Abstract. Lately, organizations have been struggling with representing their service architectures due to problems such as lack of communication between the various departments and a common language to represent that architecture. There is also a big problem regarding the size of the architecture that becomes almost incomprehensible to visualize it. In this thesis I will do a research on what has been done regarding this theme, what the discipline of Enterprise Architecture has done to support the visualization of Service Architectures, what the frameworks to model it are, and what tools can be used to help us develop a solution. We provide a solution to what we think it is fit for this problem. This work has the goal of improving service architecture visualization in line with the main Enterprise Architecture frameworks and guidelines. This work was inserted in the context of the organization Serviços Partilhados do Ministério da Saúde.

1 Introduction

Organizations have been growing and IT is keeping up with it. Due to this, there’s a need to developed and maintain the IT service architecture. Various departments in the organizations developed their own model but this leads to several problems such has different languages to represent the same architecture, redundancy of information and problems of communication between departments. Several stakeholders have different needs inside the organization, and at the moment, it’s not possible to address those needs in a single visualization model. This work have as its foundation the discipline of Enterprise Architecture [1] whilst providing the main concepts and domains. This work was developed inside the context of the organization Serviços Partilhados do Ministério da Saúde.

1.1 Problem

In the Serviços Partilhados do Ministério da Saúde, there’s a lack of communication across the various departments and there’s no agreement on how to represent their service architecture. The large heterogeneity of stakeholders leads to different needs regarding the visualization of the architecture such as different level of detail and viewpoint. Since the service architecture is constantly changing, there’s a need to create different representations and create a framework of visualization that can be flexible and easy to understand by every stakeholder. The large density of information is also a problem to take into consideration since large organizations have large service architecture with high cardinality of entities and it can be quite difficult to manage it.

1.2 Objective of work

We can summarize the objectives of work in gathering and consolidating knowledge regarding this visualization problem. Based on that knowledge obtain the critical points that helped us developing a final solution to address. After developing a theoretical solution, it was implemented in the EAMS (Enterprise Architecture Management System), which is owned and maintained by Link Consulting.
1.3 Research Methodology

Before starting the developed the solution we had to specify and structure how we would lead the research of information to approach this problem. We followed one of the most used methodologies for research in computer science, the *Design Science in Information System* [2]. This framework provides the guidelines and principles in order to develop and validation a solution to the problem proposed.

Fig. 1: SPMS layered blueprint with an high cardinality of instances and relationships

The figure 1 shows an agglomerate of the current problems this thesis had to solve.

2 Related Work

The problem lies in developing a service architecture visualization framework, so in this chapter, we give an overview of what has been done in this field of study. The analysis of this work gave us some insight to solve our problem. Initially, we provide an introduction to EA (Enterprise Architecture), what is a service and what kind of modeling frameworks and tools exist in the current present.

2.1 Background work

The background work focus primarily in defining what is Enterprise Architecture, what are its domains, what is a service and what is a service architecture.
2.1.1 Enterprise Architecture

Enterprise Architecture is a "well defined practice for conducting enterprise analysis, design, planning and implementation, using a holistic approach at all times, for the successful development and execution of strategy", according to the The Federation of Enterprise Architecture Professional Organizations [3]. It provides a set of principles and guidelines to manage the organization according to four domain: Business, information, application and technology.

2.1.1.1 Enterprise Architecture Domains

These four domains are represent as architectures in the various Enterprise Architecture frameworks. TOGAF [1] provides the following definitions:

- Business Architecture - defines the business strategy, governance, organization and key business processes.
- Information Architecture - provides the artifacts, models, relations and standards on how the data is manipulated by the Application Architecture.
- Application Architecture - describes the structure and interaction of the multiple applications as groups of capabilities that provide key business functions and manage the data assets.
- Technology Architecture - it provides the set of components, whether physical or logical, on which the global architecture infrastructure will be set.

