Visual Analysis of Enterprise Models

David Naranjo, Mario Sánchez, Jorge Villalobos
Department of Systems and Computing Engineering
Universidad de los Andes
Bogotá, Colombia
{*da-naran,mar-san1,jvillalo*}*@uniandes.edu.co*

***Abstract*—In Enterprise Architecture (EA) 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, more detailed, and start covering more domains. 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 offer an evaluation framework to assess the strengths and weaknesses of visualization tools to support EA analysis activities. The framework is based on a set of 14 requirements which are either visualization-related or specific to EA analysis, and which where harvested from experimentation and from a survey of current EA and visualization tools. The requirements were defined around the design principle of cognitive effectiveness, and they favor a broader use of visual languages in order to enhance the semantics of EA visualizations.

***Keywords*-enterprise architecture; analysis; visualization**

I. 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. Currently, analysts have very large amounts of data available, but only a few analysis tools and methodologies that support them in the task of making sense of this data.

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 [**?**], 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 on the enterprise. Acquiring knowledge of this type sometimes needs elaboration of specialized artifacts or the introduction of complementary methodologies, that despite their usefulness and pertinence, are out of the scope of an EA analysis task.

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* offer partial glimpses of the models, but most of the times they lack a sense of continuum, missing 'the big picture' as they cover specific stakeholder concerns. For example, most EA frameworks propose several architecture layers and views in order to reduce the number of artifacts per model [**?**], but this makes it difficult to see all the concepts in a holistic way[**?**].

Visualizations, the third analysis category, 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 [**?**]. Diagrams (or 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 [**?**]. Additionally, the manual creation of visualizations is an error prone and time consuming task [**?**]. There is also a need of tools to support the coordination of business and IT on various levels of abstraction [**?**], but 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.

Visual artifacts need to be supported by queries and views to be an effective vehicle for communication. 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 diagrams can be assembled [**?**].

Domain-specific model visualization [**?**] is a largely unexplored field, as it is difficult to translate visualization rules from a concrete domain into formal rules. Buckl et al. [**?**][**?**][**?**] build model visualizations with model transformations, which they use to map model elements to their graphical representation with the help of on a layered cartography metaphor, which greatly facilitates analysis tasks by using an Enterprise Architecture Management pattern catalog. Ernst et al. [**?**] offer an extensive analysis and evaluation of tool support for EA management tasks, recognizing the importance of visualization. Given the maturity that these tools have achieved in the aspect of offered features, we seek to complement this evaluation with a set of requirements that arose when we were exploring the 'big picture' concept.

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?*

The objectives of this paper are: 1) to expose a possible gap in the support of the analysis process; 2) to introduce, describe, and justify a list of 14 requirements that may be essential to an enterprise architect in order to reduce this gap; and 3) to apply these requirements to six popular EA management tools and six visualization tools, offering a comparative evaluation framework that assesses the strengths of each tool and gives space for future research on the key areas defined by the requirements.

It is out of the scope of this paper to make a comprehensive guide both of the analysis and visualization domains. Instead, in Section **??** we will address the main issues that we have encountered when trying to represent large EA models, and briefly comment on how we can achieve effective information visualization through the use of visual semantics of a model. In Section **??** we will describe the requirements of a visualization tool for analysis, from the visual and EA perspectives. Section **??** is a summary of our case study, with evaluation results of the EA and Visualization tools. Finally, Section **??** will be a space for discussion of current tool support for EA visualization.

II. Model Visualization

Diagrams are used to depict certain characteristics of a domain with a visual language. They serve two main purposes: to facilitate understanding and to communicate. There is the perception that diagram drawing is not always the best option [**?**] as these purposes are often truncated by the complications that arise - e.g., there is no perfect layout for all models/graphs, (see Fig. **??**) or the diagram ends up in being just a drawing[**?**] with no semantic correspondence.

