- The ability to derive new insights by the construction of queries.
- The Visual Economy offered by some tools, leveraging on the visual system of the user.
- Noise reduction of diagrams by using several layout algorithms.
- The ability to visualize inter-domain relationships.
- Impact Analysis and model validation capabilities.
- Flexibility in the metamodel of some of the tools.
- The effort of tools like Iteraplan for offering a good semantic correspondence of diagrams and reports.

However, we also consider that there are some subjects need to tackled by both industry and academia:

1. Simpler visual representations of concepts and relations, as current visual languages fail to support structural analysis of the whole model.

2. Mechanisms that allow the composition and automation of analysis methods, in order to answer more difficult or abstract questions.
3. New techniques where the analyst can easily detect structurally important elements of the architecture.
4. The introduction of interactive operations (e.g. fish-eye views, semantic zoom, 3D navigation) for the exploration of these models.

These points can be tackled with the evaluated visualization tools, as they offer as a whole an almost complete set of features for supporting visual analysis of enterprise models. However, all of these capabilities are not yet found in just one toolkit. Thus, an hypothetical EA Visual Analysis tool should incorporate the different approaches of these tools, that is, to serve as a common platform that uses several visualization toolkits.

Last, but not least, we can say that nowadays there is an effort to tackle some of these problems. Tendencies such as Model-driven visualization[Bul08] are key in this context. For instance, an interesting research direction is the design of a visualization language that includes the mapping between analysis questions and visual variables, or in other words, that models the intent of giving semantic significance to visualizations.

References

[AD07] S. Alam and P. Dugerdil. Evospaces visualization tool: Exploring software architecture in 3d. In *Reverse Engineering, 2007. WCRE 2007. 14th Working Conference on*, pages 269–270, Oct 2007. doi:10.1109/WCRE.2007.26.

[AS05] R.A. Amar and J.T. Stasko. Knowledge precepts for design and evaluation of information visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 11(4):432 –442, july-aug. 2005. doi:10.1109/TVCG.2005.63.

[Ato13] Atoll Technologies. Samu overview, 2013. [Online; accessed 13-November-2013]. URL: http://www.atollgroup.eu/en/solutions/samu/overview/.

[BDL05] Michael Balzer, Oliver Deussen, and Claus Lewerentz. Voronoi treemaps for the visualization of software metrics. In *Proceedings of the 2005 ACM Symposium on Software Visualization*, SoftVis '05, pages 165–172, New York, NY, USA, 2005. ACM. doi:10.1145/1056018.1056041.

[BE02] Alan F. Blackwell and Yuri Engelhardt. A meta-taxonomy for diagram research. *Diagrammatic Representation and Reasoning*, 2002. doi:10.1007/978-1-4471-0109-3_3.

[BEL+07] Sabine Buckl, Alexander M. Ernst, Josef Lankes, Christian M. Schweda, and André Wittenburg. Generating visualizations of enterprise architectures using model transformations. In *EMISA*, volume P-119 of *LNI*, pages 33–46. GI, 2007. URL: http://subs.emis.de/LNI/Proceedings/Proceedings119/article1951.html.

[Bel09] M. Bell. *SOA Modeling Patterns for Service Oriented Discovery and Analysis*. Wiley, 2009.

[Ber81] J. Bertin. *Graphics and graphic information-processing.* Walter de Gruyter, Berlin, 1981.

[Ber83] Jacques Bertin. *Semiology of graphics.* University of Wisconsin Press, 1983.

[BFKW06] T. Bucher, R. Fischer, S. Kurpjuweit, and R. Winter. Analysis and application scenarios of enterprise architecture: An exploratory study. In *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW '06. 10th IEEE International*, page 28, oct. 2006. doi:10.1109/EDOCW.2006.22.

[BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the Third International Conference on Weblogs and Social Media (ICWSM'09)*. The AAAI Press, 2009. URL: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/viewFile/154Forum/1009.

[BiZ13] BiZZdesign. Architect, 2013. [Online; accessed 13-November-2013]. URL: http://www.bizzdesign.com/tools/bizzdesign-architect/.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011. doi:10.1109/TVCG.2011.185.

[Bul08] Robert Ian Bull. *Model driven visualization: towards a model driven engineering approach for information visualization.* PhD thesis, Victoria, B.C., Canada, Canada, 2008. 978-0-494-47327-6.