2.1.2 Service Definition

According to Terlouw [4], "a service is a pattern of coordination and production acts, performed by the executor of a transaction for the benefit of its initiator, in the order as stated in the complete, universal pattern of a transaction. When implement it as the ability to get to know the coordination facts produced by the initiator and make available to the initiator the coordination facts produced by itself". But according to The Open Group in the SOA Reference Architecture [5], which is focus to the IT service, a service can be divided into logical and physical services. A logical service is just a representation of a repeatable activity that has a specified outcome. A physical service realizes the functionality of the logical service.

2.1.3 Service Architectures

Applications have been decoupling their interfaces from the implementation since it's not practical to make future integrations, so the SOA Reference Architecture provides key principles [5], that will help envision the solution, in which it can be outlined the following principles:

- Be a generic solution, so we should be able to apply this to any organization
- Promote and facilitate the alignment between IT and business
- Address multiple stakeholder’s perspectives
- Minimize the number of layers, to reproduce all possible combinations and elements of the architecture

2.2 Specific work

With the background work specified, we can now go into the details of actual related work that can help us.

2.2.1 Modeling frameworks

We approached several modeling frameworks, but I will talk about the ones that had some use for our work.

2.2.1.1 Archimate

This modeling framework is a visual language to describe the enterprise architecture of an organization [6]. Archimate implements the concept of layering and it’s based on the Enterprise Architecture discipline. The three layers are Business Layer, Application Layer and Technology Layer.

Archimate also provides a list of relationships that the user can use to specify relations between entities.

Archimate also provides a set of views and viewpoints. A view is a part of the Architecture Description that addresses a set of related concerns and is tailored for specific stakeholders. A viewpoint provides the concepts, model, analysis, techniques and visualizations. It specifies the concern of a
certain stakeholder and it provides the visualization the stakeholder wants and needs to see. But this framework alone can’t solve our problems, otherwise there wouldn’t be a problem. Archimate is a modeling framework specific for developing enterprise architecture visualizations, which would be ideal, but it’s a very technical language that doesn’t appeal to the large set of stakeholders nor it provides navigation mechanisms. Large organizations with large service architecture will have Archimate models that are unreadable due to the enormous size those visualizations get.

2.2.1.2 TOGAF As stated before is one of the most used enterprise architecture framework but had little use for our work, so I will not talk much about it since TOGAF is not a modeling framework but an enterprise architecture framework, so it provides guidelines to develop architectures and the correspondent visualizations, so we can only draw some guidelines that can help us developing a solution.

2.2.2 Modeling Tools In this section, we will refer a few modeling tools that are widely used to model enterprise architectures.

2.2.2.1 Enterprise Architecture Visualization Tool Survey This is a technical report that contains a wide state-of-the-art in EA visualizations and its tools to support it [7]. It has a vast view-driven approach to EA modeling that states how to create and configure a visualization, what are the visual parameters, the layout, storing and restoring a configuration, set up and adapt the EA information model and data importation. It is divided in the following guidelines:

- Create and configure a visualization
- Create the visualization
- Visual parameters
- Layouting
- Set-up and adapt the EA information model
- Visualization Types

This is a vast chapter and it’s a wide and useful report that gives us an insight of what is being done in this field regarding visualizations since provides a set of various languages and methods to develop several languages, either technical or more intuitive for non experience users, but it doesn’t provide a method to unify a set of languages into one that can be adaptable to all stakeholders and deal with large organizations.

2.2.2.2 EAMS - Enterprise Architect Management System EAMS is a software developed by Link Consulting for managing enterprise architectures within an organization. It works as a catalog for services but it has a visual component to represent it. This modeling software uses the concept of Blueprint. A Blueprint is a graphical representation created according to a set of rules defined by a "Blueprint Type" and the parameter provided. Blueprint Type is the implementation of the blueprint that can be designed through a visual tool designer integrated in the EAMS or define the XML that represent the blueprint structure [8].

![Fig. 2: Blueprint Example](image-url)
In the figure 2 there’s a visualization of the blueprint and you can see the correlation between the blueprint designer and an actual representation of it. These blueprints are static and the interaction with the visualization is quite limited, since you can only navigate to other blueprints by defining a set of navigation rules depending on the data type.