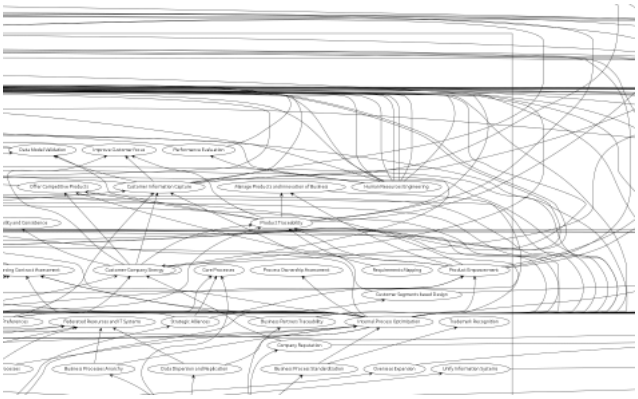

Figure 1. A very small section of an ER representation of an Enterprise Model.

In this section we will try to address some of this issues from the perspective of visualization.

A. Displaying large models

A characteristic of EA models is their large size. 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. This is not just an EA problem; we can see this in different disciplines: database schemas in Information Engineering, library dependencies in Software Engineering, or even in ontologies of big domains. This issue can be summarized with the *Database Comprehension Problem:* Usefulness of any diagram is inversely proportional to the size of the model depicted [**?**].

Representing visually these models is not an easy task, as an increase of visual elements is detrimental to human's cognitive load as well as to the system's response time [**?**]. The set of CASE tools offered require arrangements 'by hand' [**?**], which means that a great amount of time is spent dragging elements to form a decipherable diagram.

Representation of cognitive manageable large models is a known and unresolved issue, but we're interested mainly in two concrete approaches, guided by [**?**]: graph layout optimization (visual approach) and model filtering (semantic approach).

1) Graph Layouts: Optimizing graph layouts is a method for dealing with situations with high information density by structurally arranging elements based on graph properties. The most used technique is Force-Directed Layouts (FDLs) [**?**][**?**][**?**]. Most visualization tools use FDLs (see Fig. **??**) because of their flexibility, and tend to be aesthetically pleasing, exhibiting symmetries and producing crossing-free layouts for planar graphs [**?**]. For instance, some modeling environments are starting to 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, gener-

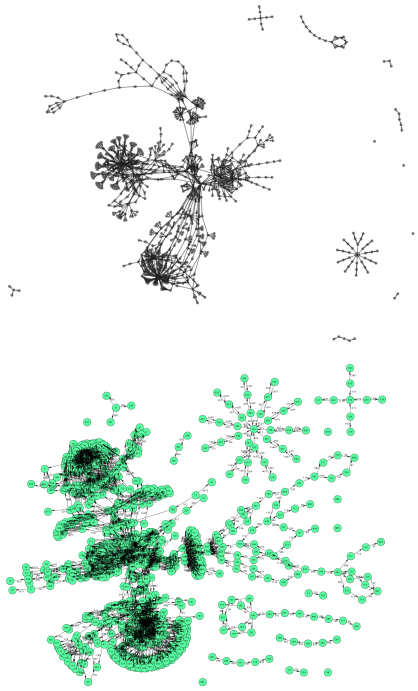

Figure 2. Two different Force-Directed layouts for an enterprise model.

ally 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 [**?**]. However, if we organize elements under some criteria, we can obtain a more clear representation (see Fig. **??**).

Other interesting approaches involve projections into non-Euclidean coordinates, such as Hyperbolic ($\mathbb{H}^2$) and Spherical ($\mathbb{S}^2$) spaces, but currently they have some limitations such as the requirement of quasi-hierarchical graphs[**?**], or that they have issues with very large graphs [**?**].

2) 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 models that have a large number of concepts is that visually all object types are considered to be of equal importance, that is, they are flat conceptual schemas [**?**]. 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 [**?**][**?**][**?**][**?**] propose mechanisms that allow designers to 'zoom out' 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 graph connected, in combination with traditional abstraction techniques, such as generalization, aggregation, and association. Recent approaches like [**?**] are interesting, because they rely on capturing the information needs from an user by the definition of a focus of interest function, filtering 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 and make heavy use of abstraction rules that certainly are not easily created and are specific to the metamodel (i.e. abstraction for a UML model is different than abstraction in BPMN or ArchiMate). This is a problem if we assume that an EA metamodel is specific to the organization and can change over time [**?**]. In addition, filtering and clustering techniques require additional user input and manual intervention or correction.