[BW05] Christian Braun and Robert Winter. A comprehensive enterprise architecture metamodel and its implementation using a metamodeling platform. In *Enterprise Modelling and Information Systems Architectures, Proc. of the Workshop in*, pages 64–79. EMISA, 2005. URL: http://subs.emis.de/LNI/Proceedings/Proceedings75/article3778.html.

[Cas13] Casewise. Casewise modeler, 2013. [Online; accessed 13-November-2013]. URL: http://www.casewise.com/products/modeler.

[Chi00] Ed H. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Visualization 2000*, INFOVIS '00, pages 69–, Washington, DC, USA, 2000. IEEE Computer Society. doi:10.1109/INFVIS.2000.885092.

[CHP96] L.J. Campbell, T.A. Halpin, and H.A. Proper. Conceptual schemas with abstractions making flat conceptual schemas more comprehensible. *Data and Knowledge Engineering*, 20(1):39 – 85, 1996. doi:10.1016/0169-023X(96)00005-5.

[CKM10] Yu-Hsuan Chan, Kimberly Keeton, and Kwan-Liu Ma. Interactive visual analysis of hierarchical enterprise data. In *Proceedings of the 12th IEEE International Conference on Commerce and Enterprise Computing*, CEC '10, pages 180–187, Washington, DC, USA, 2010. IEEE Computer Society. doi:10.1109/CEC.2010.37.

[DGOP07] Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007)*, Dresden Germany, 10 2007. URL: http://arxiv.org/abs/0803.2093.

[Dom12] Christofer Domas. ..cantor.dust.., 2012. [Online; accessed 13-November-2013]. URL: https://sites.google.com/site/xxcantorxdustxx/home.

[EGK+04] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz and dynagraph — static and dynamic graph drawing tools. In *Graph Drawing Software*, Mathematics and Visualization, pages 127–148. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-642-18638-7_6.

[Egy02] Alexander Egyed. Automated abstraction of class diagrams. *ACM Trans. Softw. Eng. Methodol.*, 11:449–491, October 2002. doi:10.1145/606612.606616.

[EHH10] Peter Eades, Weidong Huang, and Seok-Hee Hong. A force-directed method for large crossing angle graph drawing. *CoRR*, abs/1012.4559, 2010. URL: http://arxiv.org/abs/1012.4559.

[ELSW06] Alexander M. Ernst, Josef Lankes, Christian M. Schweda, and Andre Wittenburg. Tool support for enterprise architecture management - strengths and weaknesses. In *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, EDOC '06. IEEE Computer Society, 2006. doi:10.1109/EDOC.2006.60.

[Epp06] M.J. Eppler. Toward a pragmatic taxonomy of knowledge maps: Classification principles, sample typologies, and application examples. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pages 195 –204, july 2006. doi:10.1109/IV.2006.111.

[FM86] P. Feldman and D. Miller. Entity model clustering: Structuring a data model by abstraction. *The Computer Journal*, 29(4):348–360, 1986. doi:10.1093/comjnl/29.4.348.

[FR91] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21:1129–1164, November 1991. doi:10.1002/spe.4380211102.

[FR07] Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 37–54, Washington, DC, USA, 2007. IEEE Computer Society. doi:10.1109/FOSE.2007.14.

[Fra02] Ulrich Frank. Multi-perspective enterprise modeling (memo) conceptual framework and modeling languages. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 1258 – 1267, jan. 2002. doi:10.1109/HICSS.2002.993989.

[GDDS06] Dragan Gaaevic, Dragan Djuric, Vladan Devedzic, and Bran Selic. *Model Driven Architecture and Ontology Development*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[GHM08] K. Gallagher, A. Hatch, and M. Munro. Software architecture visualization: An evaluation framework and its application. *Software Engineering, IEEE Transactions on*, 34(2):260 –270, march-april 2008. doi:10.1109/TSE.2007.70757.

[Gmb14] Iteratec GmbH. Iteraplan, 2014. [Online; accessed 17-March-2014]. URL: https://www.iteraplan.de/en.

[HCL05] Jeffrey Heer, Stuart K. Card, and James A. Landay. prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 421–430, New York, NY, USA, 2005. ACM. doi:10.1145/1054972.1055031.

[HMMR13] Markus Hipp, Bela Mutschler, Bernd Michelberger, and Manfred Reichert. A framework for the intelligent delivery and user-adequate visualization of process information. In Sung Y. Shin and José Carlos Maldonado, editors, *SAC*, pages 1383–1390. ACM, 2013. doi:10.1145/2480362.2480623.