EAMS, being an enterprise architecture management and visualization tool, it’s not flawless and it doesn’t address all problems regarding the visualization of the service architecture. One problem that organizations have and EAMS can’t handle is the large sets of entities and there’s no mechanism to address high levels of cardinality.

2.3 Considerations

All of the tools and frameworks have different approaches to visualize an architecture, but there’s not a single one that can solve the problems at hand. The problems can be summarized in:

- High cardinality of instances
- High cardinality of relationships
- Disperse information
- Heterogeneity of stakeholders

Since the solution was implemented in the EAMS, we will only talk about it. As stated before, EAMS doesn’t have any mechanism with dealing with high cardinality of instances or relationships. Disperse information is a vague term but it means that the modeling tool/framework can provide mechanisms to create and visualize service architectures easy to read and interact with the visualization according to a set of properties of the data instance in question. EAMS have the option of designing blueprints so the user can arrange it in design-time, but in run-time it’s not possible. Luckily the fact that EAMS allows to develop different types of blueprints with different levels of detail so the heterogeneity of stakeholders is already addressed.

3 Solution Proposal

In this section, the goal is to present a set of features that aim to modify the representation of the service architecture. These modifications will improve the readability, the amount of information displayed to the user, allow the user to navigate according to the metamodel of the architecture and present a set of maps to visualize the service architecture of the enterprise Serviços Partilhados do Ministério da Saúde. The solution will be based on the Graph Theory, which is the structure to allow us build blueprints and it was integrated in the EAMS application.

3.1 Collapse containers

This consists in collapsing a container of data instances. This allows to hide containers with high cardinality of instances that makes the map more simple to understand and navigate. One of the problems of service architectures stated before, is the large cardinality of data instances in the blueprints, which makes the blueprint useless since the readability is affected and the user can’t understand the point the blueprint is trying to convey. The idea was to modify the initial algorithm that calculated the layout of the blueprint in the canvas, so that we could collapse the containers that had a certain amount of symbols. The algorithm was quite simple. While iterating the graph, verify the number of symbols (entities) inside, and if it’s larger than a previously specified \( \alpha \), then it called a function called \( \text{collapseContainer}(v) \) where \( v \) is the node of the graph. This function \( \text{collapseContainer} \) was recursive in the way that we had to re-fit all brothers node and its parent and recursively till the algorithm reaches the root. The following figure 3 shows a set of collapsed containers.
We also give the option to the user to expand or collapse any container he wishes as show on figure 4.

3.2 Filter blueprints based on properties

When stakeholders are visualizing the architecture, they need to identify instances according to certain properties. So we developed an algorithm that iterates the graph and simply applies the condition the user inputed. When the condition is met, it highlights the instance.

There was some discussion on how to deal with blueprints with multiple classes (data types), so we decided that the user can choose to filter by all classes or just by a specific one, giving the user the choice. The interface consists in the user specifying what classes he wants to filter, then selects the property of that class (in the case of all classes, the property list is all properties that are common between all of them), then the operator (this operator is different based on the type of property) and finally the value on which the algorithm will compare the values of the selected property.

In the following figure 5 there’s the filter interface.
But the user might want to apply more than one filter due to the diversity of information across the blueprint, so the filters are accumulative. In the following figure 6.

3.3 Sorting instances by property

According to Munzner [9], sorting is "a powerful choice for finding patterns in a dataset by interactively changing the attribute that is used to order the data". By standard, the information display is ordered by alphabetic order, and what we propose, is to give the user the choice of ordering the data instances by a property they see fit. The following figure 7 is a blueprint graph that is ordered by levels. It will help us to explain how the sorting algorithm works.
Each container has the option of ordering the elements inside it. Whether it is containers or symbols, but that arose some issues. If the user chooses to sort a container, he must face with the fact that inside there could be containers of different data types or even containers that have no data types associated (this is specified in the definition of the blueprint). So, the list of properties to sort, could vary. Also another problem was, should the user be able to sort all levels below the one he selected? To summarize, the problems are: Different data types of containers and containers with no properties associated, due to not being a data type. What was decided was: Containers with no properties associated, will only be able to sort by alphabetic order. And the user will only be able to sort the level below the container he chose. In the figure 8, we integrated on top of the container a button to open the sorting interface so the user can choose the property to sort and then the order, whether ascending or descending.