Automatic layout for large ER diagrams is difficult to manage, and current filtering techniques are not satisfactory. In addition, classical hierarchical decomposition techniques that are used for visualizing large plain graphs have not been applied or tested on conceptual diagrams [**?**]. Taking into account the benefits and limitations of both techniques, Tzizikas and Hainaut [**?**] proposed a way to 'tame' large conceptual diagrams using a ranking algorithm that filters elements, finally displaying them with a FDL. This is an example of how we can use a mixed approach when dealing with the problems described above.

However, we are not interested in hiding elements of the enterprise model. Our approach is based in a total view of this model, in order to give an overview of the architecture.

B. Giving visualizations a semantic meaning

A recurring concern when designing EA model views is the need to develop representations that are understandable by both business and technical experts [**?**], or in general, to provide a medium for communication between people with different professional backgrounds [**?**]. This often requires using a strategy for the design of these views.

Bertin [**?**][**?**] described a set of seven visual variables (position, shape, color, size, texture, value [opacity] and orientation), which are modifications to 'marks' on a given 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.

In order to increase expresiveness of EA model visualizations, our strategy is to take advantage of this mapping. An analysis process begins with questions that may be translated to a formal (query) language. These queries are processed, in order to obtain an answer under some other language, i.e. a view, a table, some text or a visualization. In our context, this processing can be seen as a transformation function that operates over the model, assigning a value to all model elements, i.e., ranking these elements. We can map

these processed values into visual values that visual variables operate to encode graphically a query response, giving a semantic meaning to this view.

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 [**?**], or visualization frameworks and patterns for choosing and using the appropriate data visualization techniques. However, our domain-specific approach (EA Analysis) has a set of requirements, which we will explain in the next section.

III. 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.

A. Overview

We defined 14 criteria for evaluating visualization tools that surfaced in the process of exploring the concept of this 'bird-eye' or 'big picture' view in the context of EA, validating them with interviews with fellow architects (see Table **??**). We separated them in two categories. *General Requirements* deal with common visualization principles around large model representation, while *EA-specific Requirements* are a (wish)list of features that an architect/analyst would like to have in his toolbox. The format is the same for all requirements; they have an identifier, a name, a description, and a visual example when possible.

These requirements are not a complete list of the elements to take into account both in Data Visualization and the Enterprise Architecture domains. Instead, and given the short amount of research in this particular matter, we offer an extensible evaluation framework that hopefully will bring additional criteria in the selection of an EA management and/or a visualization tool, in order to fill the current analysis gap that exists now.

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

B. General Visualization Requirements

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 [**?**]. However, this one to one (or even one to many) correspondence also results in complex visualizations that are heavy to decode due to the large amount of symbols involved.

ID	Name
General Visualization Requirements	
MVE	Maximize Visual Economy
EVE	Enhance Visual Expresiveness
MN	Minimize the noise
NI	Navigability and Interaction
KC	Keep the context
MMI	Maintain model integrity
GSC	Guarantee Semantic correspondence
EA-specific Requirements	
DH	Domain Hierarchy
SKC	Structural Key Concepts
FI	Focus of Interest
SD	Structural Diagnosis
SC	Semantic Characteristics
AQ	Architectural Qualities
MA	Metamodel Agnosticism and Evolution

Table I
Overview of the requirements

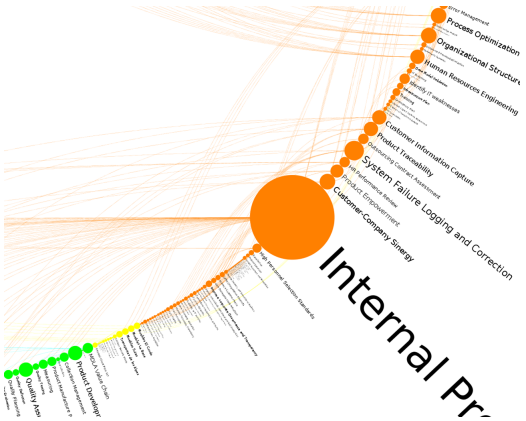

Figure 3. Detail of an example of the EVE and SKC principles.

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 from part of the user. This brainpower can be used instead for extracting conclusions from the model.