[HMR11] Markus Hipp, Bela Mutschler, and Manfred Reichert. On the context-aware, personalized delivery of process information: Viewpoints, problems, and requirements. In *ARES*, pages 390–397. IEEE, 2011.

[HS12] Jeffrey Heer and Ben Shneiderman. Interactive dynamics for visual analysis. *Queue*, 10(2):30:30–30:55, February 2012. doi:10.1145/2133416.2146416.

[Hu05] Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005. URL: http://www.mathematica-journal.com/issue/v10i1/contents/graph_draw/graph_draw.pdf.

[IBM] IBM Corporation. *IBM Rational System Architect User Guide Release 11.3*.

[IS11] N. Iliinsky and J. Steele. *Designing Data Visualizations: Representing Informational Relationships*. Oreilly and Associate Series. O'Reilly Media, 2011.

[KAPS10] Steffen Kruse, Jan Addicks, Matthias Postina, and Ulrike Steffens. Decoupling models and visualisations for practical ea tooling. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, volume 6275 of *Lecture Notes in Computer Science*, pages 62–71. Springer Berlin / Heidelberg, 2010. doi:10.1007/978-3-642-16132-2_6.

[KHL+07] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods - a survey. *ACM Comput. Surv.*, 39, November 2007. doi:10.1145/1287620.1287621.

[KSB+09] Martin I Krzywinski, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 2009. doi:10.1101/gr.092759.109.

[KW05] Stephen G. Kobourov and Kevin Wampler. Non-euclidean spring embedders. *IEEE Transactions on Visualization and Computer Graphics*, 11:757–767, November 2005. doi:10.1109/TVCG.2005.103.

[KY93] H. Koike and H. Yoshihara. Fractal approaches for visualizing huge hierarchies. In *Visual Languages, 1993., Proceedings 1993 IEEE Symposium on*, pages 55 –60, aug 1993. doi:10.1109/VL.1993.269566.

[LY05] Chun-Cheng Lin and Hsu-Chun Yen. A new force-directed graph drawing method based on edge-edge repulsion. In *Information Visualisation, 2005. Proceedings. Ninth International Conference on*, pages 329 – 334, july 2005. doi:10.1016/j.jvlc.2011.12.001.

[MLO00] E Morse, M Lewis, and K.A Olsen. Evaluating visualizations: using a taxonomic guide. *International Journal of Human-Computer Studies*, 53(5):637 – 662, 2000. doi:10.1006/ijhc.2000.0412.

[Moo97] Daniel Moody. A multi-level architecture for representing enterprise data models. In *Conceptual Modeling — ER '97*, volume 1331 of *Lecture Notes in Computer Science*, pages 184–197. Springer Berlin / Heidelberg, 1997. doi:10.1007/3-540-63699-4_16.

[Moo09] Daniel L. Moody. The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35:756–779, 2009. doi:10.1109/TSE.2009.67.

[Mun00] Tamara Macushla Munzner. *Interactive visualization of large graphs and networks*. PhD thesis, Stanford, CA, USA, 2000.

[Mun09] T. Munzner. A nested model for visualization design and validation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):921 –928, nov.-dec. 2009. doi:10.1109/TVCG.2009.111.

[NdFCSJM12] Renato Lima Novais, Glauco de F. Carneiro, Paulo R.M. Simões Júnior, and Manoel Gomes Mendonça. On the use of software visualization to analyze software evolution: An interactive differential approach. In *Enterprise Information Systems*, volume 102 of *Lecture Notes in Business Information Processing*, pages 241–255. Springer Berlin Heidelberg, 2012. URL: http://dx.doi.org/10.1007/978-3-642-29958-2_16, doi:10.1007/978-3-642-29958-2_16.

[NSV13] David Naranjo, Mario E. Sánchez, and Jorge Villalobos. Connecting the dots: Examining visualization techniques for enterprise architecture model analysis. In Janis Grabis, Marite Kirikova, Jelena Zdravkovic, and Janis Stirna, editors, *PoEM (Short Papers)*, volume 1023 of *CEUR Workshop Proceedings*, pages 29–38. CEUR-WS.org, 2013. URL: http://ceur-ws.org/Vol-1023/paper3.pdf.

[Obj14a] Object Management Group. Archimate, 2014. [Online; accessed 13-November-2013]. URL: http://www.opengroup.org/subjectareas/enterprise/archimate.