Property with no value, will be put in the end of the list if descending or the begin if ascending. All properties are divided into five categories: References, text, date, numeric and booleans. Since references and booleans aren’t ordered attributes, we can’t sort it, so in the property list of the interface, those properties never show up.

3.4 Filter the blueprint relations

This feature came up since blueprints with large cardinality of entities, tend to have a large cardinality of relationships, which makes the blueprint hard to understand and read. The relationships of a graph are can be different from the relations from the metamodel. These
relations are specified in the blueprint implementation. There are two types of relations, unidirectional and bidirectional and depending on the blueprint implementation, there can be relations between every type of graphical entity as the following figure 9 shows.

The idea was to divide the origin/destinations into two groups: The generic containers and data types. This means we can choose to only represent on the blueprint, relationships that have a origin/destination in a specific container or relationships that point to a certain data type, for example, relationships that point to Applications. The second approach was to find and outline a path from an entity given a depth, but we will explain later.

3.4.1 Relationships Filter Flowchart The following flowchart 10 depicts the process of interacting with the interface to choose a filter.

The selection of the filter is quite linear. The user first selects the origin type of the relation. He then selects if the relation is unidirectional, bidirectional or all directions. If it’s unidirectional the arrow go from the origin to the destination. If it’s bidirectional, then the origin/destination order is irrelevant. After the user selects the direction, he then selects the destination type as he did for the origin.

3.4.2 Calculate relationship path of an entity given a depth In a service architecture there’s a great component of integration between services, which creates a lot of dependencies
between services, so when a service change, the user wishes to know the impact it will have on the other services that depend on it and with a large cardinality of relations on the blueprint, it’s hard to determine what are the data instances that depend on a given data instance.

So we implemented an algorithm that search in depth given a direction, whether going towards to node or outwards, or bi-directional. The following graph 11, shows how the algorithm works. It highlights the path for a depth of 2, and going outwards.

Fig. 11: Graph with relations and data instances highlighted for a depth of 2

The following figure 12 shows how the interface for filtering is.
3.5 Defining blueprints to support the SPMS organization

All features specified before are generic solutions that can improve the user experience while visualizing any kind of blueprint in any organization but since we’re doing this thesis in the context within the organization, there was a need to develop some maps, or blueprints, that were fitted to the organization’s internal structure. The SPMS, Serviços Partilhados do Ministério da Saúde, is an organization that develops IT solutions for the public health care institutes. The organization wishes to visualize their service architecture according to the internal organizations, so before presenting the developed blueprints, we need to provide an insight of the internal organization of the SPMS.

The internal structure of the organization is divided into several departments that are aggregated by the DSI - Direção de Sistemas de Informação. Each of this department develops IT solutions. Then, we have the users, the ones that will be using those IT solutions. Basically, it’s a group of health centers, local unit of healths and hospitals that use those solutions. From this information, and always in line with the stakeholders at the SPMS, we developed four blueprints that satisfy different needs, depending on the stakeholder. The first blueprint 13, depicts the internal organization of the users and how they divide their solutions by type. They also distinguish between solutions developed by SPMS and external providers.
The next blueprint 14 is simply the organic structure of health units inside the health care environment.

This blueprint 15, is a business process layered, so it shows the interaction between business processes and the IT solutions, and what kind of artifacts and actors interact with it.
And finally the last blueprint 16 is simply the internal organization of the IT solutions inside the SPMS.

3.6 Final Considerations

The solution presented consisted in a set of features to help visualize any service architecture of any organization, since those are generic solutions that meet the problems the SPMS has. The
filters give the user a easy tool to explore the architecture. The collapse allows the user the hide or show information of interest, or not, to him. The sorting feature, gives the user the a way to compare instances given a certain property, for example, the user could sort a set of data instances by cost, from the cheapest to the most expensive.