2) ***Enhance Visual Expresiveness (EVE):*** This requirement refers to a proper use of the visual vocabulary supported by the mapping of model elements to visual variables (see Section **??**).

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

*3) **Minimize the noise (MN):*** A designer should avoid including information that is not relevant to the message he seeks to transmit [**?**]. 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 visual variables and distorting the intended message [**?**].

Multiple edge crossings and overlapped symbols can be a form of noise, as they make difficult the process of decoding visual information. Complexity management mechanisms, such as different layout algorithms (see Fig. **??**) are required. 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.

*4) **Navigability and interaction (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 [**?**], 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.

*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 [**?**], so we need mechanisms for encoding large models without losing 'the big picture'. Focus-plus-Context [**?**] views are a good example of this.

*6) **Maintain model integrity (MMI):*** Some tools allow information loss between transformations from semantic to visual models, which means visualizations are incomplete or inaccurate. This affects analysis tasks, as the user has to deal with missing information.

Flexible analyses require dynamic queries about elements and their attributes, so the way tools handle data is relevant in this context. Mechanisms that allow the user to ask different questions about the model are important, as well as the language used to formulate them. Queries need to be flexible, in order to be able to extract new information.

*7) **Guarantee Semantic Correspondence (GSC):*** When designing visualizations, there is always a set of objectives in the mind of a designer. Awareness of these objectives can differenciate a successful from an arbitrary visualization [**?**]. However, the tools available mediate with these goals, making more difficult (or less) to travel all the way back from visual meaning to the semantic model. As visualizations get more complex, the user can lose the notion of these objectives, so tools may need to offer the user a mechanism for remembering this correspondence between the graphic and conceptual domains. For instance, diagrams should include legends that help the user mapping visual variables and symbols into meaning, in the same way [**?**][**?**] that cartographers did over the past centuries.

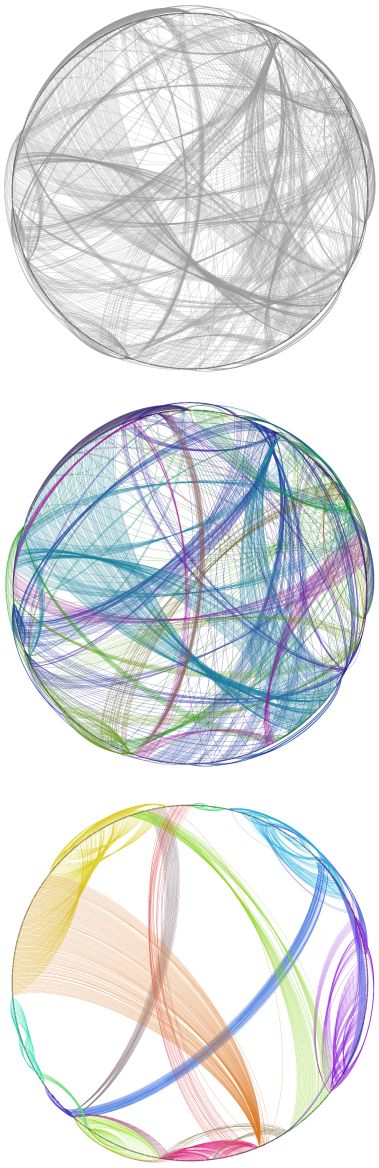

Figure 4. Using different layouts and changing relative position of elements can reduce noise and improve understanding.

C. EA-specific Requirements

*1) **Domain Hierarchy (DH):*** Domains are logical abstraction elements that group common concepts, allowing to separate the universe of discourse into manageable partitions (See Fig. **??**. For instance, classical EA frameworks are divided in Business, Applications, Technology and Information layers. However, these layers often are incomplete, inconsistent or not quite rigorous [**?**]. Visual differenciation between them is necessary, as this hierarchy gives meaning to the model. Also, the real value of most EA metamodels lies in the interdependency relations between these domains,