[Obj14b] Object Management Group. Business motivation model (bmm), 2014. [Online; accessed 13-November-2013]. URL: http://www.omg.org/spec/BMM/.

[Obj14c] Object Management Group. Business process model and notation (bpmn), 2014. [Online; accessed 13-November-2013]. URL: http://www.omg.org/spec/BPMN/.

[PdKP12] Georgios Plataniotis, Sybren de Kinderen, and Henderik A. Proper. Ea anamnesis: towards an approach for enterprise architecture rationalization. In *Proceedings of the 2012 workshop on Domain-specific modeling*, DSM '12, pages 27–32, New York, NY, USA, 2012. ACM. doi:10.1145/2420918.2420927.

[PH13] James Peters and Randima Hettiarachichi. Visual motif patterns in separation spaces. *Theory and Applications of Mathematics & Computer Science*, 3(2), 2013. URL: http://www.uav.ro/stiinte_exacte/journal/index.php/TAMCS/article/view/81.

[Rei12] Manfred Reichert. Visualizing large business process models: Challenges, techniques, applications. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops*, volume 132 of *Lecture Notes in Business Information Processing*, pages 725–736. Springer, 2012. doi:10.1007/978-3-642-36285-9_73.

[Shn96] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343. IEEE, 1996. doi:10.1109/VL.1996.545307.

[Soa07] Luís Miguel Jesus Soares. Enterprise architecture: Information systems architecture visualization. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisboa, Portugal, October 2007.

[Spa] Sparx Systems. *Enterprise Architect 10 Reviewer's Guide*.

[SSvH14] Christoph Daniel Schulze, Miro Spönemann, and Reinhard von Hanxleden. Drawing layered graphs with port constraints. *Journal of Visual Languages and Computing, Special Issue on on Diagram Aesthetics and Layout*, 25(2):89–106, 2014. doi:10.1016/j.jvlc.2013.11.005.

[Tam10] R. Tamassia. *Handbook of Graph Drawing and Visualization*. Discrete Mathematics And Its Applications. Taylor and Francis, 2010.

[TH05] Yannis Tzitzikas and Jean-Luc Hainaut. How to tame a very large er diagram (using link analysis and force-directed drawing algorithms). In *ER*, volume 3716 of *Lecture Notes in Computer Science*, pages 144–159. Springer, 2005. doi:10.1007/11568322_10.

[TH06] Yannis Tzitzikas and Jean-Luc Hainaut. On the visualization of large-sized ontologies. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 99–102, New York, NY, USA, 2006. ACM. doi:10.1145/1133265.1133285.

[The09] The Open Group. TOGAF 9 - The Open Group Architecture Framework Version 9, 2009.

[VO10] Antonio Villegas and Antoni Olivé. A method for filtering large conceptual schemas. In *Conceptual Modeling – ER 2010*, volume

6412 of *Lecture Notes in Computer Science*, pages 247–260. Springer Berlin / Heidelberg, 2010. doi:10.1007/978-3-642-16373-9_18.

[vW05] J.J. van Wijk. The value of visualization. In *Visualization, 2005. VIS 05. IEEE*, pages 79 – 86, oct. 2005. doi:10.1109/VISUAL.2005.1532781.

[WL07] Richard Wettel and Michele Lanza. Visualizing software systems as cities. In *In Proc. of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 92–99. Society Press, 2007. doi:10.1109/VISSOF.2007.4290706.

[ZF98] Michelle X. Zhou and Steven K. Feiner. Visual task characterization for automated visual discourse synthesis. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '98, pages 392–399, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. doi:10.1145/274644.274698.

About the authors

David Naranjo holds a MSc degree in Systems and Computing Engineering, and currently is a Research Assistant at the Universidad de los Andes in Bogotá, Colombia.
Contact him at da-naran@uniandes.edu.co.

Mario Sánchez is currently an Assistant Professor at the Universidad de los Andes in Bogotá, Colombia. After finishing his PhD on Executable Models and MDE for Business Process Modeling, he has focused on Enterprise Modeling and Analysis, and, more generally, on advancing tool support for enterprise architecture.
Visit him at http://sistemas.uniandes.edu.co/~mar-san1/.

Jorge Villalobos is currently an Associate Professor at the Universidad de los Andes in Bogotá, Colombia. He leads the research group in Enterprise Architecture and Modeling.
See: http://sistemas.uniandes.edu.co/~jvillalo.