The relationships filter weren’t implemented due to time and software constraints, but the value it adds is the ability to give the user the choice of only showing some relationships based on certain properties, in this case, the containers where they point and the type of relations.

The last feature developed, the Calculate and highlight path based on depth, is more than a visual feature. It actually allows the user to see the dependencies of each data instance, and how far it affects other entities. Before this feature, it was literally impossible to determine those entities on blueprints with large cardinality of instances and relationships. But the most important feature were actually the maps developed, since they portray the architecture of the organization and help the stakeholders to have an insight of the organization. The features developed help the user to navigate through the maps. A map that doesn’t accomplish a purpose and doesn’t met a stakeholders need, could have all the features to navigate but it would still be useless.

The following table 1 now show how each problem is approached by each feature.

<table>
<thead>
<tr>
<th>Problems</th>
<th>Features provided by EAMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>High cardinality of instances</td>
<td>- Collapsing containers</td>
</tr>
<tr>
<td></td>
<td>- Filtering instances</td>
</tr>
<tr>
<td>High cardinality of relationships</td>
<td>- Filter relationships by container type</td>
</tr>
<tr>
<td></td>
<td>- Calculate and highlight path based on depth</td>
</tr>
<tr>
<td>Disperse information</td>
<td>- Filtering instances</td>
</tr>
<tr>
<td></td>
<td>- Sorting instances</td>
</tr>
<tr>
<td>Heterogeneity of stakeholders</td>
<td>- Blueprints designer</td>
</tr>
</tbody>
</table>

4 Work Evaluation Methodology

To evaluate the proposal solution I’m using a research paper that is based on the Design-Science Research [2], the Artifact evaluation in information systems, Design-Science Research - A holistic View [10].

This paper focuses on the artifact evaluation in Design-Science Research. It provides guidelines and evaluation models that can be used as templates to assist us on the evaluation of the solution. There’s an hierarchy of criteria that complies multiple approaches of evaluations, depending on the type and purpose of the artifact.

5 Demonstration

For the demonstration, we made a case study in the Serviços Partilhados do Ministério da Saúde. It consisted in running some users tests with some tasks. We set up four tasks to evaluate the usability of our interface. Each task was to use one feature of what was implemented.

We used the System Usability scale, which is a scale from 0 to 100, that assesses the usability and learn-ability of a given system, based on ten questions answered by the users [11]. For the user inquiries, we had four users to test our system and then answer the SUS test. For four users, the results are quite promising but the more users we use to test the system the more trustworthy the results are. The global average score was 85.63, and according to Bangor, Aaron & Kortum, Philip & Miller, James [11], this SUS score is placed on a B grade scale which is considered to be an acceptable grade. This means the system achieved most of the users objectives and expectations for the system.
6 Conclusion

In the research of the related-work, it became more clear that there are no consensus regarding the modeling the service architecture in the organizations, and it’s due to many reasons, like lack of technical knowledge, communication and different conceptions of the same entity. This work tries to develop some features and maps to create visualizations that aid the user to visualize those architecture representations. Those features consist in filtering instances based their properties, filter relationships based on the type of relations and the origin and destination type. We developed a sorting interface that allows the user to order a set of instances based on a property. And developed some blueprints that covers the architecture of the organization and provides various maps with different perspective that approaches the heterogeneity of stakeholders. The calculating and highlighting path was an important feature that’s more than a visualizing feature mechanism but a way to navigate across the metamodel and check dependencies between each entity. All of those features and blueprints can be mashed up together in a single representation and improve the visualization experience for the user. To test those features we run some user tests to verify what the users think of the system and the feedback was quite positive with some issues raised.

6.1 Future work

With the issues raised from the user testing, we will have to verify how to make the filter interface more intuitive for the user, so we will ask the users for their opinions. The sorting icon will also be changed to a more intuitive one. And the feature that wasn’t implemented, the relationships filter would be implemented when the software constraints were solved. We still need to continue to monitor the users in order to verify what other issues could arise in order to improve the system.

References