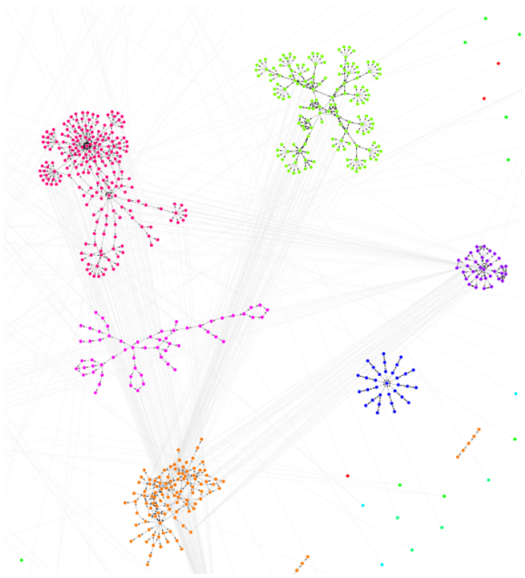

Figure 5. Example of the Domain Hierarchy principle. Each cluster represents a domain, organized with a Force Directed Layout and with outgoing direct arcs for inter-domain relations.

as they provide a bridge between the layers.

*2) **Structural Key Concepts (SKC)**:* There are concepts that have the highest semantic importance in the metamodel (i.e. Service, Process, Role, or Organization Unit), and keep the model graph connected. These concepts may appear in different EA frameworks and metamodels, and may have more significance than supportive concepts such as Activity, Primitive Data Type, Sales Order, etc.

It would be interesting that these *Archetypes*[1], or Structural Key Concepts had a visual emphasis or distinction.

*3) **Focus of Interest (FI)**:* As it happens with the *depth of field* in photography, an analyst would want to emphasize certain message on a given context, while less important elements are also present but unfocused. Views have in some way an implicit focus of interest, as they cover one or more stakeholder concerns by selecting suitable enterprise model elements and bundling this selection as a viewpoint [**?**].

However, this focus of interest should be dynamic, as context may be variable in some analyses. A limitation of this requirement is that focus on an element implies losing information about its surroundings[**?**], so it should be complemented with the General Visualization Requirement *Keep the Context* (KC).

*4) **Structural Diagnosis (SD)**:* From a MDE perspective, models are created under a set of rules (that can be represented by constraints such as the Object Constraint Language) and a common language, their metamodel. Analysts and modelers may need flexible graphical validation

[1]Common concepts or patterns that are universally recognized and understood

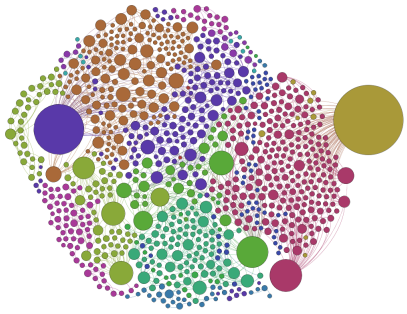

Figure 6. Example of the Architectural Qualities principle.

and visual detection of structural anomalies (e.g. metamodel conformity, attributes without values, duplicate objects, etc.).

*5) **Semantic Characteristics (SC)**:* Another method for making statements of an architecture is the inclusion of mechanisms that add new information to existing statements. This is a form of specialized queries, where the analyst can extract semantic information that is not directly available from the model. EA management tools commonly provide the user with the possibility to introduce visual elements without defined semantics in the context of the visualization[**?**], so this is an interesting requirement that may need further exploration. For instance, it would be valuable to highlight elements that will change in the 'as is'-'to be' architecture transition.

*6) **Architectural Qualities (AQ)**:* Assessments about architectures 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 them. For instance, an analyst would have questions about the orientation of an enterprise architecture; i.e., Which domain/layer is predominant in the architecture? If each domain has a color, which color describes better the architecture? (See Fig. **??**)

This requirement could be further developed with application of EA Management Patterns [**?**], and their corresponding visual validation.

*7) **Metamodel Agnosticism and Evolution (MA)**:* EA conceptual models should not come with visualization information within them; this information needs to be decoupled [**?**] [**?**] from their metamodels, as representation information adds additional noise to the model when it is defined in an implicit manner (as annotations or as concepts). In order to do so, visualizations should be agnostic of the metamodel, as they should have their own visualization metamodel. This makes possible transformation from a conceptual model to a visualization model.

Metamodel evolution is also a critical aspect. Ever-changing business needs often imply enterprise-specific models that also change over time [**?**][**?**]. This means that

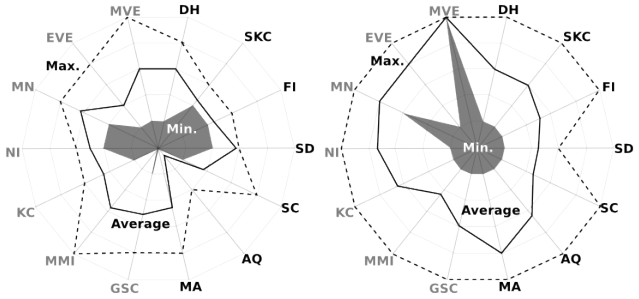

Figure 7. Overview of the EA and Visualization evaluation results.

a static metamodel is not always the best option for an architect, so flexibility in both definition and visualization of EA metamodels is also required.

IV. Evaluation

A. Overview

In order to allow visual comparison of the different tools, we make use of Kiviat diagrams [**?**] (also known as starplots [**?**]). 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 General Visualization Requirements, and should be read counter-clockwise.

An overview of EA and Visualization tool evaluations is shown in Fig. **??**, showing minimum, average and maximum values for each requirement. Tool evaluation results are then displayed (see Fig. **??** and Fig. **??**).

B. Case Study

Muebles de los Alpes[2] 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 complete Enterprise Architecture, among several other features. 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, ArchiMate, BPMN), or are customized representations of standards, frameworks such as Service Oriented Architecture, or even formalizations 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 a set of 11 visualization tools. Our approach is based in [**?**][**?**][**?**], 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. GraphML, Dot, CAIDA, or even Java). This often required designing or searching for the metamodels of these languages.

[2]If the reader wants to know more about this case study, more information is available in: http://muebles.losalpes.com.co

ID	Tool	Vendor
BA	Architect	BizzDesign
ISA	Rational System Architect	IBM
SEA	Enterprise Architect	Sparx Systems
I	Iteraplan	Iteratec
SAMU	SAMU Repository	Atoll Technologies
CCM	Corporate Modeler Suite	Casewise (SAP)

Table II
Selected EA modeling tools

ID	Tool
GV	GraphViz
P	Prefuse
GS	GraphStream
G	Gephi
C	CIRCOS
W	Walrus

Table III
Selected Visualization tools/frameworks

C. EA Tools

We selected 6 common EA management tools (see Table **??**) for evaluation. We noticed that each tool seeks differentiation by offering diverse sets of features, thus proposing a different paradigm of EA modeling and visualization. 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 0 implied that their paradigm gave no clues about a requirement or their approach did the opposite of the requirement. A score of 3 was given if there were signs of an effort to resolve issues that each requirement described, but improvements were needed or their vision was not totally clear. A score of 5 was given if there was a conscious (and in most cases successful) effort to address a requirement.

D. Visualization Tools

We selected six (see Table **??**) visualization tools or frameworks for evaluation of the requirements. Criteria for selection included capability to process the model[3] and

[3]With the exception of one tool - Walrus. This tool was released in 2001, with its last version released in 2005. It was not possible to process our model with this tool, but we considered its inclusion as it offers exceptional navigational capabilities. We analyzed this tool with another data set similar to our model.

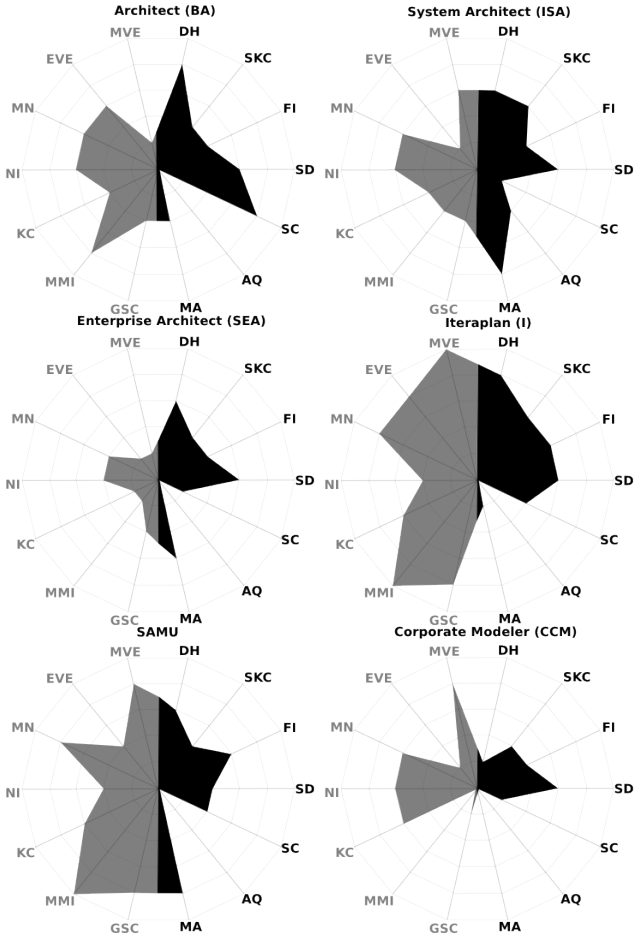

Figure 8. Evaluation results for EA tools.

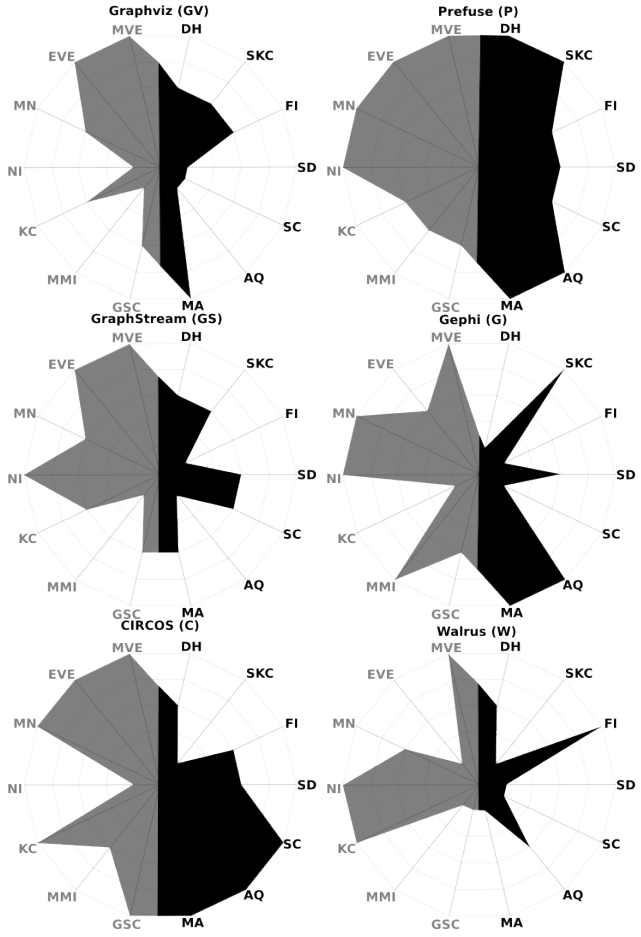

Figure 9. Evaluation results for Visualization tools.

potential expressiveness. As they are general purpose visualization tools, we relaxed our measurements to three possible values: 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.

V. Conclusion

In this paper we have highlighted 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. However, the objectives of this study do not include offering a judgement of which tool is better than another. 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, assessment that is not complete if other (non-visual) requirements are not taken into account.

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 from EA tool vendors.

On the other hand, visualization tools offer as a whole an almost complete set of features for supporting visual analysis of enterprise models. An hypothetical EA specialized visualization tool should incorporate the different approaches of these tools, or even serve as a common platform that uses those tools, serving as intermediary.

Last, but not least, visual analysis is a field with a lot of research opportunities. We can say that nowadays there is a clear effort to formalize the different aspects involved in this field, and tendencies such as Model-driven visualization[**?**] are key in this context. For instance, an interesting research direction is the design of a visualization metamodel that includes the mapping between analysis questions and visual variables, or in other words, that models the intent of giving semantic significance to visualizations.

[**?**]